Contents

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 1: Introduction and Course Description

Sept. 6 2006

These notes and all course materials will available at http://pruffle.mit.edu/3.016-2006. Students should bookmark this site use it to download lecture notes, homework assignments, and reading assignments for laboratories and lectures.

The web materials for 3.016 are revised each year. This year I am doing substantial revision on how the lecture notes are designed and some revision on lecture content. The site will develop throughout the semester.

Previous years' notes are available at `http://pruffle.mit.edu/3.016` and may be useful to you.

Preface

The subject is for undergraduate materials scientists and engineers who wish to learn about the mathematics that is essential to their chosen field.

Materials science and engineering is a discipline that combines knowledge of chemistry, mechanics, and physics and then applies them to the study of materials and their properties. It is a challenging and diverse enterprise—obtaining expertise in a large set of diverse subjects—but a career devoted to diverse study and applications will be very rewarding and fulfilling.

Mathematics is the *language* that binds together disparate topics in physics, engineering, and chemistry.

While it is possible to become an excellent materials scientist and engineer without some working knowledge of a large subset of mathematical topics, it is much easier to master this discipline with mathematics to guide you. Through mathematics, you will discover that some topics have similarities that are not obvious—and not taught to you as *being* similar. Such similarities and analogies will make learning much, much easier—and I think much more enjoyable.

MIT's Department of Materials Science and Engineering has determined that students benefit from a background and a working knowledge of many more mathematics topics that pertain specifically to MS&E than are taught in a one-semester subject in the mathematics department. It is reasonable to ask, "Is this subject a substitute for a mathematics subject such as linear algebra or partial differential equations taught by a mathematics professor as part of the mathematics department?" No, this is not a replacement for such a subject and I encourage you to take subjects in mathematics in the future.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

This subject is designed to be very broad in scope and therefore its depth in any one topic is limited.

I do believe very strongly that you will enjoy studying math more after taking this introduction and that the mathematical background you will receive this semester will make your materials science education richer and more rewarding.

I have designed this subject to help you learn as much essential mathematics as possible in a short time. To this end, this subject has several unusual aspects that you will need to know.

3.016 Mathematical Software

Symbolic mathematical computer software is a tool used by almost every applied scientist. Such software helps produce results quickly, visualizes and documents the results, and minimizes the silly errors that creep into complicated mathematical manipulations. Although there are many other good choices, I have decided to use MATHEMATICA® as a vehicle for learning and doing mathematics. It has a fairly steep learning curve, but it probably repays the time investment with powerful (once learned) language syntax and packages. Such symbolic mathematical software is an aid to help you think about and perform mathematics—it is not a replacement for mathematical understanding.

MATHEMATICA® is available for all MIT students, both on Athena (free) and via licenses for personal laptop and desktop machines ($30 for students, a useful investment for other subjects). The process to access MATHEMATICA® on Athena and the steps to download a license will be explained to you; the pertinent website is http://web.mit.edu/is/products/vsls/ http://web.mit.edu/is/products/vsls/. You will need MATHEMATICA® for your first homework set and laboratory; you should try to get it working someplace very soon. If you have a laptop, I suggest that you install MATHEMATICA® on it as soon as possible.

Laboratory assignments must be completed during the laboratory period and are to be emailed as an electronic copy of a MATHEMATICA® notebook to the instructor and the TA.

3.016 Examination Philosophy

Tests and exams are powerful motivators to get students to take a subject seriously but I think that working through homework problems better promotes learning, particularly for self-motivated students.

Therefore, there will be no exams, tests, or quizzes in 3.016. Your grade will be based on your homeworks and laboratories. These will graded carefully (described below) and there will be about one

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

homework set every week and a half while the subject is meeting (i.e., no homework will be assigned during the weeks that the laboratory 3.014 meets).

3.016 Homework

The purpose of the homework is to help you solidify your understanding of *mathematics applied to engineering and science problems* by working through examples. Some examples may be exercises in mathematics; others will be exercises in application of mathematics to solving engineering and science problems. I encourage you to use MATHEMATICA® to solve your homework problems, and you may turn in solutions as printed MATHEMATICA® notebooks. Nevertheless to appreciate what symbolic mathematics programs can do for you, there will be some exercises that I will ask you to do with pencil and paper. However, there is no harm in checking your “by-hand” results with MATHEMATICA® .

Of course when you do homework, you are not under the potentially menacing eyes of an exam proctor. This means that you can receive help in the form of:

Books Go to the library and find solutions to problems. It is good practice and you will learn quite a bit by doing so. I recommend that you attempt to find a solution before going to the library—not only will it help you appreciate the solution, it will also make your search a bit easier!

Experts By all means, consult with experts on your homework. It is a good idea as long as you understand what *you* turn in.

Classmates This is the best choice of all. I think it is both inevitable and beneficial to give and receive help. Cooperating on homework will help you learn to communicate your ideas and begin to appreciate the difficulties and rewards of teamwork.

As explained below, the homework assignments in 3.016 will be, in part, cooperative.

You will find that you are more busy some weeks than others and relying on a classmate during a busy week can be a life-saver. However, if you start slacking off and don't hold up your end of the bargain when you are able, you will engender resentment and endanger professional and friendly relationships. I leave it to your own conscience to play fairly and contribute when you can—and, while understanding that everyone experiences different kinds of pressures, to be forthright and honest with others who do not contribute consistently.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

It is fairly easy for the instructor to ascertain who is slacking and who is not. I can't say that my good opinion has any particular value, but keep in mind: it is possible, but slackers will have hard time regaining my good opinion.

If you turn in work that youdid not do, and do not attribute the solution to its rightful author(s), then you are plagiarizing. As a first assignment in this course, every one of you should read MIT's policy on academic integrity (html) or (pdf) immediately.

Homework cooperation has a potential downside because you all receive individual grades. We will attempt to mitigate this downside by dividing the homework into two parts:

Group For each homework set, a few problems will be designated as *Group Exercises*. For these problems, the entire group will turn in *one homework*. Every member of the group *who puts their name on the turned-in assignment* will receive exactly the same credit for the homework grade.

Homework groups will be *assigned* with each homework set. The groups will change from week to week and the members will be assigned randomly. Each group will be assigned a homework leader who will be responsible for arranging meetings and turning in the homework.

Individual Each problem set will contain a few problems for each student to complete individually. These problems will come out of the textbook and tend to be a bit easier than the group exercises. They are designed to maintain a sufficient amount of currency and emphasize that reading the textbook is an essential part of this course.

By putting each individual's name on a homework assignment, the group verifies that each indicated person has contributed to the assignment. By putting your own name on the group's turned-in assignment, indicates that you have reviewed *all* of the assignment; if questioned, each person should be able to describe how each problem was done. MIT's policy on academic integrity is also the policy for 3.016.

3.016 Laboratory

There will be a laboratory each week that 3.016 meets. The labs will be practical and focused on using MATHEMATICA® effectively.

There will be assigned reading from the MATHEMATICA® help browser that comes with the software. *You should always do this reading before the laboratory, or you may not be able to* FINISH

YOUR ASSIGNMENT AND TURN IT IN BEFORE THE END OF THE LABORATORY IN ORDER TO GET CREDIT.

If you stay current in the course material and do the homeworks, you should have no difficulty doing the laboratory assignments *if you do the pre-assigned reading.*

It is not necessary, but if if you have your own laptop running MATHEMATICA® , it will be helpful to bring it to the lab with you.

Grades

As stated above, all of the final grade will depend on the homeworks and the laboratory assignments. There is no fixed average grade for this course; the average will depend on the entire class performance. However, if your homework grades and your laboratory reports are consistently within the top quartile, then it is extremely likely that you will receive an **A**. Homeworks will be graded by ranking them in order from *Best Homework* to *Least Best Homework*. A decision will be made regarding how many points (out of a possible 100) the *Least Best Homework* deserves, and the homework scores will be interpolated between a score of 99 for the *Best Homework* and that of the *Least Best*.

Homeworks will be evaluated on the basis of:

Accuracy The solution must be a reasonable and correct answer to the homework question.

Exposition The solution must clearly show the reasoning that was utilized to find it and the method of solution should be clearly apparent. Exegetic solutions will be ranked higher.

Beauty Good solutions will often require graphics and, with care, graphics can often beautifully explain the solution. The layout of the page, the quality of the supporting prose, the clarity of the graphics, and all that "je ne sais quoi" is fairly subjective but very important. The reader will include a judgment of your art in the ranking of homeworks.

Observation Supplemental observations provide aids in understanding and demonstrate mastery of a topic. An example of a supplemental observation might be something like, "Note that in the limit of long times, that the total concentration goes to zero. This is sensible because the boundary condition on mass flux is directed outward everywhere on the finite domain."

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Laboratories will be graded on their completeness, demonstrated mastery of MATHEMATICA® for that assignment, and exposition.

Note that there will be times when you have two homework sets pending—this is done so that you can arrange your time conveniently.

Homework Calendar and Weighting

Homework Schedule			
Homework	**Available**	**Available**	**Due**
Assignment	**After**	**Date**	**Date**
Set	Lecture	Date	Date
1	Lect. 1	6 Sept.	14 Sept.
2	Lect. 4	13 Sept.	28 Sept.
3	Lect. 6	27 Sept.	12 Oct.
4	Lect. 13	23 Oct.	9 Nov.
5	Lect. 17	8 Nov.	30 Nov.
6	Lect. 21	20 Nov.	7 Dec.

Late Policy

Students will be allowed to turn in one homework up to 3 days late, for the individual portion only. No second late homework will be allowed without formal documentation about an unforeseeable emergency. No late group homework portions will be accepted—no exceptions.

Laboratory assignments must be turned in during the laboratory period. You must show documentation of unforeseeable emergencies that prevent you from attending a laboratory period. Any missed laboratories must be made up by special arrangement. If for some reason, you cannot complete a laboratory during the laboratory period, you should send a paragraph explaining why you could not finish.

It is your responsibility to do the assigned reading before the laboratory.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Textbook

We will use a fairly general textbook on applied mathematics (E. Kreyszig, *Advanced Engineering Mathematics*, ninth ed., J.W. Wiley, ≈ 1200 pages). You'll notice that reading assignments do not follow the table of contents—while I like the book, there are pedagogical reasons for studying mathematics in the sequence we will follow in this subject. Extra material pertaining to materials science specifically will be created and placed on the web.

I have identified 66 sections of the book (330 pages in total) as required reading. The readings for each lecture will appear in the Lecture Notes and always posted on the web at: http://pruffle.mit.edu/3.016-2006. I hope you will keep up with the reading—I think it would be wise to give the material a cursory reading prior to the lecture and then read it more carefully before starting the homework.

This course is designated as a 12 (*3-1-8*) unit subject[1] Time spent awake at lectures and recitations is less than half of your job—reading and doing homework is the greater part.

Lecture Notes

Lecture notes (like these) will be available for you to print out for each lecture. The lecture notes will be available at: http://pruffle.mit.edu/3.016-2006. These will supplement (not replace) the textbook. The lecture notes also serve as a guide to help the student understand what parts of the text are considered more relevant or important.

The specific purpose of the notes is to provide neatly typeset equations and graphics that will be used in the lecture along with a few observations. This will eliminate the time required to write and draw, perhaps a bit sloppily, for you in your notes and for me on the blackboard.

The lecture notes will have reading assignments printed at the beginning of each lecture; they will look like this:

Kreyszig **6.1, 6.2, 6.3, 6.4** (pages: 304–309, 312–318, 321–323, 331–336). Part of the units for this course involve reading. You are receiving an expensive education—you should strive to make your education valuable by doing all the required reading. Your intellect will profit even more by doing outside reading.

[1] Units at MIT are assigned under the following schema: *lec-lab-out* where *lec* is the number of lecture/recitation hours, *lab* is the number of laboratory hours, and *out* is the number of outside (reading, preparation, homework) hours per week. One MIT unit represents about 14 hours of semester work on the average.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Those concepts that are fundamental to this course will be presented in lectures by the lecturer (or in the form of welcome questions and points of clarification by the students) and some explanatory notes will be written upon the blackboard.

The notes will have places for you to fill in auxiliary discussion and explanation. Those places will look like this: You can use these notes in several ways. You could print them out before lecture and write your own lecture notes directly during the lecture. You could take lecture notes on your own paper and then neatly copy them onto a printout later. You could print them before lecture and write on them rapidly and then copy—neatly and thoughtfully—notes onto a freshly printed set of lecture. I recommend the latter for effective learning and the creation of a set of notes that might provide future reference material—but do whatever works for you.

The lecture notes will also refer to MATHEMATICA® notebooks available on the 3.016 website for downloading. These notebooks will be used as MATHEMATICA® sessions during the lectures to illustrate specific points and provide examples for you to help solve homework problems.

References to MATHEMATICA® notebooks look like the ones given at the end of this lecture's notes in section 1-0.0.15.

These examples will serve as place-holders in the lecture note when we switch from chalkboard and/or projected display of the notes to a live MATHEMATICA® session.

Lecture and Laboratory Calendar

This calendar will be updated throughout the semester: students should consult this calendar weekly to obtain the required reading assignments for the laboratory.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

. Week of 4–8 September

Lectures		
	Topics	**Reading**
M 09/04	Labor Day, No Lectures	
W 09/06 Lect. 1	Course organization and introduction to Mathematica	Mathematica Notes *I*
F 09/08 Lect. 2	Introduction to Mathematica, assignment and evaluation, rules and replacement, procedural and functional programming	Mathematica Notes *II*

Laboratory		
F 09/08 Lab 1 (Not Graded)	Getting started with Mathematica	*Mathematica Help Browser* **Online Tutorial**

Homeworks		
Homework Set	Available	Due Date
1	Wednesday 6 Sept	

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

. Week of 11-15 September

Lectures		
	Topics	**Reading**
M 09/11 Lect. 3	Mathematica graphics: basic plotting, data, two- and three-dimensional plotting, graphics primitives, formatting	Mathematica Notes *III*
W 09/13 Lect. 4	Mathematica: symbolic and numeric calculations, linear algebra, roots of equations	Mathematica Notes *IV*
F 09/15 Lect. 5	Mathematica: functional programming, packages, and file input/output	Mathematica Notes *V*

Laboratory		
F 09/15 Lab 2	Symbolic calculations and plotting	*Mathematica Help Browser* **Mathematica Book: sections 1.4.2, 1.7.1; Functions: Integrate, Simplify, NIntegrate, Plot, Plot3D, ContourPlot**

Homeworks		
Homework Set	Available	Due Date
1		Thursday 14 Sept
1	Wednesday 13 Sept	

. Week of 18-22 September

3.014 Laboratory Week: 3.016 does not meet.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

. Week of 25-29 September

Lectures

	Topics	**Reading**
M 09/25	MIT Holiday, No Lectures	
W 09/27 Lect. 6	Linear algebra: matrix operations, interpretations of matrix operations, multiplication, transposes, index notation	*Kreyszig* **7.1, 7.2, 7.3, 7.4** (pages: 272–276, 278–286, 287–294, 296–301)
F 09/29 Lect. 7	Linear algebra: solutions to linear systems of equations, determinants, matrix inverses, linear transformations and vector spaces	*Kreyszig* **7.5, 7.6, 7.7, 7.8, 7.9** (pages: 302–305, 306–307, 308–314, 315–323, 323–329)

Laboratory

F 09/29 Lab 3	Solving linear systems of equations	*Mathematica Help Browser* **Mathematica Book Section 1.8.3, Functions: Inverse, Transpose, Eigensystem, matrix multiplication "."**

Homeworks

Homework Set	Available	Due Date
2		Thursday 28 Sept
3	Wednesday 27 Sept	

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

. Week of 2–6 Oct.

Lectures		
	Topics	**Reading**
M 10/02 Lect. 8	Complex numbers: complex plane, addition and multiplication, complex conjugates, polar form of complex numbers, powers and roots, exponentiation, hyperbolic and trigonometric forms	*Kreyszig* **13.1, 13.2, 13.5, 13.6** (pages: 602–606, 607–611, 623–626, 626–629)
W 10/04 Lect. 9	Matrix eigenvalues: eigenvalue/eigenvector definitions, invariants, principal directions and values, symmetric, skew-symmetric, and orthogonal systems, orthogonal transformations	*Kreyszig* **8.1, 8.2, 8.3** (pages: 334–338, 340–343, 345–348)
F 10/06 Lect. 10	Hermitian forms, similar matrices, eigenvalue basis, diagonal forms	*Kreyszig* **8.4, 8.5** (pages: 349–354, 356–361)
Laboratory		
F 10/06 Lab 4	File input/output, plotting data	*Mathematica Help Browser* **Mathematica Book 2.12.7, 2.12.8; Functions: Dimensions, Append. AppendTo, Do, Mean, StandardDeviation, ListPlot, Table, Graphics‘MultipleListPlot, Fit**

3.016

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

. Week of 9–13 October

Lectures		
	Topics	**Reading**
M 10/09	Holiday, No Lectures	
W 10/11 Lect. 11	Vector calculus: vector algebra, inner products, cross products, determinants as triple products, derivatives of vectors	*Kreyszig* **9.1, 9.2, 9.3, 9.4** (pages: 364–369, 371–374, 377–383, 384–388)
F 10/13	3.014 Laboratory, no 3.016 lecture	

Homeworks		
Homework Set	Available	Due Date
3		Thursday 12 Oct

. Week of 16–20 October

Lectures		
	Topics	**Reading**
M 10/16	3.014 Laboratory, no 3.016 lecture	
W 10/18	3.014 Laboratory, no 3.016 lecture	
F 10/20 Lect. 12	Multi-variable calculus: curves and arc length, differentials of scalar functions of vector arguments, chain rules for several variables, change of variable and thermodynamic notation, gradients and directional derivatives	*Kreyszig* **9.5, 9.6, 9.7** (pages: 389–398, 400–403, 403–409)

Laboratory		
F 10/20 Lab 5	Statistics, fitting data, error analysis	*Mathematica Help Browser* **Mathematica Book: 3.8.2; Functions: Fit, FindFit; Package: Statistics‘NonlinearFit**

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

. Week of 23–27 October

Lectures

	Topics	**Reading**
M 10/23 Lect. 13	Vector differential operations: divergence and its interpretation, curl and its interpretation	*Kreyszig* **9.8, 9.9** (pages: 410–413, 414–416)
W 10/25 Lect. 14	Path integration: integral over a curve, change of variables, multidimensional integrals	*Kreyszig* **10.1, 10.2, 10.3** (pages: 420–425, 426–432, 433–439)
F 10/27 Lect. 15	Multidimensional forms of the Fundamental theorem of calculus: Green's theorem in the plane, surface representations and integrals	*Kreyszig* **10.4, 10.5, 10.6, 10.7** (pages: 439–444, 445–448, 449–458, 459–462)

Laboratory

F 10/27 Lab 6	Graphical representations in three and higher dimensions	*Mathematica Help Browser* **Mathematica Book: 1.9.1—1.9.7, 1.9.9—1.9.11**

Homeworks

Homework Set	Available	Due Date
4	Thursday 23 Oct	

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

. Week of 30 Sept–3 Nov

Lectures		
	Topics	**Reading**
M 10/30 Lect 16	Multi-variable calculus: triple integrals and divergence theorem, applications and interpretation of the divergence theorem, Stokes' theorem.	*Kreyszig* **10.8, 10.9** (pages: 463–467, 468–473)
W 11/01 Lect. 17	Periodic functions: Fourier series, Interpretation of Fourier coefficients, convergence, odd and even expansions	*Kreyszig* **11.1, 11.2, 11.3** (pages: 478–485, 487–489, 490–495)
F 11/03 Lect. 18	Fourier theory: complex form of Fourier series, Fourier integrals, Fourier cosine and sine transforms, the Fourier transforms	*Kreyszig* **11.4, 11.7, 11.8, 11.9** (pages: 496–498, 506–512 513–517, 518–523)
Laboratory		
F 11/03 Lab 7	Review of Mathematica functions and graphics	*Mathematica Help Browser* **Mathematica Book: 1.9.1–1.9.9, 2.1.1, 2.2.1. 2.3.1, 2.4.1, 2.5.1, 2.6.1, 2.7.1**

. Week of 6–10 November

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lectures		
M 11/06 Lect 19	Ordinary differential equations: physical interpretations, geometrical interpretations, separable equations	*Kreyszig* **1.1, 1.2, 1.3** (pages: 2–8, 9–11, 12–17)
W 11/08 Lect. 20	ODEs: derivations for simple models, exact equations and integrating factors, the Bernoulli equation	*Kreyszig* **1.4, 1.5** (pages: 19–25, 26–32)
F 11/10	Holiday, no 3.016 lecture	

Homeworks		
Homework Set	Available	Due Date
4		Thursday 9 Nov
5	Wednesday 8 Nov	

. Week of 13–17 November

3.014 Laboratory Week: 3.016 does not meet.

. Week of 20–24 November

Lectures		
	Topics	**Reading**
M 11/20 Lect. 21	Higher order differential equations: homogeneous second order, initial value problems, second order with constant coefficients, solution behavior	*Kreyszig* **2.1, 2.2** (pages: 45–52, 53–58)
W 11/22 Lect. 22	Differential operators, damped and forced harmonic oscillators, non-homogeneous equations	*Kreyszig* **2.3,2.4, 2.7** (pages: 59–60, 61–69, 78–83)
F 11/24	Holiday, no 3.016 lecture	

Homeworks		
Homework Set	Available	Due Date
6	Monday 20 Nov	

. Week of 27 Nov–1 Dec

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lectures

	Topics	Reading
M 11/27 Lect. 23	Resonance phenomena, higher order equations, beam theory	*Kreyszig* **2.8, 2.9, 3.1, 3.2, 3.3** (pages: 84–90, 91–96, 105–111, 111–115, 116–121)
W 11/29 Lect. 24	Systems of differential equations, linearization, stable points, classification of stable points	*Kreyszig* **4.1, 4.2** (pages: 131–135, 136–139)
F 12/01 Lect. 25	Linear differential equations: phase plane analysis and visualization	*Kreyszig* **4.3, 4.4** (pages: 139–146, 147–150)

Laboratory

F 12/01 Lab 8	Solutions to ordinary differential equations	*Mathematica Help Browser* **Mathematica Book: 1.5.9, 3.5.11; Function: DSolve, NDSolve, NIntegrate**

Homeworks

Homework Set	Available	Due Date
5		Thursday 30 Nov

. Week of 4–8 December

Lectures		
	Topics	**Reading**
M 12/04 Lect. 26	Solutions to differential equations: Legendre's equation, orthogonality of Legendre polynomials, Bessel's equation and Bessel functions	*Kreyszig* **5.3, 5.5, 5.6** (pages: 177–180, 189–197, 198–202)
W 12/06 Lect. 27	Sturm-Louiville problems: eigenfunction, orthogonal functional series, eigenfunction expansions	*Kreyszig* **5.7, 5.8** (pages: 203–208, 210–216)
F 12/08	3.014 Laboratory continues, No more Maths lectures	

Homeworks		
Homework Set	Available	Due Date
6		Thursday 7 Dec

. Week of 11–15 December

3.014 Laboratory Week: 3.016 does not meet.

Beginners to Mathematica

Beginners to Mathematica® tend to make the same kinds of mistakes. I've been collecting a list of such mistakes and present them to you as a reference tool.

3.016
3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

Lecture 01 Mathematica® Example 1

Common Mathematica Mistakes

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A list of common beginner Mathematica mistakes. The entries here are typical **mistakes**. I welcome input from others to might add to this list

1-4 are examples of confusing usages of parentheses (—), curlies {—}, and square brackets [—].

1: Parentheses are used for logical grouping, not for function calls (first line) or lists (lines 2–3).

2: Curlies usually group lists of things; in item two multiplcation of a list of length 1 is left-multiplied by a list of length 3 which may not be what was intended.

3: Single brackets are for function arguments—and not for list extraction as in line 2 or grouping in line 3.

4: Double brackets are usually for list extraction, not function calls.

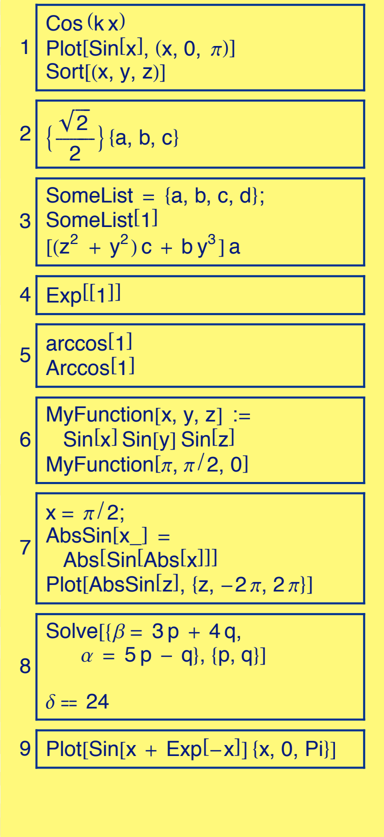
5: Mathematica® is case sensitive and functions are usually made by concatenating words with their first letters capitalized.

6: Functions are *usually* created designed with patterns (i.e., x_, y_) for variables. This is an error if `x` is a defined variable. This line is correct in using the appropriate *delayed assignment* :=.

7: Here a function is defined with a direct assignemt (=) and not *delayed assignment* :=.

8: In the first line, assignments (=) are used instead of the double equals (==) which is a *logical equality*. In the second line, a logical equality is queried and no value is assigned to δ.

9: Missing commas: `Plot` requires at least two arguments separated by commas or it doesn't know what to return.



```
1   Cos (k x)
    Plot[Sin[x], (x, 0, π)]
    Sort[(x, y, z)]

2   {√2/2} {a, b, c}

3   SomeList = {a, b, c, d};
    SomeList[1]
    [(z^2 + y^2) c + b y^3] a

4   Exp[[1]]

5   arccos[1]
    Arccos[1]

6   MyFunction[x, y, z] :=
      Sin[x] Sin[y] Sin[z]
    MyFunction[π, π/2, 0]

7   x = π/2;
    AbsSin[x_] =
      Abs[Sin[Abs[x]]]
    Plot[AbsSin[z], {z, −2 π, 2 π}]

8   Solve[{β = 3 p + 4 q,
        α = 5 p − q}, {p, q}]

    δ == 24

9   Plot[Sin[x + Exp[−x]] {x, 0, Pi}]
```

3.016 Home

⏮ ◀ ▶ ⏭

Full Screen

Close

Quit

Common Mathematica Mistakes

notebook (non-evaluated) pdf (evaluated) html (evaluated)

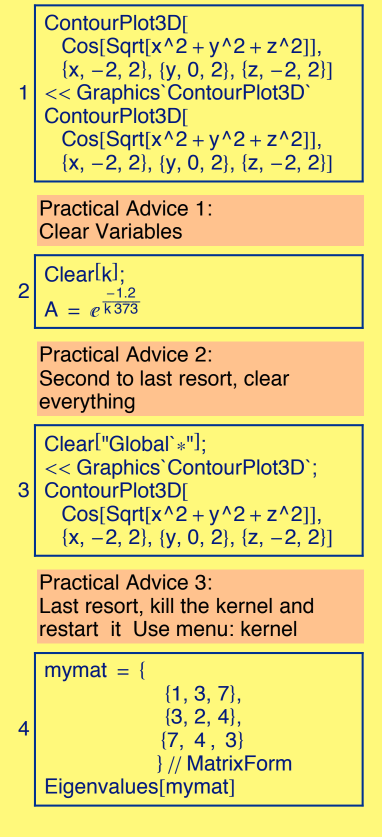
(continued) list of common beginner MATHEMATICA® mistakes. The entries here are typical **mistakes**.

1: Because `x` was defined above, it retains its value; so the function can't iterate over the x-coordinate. Also, because `ContourPlot3D` is defined in the `Graphics` package, it is unknown to the system. Loading the packing is the right thing to do, but because their are two symbols `ContourPlot3D` defined, their will be ambiguity for *which one to use.*

2: Practical advice is to clear the variable definitions with `Clear`.

3: More powerful practical advice, but slight overkill, is to clear all user-defined variables. As a last resort when everything seems awry, kill the kernel with the menu and restart it. This starts up a new MATHEMATICA® session, but does not destroy the text in the `Notebook`.

4: Sometimes the form of an expression is part of its definition. In this case, a `MatrixForm` of a matrix *is not* a matrix and so matrix operations are not defined.



1
```
ContourPlot3D[
  Cos[Sqrt[x^2 + y^2 + z^2]],
  {x, −2, 2}, {y, 0, 2}, {z, −2, 2}]
<< Graphics`ContourPlot3D`
ContourPlot3D[
  Cos[Sqrt[x^2 + y^2 + z^2]],
  {x, −2, 2}, {y, 0, 2}, {z, −2, 2}]
```

Practical Advice 1:
Clear Variables

2
```
Clear[k];
```
$A = e^{\frac{-1.2}{k\,373}}$

Practical Advice 2:
Second to last resort, clear everything

3
```
Clear["Global`*"];
<< Graphics`ContourPlot3D`;
ContourPlot3D[
  Cos[Sqrt[x^2 + y^2 + z^2]],
  {x, −2, 2}, {y, 0, 2}, {z, −2, 2}]
```

Practical Advice 3:
Last resort, kill the kernel and restart it Use menu: kernel

4
```
mymat = {
             {1, 3, 7},
             {3, 2, 4},
             {7,  4 ,  3}
          } // MatrixForm
Eigenvalues[mymat]
```

3.016 Home
Full Screen
Close
Quit

Lecture 01 MATHEMATICA® Example 3

Common Mathematica Mistakes

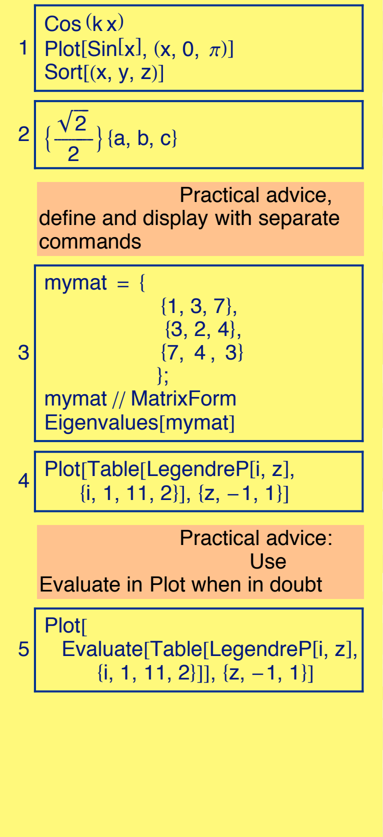
notebook (non-evaluated) pdf (evaluated) html (evaluated)

(continued) list of common beginner MATHEMATICA® mistakes. The entries here are typical **mistakes**.

1: Practical advice is to separate the definition from the display of the assigned variable. Here a matrix is defined; its `MatrixForm` is display, and `Eigenvalues` of the matrix are calculated.

2: Some functions, such as `Plot`, evaluate their arguments in a round-about way. Graphical output does not appear here, because `Plot` doesn't bother evaluating the `Table` function first.

3: If a computationally intensive function is not doing what you expect, then try to wrap an expression in an `Evaluate` function.



1
```
Cos (k x)
Plot[Sin[x], (x, 0, π)]
Sort[(x, y, z)]
```

2 $\{\frac{\sqrt{2}}{2}\}\{a, b, c\}$

Practical advice,
define and display with separate commands

3
```
mymat = {
            {1, 3, 7},
            {3, 2, 4},
            {7,  4 ,  3}
           };
mymat // MatrixForm
Eigenvalues[mymat]
```

4
```
Plot[Table[LegendreP[i, z],
    {i, 1, 11, 2}], {z, -1, 1}]
```

Practical advice:
Use
Evaluate in Plot when in doubt

5
```
Plot[
  Evaluate[Table[LegendreP[i, z],
    {i, 1, 11, 2}]], {z, -1, 1}]
```

Lecture 2: Introduction to Mathematica

Expressions and Evaluation

There are very many ways to learn how to use MATHEMATICA® . Nearly all of the best ways involve performing examples from the very beginning. That is how we are going to start—with examples. Using MATHEMATICA® 's *FrontEnd* you may execute a command by pressing Shift-Enter; simply pressing Enter tells MATHEMATICA® 's that you merely wish to have a "carriage return" on the screen.

Mathematica's syntax will feel fairly natural after a while. Use the following notebook to get started. Execute a few commands until you get a sense for what output MATHEMATICA® will produce; try editing the commands; try to make MATHEMATICA® do something strange—just try playing with it and you will soon get the hang of what is going on.

One way to use MATHEMATICA® is simply as a calculator that allows symbols to get carried along. MATHEMATICA® will usually try to resolve every symbol and return precise information about it. If something is undefined to MATHEMATICA® , it simply returns it as a symbolic expression.

A number is not returned until all of the symbols in an expression are defined as numbers. MATHEMATICA® will try to be exact—it does not calculate $\frac{1}{3}+\frac{1}{2}$ by adding $0.33333\cdots + 0.5 = 0.83333\ldots$, it has an algorithm for adding rational numbers and gives $\frac{5}{6}$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 02 Mathematica® Example 1

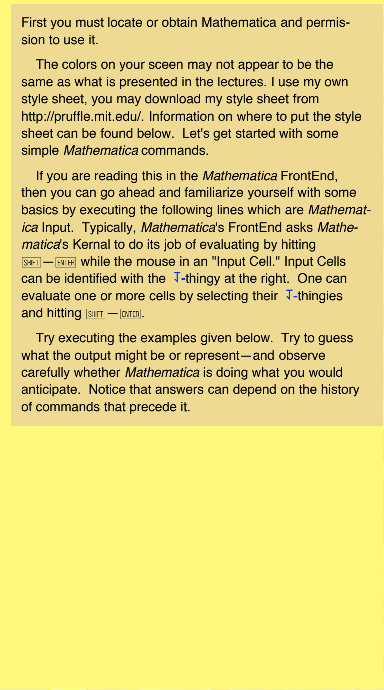
Getting Started

notebook (non-evaluated) pdf (evaluated) html (evaluated)

There are a variety of ways to get Mathematica® started and these are specific to the operating system your computer uses. A license must be purchased to run Mathematica® code, but free Mathematica® -display tools can be obtained from Wolfram.

The FrontEnd is the graphical interface between the user and Mathematica® —you arrange your Mathematica® input, sometimes with text-like comments, in the FrontEnd. The user must request the FrontEnd to pass something to Mathematica® 's *kernel*, by pressing Shift-Enter. The kernel is the resident symbolic algebra software engine behind Mathematica® .

The appearance of the FrontEnd depends on either provided or user-designed *StyleSheets*. The StyleSheet for this course can be downloaded from the course website. The course style is particulary ugly—it is hoped that this will provide an incentive for students to create their own style.



First you must locate or obtain Mathematica and permission to use it.

The colors on your sceen may not appear to be the same as what is presented in the lectures. I use my own style sheet, you may download my style sheet from http://pruffle.mit.edu/. Information on where to put the style sheet can be found below. Let's get started with some simple *Mathematica* commands.

If you are reading this in the *Mathematica* FrontEnd, then you can go ahead and familiarize yourself with some basics by executing the following lines which are *Mathematica* Input. Typically, *Mathematica*'s FrontEnd asks *Mathematica*'s Kernal to do its job of evaluating by hitting SHIFT — ENTER while the mouse in an "Input Cell." Input Cells can be identified with the ⌉-thingy at the right. One can evaluate one or more cells by selecting their ⌉-thingies and hitting SHIFT — ENTER.

Try executing the examples given below. Try to guess what the output might be or represent—and observe carefully whether *Mathematica* is doing what you would anticipate. Notice that answers can depend on the history of commands that precede it.

Lecture 02 MATHEMATICA® Example 2

Basic Input and Assignment

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The methods of assigning expressions (`expr`) to symbolic variables (`SomeVariable`) via `expr = SomeVariable`. Differences between exact (symbolic) objects and numerical objects. Logical equalities (`==`) and Clearing symbols. Many bugs crawl into MATHEMATICA® from "uncleared" symbols.

1: A symbol is assigned to an expression with an equals sign `=`. Some symbols, such as π, are already defined—in MATHEMATICA® it is *exactly* the ratio of a circle's circumference to its diameter.

2: If a previously assigned symbol is used in a new assignment, MATHEMATICA® will usually incorporate the previously assigned symbol's properties. Here, the variable is given a descriptive and long name: this is good practice if you want your code to self-document. Your definitions can be recalled and used to build up more complicated expressions.

6: It is possible to assign symbols to numberical objects; here `b` is a different kind of object (numerical) than the previously defined `a` (symbolic). There are built-in object-types created from combinations of other types.

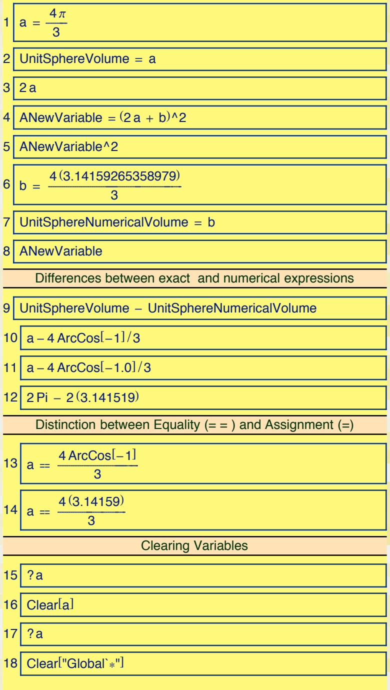
10: MATHEMATICA® will to be exact—the `ArcCos` of -1 is exactly π.

11: However, the `ArcCos` of -1.000 is a numerical approximation to π—the result will be numerical (not exact).

13: A `Rational` expression is another type of MATHEMATICA® object. There are built-in rules to treat rational expressions exactly. The result of applying `N` to the rational $\frac{5}{6}$ is a numerical approximation to 5/6.

14: Assignment `=` is *different* from *logical equality* `==`. For a logical equality, the result is either `True`, `False`, or MATHEMATICA® cannot determine and returns an expression.

17: It is possible to clear *all* user-made definitions. A last resort would be to instruct the FrontEnd to kill MATHEMATICA® 's kernel and restart it. clearing symbols.



1 $a = \frac{4\pi}{3}$

2 UnitSphereVolume = a

3 2 a

4 ANewVariable = (2 a + b)^2

5 ANewVariable^2

6 $b = \frac{4\,(3.14159265358979)}{3}$

7 UnitSphereNumericalVolume = b

8 ANewVariable

Differences between exact and numerical expressions

9 UnitSphereVolume − UnitSphereNumericalVolume

10 a − 4 ArcCos[−1]/3

11 a − 4 ArcCos[−1.0]/3

12 2 Pi − 2 (3.141519)

Distinction between Equality (= =) and Assignment (=)

13 $a == \frac{4\,\mathrm{ArcCos}[-1]}{3}$

14 $a == \frac{4\,(3.14159)}{3}$

Clearing Variables

15 ?a

16 Clear[a]

17 ?a

18 Clear["Global`*"]

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 02 MATHEMATICA® Example 3

Built-in Functions and Operations on Expressions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

MATHEMATICA® has many, *very many*, built-in mathematical functions; it's probably impossible to memorize all of them. You can usually find what you need by using the Help Browser, or by querying `?` with a wildcard `*` such as in `?*Geom*`.

1: MATHEMATICA® has a fairly consistent function naming strategy The first letter of a word is always captialized; compound words are concatenated together while maintaining the first letter capitalization; thus `InverseBetaRegularized`. A function is just another symbol—if a symbol is followed by square brackets `[]` the stuff inside the brackets become the argument(s) for the function.

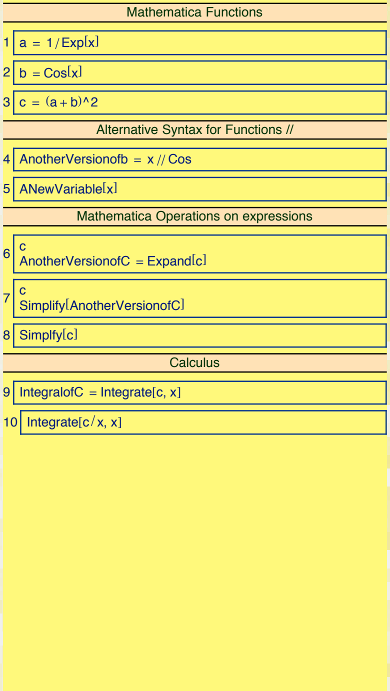
4: There are usually more than one way to do things. The operator `//` is a short-hand way of applying the function that follows `//` to the expression that proceeds it (e.g., `(Pi/2)//Sin`). You can also use `@` to prefix a function (e.g. `Sin@(Pi/2)`).

6: There are methods designed to improve or alter the appearance of complicated expressions.

8: Spelling errors can be very difficult to spot and debug—as a matter of practical advice, look for MATHEMATICA® 's spelling warnings. They can be turned off with `Off[General::spell]`, but it is not generally a good idea.

9: MATHEMATICA® has a sophisticated symbolic integration algorithm. . .

10: even if sometimes it expresses it in special functions



Mathematica Functions

```
1  a = 1/Exp[x]
2  b = Cos[x]
3  c = (a + b)^2
```

Alternative Syntax for Functions //

```
4  AnotherVersionofb = x // Cos
5  ANewVariable[x]
```

Mathematica Operations on expressions

```
6  c
   AnotherVersionofC = Expand[c]
7  c
   Simplify[AnotherVersionofC]
8  Simplfy[c]
```

Calculus

```
9  IntegralofC = Integrate[c, x]
10 Integrate[c/x, x]
```

Lecture 02 MATHEMATICA® Example 4

Calculus and Plotting

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The derivative and integration methods are introduced. Simple plotting methods are demonstrated with an example of annotating a plot.

2: If MATHEMATICA® can't differentiate or integrate a function, it will be left in a symbolic form.

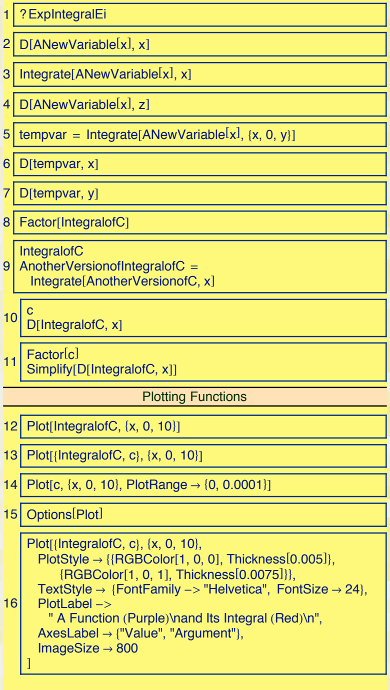
5: MATHEMATICA® applies the fundamental theorems of calculus. . .

10: The calculus operations will often create long and complicated expressions. That two expressions are equivalent can *sometimes* be shown with built-in functions such as `Simplify`, `FullSimplify`, `Factor`, `Expand`, `Collect`, etc., but sometimes it is an art to turn an expression into an aesthetic form.

12: This is the simplest form of `Plot`. The second argument is a list giving the variable and its bounds. The first argument should have a numerical value at most of the points within the variable's bounds.

15: To find all the possible options for a function with their default values, the `Options` function provides a way to decipher what aspects of a plot can be changed easily. Demonstration of a function that MATHEMATICA® does not know how to integrate or differentiate.

16: Here is an example with a plot title, axes labels, different colors and thickness for the curves.



```
1  ?ExpIntegralEi
2  D[ANewVariable[x], x]
3  Integrate[ANewVariable[x], x]
4  D[ANewVariable[x], z]
5  tempvar = Integrate[ANewVariable[x], {x, 0, y}]
6  D[tempvar, x]
7  D[tempvar, y]
8  Factor[IntegralofC]
   IntegralofC
9  AnotherVersionofIntegralofC =
     Integrate[AnotherVersionofC, x]
   c
10 D[IntegralofC, x]
   Factor[c]
11 Simplify[D[IntegralofC, x]]
```

Plotting Functions

```
12 Plot[IntegralofC, {x, 0, 10}]
13 Plot[{IntegralofC, c}, {x, 0, 10}]
14 Plot[c, {x, 0, 10}, PlotRange → {0, 0.0001}]
15 Options[Plot]
   Plot[{IntegralofC, c}, {x, 0, 10},
     PlotStyle → {{RGBColor[1, 0, 0], Thickness[0.005]},
         {RGBColor[1, 0, 1], Thickness[0.0075]}},
     TextStyle → {FontFamily -> "Helvetica", FontSize → 24},
16   PlotLabel ->
       " A Function (Purple)\nand Its Integral (Red)\n",
     AxesLabel → {"Value", "Argument"},
     ImageSize → 800
   ]
```

Lecture 02 MATHEMATICA® Example 5

Lists, Lists of Lists, and Operations on Lists

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Lists are useful ways to keep related information together, and MATHEMATICA® uses them extensively. Lists could be created in MATHEMATICA® by using the `List` function, but they are usually entered in with curly-brackets {}.

3: Some functions, such as `Cos` here, are *threadable functions*; when called on a list-argument, they will produce a list of that function applied to each list element.

4: A list's parts (or, elements) can be picked out in a variety of ways. The `Part` function has a shorthand double-bracket form.

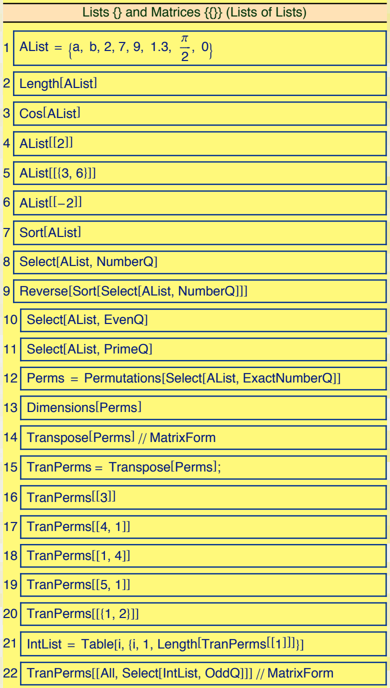
7: There are plenty of functions designed to operate on lists.

8: Logical operations, such as `NumberQ`, can be used to select elements by their characteristics.

13: A list's elements can be lists themselves. For example, a matrix is represented by list of a list. And their are higher-dimensional structures such as tensors. `Dimensions` is a useful way of learning about such structures.

14: Here, the *post-fix operator* for a function is used to change the way a matrix is displayed. *Note, the result is **not** a matrix, but a `DisplayForm` of a matrix.*

22: This is a fairly advanced example of extracting the odd-numbered columns of a matrix. The list `IntList` is simply the integers for each column; its odd-numbered members are selected and become the second (i.e., column) argument of the `Part` selection. The first argument is `All`, so the entire row is captured for each selected column.



Lists {} and Matrices {{}} (Lists of Lists)

```
1  AList = {a, b, 2, 7, 9, 1.3, π/2, 0}
2  Length[AList]
3  Cos[AList]
4  AList[[2]]
5  AList[[{3, 6}]]
6  AList[[−2]]
7  Sort[AList]
8  Select[AList, NumberQ]
9  Reverse[Sort[Select[AList, NumberQ]]]
10 Select[AList, EvenQ]
11 Select[AList, PrimeQ]
12 Perms = Permutations[Select[AList, ExactNumberQ]]
13 Dimensions[Perms]
14 Transpose[Perms] // MatrixForm
15 TranPerms = Transpose[Perms];
16 TranPerms[[3]]
17 TranPerms[[4, 1]]
18 TranPerms[[1, 4]]
19 TranPerms[[5, 1]]
20 TranPerms[[{1, 2}]]
21 IntList = Table[i, {i, 1, Length[TranPerms[[1]]]}]
22 TranPerms[[All, Select[IntList, OddQ]]] // MatrixForm
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

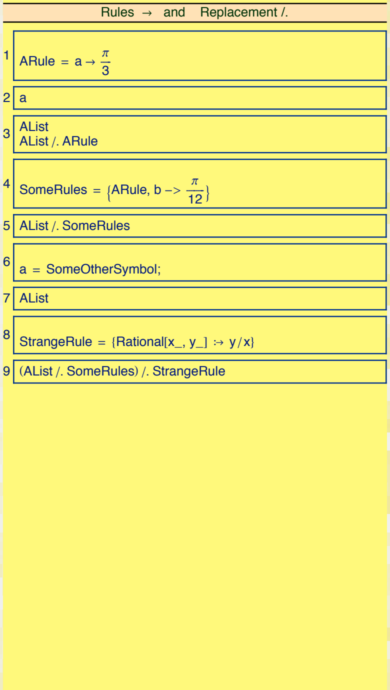
Quit

Lecture 02 MATHEMATICA® Example 6

Rules (→) and Replacement (/.)

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A rule `leftvar` → `rightvar` is *similar* to assignment in that it associates a new symbol (`leftvar`) with something else, but it the value is not assigned—it does not effect future values of the left-hand-side symbol. Rules are often used in conjunction with replacements. Many of MATHEMATICA® functions, (e.g., `Solve`) return rules as a result.



Rules → and Replacement /.

```
1 ARule = a → π/3
2 a
3 AList
  AList /. ARule
4 SomeRules = {ARule, b -> π/12}
5 AList /. SomeRules
6 a = SomeOtherSymbol;
7 AList
8 StrangeRule = {Rational[x_, y_] :→ y/x}
9 (AList /. SomeRules) /. StrangeRule
```

1: The *rule* `a` → $\pi/3$ is *assigned* to the symbol `ARule`

3: A rule can be applied with the function Replace, but the syntax (.) is typically used instead.

4: Rules can be collected into lists.

5: Assigment of `a` is reflected in the form of `ARule`.

8: Rules are necessary for manipulations in MATHEMATICA® , but can be used to generate "mistakes." Think of Rule and Replace acting on an expression as "What would the expression be if a certain rule were applied to it?" If the rule is wrong, the resulting expression will be as well.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Getting Help on Mathematica

MATHEMATICA® ’s built-in help functions are very useful. This was true even before the whole MATHEMATICA® manual was incorporated into the Help Browser. In the old days, one would memorize large portions of the MATHEMATICA® book—which has grown continuously heavier since its first publication in the early 1990’s—and rely on the useful ”?” and ”??” operators. The use of ”?” with the wildcard ”” enabled a beginning user to track down almost any MATHEMATICA® function. The Options function is also a very efficient way to discover alternative ways of getting results.

I would have recommended ‘scanning’ the *entire* MATHEMATICA® manual in a single three hour sitting (about 600 pages per hour) as an effective way to acquire a working familiarity with the software, but I don’t because the built-in browser is so easy to use.

I encourage you to idly explore the MATHEMATICA® Help Browser. You will not only learn about MATHEMATICA® , but also about mathematics.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 3: Introduction to Mathematica II

Sept. 11 2006

Functions and Rules

Besides Mathematica® 's large set of built-in mathematical and graphics functions, the most powerful aspects of Mathematica® are its ability to recognize and replace patterns and to build functions based on patterns. Learning to *program* in Mathematica® is very useful and to learn to program, the basic programmatic elements must be acquired.

The following are common to almost any programming language:

Variable Storage A mechanism to define variables, and subsequently read and write them from memory.

Loops Program structures that iterate. A well-formulated loop will always be guaranteed to exit[2].

Variable Scope When a variable is defined, what other parts of the program (or other programs) will be able to read its value or change it? The scope of a variable is, roughly speaking, the extent to which it is available.

Switches These are commands with outcomes that depend on a quality of variable, but it is unknown, when the program is written, what the variable's value will be. Common names are If, Which, Switch, IfThenElse and so on.

Functions Reusable sets of commands that are stored away for future use.

All of the above are, of course, available in Mathematica® .

The following are common to Symbolic and Pattern languages, like Mathematica® .

Patterns This is a way of identifying common structures and make them available for subsequent computation.

[2] Here is a joke: "Did you hear about the computer scientist who got stuck in the shower?" "Her shampoo bottle's directions said, 'wet hair, apply shampoo, rinse, repeat'."

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Recursion This is a method to define function that obtains its value by calling itself. An example is the Fibonacci number $F_n \equiv F_{n-1}+F_{n-2}$ (The value of F is equal to the sum of the two values that preceded it.) F_n cannot be calculated until earlier values have been calculated. So, a function for Fibonacci must call itself recursively. It stops when it reaches the end condition $F_1 = F_2 = 1$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 03 Mathematica® Example 1

Procedural Programming

notebook (non-evaluated)   pdf (evaluated)   html (evaluated)

Simple programs can be developed by sequences of variable assignment.

1: Here is a simple program that is just a sequence of statements that re-assigns `a`. In Mathematica® , a semicolon— ;—just indicates that output should be suppressed.

4: However, it would be cumbersome and unaesthetic if the example had to be typed many many times. This is where *program loops* come in. `Do` is a simple way to loop over an expression a fixed number of times. This is equivalent to item 1, but could be easily generalized to more iterations.

6: Here an equivalent example, but extra `Print` statements are added so that *intermediate output* can be observed.

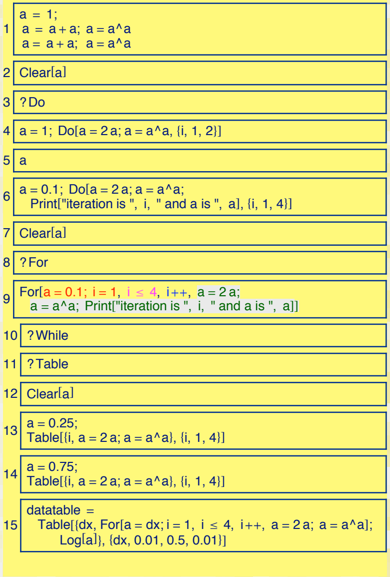
9: A `For` loop is another loop structure that enforces good programming style: Its arguments provide: an initialization, an exit condition, an iteration operator, and a function statement, and is equivalent to item 6.

10: The are many types of loop constructs; `While` is yet another.

11: `Table` is a very useful Mathematica® function. While it iterates, it leaves intermediate results in a List structure.

13: Except for the *intial iteration value* of `a`, this is programmatically equivalent to items 1, 4, 6, and 9, but each iteration's result is a member of a List.

15: Here we generalize, but putting a `Table` and a `For` together. The result is a list of (lists of length 2). The first entry in each list is the initial increment value and the second entry is the result of the `For`-loop after four iterations. A special *increment structure* is utilized—it sets initial and final values as well as the increment size.



```
1   a = 1;
     a = a + a; a = a^a
     a = a + a;  a = a^a
2   Clear[a]
3   ? Do
4   a = 1; Do[a = 2 a; a = a^a, {i, 1, 2}]
5   a
6   a = 0.1; Do[a = 2 a; a = a^a;
      Print["iteration is ", i, " and a is ", a], {i, 1, 4}]
7   Clear[a]
8   ? For
9   For[a = 0.1; i = 1, i ≤ 4, i++, a = 2 a;
      a = a^a; Print["iteration is ", i, " and a is ", a]]
10  ? While
11  ? Table
12  Clear[a]
13  a = 0.25;
    Table[{i, a = 2 a; a = a^a}, {i, 1, 4}]
14  a = 0.75;
    Table[{i, a = 2 a; a = a^a}, {i, 1, 4}]
15  datatable =
      Table[{dx, For[a = dx; i = 1, i ≤ 4, i++, a = 2 a; a = a^a];
          Log[a]}, {dx, 0.01, 0.5, 0.01}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 03 MATHEMATICA® Example 2

Plotting Lists of Data and Examples of Deeper MATHEMATICA® Functionality

notebook (non-evaluated) pdf (evaluated) html (evaluated)

This demonstrates how visualizing data can be combined with other functions to perform analysis.

1: The data produced from the last example can be plotted. It is apparent that there is a minimum between initial values of 0.1 and 0.3. But, it will difficult to see unless the visualization of the plot can be controlled.

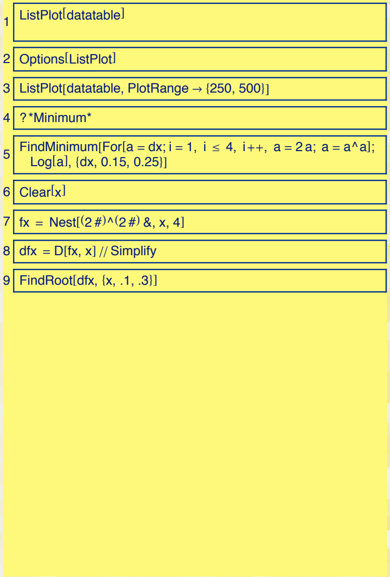
3: By specifying one of `ListPlot`'s option for the range of the y-like variable, the character of minimum can be approximately quantified.

5: `FindMinimum` is a fairly sophisticated function to obtain the minimum of an expression in a specified range, even if the function only returns a numerical result. Here `FindMinimum` is used, to find a very high precision approximation to the minimum observed in item 3.

7: This is a fairly advanced example—beginning students should not worry about understanding it yet. `Nest` is a sophisticated method of repeated application of a function (i.e., $f(f(f(x)))$ is nesting the function f three on an argument x). It is equivalent to the previous methods of producing the iterative stucture. This concept uses *Pure Functions*.

8: The minimum of the function can be analyzed the standard way, here by taking derivatives with `D`.

9: `FindRoot` is sophisticated numerical method to obtain the zero of an expression in a specified range.



```
1 ListPlot[datatable]

2 Options[ListPlot]

3 ListPlot[datatable, PlotRange → {250, 500}]

4 ?*Minimum*

5 FindMinimum[For[a = dx; i = 1, i ≤ 4, i++, a = 2 a; a = a^a];
    Log[a], {dx, 0.15, 0.25}]

6 Clear[x]

7 fx = Nest[(2 #)^(2 #) &, x, 4]

8 dfx = D[fx, x] // Simplify

9 FindRoot[dfx, {x, .1, .3}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Very complex expressions and concepts can be built-up by loops, but within MATHEMATICA® the complexity can be buried so that only the interesting parts are apparent and shown to the user.

Sometimes, as complicated expressions are being built up, intermediate variables are used. Consider the value of `i` after running the program
`FindMinimum[For[a = dx; i = 1, i ≤ 4, i++, a = 2a; a = a∧a]; Log[a], {dx, 0.15, 0.25}]`,
the value of `i` (in this case 5) is has no useful meaning anymore. If you had defined a symbol such as `x = 2i` previously, then now `x` would have the value of 10, which is probably not what was intended. It is much safer to localize variables—in other words, to limit the scope of their visibility to only those parts of the program that need the variable and this is demonstrated in the next example. Sometimes this is called a “Context” for the variable in a programming language; MATHEMATICA® has contexts as well, but should probably be left as an advanced topic.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

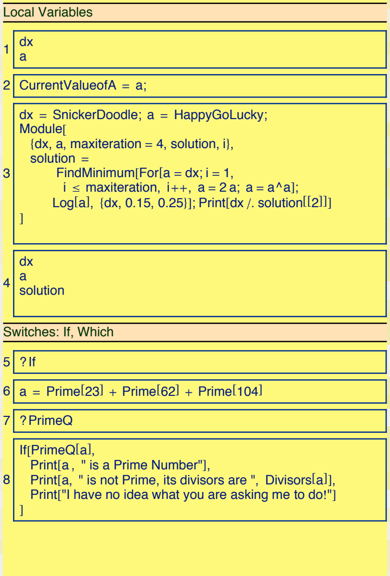
Lecture 03 MATHEMATICA® Example 3

Making Variables Local and Using Switches to Control Procedures

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Describes the use of `Module` to “hide” a variable: consider the variable `a` from the first item in the above example, its intermediate values during iteration are not always important. Suppose you wish to use the symbol `a` later—that it played an intermediate role and then was not used may easily be forgotten. It is good practice to make such variables ‘local’ to their own functions.

A example of a logical switches is demonstrated for `If`.



Local Variables

```
1  dx
   a
```

```
2  CurrentValueofA = a;
```

```
   dx = SnickerDoodle; a = HappyGoLucky;
   Module[
     {dx, a, maxiteration = 4, solution, i},
     solution =
3          FindMinimum[For[a = dx; i = 1,
             i ≤ maxiteration, i++, a = 2 a; a = a^a];
           Log[a], {dx, 0.15, 0.25}]; Print[dx /. solution[[2]]]
   ]
```

```
   dx
4  a
   solution
```

Switches: If, Which

```
5  ? If
```

```
6  a = Prime[23] + Prime[62] + Prime[104]
```

```
7  ? PrimeQ
```

```
   If[PrimeQ[a],
     Print[a , " is a Prime Number"],
8    Print[a, " is not Prime, its divisors are ", Divisors[a]],
     Print["I have no idea what you are asking me to do!"]
   ]
```

1: The symbols `dx` and `a` are left over from the last example, even though they played only an intermediate role for a final result.

2: This could lead us to mistakenly use its value later as though it might be undefined. This is a common error.

3: The production of such errors can be reduced with a programming praction known as *localized varibles* (also known as *scoping!of variables*). The idea is to hide the variable within its own structure—the variable is said to have a limited *scope*. `Module` provides a function for doing this. Here symbols `dx` and `a` have set values before the call to `Module`, but any value that is changed *inside* of `Module` has no effect on its “global” value.

4: Even though `Module` changed symbols `dx`, `a`, and used `solution`, their should be no effect outside of `Module`.

6: Here, a simple example of the use of `If` will be applied to a symbol which is the sum of the 23rd, 62nd, and 104th prime numbers.

8: Here is a simple program. First, it first checks if `a` is prime. If the check is true, then it prints a message saying so, and then returns control to the MATHEMATICA® kernel. If the check is false, then it prints out a message and some more useful information about the fact it isn't prime. If the statement cannot be determined to be true or false, a message to that effect is printed.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Patterns are extremely important in mathematics and in Mathematica® . In Mathematica® , the use of the underscore, _, means “this is a placeholder for something that will be used later.” It is a bit like teaching like teaching a dog to fetch—you cock an arm as if to throw _something_, and then when something gets thrown your dog runs after the “something.” The first _something_ is a place holder for an object, say anything from a stick to a ball to the morning paper. The second something is the actual object that is actually tossed, that finally becomes the “something” your dog uses as the actual object in the performance of her ritual response to the action of throwing.

Usually, one needs to name to call the pattern to make it easier to refer to later. The pattern gets named by adding a head to the underscore, such as `SomeVariableName_`, and then you can refer to what ever pattern matched it with the name `SomeVariableName`.

This is a bit abstract and probably difficult to understand without the aid of a few examples:

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

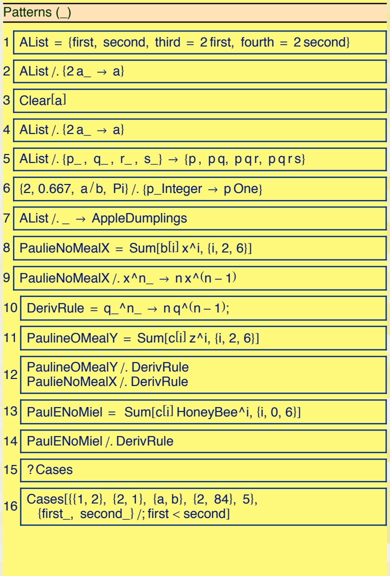
Quit

Lecture 03 MATHEMATICA® Example 4

Operating with Patterns

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Learning to use expression and variable patterns is the beginning of intermediate use of MATHEMATICA® . Patterns are identified by the underscore character _, the matched pattern can be named for later use (e.g., `thematch_`) and it can be further qualified as demonstrated below.



Patterns (_)

```
1  AList = {first, second, third = 2 first, fourth = 2 second}
2  AList /. {2 a_ → a}
3  Clear[a]
4  AList /. {2 a_ → a}
5  AList /. {p_ , q_ , r_ , s_} → {p , p q, p q r, p q r s}
6  {2, 0.667, a/b, Pi} /. {p_Integer → p One}
7  AList /. _ → AppleDumplings
8  PaulieNoMealX = Sum[b[i] x^i, {i, 2, 6}]
9  PaulieNoMealX /. x^n_ → n x^(n – 1)
10 DerivRule = q_^n_ → n q^(n – 1);
11 PaulineOMealY = Sum[c[i] z^i, {i, 2, 6}]
12 PaulineOMealY /. DerivRule
   PaulieNoMealX /. DerivRule
13 PaulENoMiel = Sum[c[i] HoneyBee^i, {i, 0, 6}]
14 PaulENoMiel /. DerivRule
15 ? Cases
16 Cases[{{1, 2}, {2, 1}, {a, b}, {2, 84}, 5},
     {first_, second_} /; first < second]
```

2: Here a rule is applied to `AList` through the use of the operator `/.` (short-hand for `ReplaceAll`). The pattern here is "two multiplied by something." The symbol `a` should a placeholder for *something*, but `a` was already defined and so the behavior is probably not what was wanted. Another (probably better) usage is the *delayed ruleset* `:->`.

4: After `a` has been cleared, the symbol `a` is free to act as a placeholder; so the effect of applying the rule is that 2× *all somethings* are replaced by the pattern represented by `a`.

6: The types of things that get pattern-matched can be restricted by adding a *pattern qualifiers* to the end of the underscore.

8: For a simple (incomplete) example of the use of patterns, an example producing symbolic derivative of a polynomial will be developed. Here, a polynomial `PaulNoMealX` in `x` is defined using `Sum`.

9: A rule is applied, which replaces patterns `x` to a power with a derivative rule. Only the power is used later, so it is given a place-holder name `n`. This technique would only work on polynomials in `x`.

10: To generalize, a place-holder is defined the dependent variable.

14: This will not work for the constant and linear terms in a polynomial. This could be fixed, but the example would become too complicated and not as good as MATHEMATICA® 's built-in differention rules.

16: Patterns can also be used in conjunction with `Condition` operator `/;`. Here is an example of its use in `Cases`. The pattern is any two-member list *subject to the condition* that teh first member is less than the second.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

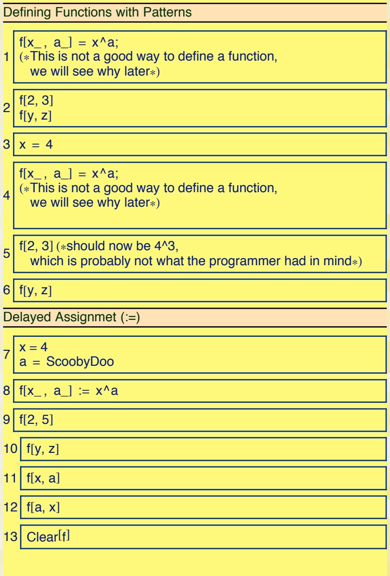
Quit

Lecture 03 MATHEMATICA® Example 5

Creating Functions using Patterns and Delayed Assignment

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The real power of patterns and replacement is obtained when defining functions. Examples of how to define functions are presented.



```
Defining Functions with Patterns

1  f[x_ , a_] = x^a;
   (*This is not a good way to define a function,
     we will see why later*)

2  f[2, 3]
   f[y, z]

3  x = 4

4  f[x_ , a_] = x^a;
   (*This is not a good way to define a function,
     we will see why later*)

5  f[2, 3] (*should now be 4^3,
     which is probably not what the programmer had in mind*)

6  f[y, z]

Delayed Assignmet (:=)

7  x = 4
   a = ScoobyDoo

8  f[x_ , a_] := x^a

9  f[2, 5]

10 f[y, z]

11 f[x, a]

12 f[a, x]

13 Clear[f]
```

1: Here is an example of a pattern: a symbol `f` is defined such that if it is called as a function with a pattern of two named arguments `x_` and `a_`, then the result is what ever x^a evaluated to be when the function was defined. **Don't emulate this example—it is not usually the best way to define a function.**

4: This example shows why this can be a bad idea. `f` with two pattern-arguments, is assigned *when it is defined*, and therefore if either `x` or `a` was previously defined, then the definition will permanently reflect that definition.

5: Calling the function now, doesn't produce the result the user probably expected.

8: For beginning users to MATHEMATICA® , this is the best way to define functions. This involves use *delayed assignment*. In a delayed assignment, the right-hand-side is *not evaluated until the function is called* and then the patterns become *transitory until the function returns its result*. This is usually what we mean when we write $y(x) = ax^2$ mathematically—if y is given a value x, then it operates and returns a value related to that x *and not any other* x *that might have been used earlier*. **This is the prototype for function definitions.**

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

It is probably a good idea to define all function with delayed assignment (`:=`) instead of immediate assignment (`=`). With delayed assignment, Mathematica® does not evaluate the right-hand-side *until* you ask it to perform the function. With immediate assignment, the right-hand-side is evaluated when the function is defined making it much less flexible because your name for the pattern may get "evaluated away."

Defining functions are essentially a way to eliminate repetitive typing and to "compactify" a concept. This "compactification" is essentially what we do when we define some function or operation (e.g., $\cos(\theta)$ or $\int f(x)dx$) in mathematics—the function or operation is a placeholder for something perhaps complicated to describe completely, but sufficiently understood that we can use a little picture to identify it.

Of course, it is desirable for the function to do the something reasonable even if asked to do something that might be unreasonable. No one would buy a calculator that would try to return a very big number when division by zero occurs—or would give a real result when the arc-cosine of 1.1 is demanded. Functions should probably be defined so that they can be reused, either by you or someone else. The conditions for which the function can work should probably be encoded into the function. In Mathematica® this can be done with restricted patterns.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 03 MATHEMATICA® Example 6

Functional Programming with Rules and Pattern Restrictions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Demonstration of increasingly careful factorial function defintions which will result with something sensible for non-integer or negative arguments.

1: This is a functional definition that will produce the factorial function by recursion because $(n+1)! = (n+1)n!$. However, trying this function now will not give a satisfactory result because…

2: It is necessary to define a place for the recursion to stop. This is done by *defining* the factorial of zero to be unity.

3: So that recursive functions don't run for ever, leaving no way to get contact MATHEMATICA® 's kernel, a sensible limit is placed on the number of times a function can call itself.

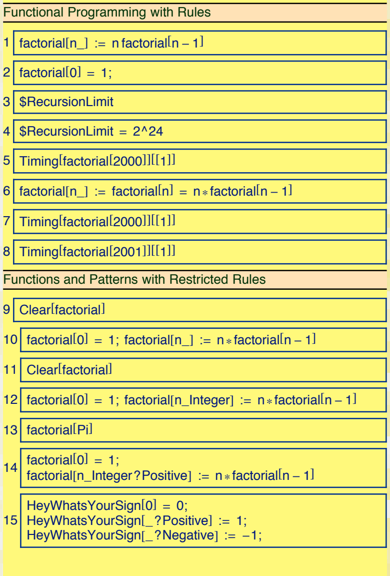
5: Consider using the function to find the factorial of 2000, the currently-defined function must call itself about 2000 times to return a value. Suppose a short time later, value of 2001! is requested. The function must again call itself about 2000 times, even though all the factorials less than 2001's were calculated previously. Unless computer memory is scarce, it seems like a waste of effort to repeat the same calculations over and over.

6: Here is an example where computation speed is purchased at the cost of memory. When the function is called, it makes an assigment as well as the computation.

12: However, what if the previously-defined function were called on a value such as π? It would recursively call $(\pi - 1)!$ which would call $(\pi - 2)!$ and so on. This potential misuse can be eliminated by placing a pattern restriction on the argument of *factorial* so that it is only defined for integer arguments.

14: To prevent unbounded recursion with a call on the previous definition for negative integers, a case switch on the pattern restriction is used.

15: An example of a function that returns the `Sign` of a number if it can.



Functional Programming with Rules

```
1  factorial[n_] := n factorial[n – 1]
2  factorial[0] = 1;
3  $RecursionLimit
4  $RecursionLimit = 2^24
5  Timing[factorial[2000]][[1]]
6  factorial[n_] := factorial[n] = n∗factorial[n – 1]
7  Timing[factorial[2000]][[1]]
8  Timing[factorial[2001]][[1]]
```

Functions and Patterns with Restricted Rules

```
9  Clear[factorial]
10 factorial[0] = 1; factorial[n_] := n∗factorial[n – 1]
11 Clear[factorial]
12 factorial[0] = 1; factorial[n_Integer] := n∗factorial[n – 1]
13 factorial[Pi]
14 factorial[0] = 1;
   factorial[n_Integer?Positive] := n∗factorial[n – 1]
15 HeyWhatsYourSign[0] = 0;
   HeyWhatsYourSign[_?Positive] := 1;
   HeyWhatsYourSign[_?Negative] := –1;
```

3.016 Home

Full Screen

Close

Quit

Lecture 4: Introduction to Mathematica III

Simplifying and Picking Apart Expression, Calculus, Numerical Evaluation

A great advantage of using a symbolic algebra software package like MATHEMATICA® is that it reduces or even eliminates errors that inevitably creep into pencil and paper calculations. However, this advantage does come with a price: what was once a simple task of arranging an expression into a convenient form is something that has to be negotiated with MATHEMATICA® . In fact, there are cases where you cannot even coerce MATHEMATICA® into representing an expression the way that *you* want it.

A MATHEMATICA® session often results in very cumbersome expressions. You can decide to live with them, or use one of MATHEMATICA® 's many simplification algorithms. Section 1.4.5 (or Help Browser/The Mathematica Book/A Practical Introduction/Algebraic Calculations/Advanced Topic: Putting Expressions into Different Forms) of the MATHEMATICA® book has a good summary of frequently used simplification algorithms. Another method is to identify patterns and replace them with your own definitions.

MATHEMATICA® has its own internal representation for rational functions (i.e., $\frac{\text{numerator expression}}{\text{denominator expression}}$) and has special operations for dealing with these. Generally, advanced simplification methods usually require a working knowledge of of MATHEMATICA® 's internal representations.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 04 MATHEMATICA® Example 1

Operations on Polynomials

notebook (non-evaluated) pdf (evaluated) html (evaluated)

There are built-in simplification operations, such as `Simplify`, but they will not always result in a form that is most useful to the user. Crafting an expression into a pleasing form is an artform.

2: `Expand` performs all multiplication and leaves the result as a sum.

3: `Factor` has an algorithm to find common terms in a sum and write the result of a factor and a cofactor.

4: `Collect` will turn in an expression into a polynomial in a user-selected variable.

5: `Coefficient` picks out coefficients of user-specified powers of a variable.

6: This is an example of using `Simplify` to operate only upon on a polynomial coefficients

8: Besides polynomial, other frequently encountered forms are *rational forms*.

9: `Apart` will re-express a rational form as a sum.

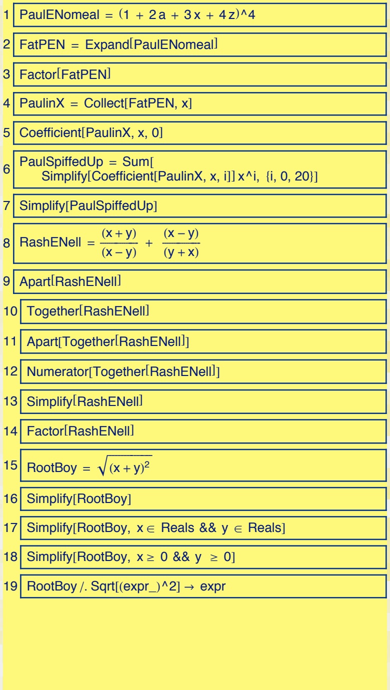
10: `Together` will collect all terms in a sum into a single rational form.

15: MATHEMATICA® is fastidious about simplifying roots and makes no assumptions—unless they are specified— about whether a variable is real, complex, positive, or negative.

16: Many users become frustrated that `Simplify` doesn't do what the user thinks must be correct…

17: `Simplify` will accept `Assumptions`.

19: This is brute force—and not really a good idea.



```
1  PaulENomeal = (1 + 2 a + 3 x + 4 z)^4
2  FatPEN = Expand[PaulENomeal]
3  Factor[FatPEN]
4  PaulinX = Collect[FatPEN, x]
5  Coefficient[PaulinX, x, 0]
6  PaulSpiffedUp = Sum[
        Simplify[Coefficient[PaulinX, x, i]] x^i, {i, 0, 20}]
7  Simplify[PaulSpiffedUp]
               (x + y)   (x - y)
8  RashENell = ------- + -------
               (x - y)   (y + x)
9  Apart[RashENell]
10 Together[RashENell]
11 Apart[Together[RashENell]]
12 Numerator[Together[RashENell]]
13 Simplify[RashENell]
14 Factor[RashENell]
                ___________
15 RootBoy = √ (x + y)^2
16 Simplify[RootBoy]
17 Simplify[RootBoy, x ∈ Reals && y ∈ Reals]
18 Simplify[RootBoy, x ≥ 0 && y ≥ 0]
19 RootBoy /. Sqrt[(expr_)^2] → expr
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 04 MATHEMATICA® Example 2

A Second Look at Calculus: Limits, Derivatives, Integrals

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of `Limit` and calculus with built-in assumptions

2: This would be a challenging limit to find for most first-year calculus students.

5: This definite integral results in a fairly complicated symbolic form which *should* be real because it should be the value of `AMessyExpression` at $x = e$ but not obviously so by inspection. Looking at the numerical version gives a hint. (n.b. `Chop` is a useful way to remove small numerical inaccuracies.)

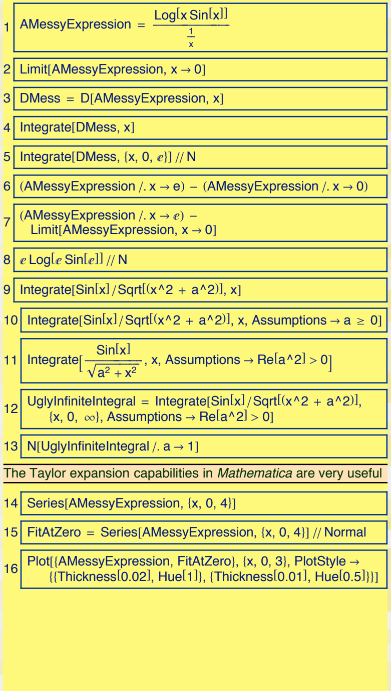
7: It took some effort, but this gives an aesthestic form for the solution.

9: Some indefinite do not have closed-form solutions, even with extra assumptions. . .

12: But, in some cases, the definite integral will have a closed-form solution.

14: `Series` is one of the most useful and powerful MATHEMATICA® functions; especially to replace a complicated function with a simpler approximation in the neighborhood`Series` of a point.

15: `Normal` converts a `SeriesData` form by chopping off the *trailing order* function `O`.



```
1  AMessyExpression = Log[x Sin[x]] / (1/x)
2  Limit[AMessyExpression, x → 0]
3  DMess = D[AMessyExpression, x]
4  Integrate[DMess, x]
5  Integrate[DMess, {x, 0, ℯ}] // N
6  (AMessyExpression /. x → e) − (AMessyExpression /. x → 0)
7  (AMessyExpression /. x → ℯ) −
     Limit[AMessyExpression, x → 0]
8  ℯ Log[ℯ Sin[ℯ]] // N
9  Integrate[Sin[x] / Sqrt[(x^2 + a^2)], x]
10 Integrate[Sin[x] / Sqrt[(x^2 + a^2)], x, Assumptions → a ≥ 0]
11 Integrate[Sin[x] / √(a^2 + x^2), x, Assumptions → Re[a^2] > 0]
12 UglyInfiniteIntegral = Integrate[Sin[x] / Sqrt[(x^2 + a^2)],
     {x, 0, ∞}, Assumptions → Re[a^2] > 0]
13 N[UglyInfiniteIntegral /. a → 1]
```

The Taylor expansion capabilities in *Mathematica* are very useful

```
14 Series[AMessyExpression, {x, 0, 4}]
15 FitAtZero = Series[AMessyExpression, {x, 0, 4}] // Normal
16 Plot[{AMessyExpression, FitAtZero}, {x, 0, 3}, PlotStyle →
     {{Thickness[0.02], Hue[1]}, {Thickness[0.01], Hue[0.5]}}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 04 MATHEMATICA® Example 3

Solving Equations

notebook (non-evaluated) pdf (evaluated) html (evaluated)

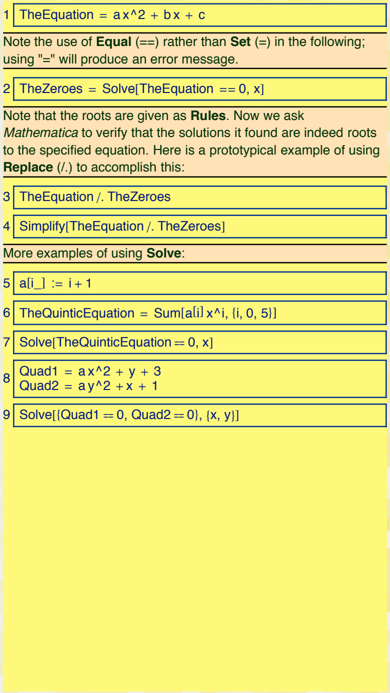
`Solve`, its resulting rules, and how to extract solutions from the rules.

2: `Solve` takes a *logical equality* (or logical equalities) as a first argument. It returns a list of solutions in the form of rules. Here, the list of ruls is assigned to `TheZeroes`.

3: If the resulting solution rules are applied original equation, the zeroes specified in the logical equality in item 2 should result.

7: The zeroes of a quintic polynomial do not have general closed forms. Here MATHEMATICA® will return a symbolic representation of the solution rules. This representation indicates that the solution doesn't have a closed form, but the form is suitable for subsequent numerical analysis.

9: This is an example of a solution to coupled quadratic equations: there are four solutions.



```
1 TheEquation = a x^2 + b x + c
```

Note the use of **Equal** (==) rather than **Set** (=) in the following; using "=" will produce an error message.

```
2 TheZeroes = Solve[TheEquation == 0, x]
```

Note that the roots are given as **Rules**. Now we ask *Mathematica* to verify that the solutions it found are indeed roots to the specified equation. Here is a prototypical example of using **Replace** (/.) to accomplish this:

```
3 TheEquation /. TheZeroes
4 Simplify[TheEquation /. TheZeroes]
```

More examples of using **Solve**:

```
5 a[i_] := i + 1
6 TheQuinticEquation = Sum[a[i] x^i, {i, 0, 5}]
7 Solve[TheQuinticEquation == 0, x]
8 Quad1 = a x^2 + y + 3
  Quad2 = a y^2 + x + 1
9 Solve[{Quad1 == 0, Quad2 == 0}, {x, y}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Sometimes, no closed form solution is possible. MATHEMATICA® will try to give you rules (in perhaps a very strange form) but it really means that you don't have a solution to work with. One usually resorts to a numerical technique when no closed form solution is possible— MATHEMATICA® has a large number of built-in numerical techniques to help out. A numerical solution is an approximation to the actual answer. Good numerical algorithms can anticipate where numerical errors creep in and account for them, but it is always a good idea to check a numerical solution to make sure it approximates the solution the original equation.

Of course, to get a numerical solution, the equation in question must evaluate to a number. This means if you want to know the numerical approximate solutions $x(b)$ that satisfy $x^6 + 3x^2 + bx = 0$, you have to iterate over values of b and "build up" your function $x(b)$ one b at a time.

Sections 1.6.1–1.6.7 (or Help Browser/The Mathematica Book/A Practical Introduction/Algebraic Calculations/Numerical Mathematics) of the MATHEMATICA® book have an overview of frequently used numerical algorithms.

Lecture 04 MATHEMATICA® Example 4

Numerical Algorithms and Solutions

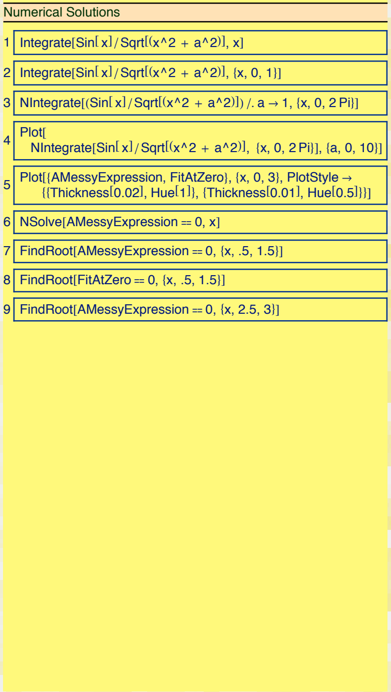
notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of numerical algorithms `NIntegrate` `FindRoot`

3: `NIntegrate` can find solutions in cases where `Integrate` cannot find a closed-form solutions. It is necessary that the integrand should evaluate to a number at all points in the domain of integration (it is possible that the integrand could have singularities at a limited set of isolated points).

4: `NSolve` will find roots to *polynomial forms*, but not for more general expressions.

5: `FindRoot` will operate on general expressions and find solutions, but additional information is required to inform where to search.



Numerical Solutions

```
1 Integrate[Sin[x]/Sqrt[(x^2 + a^2)], x]
2 Integrate[Sin[x]/Sqrt[(x^2 + a^2)], {x, 0, 1}]
3 NIntegrate[(Sin[x]/Sqrt[(x^2 + a^2)]) /. a → 1, {x, 0, 2 Pi}]
4 Plot[
    NIntegrate[Sin[x]/Sqrt[(x^2 + a^2)],  {x, 0, 2 Pi}], {a, 0, 10}]
5 Plot[{AMessyExpression, FitAtZero}, {x, 0, 3}, PlotStyle →
      {{Thickness[0.02], Hue[1]}, {Thickness[0.01], Hue[0.5]}}]
6 NSolve[AMessyExpression == 0, x]
7 FindRoot[AMessyExpression == 0, {x, .5, 1.5}]
8 FindRoot[FitAtZero == 0, {x, .5, 1.5}]
9 FindRoot[AMessyExpression == 0, {x, 2.5, 3}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

1. You will want to save your work.
2. You will want to modify your old saved work
3. You will want to use your output as input to another program
4. You will want to use the output of another program as input to Mathematica® .

You have probably learned that you can save your Mathematica® notebook with a menu. This is one way to take care of the first two items above. There are more ways to do this and if you want to do something specialized like the last two items, then you will have to make Mathematica® interact with files. Because an operating system has to allow many different kinds of programs interact with its files, the internal operations to do input/output (I/O) seem somewhat more complicated than they should be. Mathematica® has a few simple ways to do I/O—and it has some more complex ways to do it as well.

It is useful to have a few working examples that you can modify for your purposes. The examples will serve you well about 90% of the time. For the rest of the 10%, one has to take up the task of learning the guts of I/O—hopefully, beginners can ignore the gory bits.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 04 MATHEMATICA® Example 5

Interacting with the Filesystem

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Reading and writing data directly and through the use of a *file stream*. A user should check (and sometimes) change the *working directory* to interact with files using `Directory` or `SetDirectory`. Otherwise, the full path to a file must be given.

1: Simple redirection of an expression into a file is acheived with `>>` The working directory must be *writable*. Selected symbols can be saved in files all at once using `Save`.

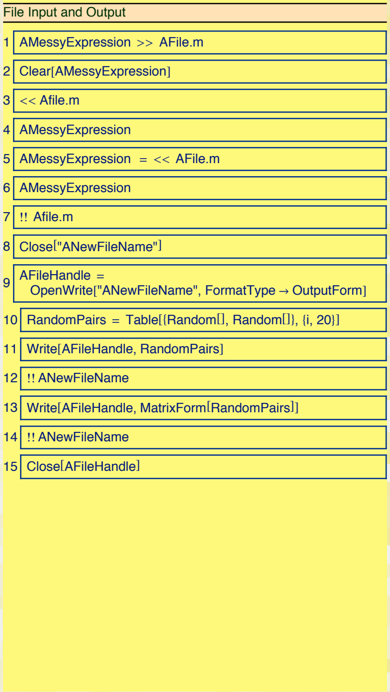
3: A file containing a MATHEMATICA® expression can be read in with `<<` The file must be *readable*.

7: The contents of a file can be displayed using `!!`.

9: This opens a filestream for subsequent use. The use of filestreams is useful for cases where data is written incrementally during a calculation and this method can be generalized to different kinds of devices. Another use of file streams is when the user wants to have the program compute the file name.

11: An example of writing with a file stream.

15: It is good practice to close open file streams when writing is finished.



File Input and Output

```
1  AMessyExpression >> AFile.m
2  Clear[AMessyExpression]
3  << Afile.m
4  AMessyExpression
5  AMessyExpression = << AFile.m
6  AMessyExpression
7  !! Afile.m
8  Close["ANewFileName"]
9  AFileHandle =
     OpenWrite["ANewFileName", FormatType → OutputForm]
10 RandomPairs = Table[{Random[], Random[]}, {i, 20}]
11 Write[AFileHandle, RandomPairs]
12 !! ANewFileName
13 Write[AFileHandle, MatrixForm[RandomPairs]]
14 !! ANewFileName
15 Close[AFileHandle]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

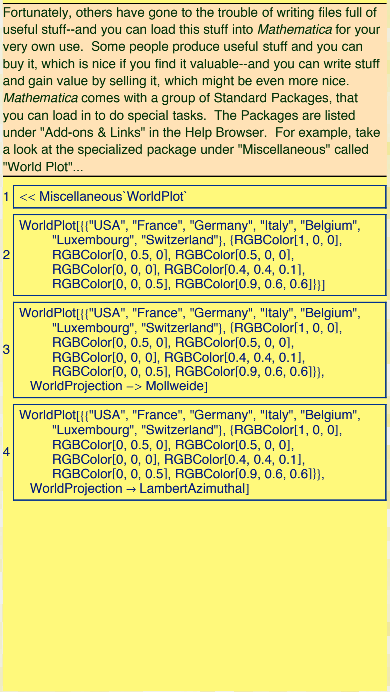
Lecture 04 MATHEMATICA® Example 6

Using Packages

notebook (non-evaluated) pdf (evaluated) html (evaluated)

There are a number of packages that come with MATHEMATICA® (and more that can be bought for special purposes). You should look through the various packages in the help browser to get an idea of what is there—it is also a good idea to take a look at the inside of a package by editing a package file with an editor. By doing this, you will see some of internal structure of MATHEMATICA® and good examples of professional programming.

1: A package is read in using the input operator `<<` or with `Needs`. After reading a bit about a package, it is straightforward to construct simple examples such as in this example of the `WorldPlot` subpackage in the `Miscellaneous` package.



Fortunately, others have gone to the trouble of writing files full of useful stuff--and you can load this stuff into *Mathematica* for your very own use. Some people produce useful stuff and you can buy it, which is nice if you find it valuable--and you can write stuff and gain value by selling it, which might be even more nice. *Mathematica* comes with a group of Standard Packages, that you can load in to do special tasks. The Packages are listed under "Add-ons & Links" in the Help Browser. For example, take a look at the specialized package under "Miscellaneous" called "World Plot"...

```
1  << Miscellaneous`WorldPlot`

2  WorldPlot[{{"USA", "France", "Germany", "Italy", "Belgium",
        "Luxembourg", "Switzerland"}, {RGBColor[1, 0, 0],
        RGBColor[0, 0.5, 0], RGBColor[0.5, 0, 0],
        RGBColor[0, 0, 0], RGBColor[0.4, 0.4, 0.1],
        RGBColor[0, 0, 0.5], RGBColor[0.9, 0.6, 0.6]}}]

3  WorldPlot[{{"USA", "France", "Germany", "Italy", "Belgium",
        "Luxembourg", "Switzerland"}, {RGBColor[1, 0, 0],
        RGBColor[0, 0.5, 0], RGBColor[0.5, 0, 0],
        RGBColor[0, 0, 0], RGBColor[0.4, 0.4, 0.1],
        RGBColor[0, 0, 0.5], RGBColor[0.9, 0.6, 0.6]}},
    WorldProjection –> Mollweide]

4  WorldPlot[{{"USA", "France", "Germany", "Italy", "Belgium",
        "Luxembourg", "Switzerland"}, {RGBColor[1, 0, 0],
        RGBColor[0, 0.5, 0], RGBColor[0.5, 0, 0],
        RGBColor[0, 0, 0], RGBColor[0.4, 0.4, 0.1],
        RGBColor[0, 0, 0.5], RGBColor[0.9, 0.6, 0.6]}},
    WorldProjection → LambertAzimuthal]
```

3.016 Home

Full Screen

Close

Quit

Lecture 5: Introduction to Mathematica IV

Graphics

Graphics are an important part of exploring mathematics and conveying its results. An informative plot or graphic that conveys a complex idea succinctly and naturally to an educated observer is a work of creative art. Indeed, art is sometimes defined as "an elevated means of communication," or "the means to inspire an observation, heretofore unnoticed, in another." Graphics are art; they are necessary. And, I think they are fun.

For graphics, we are limited to two- and three-dimensions, but, with the added possibility of animation, sound, and perhaps other sensory input in advanced environments, it is possible to usefully visualize more than three dimensions. Mathematics is not limited to a small number of dimensions; so, a challenge —or perhaps an opportunity—exists to uses artfulness to convey higher dimensional ideas graphically.

The introducetion to basic graphics starts with two-dimensional plots.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 05 MATHEMATICA® Example 1

Two-dimensional Plots I

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of simple x-y plots and how to decorate them.

1: When `Plot` gets a list of expressions as it first argument, it will superpose the curves obtained from each. In this example, the y-variable's display is controlled with `PlotRange`, and the curves' colors and thicknesses are controlled with a list for `PlotStyle`. Note that the rule `PlotStyle` is a list of descriptions such as `Hue`, `RGBColor`, `Thickness`, `Dashing`, etc.

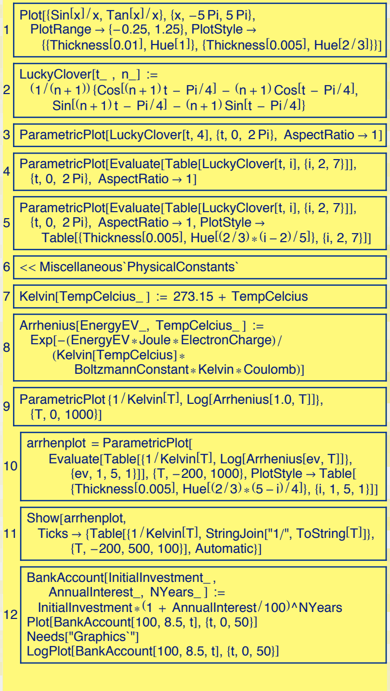
3: To plot a curve of the form, $(x(t), y(t))$ as a function of a parameter t, `ParamatricPlot` is called with its first argument being a list of x- and y-functions.

4: Superposition of parametric plots is obtained with a list of two-member lists.

9: With the physical constants package loaded, a function to convert degrees-Celcius to degrees Kelvin, and a function to calculate the Arrhenius function, an Arrhenius plot can be obtain with `ParametricPlot`.

11: By naming a plot, it can referenced and combined with more *Graphics Objects* with `Show`. In this case, a specialized "tick-scheme" is employed with "smart" labels for the $1/T$ axis.

12: `LogPlot` needs `Needs[” Graphics’”]`. Here is an example of a annually compounded bank account.



```
1  Plot[{Sin[x]/x, Tan[x]/x}, {x, −5 Pi, 5 Pi},
     PlotRange → {−0.25, 1.25}, PlotStyle →
       {{Thickness[0.01], Hue[1]}, {Thickness[0.005], Hue[2/3]}}]

2  LuckyClover[t_ , n_] :=
     (1/(n + 1)) {Cos[(n + 1) t − Pi/4] − (n + 1) Cos[t − Pi/4],
         Sin[(n + 1) t − Pi/4] − (n + 1) Sin[t − Pi/4]}

3  ParametricPlot[LuckyClover[t, 4], {t, 0, 2 Pi}, AspectRatio → 1]

4  ParametricPlot[Evaluate[Table[LuckyClover[t, i], {i, 2, 7}]],
     {t, 0, 2 Pi}, AspectRatio → 1]

5  ParametricPlot[Evaluate[Table[LuckyClover[t, i], {i, 2, 7}]],
     {t, 0, 2 Pi}, AspectRatio → 1, PlotStyle →
       Table[{Thickness[0.005], Hue[(2/3) ∗ (i − 2)/5]}, {i, 2, 7}]]

6  << Miscellaneous`PhysicalConstants`

7  Kelvin[TempCelcius_ ] := 273.15 + TempCelcius

8  Arrhenius[EnergyEV_, TempCelcius_ ] :=
     Exp[−(EnergyEV ∗ Joule ∗ ElectronCharge) /
         (Kelvin[TempCelcius] ∗
             BoltzmannConstant ∗ Kelvin ∗ Coulomb)]

9  ParametricPlot {1/Kelvin[T], Log[Arrhenius[1.0, T]]},
     {T, 0, 1000}]

10 arrhenplot = ParametricPlot[
       Evaluate[Table[{1/Kelvin[T], Log[Arrhenius[ev, T]]},
           {ev, 1, 5, 1}]], {T, −200, 1000}, PlotStyle → Table[
           {Thickness[0.005], Hue[(2/3) ∗ (5 − i)/4]}, {i, 1, 5, 1}]]

11 Show[arrhenplot,
     Ticks → {Table[{1/Kelvin[T], StringJoin["1/", ToString[T]]},
           {T, −200, 500, 100}], Automatic}]

12 BankAccount[InitialInvestment_ ,
       AnnualInterest_, NYears_ ] :=
     InitialInvestment ∗ (1 + AnnualInterest/100)^NYears
   Plot[BankAccount[100, 8.5, t], {t, 0, 50}]
   Needs["Graphics`"]
   LogPlot[BankAccount[100, 8.5, t], {t, 0, 50}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 05 MATHEMATICA® Example 2

Two-dimensional Plots II

notebook (non-evaluated) pdf (evaluated) html (evaluated)

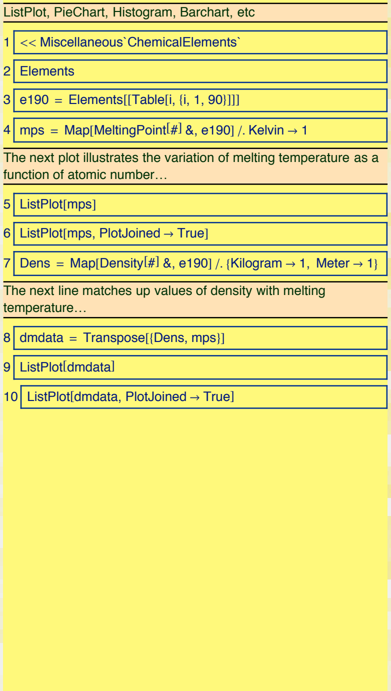
Examples of incorporating data into x-y plots. Sometimes you will want to plot numbers that come from elsewhere—otherwise known as data. Presumably, data will be imported with file I/O.

2: The chemical elements and information about them is accessible via the package `Miscellaneous‘ChemicalElements‘`. Here, this data will be used to make plots for the 1^{st}–90^{th} elements.

6: Subsequent to extracting the melting points by `Map`ing the function `MeltingPoint` onto the element-list, the trends in melting point with atomic number is visualized. Using `PlotJoined` set to true in `ListPlot`, makes the trend more visible.

8: `Dens` and `mps` are each lists with 90 number-like objects. Therefore {`Dens`, `mps`} is a list of two lists—it has dimensions 2 × 90. `ListPlot` will take data of the form $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_N, y_N\}\}$, i.e., dimensions 90×2. `Transform` will convert the data to the correct form for `ListPlot`.

10: By joining the data points with line-segments, a relationship between density and melting point becomes visible.



ListPlot, PieChart, Histogram, Barchart, etc

```
1  << Miscellaneous`ChemicalElements`
2  Elements
3  e190 = Elements[[Table[i, {i, 1, 90}]]]
4  mps = Map[MeltingPoint[#] &, e190] /. Kelvin → 1
```

The next plot illustrates the variation of melting temperature as a function of atomic number…

```
5  ListPlot[mps]
6  ListPlot[mps, PlotJoined → True]
7  Dens = Map[Density[#] &, e190] /. {Kilogram → 1, Meter → 1}
```

The next line matches up values of density with melting temperature…

```
8  dmdata = Transpose[{Dens, mps}]
9  ListPlot[dmdata]
10 ListPlot[dmdata, PlotJoined → True]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 05 MATHEMATICA® Example 3

Three Dimensional Graphics

notebook (non-evaluated) pdf (evaluated) html (evaluated)

It would be better to say, 3D graphics projected onto a 2D screen.

Using different `ViewPoints` and perspective, one can obtain informative 3D information on a screen. Unfortunately, MATHEMATICA® 's front end does not have the capability of spinning or flying-around 3D graphics (yet). But, such things are possible by exporting this information into other formats. An example of such can be found here: http://pruffle.mit.edu/c̃carter/talks/Stuttgart_INCEMS/node68.html

2: This is a function that computes the electrostatic potential over a 11×11 square-lattice of point-charges centered on the z-plane as a function of x, y, and z. In this example, some simpler methods of visualizing this four-dimensional object will be examined.

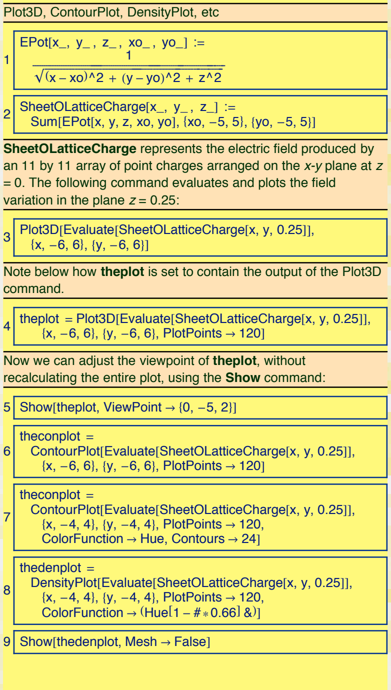
4: With sufficiently many `PlotPoints` the structure of the potential at a fixed distance $z = 0.25$ is made apparant.

5: Without recomputing all the data, the `ViewPoint` can be changed if the `SurfaceGraphics` object is assigned to a symbol that can be passed to `Show`. There is a handy *3D ViewPoint Selector* in MATHEMATICA® 's Input menu.

6: By computing *isopotentials* or *contours of constant potential* for $z = 0.25$, and using color. The `ContourPlot` function produces something like a *topographic map*.

7: The `ContourPlot` can be easily colorized by setting the `ColorFunction`-option to `Hue`. . .

8: But, the `Hue` cycles from red to green to blue—and then back to red again. Here is a method to remove redundant colors.



Plot3D, ContourPlot, DensityPlot, etc

```
1  EPot[x_, y_ , z_ , xo_ , yo_] :=
                         1
     ──────────────────────────────────
     √(x – xo)^2 + (y – yo)^2 + z^2

2  SheetOLatticeCharge[x_, y_ , z_] :=
     Sum[EPot[x, y, z, xo, yo], {xo, –5, 5}, {yo, –5, 5}]
```

SheetOLatticeCharge represents the electric field produced by an 11 by 11 array of point charges arranged on the *x-y* plane at *z* = 0. The following command evaluates and plots the field variation in the plane *z* = 0.25:

```
3  Plot3D[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
     {x, –6, 6}, {y, –6, 6}]
```

Note below how **theplot** is set to contain the output of the Plot3D command.

```
4  theplot = Plot3D[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
       {x, –6, 6}, {y, –6, 6}, PlotPoints → 120]
```

Now we can adjust the viewpoint of **theplot**, without recalculating the entire plot, using the **Show** command:

```
5  Show[theplot, ViewPoint → {0, –5, 2}]

6  theconplot =
     ContourPlot[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
       {x, –6, 6}, {y, –6, 6}, PlotPoints → 120]

7  theconplot =
     ContourPlot[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
       {x, –4, 4}, {y, –4, 4}, PlotPoints → 120,
       ColorFunction → Hue, Contours → 24]

8  thedenplot =
     DensityPlot[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
       {x, –4, 4}, {y, –4, 4}, PlotPoints → 120,
       ColorFunction → (Hue[1 – # ∗ 0.66] &)]

9  Show[thedenplot, Mesh → False]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 05 MATHEMATICA® Example 4

Graphics Primitives and Graphical Constructions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of placing *Graphics Primitives* onto the display are developed and a tidy graphical demonstration of a *Wulff construction* is presented. Because PostScript is one of the graphics primitives, you can draw anything that can be imaged in another application. You can also import your own drawing and images into MATHEMATICA®
.

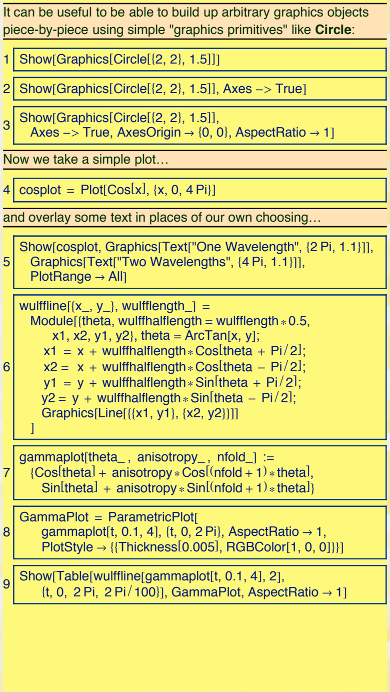
3: A `Circle` is a graphics primitive. `Graphics` takes graphics primitives as arguments and converts them to a *graphics object*. `Show` takes the graphics object and sends it to the display. As MATHEMATICA® will pick an intersection for the axis display and a convenient scaling for both axes, `AxesOrigin` and `AspectRatio` should be specified in order to obtain the desired rendering.

5: Graphics primitives, such as `Text`, can be combined with two-dimensional plots to improve their *graphical content or exposition.*

6: The Wulff construction is a famous thermodynamic construction that predicts the equilibrium enclosing surface of an anisotropic isolated body. The anistropic surface tension, $\gamma(\hat{n})$, is the amount of work (per unit area) required to produce a planar surface with outward normal $\hat{n}$. The construction proceeds by drawing a bisecting plane at each point of the polar plot $\gamma(\hat{n})\hat{n}$. The interior of all bisectors is the resulting *Wulff shape.*

8: This is an example $\gamma(\hat{n})$ with the surface tension being smaller in the $\langle 11 \rangle$-directions.

9: By combining the graphics primitives from the *wulffline* function with the γ-plot. the equilibrium shape is visualized.



It can be useful to be able to build up arbitrary graphics objects piece-by-piece using simple "graphics primitives" like **Circle**:

```
1 Show[Graphics[Circle[{2, 2}, 1.5]]]
```

```
2 Show[Graphics[Circle[{2, 2}, 1.5]], Axes –> True]
```

```
3 Show[Graphics[Circle[{2, 2}, 1.5]],
    Axes –> True, AxesOrigin → {0, 0}, AspectRatio → 1]
```

Now we take a simple plot…

```
4 cosplot = Plot[Cos[x], {x, 0, 4 Pi}]
```

and overlay some text in places of our own choosing…

```
5 Show[cosplot, Graphics[Text["One Wavelength", {2 Pi, 1.1}]],
    Graphics[Text["Two Wavelengths", {4 Pi, 1.1}]],
    PlotRange → All]
```

```
6 wulffline[{x_, y_}, wulfflength_] =
    Module[{theta, wulffhalflength = wulfflength * 0.5,
          x1, x2, y1, y2}, theta = ArcTan[x, y];
        x1 = x + wulffhalflength * Cos[theta + Pi/2];
        x2 = x + wulffhalflength * Cos[theta − Pi/2];
        y1 = y + wulffhalflength * Sin[theta + Pi/2];
       y2 = y + wulffhalflength * Sin[theta − Pi/2];
       Graphics[Line[{{x1, y1}, {x2, y2}}]]
    ]
```

```
7 gammaplot[theta_ , anisotropy_ , nfold_] :=
    {Cos[theta] + anisotropy * Cos[(nfold + 1) * theta],
       Sin[theta] + anisotropy * Sin[(nfold + 1) * theta]}
```

```
8 GammaPlot = ParametricPlot[
       gammaplot[t, 0.1, 4], {t, 0, 2 Pi}, AspectRatio → 1,
       PlotStyle → {{Thickness[0.005], RGBColor[1, 0, 0]}}]
```

```
9 Show[Table[wulffline[gammaplot[t, 0.1, 4], 2],
       {t, 0, 2 Pi, 2 Pi/100}], GammaPlot, AspectRatio → 1]
```

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

Lecture 05 Mathematica® Example 5

Animation

notebook (non-evaluated) pdf (evaluated) html (evaluated)

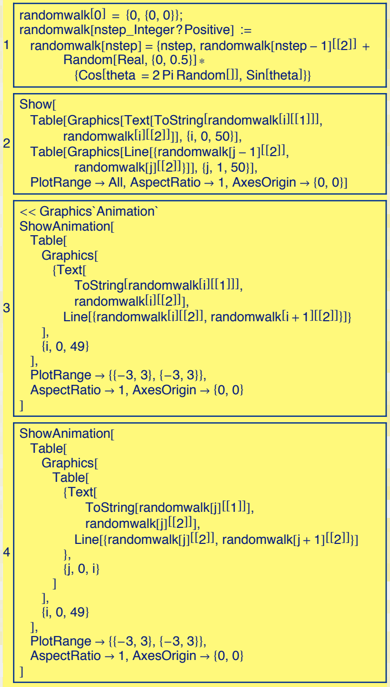
A *random walk* process is an important concept in diffusion and other statistical phenomena. An animation of a 2D random walk process is developed.

1: This is a recursive function that simulates a random walk process. Each step in the random walk is recorded as a list structure, {iteration number, $\{x, y\}$}, and assigned to *randomwalk* [iteration number]. For each step (or iteration), a number between 0 and 1/2 is (for the magnitude of the displacement) and an angle between 0 and 2π (for the direction) are selected randomly from a uniform distribution.

2: This shows the history of a random walk after 50 iterations by using graphics primitives.

3: This will produce a sequence of images which can be grouped together and then collapsing the cell by double-clicking it. The collapsed cell can be animated by using a menu item under the Cell-menu. Here, the step is depicted with a number at the current position and a line segment to the subsequent position.

4: This is a similar animation, but the history of each previous step is included in the graphical display.



```
1
randomwalk[0] = {0, {0, 0}};
randomwalk[nstep_Integer?Positive] :=
  randomwalk[nstep] = {nstep, randomwalk[nstep − 1][[2]] +
        Random[Real, {0, 0.5}] ∗
          {Cos[theta = 2 Pi Random[]], Sin[theta]}}
```

```
2
Show[
  Table[Graphics[Text[ToString[randomwalk[i][[1]]],
        randomwalk[i][[2]]]], {i, 0, 50}],
  Table[Graphics[Line[{randomwalk[j − 1][[2]],
          randomwalk[j][[2]]}]], {j, 1, 50}],
  PlotRange → All, AspectRatio → 1, AxesOrigin → {0, 0}]
```

```
3
<< Graphics`Animation`
ShowAnimation[
  Table[
    Graphics[
      {Text[
          ToString[randomwalk[i][[1]]],
          randomwalk[i][[2]]],
        Line[{randomwalk[i][[2]], randomwalk[i + 1][[2]]}]}
    ],
    {i, 0, 49}
  ],
  PlotRange → {{−3, 3}, {−3, 3}},
  AspectRatio → 1, AxesOrigin → {0, 0}
]
```

```
4
ShowAnimation[
  Table[
    Graphics[
      Table[
        {Text[
            ToString[randomwalk[j][[1]]],
            randomwalk[j][[2]]],
          Line[{randomwalk[j][[2]], randomwalk[j + 1][[2]]}]
        },
        {j, 0, i}
      ]
    ],
    {i, 0, 49}
  ],
  PlotRange → {{−3, 3}, {−3, 3}},
  AspectRatio → 1, AxesOrigin → {0, 0}
]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 6: Linear Algebra I

Reading:
Kreyszig Sections: 7.5, 7.6, 7.7, 7.8, 7.9 (pages302–305, 306–307, 308–314, 315–323, 323–329)

Vectors

Vectors as a list of associated information

$$\vec{x} = \begin{pmatrix} \text{number of steps to the east} \\ \text{number of steps to the north} \\ \text{number steps up vertical ladder} \end{pmatrix} \tag{6-1}$$

$$\vec{x} = \begin{pmatrix} 3 \\ 2.4 \\ 1.5 \end{pmatrix} \qquad \text{determines position} \qquad \begin{pmatrix} x_{\text{east}} \\ x_{\text{north}} \\ x_{\text{up}} \end{pmatrix} \tag{6-2}$$

The vector above is just one example of a position vector. We could also use coordinate systems that differ from the Cartesian (x, y, z) to represent the location. For example, the location in *cylindrical coordinate system* could be written as

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\cos\theta \\ r\sin\theta \\ z \end{pmatrix} \tag{6-3}$$

as a *Cartesian vector* in terms of the cylindrical coordinates (r, θ, z).

The position could also be written as a cylindrical, or *polar* vector

$$\vec{x} = \begin{pmatrix} r \\ \theta \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}\frac{y}{x} \\ z \end{pmatrix} \tag{6-4}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

where the last term is the polar vector in terms of the Cartesian coordinates. Similar rules would apply for other coordinate systems like spherical, elliptic, etc.

However, vectors need not represent position at all, for example:

$$\vec{n} = \begin{pmatrix} \text{number of Hydrogen atoms} \\ \text{number of Helium atoms} \\ \text{number of Lithum atoms} \\ \vdots \\ \text{number of Plutonium atoms} \\ \vdots \end{pmatrix} \tag{6-5}$$

Scalar multiplication

$$\frac{1}{\mathrm{N}_{\text{avag.}}}\vec{n} \equiv \begin{pmatrix} \frac{\text{number of H}}{\mathrm{N}_{\text{avag.}}} \\ \frac{\text{number of He}}{\mathrm{N}_{\text{avag.}}} \\ \frac{\text{number of Li}}{\mathrm{N}_{\text{avag.}}} \\ \vdots \\ \frac{\text{number of Pu}}{\mathrm{N}_{\text{avag.}}} \\ \vdots \end{pmatrix} = \begin{pmatrix} \text{moles of H} \\ \text{moles of He} \\ \text{moles of Li} \\ \vdots \\ \text{moles of Pu} \\ \vdots \end{pmatrix} = \vec{m} \tag{6-6}$$

Vector norms

$$\|\vec{x}\| \equiv x_1^2 + x_2^2 + \ldots x_k^2 = \text{euclidean separation} \tag{6-7}$$

$$\|\vec{n}\| \equiv n_{\mathrm{H}} + n_{\mathrm{He}} + \ldots n_{132?} = \text{total number of atoms} \tag{6-8}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Unit vectors

$$\text{unit direction vector} \qquad\qquad \text{mole fraction composition} \tag{6-9}$$

$$\hat{x} = \frac{\vec{x}}{\|\vec{x}\|} \qquad\qquad \hat{m} = \frac{\vec{m}}{\|\vec{m}\|} \tag{6-10}$$

Extra Information and Notes

Potentially interesting but currently unnecessary

If $\Re$ stands for the set of all real numbers (i.e., 0, -1.6, $\pi/2$, etc.), then can use a shorthand to specify the position vector, $\vec{x} \in \Re^N$ (e.g., each of the N entries in the vector of length N must be a real number—or in the set of real numbers. $\|\vec{x}\| \in \Re$.

For the unit (direction) vector: $\hat{x} = \{\vec{x} \in \Re^3 \mid \|\vec{x}\| = 1\}$ (i.e, the unit direction vector is the set of all position vectors such that their length is unity—-or, the unit direction vector is the subset of all position vectors that lie on the unit sphere. $\vec{x}$ and $\hat{x}$ have the same number of entries, but compared to $\vec{x}$, the number of independent entries in $\hat{x}$ is smaller by one.

For the case of the composition vector, it is strange to consider the case of a negative number of atoms, so the mole fraction vector $\vec{n} \in (\Re^+)^{elements}$ ($\Re^+$ is the real non-negative numbers) and $\hat{m} \in (\Re^+)^{(elements\text{-}1)}$.

Matrices and Matrix Operations

Consider methane (CH_4), propane (C_3H_8), and butane (C_4H_{10}).

$$\underline{M_{HC}} = \begin{array}{cc|l} \text{H-column} & \text{C-column} & \\ \frac{\text{number of H}}{\text{methane molecule}} & \frac{\text{number of C}}{\text{methane molecule}} & \text{methane row} \\ \frac{\text{number of H}}{\text{propane molecule}} & \frac{\text{number of C}}{\text{propane molecule}} & \text{propane row} \\ \frac{\text{number of H}}{\text{butane molecule}} & \frac{\text{number of C}}{\text{butane molecule}} & \text{butane row} \end{array} \tag{6-11}$$

$$\underline{M_{HC}} = \begin{pmatrix} 4 & 1 \\ 8 & 3 \\ 10 & 4 \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \\ M_{31} & M_{32} \end{pmatrix} \tag{6-12}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Matrices as a linear transformation of a vector

$$\begin{aligned}
\vec{N_{HC}} &= (\text{number of methanes}, \text{number of propanes}, \text{number of butanes}) && \text{(6-13)}\\
&= (N_{HC\,m}, N_{HC\,p}, N_{HC\,b}) && \text{(6-14)}\\
&= (N_{HC\,1}, N_{HC\,2}, N_{HC\,3}) && \text{(6-15)}\\
& && \text{(6-16)}
\end{aligned}$$

$$\vec{N_{HC}}\underline{M_{HC}} \equiv \sum_{i=1}^{3} N_{HC\,i} M_{HC\,ij} = \vec{N} \qquad \text{(6-17)}$$

The "summation" convention is often used, where a repeated index is summed over all its possible values:

$$\sum_{i=1}^{p} N_{HC\,i} M_{HC\,ij} \equiv N_{HC\,i} M_{HC\,ij} = N_j \qquad \text{(6-18)}$$

For example, suppose

$$\vec{N_{HC}} = (1.2 \times 10^{12} \text{ molecules methane}, 2.3 \times 10^{13} \text{ molecules propane}, 3.4 \times 10^{14} \text{ molecules butane}) \qquad \text{(6-19)}$$

$$\begin{aligned}
&\vec{N_{HC}}\underline{M_{HC}} = \\
&(1.2 \times 10^{14} \text{ methanes}, 2.3 \times 10^{13} \text{ propanes}, 3.4 \times 10^{12} \text{ butanes}) \begin{pmatrix} \frac{4 \text{ atoms H}}{\text{methane}} & \frac{1 \text{ atoms C}}{\text{methane}} \\ \frac{8 \text{ atoms H}}{\text{propane}} & \frac{3 \text{ atoms C}}{\text{propane}} \\ \frac{10 \text{ atoms H}}{\text{butane}} & \frac{\text{atoms C}}{\text{butane}} \end{pmatrix} \\
=&(7.0 \times 10^{14} \text{ atoms H}, 2.0 \times 10^{14} \text{ atoms C})
\end{aligned} \qquad \text{(6-20)}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Matrix transpose operations

Above the lists (or vectors) of atoms were stored as rows, often it is convenient to store them as columns. The operation to take a row to a column (and vice-versa) is a "transpose".

$$\underline{M_{HC}}^T = \begin{array}{c} \\ \\ \end{array}\begin{array}{ccc} \text{methane-column} & \text{propane-column} & \text{butane-column} \\ \left(\begin{array}{c} \frac{\text{number of H}}{\text{methane molecule}} \\ \frac{\text{number of C}}{\text{methane molecule}} \end{array} \right. & \begin{array}{c} \frac{\text{number of H}}{\text{propane molecule}} \\ \frac{\text{number of C}}{\text{propane molecule}} \end{array} & \left. \begin{array}{c} \frac{\text{number of H}}{\text{butane molecule}} \\ \frac{\text{number of C}}{\text{butane molecule}} \end{array} \right) \end{array} \begin{array}{c} \\ \text{hydrogen row} \\ \text{carbon row} \end{array} \tag{6-21}$$

$$\vec{N_{HC}}^T = \begin{pmatrix} \text{number of methanes} \\ \text{number of propanes} \\ \text{number of butanes} \end{pmatrix} = \begin{pmatrix} N_{HC\,m} \\ N_{HC\,p} \\ N_{HC\,b} \end{pmatrix} \tag{6-22}$$

$$\underline{M_{HC}}^T \vec{N_{HC}}^T = \vec{N}^T \begin{pmatrix} 4 & 8 & 10 \\ 1 & 3 & 4 \end{pmatrix} \begin{pmatrix} \text{number of methanes} \\ \text{number of propanes} \\ \text{number of butanes} \end{pmatrix} = \begin{pmatrix} \text{number of H-atoms} \\ \text{number of C-atoms} \end{pmatrix} \tag{6-23}$$

Matrix Multiplication

The next example supposes that some process produces hyrdocarbons and be modeled with the pressure P and temperature T. Suppose (this is an artificial example) that the number of hydrocarbons produced in one millisecond can be related linearly to the pressure and temperature:

$$\begin{aligned} \text{number of methanes} &= \alpha P + \beta T \\ \text{number of propanes} &= \gamma P + \delta T \\ \text{number of butanes} &= \epsilon P + \phi T \end{aligned} \tag{6-24}$$

or

$$\vec{N_{HC}}^T = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \\ \epsilon & \phi \end{pmatrix} \begin{pmatrix} P \\ T \end{pmatrix} \tag{6-25}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 06 MATHEMATICA® Example 1

Matrices

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Here is an example operation that takes us from the processing vector $(P, T)^T$ to the number of hydrogens and carbons.

1: The matrix (Eq. 6-12) is entered as a list of sublists. The sub-lists are the rows of the matrix. The first elements of each row-sublist form the first column; the second elements are the second column and so on.

The `Length` of a matrix-object gives the number of row, and the second member of the result of `Dimensions` gives the number of columns.

All sublists of a matrix must have the same dimensions.

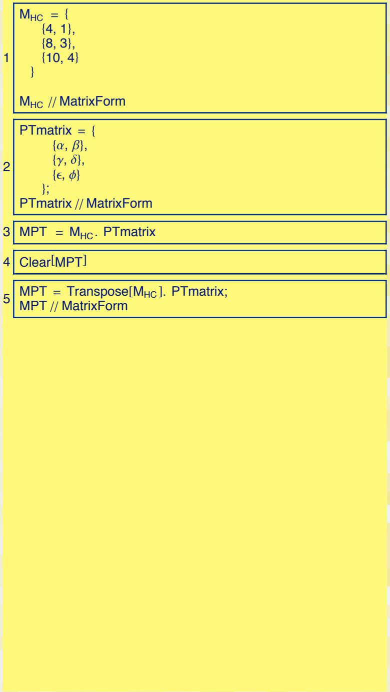
It is good practice to enter a matrix and then display it separately using `MatrixForm`. Otherwise, there is a risk of defining a symbol as a `MatrixForm`-object *and not as a matrix which was probably the intent.*

3: This command will generate an error.

Matrix multiplication in MATHEMATICA® is produced by the "dot" . operator. For matrix multiplication, $\underline{A}.\underline{B}$, the number of columns of $\underline{A}$ must be equal to the number of rows of $\underline{B}$.

5: The `Transpose` "flips" a matrix by producing a new matrix which has the original's i$^{\text{th}}$ row as the new matrix's i$^{\text{th}}$ column (or, equivalently the j$^{\text{th}}$ column as the new j$^{\text{th}}$ row). In this example, a 3 × 2-matrix ($\underline{PTmatrix}$) is being left-multiplied by a a 2 × 3-matrix.

The resulting matrix would map a vector with values P and T to a vector for the rate of production of C and H.



1
```
M_HC = {
    {4, 1},
    {8, 3},
    {10, 4}
  }

M_HC // MatrixForm
```

2
```
PTmatrix = {
      {α, β},
      {γ, δ},
      {ϵ, ϕ}
    };
PTmatrix // MatrixForm
```

3
```
MPT = M_HC . PTmatrix
```

4
```
Clear[MPT]
```

5
```
MPT = Transpose[M_HC]. PTmatrix;
MPT // MatrixForm
```

3.016

3.016 Home

Full Screen

Close

Quit

Matrix multiplication is defined by:

$$\underline{AB} = \sum_i A_{ki} B_{ij} \tag{6-26}$$

The indices of the matrix defined by the multiplication $\underline{AB} = \underline{C}$ are C_{kj}.

Matrix Inversion

Sometimes what we wish to know, "What vector is it ($\vec{x}$), when transformed by some matrix ($\underline{A}$) gives us a particular result ($\vec{b} = \underline{A}\vec{x}$)?"

$$\begin{aligned} \underline{A}\vec{x} &= \vec{b} \\ \underline{A}^{-1}\underline{A}\vec{x} &= \underline{A}^{-1}\vec{b} \\ \vec{x} &= \underline{A}^{-1}\vec{b} \end{aligned} \tag{6-27}$$

The inverse of a matrix is defined as something that when multiplied with the matrix leaves a product that has no effect on any vector. This special product matrix is called the *identity matrix*.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 04 MATHEMATICA® Example 2

Inverting Matrices

notebook (non-evaluated) pdf (evaluated) html (evaluated)

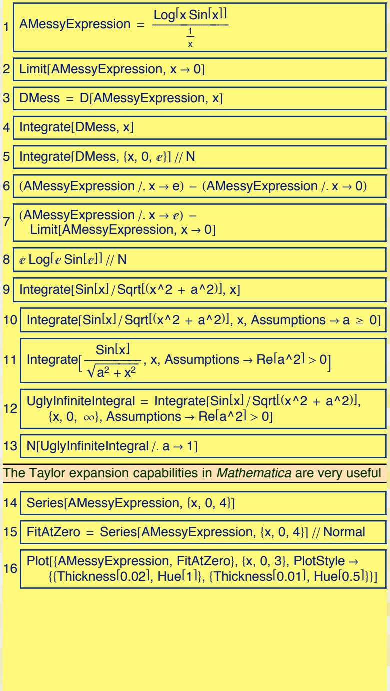
Our last example produced a linear operation that answered the question, “given a particular P and T, at what rate will C and H be produced?”

To answer the converse question, “If I want a particular rate of production for C and H, at what P and T should the process be carried out?”

To invert the question on linear processes, the matrix is inverted.

1: Inverting a matrix by hand is tedious and prone to error, `Inverse` does this in MATHEMATICA® . In this example, `Factor` is called on the result of `Inverse`. `Factor` is an example of a *threadable function*—it recursively operates on all members of any argument that is a list-object.

2: The *determinant* of a matrix is fundamentally linked to the existance of its inverse. In this example, it is observed that if the `Det` of a matrix vanishes, then the entries of its inverse are indeterminant.



```
1   AMessyExpression = Log[x Sin[x]]
                       ─────────────
                             1
                             ─
                             x
2   Limit[AMessyExpression, x → 0]
3   DMess = D[AMessyExpression, x]
4   Integrate[DMess, x]
5   Integrate[DMess, {x, 0, ⅇ}] // N
6   (AMessyExpression /. x → e) − (AMessyExpression /. x → 0)
7   (AMessyExpression /. x → ⅇ) −
      Limit[AMessyExpression, x → 0]
8   ⅇ Log[ⅇ Sin[ⅇ]] // N
9   Integrate[Sin[x] / Sqrt[(x^2 + a^2)], x]
10  Integrate[Sin[x] / Sqrt[(x^2 + a^2)], x, Assumptions → a ≥ 0]
11  Integrate[    Sin[x]     , x, Assumptions → Re[a^2] > 0]
              ───────────
              √(a² + x²)
12  UglyInfiniteIntegral = Integrate[Sin[x] / Sqrt[(x^2 + a^2)],
        {x, 0, ∞}, Assumptions → Re[a^2] > 0]
13  N[UglyInfiniteIntegral /. a → 1]
```

The Taylor expansion capabilities in *Mathematica* are very useful

```
14  Series[AMessyExpression, {x, 0, 4}]
15  FitAtZero = Series[AMessyExpression, {x, 0, 4}] // Normal
16  Plot[{AMessyExpression, FitAtZero}, {x, 0, 3}, PlotStyle →
        {{Thickness[0.02], Hue[1]}, {Thickness[0.01], Hue[0.5]}}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

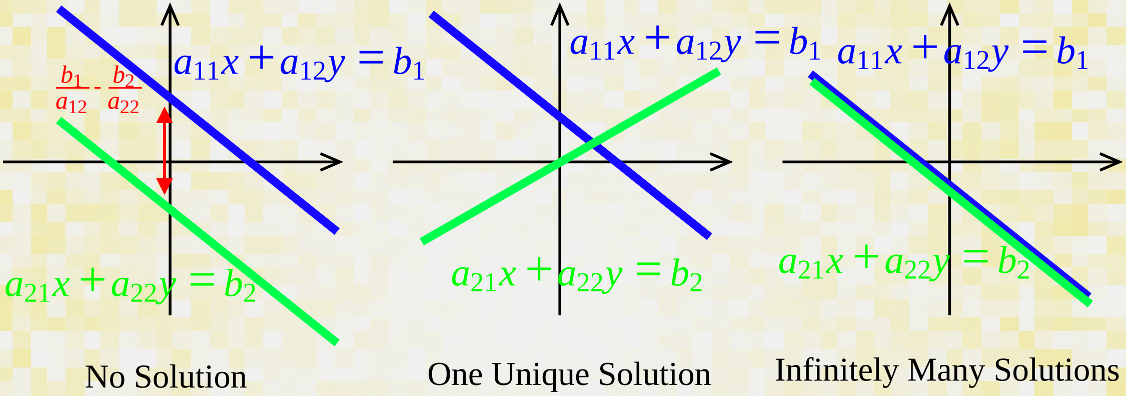
Full Screen

Close

Quit

Linear Independence: When solutions exist

$$\underline{A}\vec{x} = \vec{b}$$
$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{6-28}$$



$a_{11}x + a_{12}y = b_1$
$\frac{b_1}{a_{12}} - \frac{b_2}{a_{22}}$
$a_{21}x + a_{22}y = b_2$
No Solution
$a_{11}x + a_{12}y = b_1$
$a_{21}x + a_{22}y = b_2$
One Unique Solution
$a_{11}x + a_{12}y = b_1$
$a_{21}x + a_{22}y = b_2$
Infinitely Many Solutions

Figure 6-1: Geometric interpretation of solutions in two dimensions

Lecture 06 MATHEMATICA® Example 3

Eliminating redundant equations or variables

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Consider liquid water near the freezing point—dipole interactions will tend to make water molecules form clusters such as H_2O and H_4O_2.
Then the mapping from molecules to the number of atoms becomes:

$$\begin{pmatrix} 2 & 4 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} N_{H_2O} \\ N_{H_4O_2} \end{pmatrix} = \begin{pmatrix} N_H \\ N_O \end{pmatrix} \tag{6-29}$$

This example treats this case where the columns are not linearly independent.

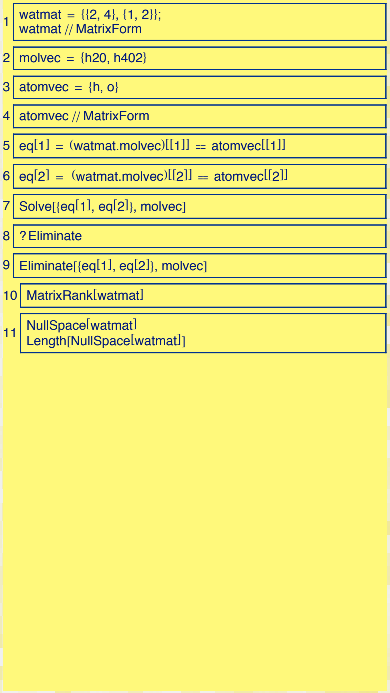
5: This equation is the same as the first row of $\underline{A}\vec{x}$ being set to the first entry of $\vec{b}$ for $\underline{A}\vec{x} = \vec{b}$.

7: This is an attempt to find the number of H_2O- and H_4O_2-molecules, given the number of H- and O-atoms. Of course, it has to fail.

9: `Eliminate` produces a logical equality for each redundancy in a set of equations. In this case, the result expresses the fact that 2 × (second row) is the same as the (first row).

10: The rank of a matrix, obtained with `MatrixRank`, gives the number of linearly independent rows.

11: The null space of a matrix, $\underline{A}$, is a linearly independent set of vectors $\vec{x}$, such that $\underline{A}\vec{x}$ is the zero-vector; this list can be obtained with `NullSpace`. The *nullity* is the number of vectors in a matrix's null space. The rank and the nullity must add up to the number of columns of $\underline{A}$



```
1  watmat = {{2, 4}, {1, 2}};
   watmat // MatrixForm
2  molvec = {h20, h402}
3  atomvec = {h, o}
4  atomvec // MatrixForm
5  eq[1] = (watmat.molvec)[[1]] == atomvec[[1]]
6  eq[2] = (watmat.molvec)[[2]] == atomvec[[2]]
7  Solve[{eq[1], eq[2]}, molvec]
8  ?Eliminate
9  Eliminate[{eq[1], eq[2]}, molvec]
10 MatrixRank[watmat]
11 NullSpace[watmat]
   Length[NullSpace[watmat]]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 7: Linear Algebra

Reading:
Kreyszig Sections: 13.1, 13.2, 13.5, 13.6 (pages602–606, 607–611, 623–626, 626–629)

Uniqueness and Existence of Linear System Solutions

It would be useful to use the Mathematica Help Browser and look through the section in the Mathematica Book: Advanced Mathematics/ Linear Algebra/Solving Linear Equations

A linear system of m equations in n variables $(x_1, x_2, \ldots, x_n)$ takes the form

$$\begin{aligned}
A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + \ldots + A_{1n}x_n &= b_1 \\
A_{21}x_1 + A_{22}x_2 + A_{23}x_3 + \ldots + A_{2n}x_n &= b_2 \\
\vdots &= \vdots \\
A_{k1}x_1 + A_{k2}x_2 + A_{k3}x_3 + \ldots + A_{kn}x_n &= b_k \\
\vdots &= \vdots \\
A_{m1}x_1 + A_{m2}x_2 + A_{m3}x_3 + \ldots + A_{mn}x_n &= b_m
\end{aligned} \tag{7-1}$$

and is fundamental to models of many systems.

The coefficients, A_{ij}, form a matrix and such equations are often written in an "index" short-hand known as the Einstein summation convention:

$$A_{ij}x_i = b_j \qquad \text{(Einstein summation convention)} \tag{7-2}$$

where *if an index* (here i) *is repeated in any set of multiplied terms,* (here $A_{ij}x_i$) *then a summation over all values of that index is implied.* Note that, by multiplying and summing in Eq. 7-2, the repeated

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

index i disappears from the right-hand-side. It can be said that the repeated index in “contracted” out of the equation and this idea is emphasized by writing Eq. 7-2 as

$$x_i A_{ij} = b_j \qquad \text{(Einstein summation convention)} \tag{7-3}$$

so that the “touching” index, i, is contracted out leaving a matching j-index on each side. In each case, Eqs. 7-2 and 7-3 represent m equations ($j = 1, 2, \ldots, m$) in the n variables ($i = 1, 2, \ldots, n$) that are contracted out in *each equation*. The summation convention is especially useful when the dimensions of the coefficient matrix is larger than two; for a linear elastic material, the elastic energy density can be written as:

$$E_{\text{elast}} = \frac{1}{2}\epsilon_{ij} C_{ijkl} \epsilon_{kl} = \frac{1}{2}\sigma_{pq} S_{pqrs} \sigma_{rs} \tag{7-4}$$

where C_{ijkl} and ϵ_{ij} are the fourth-rank *stiffness* tensor and second-rank elastic *strain* tensor; where S_{ijkl} and σ_{ij} are the fourth-rank compliance tensor and second-rank stress tensor;

In physical problems, the goal is typically to find the n x_i for a given m b_j in Eqs. 7-2, 7-3, or 7-1. This goal is conveniently represented in matrix-vector notation:

$$\underline{A}\vec{x} = \vec{b} \tag{7-5}$$

Lecture 07 MATHEMATICA® Example 1

Solving Linear Sets of Equations

notebook (non-evaluated) pdf (evaluated) html (evaluated)

This example is kind of backwards. We will take a matrix

$$\underline{A} = \begin{pmatrix} 1 & 2 & 1 & 1 \\ -1 & 4 & -2 & 0 \\ 1 & 2 & 4 & 5 \\ 1 & 0 & 1 & 1 \end{pmatrix} \text{ unknown vector } \vec{x} = \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} \text{ and known vector } \vec{b} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \qquad (7\text{-}6)$$

and extract four equations for input to `Solve` to obtain the solution to $\vec{x}$.

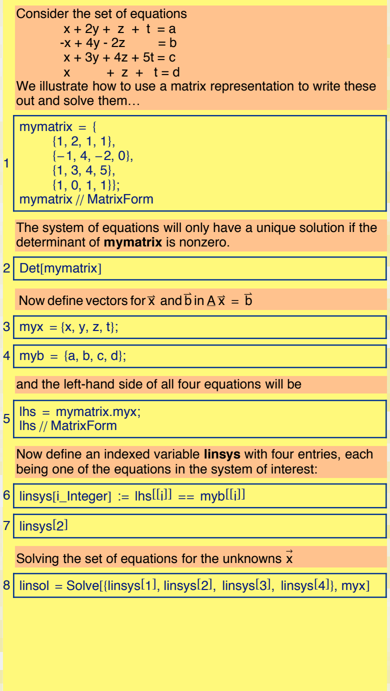
1: The coefficient matrix is a list of row-lists.

2: A unique solution will exist if the determinant, computed with `Det`, is non-zero.

5: The left-hand-side is a column-vector with four entries.

6: This function creates *logical equalities* for each corresponding entry on the left- and right-hand-sides.

8: Here, the function creates the input four equations and the `myx` contains the unknowns.



Consider the set of equations

```
  x + 2y +  z  +  t  = a
 -x + 4y - 2z        = b
  x + 3y + 4z + 5t = c
  x         +  z  +   t = d
```

We illustrate how to use a matrix representation to write these out and solve them…

```
1  mymatrix = {
          {1, 2, 1, 1},
          {−1, 4, −2, 0},
          {1, 3, 4, 5},
          {1, 0, 1, 1}};
   mymatrix // MatrixForm
```

The system of equations will only have a unique solution if the determinant of **mymatrix** is nonzero.

```
2  Det[mymatrix]
```

Now define vectors for $\vec{x}$ and $\vec{b}$ in $\underline{A}\,\vec{x} = \vec{b}$

```
3  myx = {x, y, z, t};
```

```
4  myb = {a, b, c, d};
```

and the left-hand side of all four equations will be

```
5  lhs = mymatrix.myx;
   lhs // MatrixForm
```

Now define an indexed variable **linsys** with four entries, each being one of the equations in the system of interest:

```
6  linsys[i_Integer] := lhs[[i]] == myb[[i]]
```

```
7  linsys[2]
```

Solving the set of equations for the unknowns $\vec{x}$

```
8  linsol = Solve[{linsys[1], linsys[2], linsys[3], linsys[4]}, myx]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 07 MATHEMATICA® Example 2

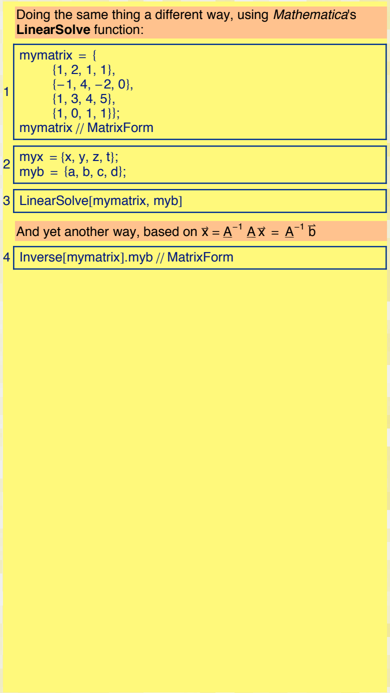
Inverting Matrices or Just Solving for the Unknown Vector

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Continuing the last example, it is much more compact to invert a matrix symbolically or numerically. If a matrix inverse is going to be used over and over again, it is probably best to compute and store the inverse once. However, if a one-time only solution for $\vec{x}$ in $\underline{A}\vec{x} = \vec{b}$ is needed, then computing the inverse is computationally less efficient than using an algorithm designed to solve for $\vec{x}$ directly. Here is an example of both methods.

3: `LinearSolve` can take two arguments, $\underline{A}$ and $\vec{b}$, and returns $\vec{x}$ that solves $\underline{A}\vec{x} = \vec{b}$. It will be noticibly faster than the following inversion method, especially for large matrices.

4: The matrix inverse is obtained with `Inverse` and a subsequent multiplication by the right-hand-side gives the solution.



Doing the same thing a different way, using *Mathematica*'s **LinearSolve** function:

```
1  mymatrix = {
         {1, 2, 1, 1},
         {-1, 4, -2, 0},
         {1, 3, 4, 5},
         {1, 0, 1, 1}};
   mymatrix // MatrixForm

2  myx = {x, y, z, t};
   myb = {a, b, c, d};

3  LinearSolve[mymatrix, myb]
```

And yet another way, based on $\vec{x} = \underline{A}^{-1}\,\underline{A}\,\vec{x} = \underline{A}^{-1}\,\vec{b}$

```
4  Inverse[mymatrix].myb // MatrixForm
```

3.016 Home

Full Screen

Close

Quit

Uniqueness of solutions to the nonhomogeneous system

$$\underline{A}\vec{x} = \vec{b} \tag{7-7}$$

Uniqueness of solutions to the homogeneous system

$$\underline{A}\vec{x_o} = \vec{0} \tag{7-8}$$

Adding solutions from the nonhomogeneous and homogenous systems

You can add any solution to the homogeneous equation (if they exist there are infinitely many of them) to any solution to the nonhomogeneous equation and the result is still a solution to the nonhomogeneous equation.

$$\underline{A}(\vec{x} + \vec{x_o}) = \vec{b} \tag{7-9}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 07 MATHEMATICA® Example 3

Determinants, Rank, and Nullity

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Several examples of determinant calculations are provided to illustrate the properties of determinants. When a determinant vanishes (i.e., $\det \underline{A} = 0$, there is no solution to the inhomogeneous equation $\det \underline{A} = \vec{b}$, but there will be an infinity of solutions to $\det \underline{A} = 0$; the infinity of solutions can be characterized by solving for a number *rank* of the entries of $\vec{x}$ in terms of the *nullity* of other entries of $\vec{x}$

1: A matrix is created where the third row is the sum of $p \times$ first row, $q \times$ second row, and $r \times$ fourth row. In other words, one row is a linear combination of the others.

2: The determinant is computed with `Det`.

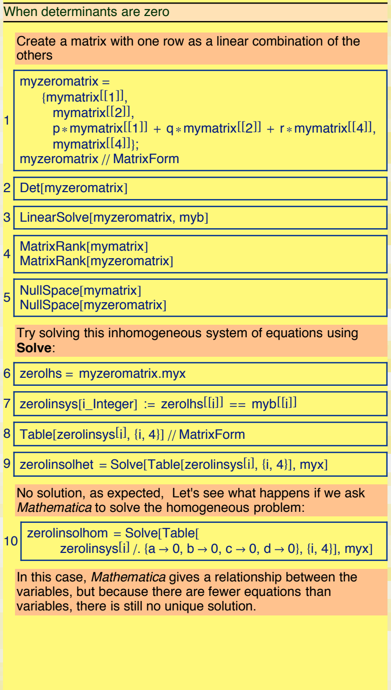
3: An attempt to solve the linear inhomogeneous equation should fail.

4: When the determinant is zero, there may still be some linearly independent rows or columns. The rank gives the number of linearly independent rows or columns and is computed with `MatrixRank`.

5: The *null space* of a matrix, $\underline{A}$, is a set of linearly independent vectors that, if left-multiplied by $\underline{A}$ gives a zero vector. The nullity is how many linearly independent vectors there are in the null space. Sometimes, vectors in the null space are called *killing vectors*.

9: Here, an attempt to use `Solve` for the heterogeneous system is attempted, but of course it is bound to fail...

10: However, this is the solution the singular homogeneous problem ($\underline{A}\vec{x} = \vec{0}$, where $\det \underline{A} = 0$. The solution is three (the rank) dimensional surface embedded in four dimensions (the rank plus the nullity). Notice that the solution is a multiple of the null space.



When determinants are zero

Create a matrix with one row as a linear combination of the others

```
1  myzeromatrix =
     {mymatrix[[1]],
       mymatrix[[2]],
       p∗mymatrix[[1]] + q∗mymatrix[[2]] + r∗mymatrix[[4]],
       mymatrix[[4]]};
   myzeromatrix // MatrixForm
2  Det[myzeromatrix]
3  LinearSolve[myzeromatrix, myb]
4  MatrixRank[mymatrix]
   MatrixRank[myzeromatrix]
5  NullSpace[mymatrix]
   NullSpace[myzeromatrix]
```

Try solving this inhomogeneous system of equations using **Solve**:

```
6  zerolhs = myzeromatrix.myx
7  zerolinsys[i_Integer] := zerolhs[[i]] == myb[[i]]
8  Table[zerolinsys[i], {i, 4}] // MatrixForm
9  zerolinsolhet = Solve[Table[zerolinsys[i], {i, 4}], myx]
```

No solution, as expected, Let's see what happens if we ask *Mathematica* to solve the homogeneous problem:

```
10 zerolinsolhom = Solve[Table[
        zerolinsys[i] /. {a → 0, b → 0, c → 0, d → 0}, {i, 4}], myx]
```

In this case, *Mathematica* gives a relationship between the variables, but because there are fewer equations than variables, there is still no unique solution.

3.016 Home

Full Screen

Close

Quit

Properties and Roles of the Matrix Determinant

In example 07-1, it was stated (item 2) that a unique solution exists if the matrix's determinant was non-zero. The solution,

$$\vec{x} = \begin{pmatrix} \frac{2a+2b-4c+18d}{\det \underline{A}} \\ \frac{7a-7d}{\det \underline{A}} \\ \frac{13a-8b+2c-23d}{\det \underline{A}} \\ \frac{-15a+6b+2c+19d}{\det \underline{A}} \end{pmatrix} \tag{7-10}$$

indicates why this is the case and also illustrates the role that the determinant plays in the solution. Clearly, if the determinant vanishes, then the solution is undetermined unless $\vec{b}$ is a zero-vector $\vec{0} = (0, 0, 0, 0)$. Considering the *algebraic equation*, $ax = b$, the determinant plays the role for matrices that the condition $a = 0$ plays for algebra: the inverse exists when $a \neq 0$ or $\det \underline{A} \neq 0$.

The determinant is only defined for square matrices; it derives from the elimination of the n unknown entries in $\vec{x}$ using all n equation (or rows) of

$$\underline{A}\vec{x} = 0 \tag{7-11}$$

For example, eliminating x and y from

$$\begin{gathered} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \text{gives the expression} \\ \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \equiv a_{11}a_{22} - a_{12}a_{21} = 0 \end{gathered} \tag{7-12}$$

and eliminating x, y, and z from

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

gives the expression

$$\det\underline{A} \equiv a_{11}a_{22}a_{33} - a_{11}a_{32}a_{23} + a_{21}a_{32}a_{13} - a_{21}a12a_{33} + a_{31}a_{12}a_{23} - a_{31}a_{22}a_{13} = 0 \tag{7-13}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The following general and true statements about determinants are plausible given the above expressions:

- The terms in the determinant's sum are products of a terms; one term comes from each column.
- Each term is one of all possible the products of an entry from each column.
- There is a plus or minus in front each term in the sum, $(-1)^p$, where p is the number of *neighbor exchanges required to put the rows in order* in each term written as an ordered product of their columns (as in example Eqs. 7-12 and 7-13).

These and the observation that it is impossible to eliminate $\vec{x}$ in Eqs. 7-12 and 7-13 if the information in the rows is redundant (i.e., there is not enough information—or independent equations—to solve for the $\vec{x}$) yield the general properties of determinants that are illustrated in the following example.

3.016 Home

Full Screen

Close

Quit

Lecture 07 Mathematica® Example 4

Properties of Determinants

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Rules corresponding how det A changes when the columns of $\underline{A}$ are permuted, or multiplied by a constant are demonstrated, along with $\det(\underline{AB}) = \det \underline{A} \det \underline{B}$ and $\underline{AB} \neq \underline{BA}$.

2: A matrix with random real entries between -1 and 1 is created.

5: Multiplying *one* column of a matrix by a constant a, multiplies the matrix's determinant by *one factor of* a.

7: Because the matrix has one linearly-dependent column, its determinant should vanish. This example demonstrates what happens with limited numerical precision operations on real numbers. The determinant is not zero, but could be considered *effectively zero*.

8: Problems with numerical imprecision can usually be alleviated with `Chop` which sets small magnitude numbers to zero.

10: Using, `Permutations`, create all possible permutations of two sets of three identical objects for subsequent construction of a symbolic matrix.

12: The symbolic matrix has a fairly simple determinant.

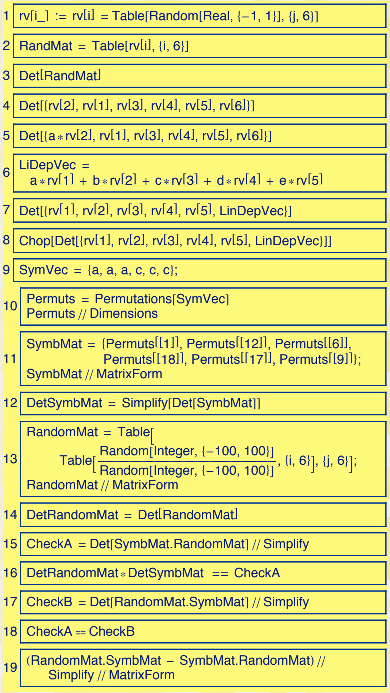
13: A matrix with random rational numbers is created...

14: And, of course, its determinant is also a rational number.

16: This demonstrates that determinant of a product is the product of determinants...

18: And, this (reduntantly) shows that the order of matrix multiplication does not affect the product rule for determinants.

19: However, the result of multiplying two matrix *does* depend on the order of multiplication: $\underline{AB} \neq \underline{BA}$, in general.



```
1  rv[i_] := rv[i] = Table[Random[Real, {-1, 1}], {j, 6}]
2  RandMat = Table[rv[i], {i, 6}]
3  Det[RandMat]
4  Det[{rv[2], rv[1], rv[3], rv[4], rv[5], rv[6]}]
5  Det[{a*rv[2], rv[1], rv[3], rv[4], rv[5], rv[6]}]
6  LiDepVec =
     a*rv[1] + b*rv[2] + c*rv[3] + d*rv[4] + e*rv[5]
7  Det[{rv[1], rv[2], rv[3], rv[4], rv[5], LinDepVec}]
8  Chop[Det[{rv[1], rv[2], rv[3], rv[4], rv[5], LinDepVec}]]
9  SymVec = {a, a, a, c, c, c};
10 Permuts = Permutations[SymVec]
   Permuts // Dimensions
11 SymbMat = {Permuts[[1]], Permuts[[12]], Permuts[[6]],
                Permuts[[18]], Permuts[[17]], Permuts[[9]]};
   SymbMat // MatrixForm
12 DetSymbMat = Simplify[Det[SymbMat]]
13 RandomMat = Table[
          Table[Random[Integer, {-100, 100}]
                ----------------------------, {i, 6}], {j, 6}];
                Random[Integer, {-100, 100}]
   RandomMat // MatrixForm
14 DetRandomMat = Det[RandomMat]
15 CheckA = Det[SymbMat.RandomMat] // Simplify
16 DetRandomMat*DetSymbMat == CheckA
17 CheckB = Det[RandomMat.SymbMat] // Simplify
18 CheckA == CheckB
19 (RandomMat.SymbMat - SymbMat.RandomMat) //
       Simplify // MatrixForm
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Vector Spaces

Consider the position vector

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{7-14}$$

The vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ can be used to generate any general position by suitable scalar multiplication and vector addition:

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = x \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + y \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + z \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{7-15}$$

Thus, three dimensional real space is "spanned" by the three vectors: $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. These three vectors are candidates as "basis vectors for $\Re^3$."

Consider the vectors $(a, -a, 0)$, $(a, a, 0)$, and $(0, a, a)$ for real $a \neq 0$.

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{x+y}{2a} \begin{pmatrix} a \\ -a \\ 0 \end{pmatrix} + \frac{x-y}{2a} \begin{pmatrix} a \\ a \\ 0 \end{pmatrix} + \frac{x-y+2z}{2a} \begin{pmatrix} 0 \\ a \\ a \end{pmatrix} \tag{7-16}$$

So $(a, -a, 0)$, $(a, a, 0)$, and $(0, a, a)$ for real $a \neq 0$ also are basis vectors and can be used to span $\Re^3$.

The idea of basis vectors and vector spaces comes up frequently in the mathematics of materials science. They can represent abstract concepts as well as shown by the following two dimensional basis set:

3.016

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

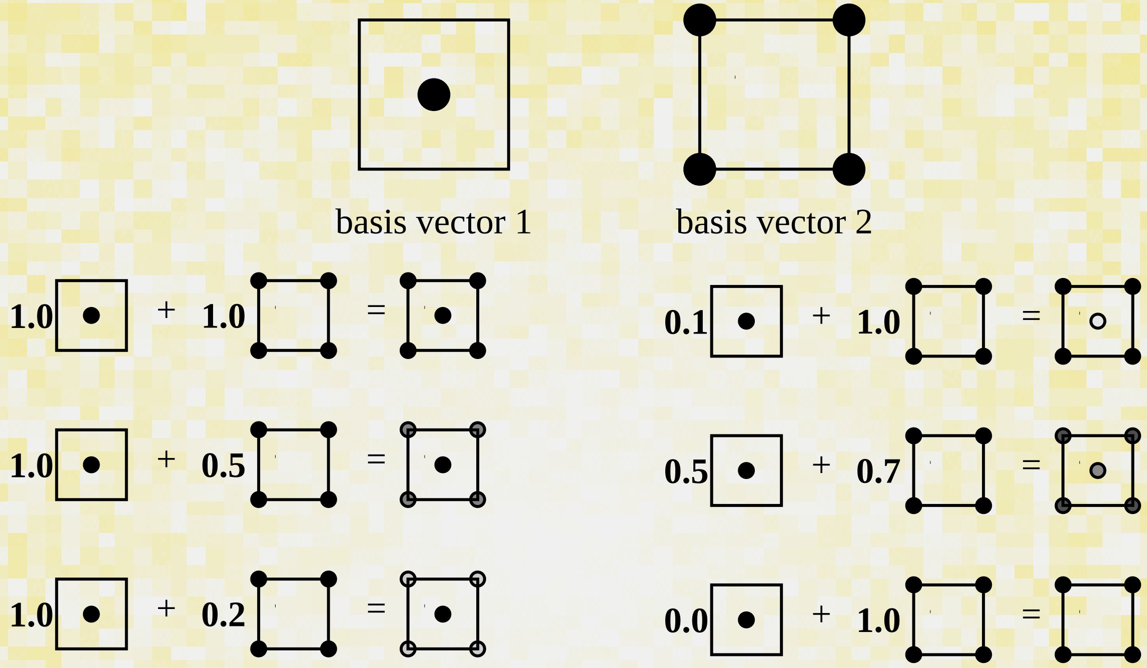


Figure 7-2: A vector space for two-dimensional CsCl structures. Any combination of center-site concentration and corner-site concentration can be represented by the sum of two basis vectors (or basis lattice). The set of all grey-grey patterns is a vector space of patterns.

3.016 Home
Full Screen
Close
Quit

Lecture 07 MATHEMATICA® Example 5

Visualization of linear transformations

notebook (non-evaluated) pdf (evaluated) html (evaluated)

An simple octagon with different colored faces is transformed by operating on all of it vertices with a matrix. This example demonstrates how symmetry operations, like rotations reflections, can be represented as a matrix multiplication, and how to visualize the results of linear transformations generally.

1: The package `Polyhedra` contains `Graphics Objects` with the coordinates of many common polyhedra.

2: This demonstrates how an `Octahedron` can be drawn on the screen.

3: The `ViewPoint` option to `Show` allows viewing from different points in 3D space.

4: `InputForm` reveals how the coordinates of the polydra are stored. . .

5: And, this can be mimicked to create a face-colored polyhedron with the `Hue` graphics directive.

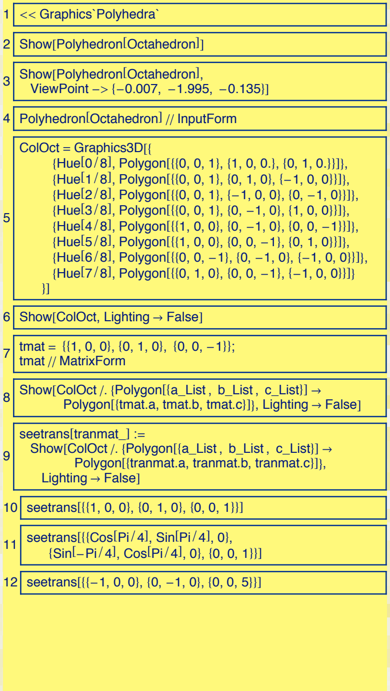
7: This is a matric which would create the mirror image across the z-axis of any point it multiplies.

8: This is a moderately sophisticated example of rule usage: it looks for triangles (`Polygons` with three points); names the points; and then multiplies a matrix by each of the points. The result in this case is a mirror operation.

9: This generalizes the previous example, by creating a function that takes a matrix as an argument.

11: This visualizes a rotation of $\pi/4$ around the z-axis.

12: This mirrors across the x- and y-axis and performs a linear expansion by a factor of 5 along the z-direction. The octagon volume increases by the determinant of the transformation matrix.



```
1   << Graphics`Polyhedra`

2   Show[Polyhedron[Octahedron]]

3   Show[Polyhedron[Octahedron],
      ViewPoint −> {−0.007, −1.995, −0.135}]

4   Polyhedron[Octahedron] // InputForm

5   ColOct = Graphics3D[{
            {Hue[0/8], Polygon[{{0, 0, 1}, {1, 0, 0.}, {0, 1, 0.}}]},
            {Hue[1/8], Polygon[{{0, 0, 1}, {0, 1, 0}, {−1, 0, 0}}]},
            {Hue[2/8], Polygon[{{0, 0, 1}, {−1, 0, 0}, {0, −1, 0}}]},
            {Hue[3/8], Polygon[{{0, 0, 1}, {0, −1, 0}, {1, 0, 0}}]},
            {Hue[4/8], Polygon[{{1, 0, 0}, {0, −1, 0}, {0, 0, −1}}]},
            {Hue[5/8], Polygon[{{1, 0, 0}, {0, 0, −1}, {0, 1, 0}}]},
            {Hue[6/8], Polygon[{{0, 0, −1}, {0, −1, 0}, {−1, 0, 0}}]},
            {Hue[7/8], Polygon[{{0, 1, 0}, {0, 0, −1}, {−1, 0, 0}}]}
        }]

6   Show[ColOct, Lighting → False]

7   tmat = {{1, 0, 0}, {0, 1, 0}, {0, 0, −1}};
    tmat // MatrixForm

8   Show[ColOct /. {Polygon[{a_List , b_List , c_List}] →
            Polygon[{tmat.a, tmat.b, tmat.c}]}, Lighting → False]

9   seetrans[tranmat_] :=
      Show[ColOct /. {Polygon[{a_List , b_List , c_List}] →
              Polygon[{tranmat.a, tranmat.b, tranmat.c}]},
        Lighting → False]

10  seetrans[{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}]

11  seetrans[{{Cos[Pi/4], Sin[Pi/4], 0},
        {Sin[−Pi/4], Cos[Pi/4], 0}, {0, 0, 1}}]

12  seetrans[{{−1, 0, 0}, {0, −1, 0}, {0, 0, 5}}]
```

3.016 Home

Full Screen

Close

Quit

Lecture 8: Complex Numbers and Euler's Formula

Reading:
Kreyszig Sections: 8.1, 8.2, 8.3 (pages334–338, 340–343, 345–348)

Complex Numbers and Operations in the Complex Plane

Consider, the number zero: it could be *operationally* defined as the number, which when multiplied by any other number always yields itself; and its other properities would follow.

Negative numbers could be defined operationally as something that gives rise to simple patterns. Multiplying by -1 gives rise to the pattern $1, -1, 1, -1, \ldots$ In the same vein, a number, $\imath$, can be created that doubles the period of the previous example: multiplying by $\imath$ gives the pattern: $1, \imath, -1, -\imath, 1, \imath, -1, -\imath, \ldots$ Combining the imaginary number, $\imath$, with the real numbers, arbitrarily long periods can be defined by multiplication; applications to periodic phenonena is probably where complex numbers have their greatest utility in science and engineering

With $\imath \equiv \sqrt{-1}$, the complex numbers can be defined as the space of numbers spanned by the vectors:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 \\ \imath \end{pmatrix} \tag{8-1}$$

so that any complex number can be written as

$$z = x \begin{pmatrix} 1 \\ 0 \end{pmatrix} + y \begin{pmatrix} 0 \\ \imath \end{pmatrix} \tag{8-2}$$

or just simply as

$$z = x + iy \tag{8-3}$$

where x and y are real numbers. $\text{Re}z \equiv x$ and $\text{Im}z \equiv y$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 08 MATHEMATICA® Example 1

Operations on complex numbers

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Straightforward examples of addition, subtraction, multiplication, and division of complex numbers are demonstrated. An example that demonstrates that MATHEMATICA® doesn't make *a priori* assumptions about whether a symbol is real or complex. An example function that converts a complex number to its polar form is constructed.

2: Just like `Pi` is a *mathematical constant*, the imaginary number is defined in MATHEMATICA® as something with the properties of $\imath$

3: Here, two numbers that are *potentially, but not necessarily* complex are defined.

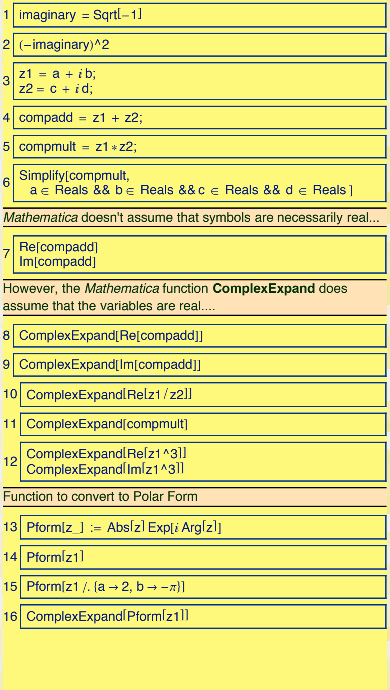
4: Addition and multiplication are defined as for any symbol; here the results do not appear to be very interesting *because* the other symbols could themselves be complex...

5: And, `Simplify` doesn't help much even with assumptions.

6: The real and imaginary parts of a complex entity can be extracted with `Re` and `Im`. This demonstrates that MATHEMATICA® hasn't made assumptions about `a`, `b`, `c`, and `d`.

8-12 However, `ComplexExpand` does make assumptions that symbols are real and, here, demonstrate the rules for addition, multiplication, division, and exponentiation.

13: `Abs` calculates the magnitude (also known as modulus or absolute value) and `Arg` calculates the argument (or angle) of a complex number. Here, they are used to define a function to convert and expression to an equivalent *polar form of a complex number.*



```
1  imaginary = Sqrt[-1]
2  (-imaginary)^2
3  z1 = a + ⅈ b;
   z2 = c + ⅈ d;
4  compadd = z1 + z2;
5  compmult = z1 * z2;
6  Simplify[compmult,
     a ∈ Reals && b ∈ Reals && c ∈ Reals && d ∈ Reals ]
```

Mathematica doesn't assume that symbols are necessarily real...

```
7  Re[compadd]
   Im[compadd]
```

However, the *Mathematica* function **ComplexExpand** does assume that the variables are real....

```
8  ComplexExpand[Re[compadd]]
9  ComplexExpand[Im[compadd]]
10 ComplexExpand[Re[z1 / z2]]
11 ComplexExpand[compmult]
12 ComplexExpand[Re[z1 ^3]]
   ComplexExpand[Im[z1 ^3]]
```

Function to convert to Polar Form

```
13 Pform[z_] := Abs[z] Exp[ⅈ Arg[z]]
14 Pform[z1]
15 Pform[z1 /. {a → 2, b → -π}]
16 ComplexExpand[Pform[z1]]
```

3.016 Home

Full Screen

Close

Quit

Complex Plane and Complex Conjugates

Because the complex basis can be written in terms of the vectors in Equation 8-1, it is natural to plot complex numbers in two dimensions—typically these two dimensions are the "complex plane" with $(0, \imath)$ associated with the y-axis and $(1, 0)$ associated with the x-axis.

The reflection of a complex number across the real axis is a useful operation. The image of a reflection across the real axis has some useful qualities and is given a special name—"the complex conjugate."

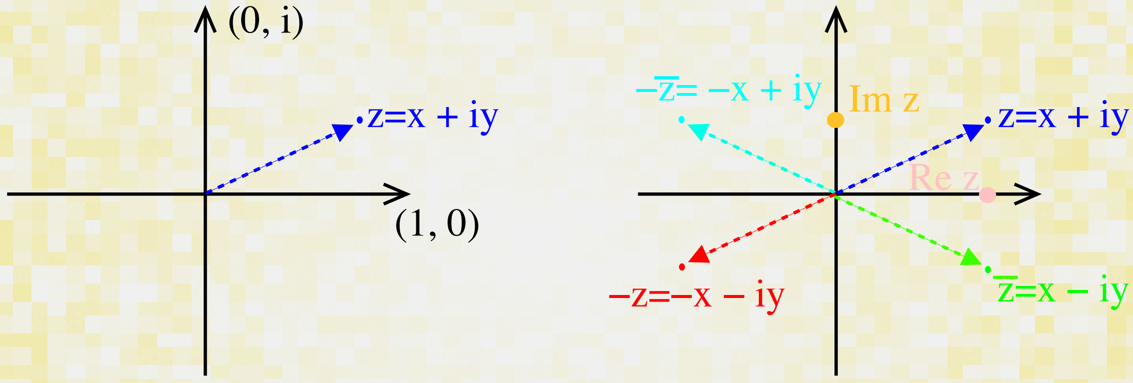


Figure 8-3: Plotting the complex number z in the complex plane: The complex conjugate $(\bar{z})$ is a *reflection* across the real axis; the minus $(-z)$ operation is an *inversion* through the origin; therefore $-(\bar{z}) = (\overline{-z})$ is equivalent to either a reflection across the imaginary axis or an inversion followed by a reflection across the real axis.
The real part of a complex number is the projection of the displacement in the real direction and also the average of the complex number and its conjugate: $\text{Re}z = (z + \bar{z})/2$. The imaginary part is the displacement projected onto the imaginary axis, or the complex average of the complex number and its reflection across the imaginary axis: $\text{Im}z = (z - \bar{z})/(2\imath)$.

Polar Form of Complex Numbers

There are physical situations in which a transformation from Cartesian (x, y) coordinates to polar (*or cylindrical*) coordinates (r, θ) simplifies the algebra that is used to describe the physical problem.

An equivalent coordinate transformation for complex numbers, $z = x + \imath y$, has an analogous simplifying effect for *multiplicative operations* on complex numbers. It has been demonstrated how the complex conjugate, $\bar{z}$, is related to a reflection—multiplication is related to a **counter-clockwise** rotation in the complex plane. Counter-clockwise rotation corresponds to increasing θ.

The transformations are:

$$\begin{aligned} (x, y) \to (r, \theta) & \begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases} \\ (r, \theta) \to (x, y) & \begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = \arctan\frac{y}{x} \end{cases} \end{aligned} \tag{8-4}$$

where arctan $\in (-\pi, \pi]$.

Multiplication, Division, and Roots in Polar Form

One advantage of the polar complex form is the simplicity of multiplication operations:

DeMoivre's formula:

$$z^n = r^n(\cos n\theta + \imath \sin n\theta) \tag{8-5}$$

$$\sqrt[n]{z} = \sqrt[n]{z}(\cos\frac{\theta + 2k\pi}{n} + \imath \sin\frac{\theta + 2k\pi}{n}) \tag{8-6}$$

3.016 Home

Full Screen

Close

Quit

Lecture 08 MATHEMATICA® Example 2

Numerical Properties of Operations on Complex Numbers

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Several examples demonstrate issues that arise when complex numbers are evaluated numerically.

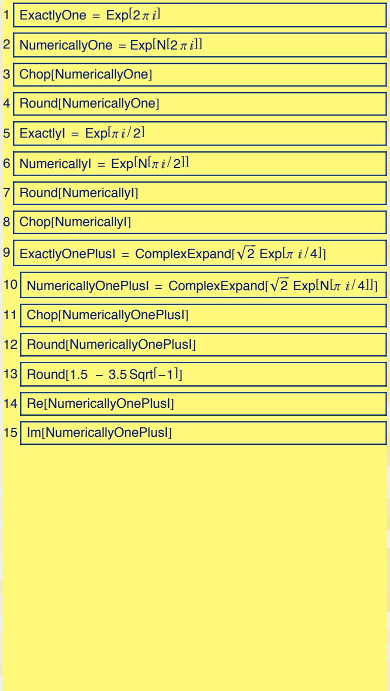
1: The relationship $e^{2\pi i} = 1$ is exact.

2: However, $e^{2.0\pi i}$ is *numerically* 1.

3: `Chop` removes small evalues that are presumed to be the result of numerical imprecision; it operates on complex numbers as well.

5–8 Here, the difference between something that is exactly $\imath$ and is numerically $1.0\imath$ is demonstrated…

9-12 And, this is similar demostration for $1+\imath$ using its polar form as a starting point.



```
1  ExactlyOne = Exp[2 π ⅈ]
2  NumericallyOne = Exp[N[2 π ⅈ]]
3  Chop[NumericallyOne]
4  Round[NumericallyOne]
5  ExactlyI = Exp[π ⅈ / 2]
6  NumericallyI = Exp[N[π ⅈ / 2]]
7  Round[NumericallyI]
8  Chop[NumericallyI]
9  ExactlyOnePlusI = ComplexExpand[√2 Exp[π ⅈ / 4]]
10 NumericallyOnePlusI = ComplexExpand[√2 Exp[N[π ⅈ / 4]]]
11 Chop[NumericallyOnePlusI]
12 Round[NumericallyOnePlusI]
13 Round[1.5 − 3.5 Sqrt[−1]]
14 Re[NumericallyOnePlusI]
15 Im[NumericallyOnePlusI]
```

3.016 Home

Full Screen

Close

Quit

Exponentiation and Relations to Trignometric Functions

Exponentiation of a complex number is defined by:

$$e^{z} = e^{x+iy} = e^{x}(\cos y + \imath \sin y) \tag{8-7}$$

Exponentiation of a purely imaginary number advances the angle by rotation:

$$e^{\imath y} = \cos y + \imath \sin y \tag{8-8}$$

combining Eq. 8-8 with Eq. 8-7 gives the particularly useful form:

$$z = x + \imath y = re^{\imath\theta} \tag{8-9}$$

and the useful relations (obtained simply by considering the complex plane's geometry)

$$e^{2\pi\imath} = 1 \quad e^{\pi\imath} = -1 \quad e^{-\pi\imath} = -1 \quad e^{\frac{\pi}{2}\imath} = \imath \quad e^{-\frac{\pi}{2}\imath} = -\imath \tag{8-10}$$

Subtraction of powers in Eq. 8-8 and generalization gives known relations for trigonometric functions:

$$\begin{aligned} \cos z &= \frac{e^{\imath z} + e^{-\imath z}}{2} & \sin z &= \frac{e^{\imath z} - e^{-iz}}{2\imath} \\ \cosh z &= \frac{e^{z} + e^{-z}}{2} & \sinh z &= \frac{e^{z} - e^{-z}}{2} \\ \cos z &= \cosh \imath z & \imath \sin z &= \sinh \imath z \\ \cos \imath z &= \cosh z & \sin \imath z &= \imath \sinh z \end{aligned} \tag{8-11}$$

Complex Numbers in Roots to Polynomial Equations

Complex numbers frequently arise when solving for the roots of a polynomial equation. There are many cases in which a model of system's physical behavior depends on whether the roots of a polynomial are real or imaginary, and if the real part is positive. While evaluating the nature of the roots is straightforward conceptually, this often creates difficulties computationally. Frequently, ordered lists of solutions are maintained and the behavior each solution is followed.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 08 MATHEMATICA® Example 3

Complex Roots of Polynomial Equations

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Here we construct an artificial example of a model that depends on a single parameter in a quadratic polynomial and illustrate methods to analyze and visualize its roots. Methods to "peek" at the form of long expressions are also demonstrated.

1-6 Using a prototype fourth order equation, a list of solutions are obtained; the real and imaginary parts are computed.

7: The above is generalized to a single parameter b in the quartic equation; the conditions that the roots are real will be visualized. `bsols`, the list of solution rule-lists is long and complicated.

8: First, one must consider the structure of `bsols`, `Dimensions` indicates it is a list of four lists, each of length 1.

9: `Short` is a practical method to observe the structure without filling up the screen display.

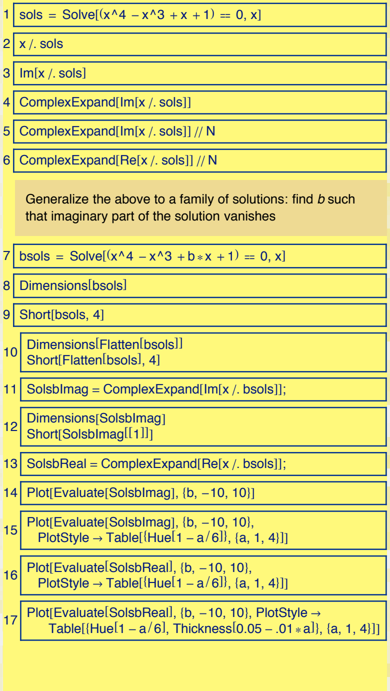
10: `Flatten` converts lists of lists into a single list—it is especially useful with the lists of rule lists that are returned from `Solve`.

10: Here, the real and complex parts of *each* of the solutions is obtained with `Re` and `Im` where the parameter b is assumed to be real via the use of `ComplexExpand`. **This may take a long time to evaluate on some computers.**

12: Which of the solutions (i.e., 1,2,3, or 4) is identified by using `Hue`.

13: Similarly, the real parts appear to converge to a single value when the imaginary parts (from above) appear. . .

14: But, the acual behavior is best illustrated by using `Thickness` to distinguish superposed values. The behavior of real parts of this solution have what is called a *pitchfork structure*.



```
1  sols = Solve[(x^4 - x^3 + x + 1) == 0, x]
2  x /. sols
3  Im[x /. sols]
4  ComplexExpand[Im[x /. sols]]
5  ComplexExpand[Im[x /. sols]] // N
6  ComplexExpand[Re[x /. sols]] // N
```

Generalize the above to a family of solutions: find *b* such that imaginary part of the solution vanishes

```
7  bsols = Solve[(x^4 - x^3 + b * x + 1) == 0, x]
8  Dimensions[bsols]
9  Short[bsols, 4]
10 Dimensions[Flatten[bsols]]
   Short[Flatten[bsols], 4]
11 SolsbImag = ComplexExpand[Im[x /. bsols]];
12 Dimensions[SolsbImag]
   Short[SolsbImag[[1]]]
13 SolsbReal = ComplexExpand[Re[x /. bsols]];
14 Plot[Evaluate[SolsbImag], {b, -10, 10}]
15 Plot[Evaluate[SolsbImag], {b, -10, 10},
     PlotStyle → Table[{Hue[1 - a/6]}, {a, 1, 4}]]
16 Plot[Evaluate[SolsbReal], {b, -10, 10},
     PlotStyle → Table[{Hue[1 - a/6]}, {a, 1, 4}]]
17 Plot[Evaluate[SolsbReal], {b, -10, 10}, PlotStyle →
       Table[{Hue[1 - a/6], Thickness[0.05 - .01 * a]}, {a, 1, 4}]]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 9: Eigensystems of Matrix Equations

Reading:
Kreyszig Sections: 8.1, 8.2, 8.3 (pages334–338, 340–343, 345–348)

Eigenvalues and Eigenvectors of a Matrix

The conditions for which general linear equation

$$\underline{A}\vec{x} = \vec{b} \tag{9-1}$$

has solutions for a given matrix $\underline{A}$, fixed vector $\vec{b}$, and unknown vector $\vec{x}$ have been determined.

The operation of a matrix on a vector—whether as a physical process, or as a geometric transformation, or just a general linear equation—has also been discussed.

Eigenvalues and eigenvectors are among the most important mathematical concepts with a very large number of applications in physics and engineering.

An eigenvalue problem (associated with a matrix $\underline{A}$) relates the operation of a matrix multiplication on a particular vector $\vec{x}$ to its multiplication by a particular scalar λ.

$$\underline{A}\vec{x} = \lambda\vec{x} \tag{9-2}$$

This equation bespeaks that the matrix operation can be replaced—or is equivalent to—a stretching or contraction of the vector: "$\underline{A}$ has some vector $\vec{x}$ for which its multiplication is simply a scalar multiplication operation by λ." $\vec{x}$ is an *eigenvector* of $\underline{A}$ and λ is $\vec{x}$'s associated *eigenvalue*.

The condition that Eq. 9-2 has solutions is that its associated homogeneous equation:

$$(\underline{A} - \lambda\underline{I})\vec{x} = \vec{0} \tag{9-3}$$

has a zero determinant:

$$\det(\underline{A} - \lambda\underline{I}) = 0 \tag{9-4}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Eq. 9-4 is a polynomial equation in λ (the power of the polynomial is the same as the size of the square matrix).

The eigenvalue-eigenvector system in Eq. 9-2 is solved by the following process:

1. Solve the characteristic equation (Eq. 9-4) for each of its roots λ_i.

2. Each root λ_i is used as an eigenvalue in Eq. 9-2 which is solved for its associated eigenvector $\vec{x_i}$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 09 MATHEMATICA® Example 1

Matrix eigensystems and their geometrical interpretation

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The symbolic computatation of eigenvalues and eigenvectors is demonstrated. In this example, a “cheat” is employed so that a matrix with “interesting” eigenvalues and eigenvectors is used as computation fodder. Just why the “cheat” works is demonstrated in Example 09-2.

1-3 These are initialization steps used to define `mymatrix` below. The idea here is to produce a non-trivial matrix with simple eigenvalues.

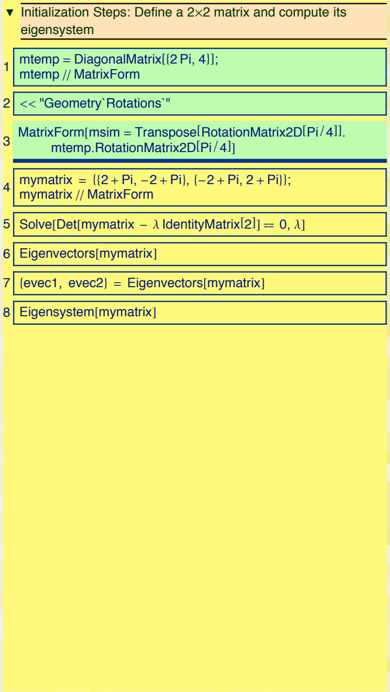
4: A “typical” 2×2 matrix $\underline{A}$ is defined. . .

5: Its eigenvalues are obtained by by first solving equation Eq. 9-4 for λ. . .

6: And, its eigenvectors could be obtained by putting each eigenvalue back into Eq. 9-2 and then solving $\vec{x}$ for each distinct λ. However, this tedious procedure can also be performed with `Eigenvectors`

7: Here, a matrix of eigenvectors is defined with named rows `evec1` and `evec2`.

8: `Eigensystem` generates the same results as `Eigenvectors` and `Eigenvalues` in one step.



▼ Initialization Steps: Define a 2×2 matrix and compute its eigensystem

```
1 mtemp = DiagonalMatrix[{2 Pi, 4}];
  mtemp // MatrixForm
2 << "Geometry`Rotations`"
3 MatrixForm[msim = Transpose[RotationMatrix2D[Pi / 4]].
        mtemp.RotationMatrix2D[Pi / 4]]
4 mymatrix = {{2 + Pi, −2 + Pi}, {−2 + Pi, 2 + Pi}};
  mymatrix // MatrixForm
5 Solve[Det[mymatrix − λ IdentityMatrix[2]] == 0, λ]
6 Eigenvectors[mymatrix]
7 {evec1, evec2} = Eigenvectors[mymatrix]
8 Eigensystem[mymatrix]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The matrix operation on a vector that returns a vector that is in the same direction is an eigensystem. A physical system that is associated can be interpreted in many different ways:

geometrically The vectors $\vec{x}$ in Eq. 9-2 are the ones that are unchanged by the linear transformation on the vector.

iteratively The vector $\vec{x}$ that is processed (either forward in time or iteratively) by $\underline{A}$ increases (or decreases if $\lambda < 1$) along its direction.

In fact, the eigensystem can be (and will be many times) generalized to other interpretations and generalized beyond linear matrix systems.

Here are some examples where eigenvalues arise. These examples generalize beyond matrix eigenvalues.

- As an analogy that will become real later, consider the "harmonic oscillator" equation for a mass, m, vibrating with a spring-force, k, this is simply Newton's equation:

$$m\frac{d^2x}{dt^2} = kx \tag{9-5}$$

  If we treat the second derivative as some linear operator, $\mathcal{L}_{\text{spring}}$ on the position x, then this looks like an eigenvalue equation:

$$\mathcal{L}_{\text{spring}} x = \frac{k}{m} x \tag{9-6}$$

- Letting the positions x_i form a vector $\vec{x}$ of a bunch of atoms of mass m_i, the harmonic oscillator can be generalized to a bunch of atoms that are interacting as if they were attached to each other by springs:

$$m_i \frac{d^2x_i}{dt^2} = \sum_{i\text{'s near neighbors } j} k_{ij}(x_i - x_j) \tag{9-7}$$

3.016 Home

Full Screen

Close

Quit

For each position i, the j-terms can be added to each side, leaving and operator that looks like:

$$\mathcal{L}_{\text{lattice}} = \begin{pmatrix} m_1\frac{d^2}{dt^2} & -k_{12} & 0 & -k_{14} & \dots & & 0 \\ -k_{21} & m_2\frac{d^2}{dt^2} & -k_{23} & 0 & & \dots & 0 \\ \vdots & & \ddots & & & & \vdots \\ \vdots & & & m_i\frac{d^2}{dt^2} & & & \vdots \\ & & & & \ddots & & \\ & & & & & m_{N-1}\frac{d^2}{dt^2} & -k_{N-1\ N} \\ 0 & 0 & \dots & & & -k_{N\ N-1} & m_N\frac{d^2}{dt^2} \end{pmatrix} \qquad (9\text{-}8)$$

The operator $\mathcal{L}_{\text{lattice}}$ has diagonal entries that have the spring (second-derivative) operator and one off-diagonal entry for each other atom that interacts with the atom associated with row i. The system of atoms can be written as:

$$\underline{k}^{-1}\mathcal{L}_{\text{lattice}}\vec{x} = \vec{x} \qquad (9\text{-}9)$$

which is another eigenvalue equation and solutions are constrained to have unit eigenvalues—these are the 'normal modes.'

- To make the above example more concrete, consider a system of three masses connected by springs.

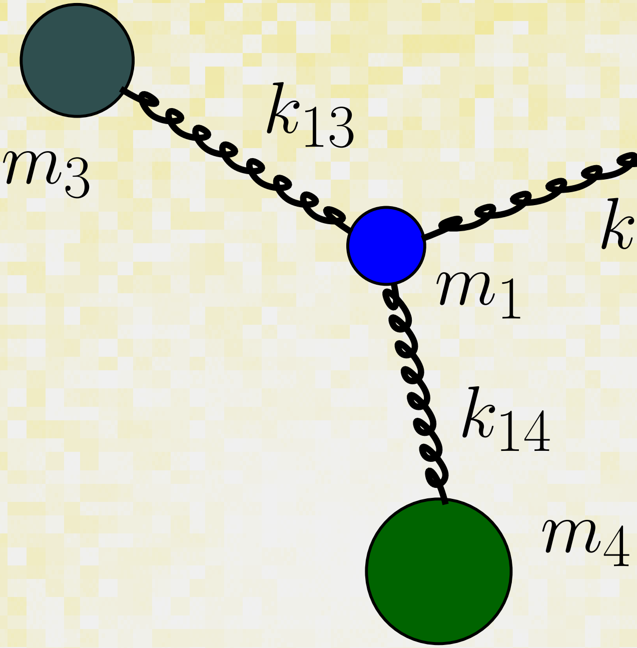


Figure 9-4: Four masses connected by three springs

The equations of motion become:

$$\begin{pmatrix} m_1 \frac{d^2}{dt^2} & -k_{12} & -k_{13} & -k_{14} \\ -k_{12} & m_2 \frac{d^2}{dt^2} & 0 & 0 \\ -k_{13} & 0 & m_2 \frac{d^2}{dt^2} & 0 \\ -k_{14} & 0 & 0 & m_2 \frac{d^2}{dt^2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} k_{12}+k_{13}+k_{14} & 0 & 0 & 0 \\ 0 & k_{12} & 0 & 0 \\ 0 & 0 & k_{13} & 0 \\ 0 & 0 & 0 & k_{14} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{9-10}$$

which can be written as

$$\mathcal{L}_{4\times 4}\vec{x} = \underline{k}\vec{x} \tag{9-11}$$

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

or

$$\underline{k}^{-1}\mathcal{L}_{4\times4}\vec{x} = \vec{x} \tag{9-12}$$

As will be discussed later, this system of equations can be "diagonalized" so that it becomes four independent equations. Diagonalization depends on finding the eigensystem for the operator.

- The one-dimensional Shrödinger wave equation is:

$$-\frac{\hbar}{2m}\frac{d^2\psi(x)}{dx^2} + U(x)\psi(x) = E\psi(x) \tag{9-13}$$

where the second derivative represents the kinetic energy and $U(x)$ is the spatial-dependent potential energy. The "Hamiltonian Operator" $\mathcal{H} = -\frac{\hbar}{2m}\frac{d^2}{dx^2} + U(x)$, operates on the wavefunction $\psi(x)$ and returns the wavefunction's total energy multiplied by the wavevector;

$$\mathcal{H}\psi(x) = E\psi(x) \tag{9-14}$$

This is another important eigenvalue equation (and concept!)

Symmetric, Skew-Symmetric, Orthogonal Matrices

Three types of matrices occur repeatedly in physical models and applications. They can be placed into three categories according to the conditions that are associated with their eigenvalues:

All real eigenvalues Symmetric matrices—those that have a "mirror-plane" along the northwest–southeast diagonal ($\underline{A} = \underline{A}^T$)—must have all real eigenvalues.

Hermetian matrices—the complex analogs of symmetric matrices—in which the reflection across the diagonal is combined with a complex conjugate operation ($a_{ij} = \bar{a_{ji}}$), must also have all real eigenvalues.

All imaginary eigenvalues Skew-symmetric (diagonal mirror symmetry combined with a minus) matrices ($-\underline{A} = \underline{A}^T$) must have all complex eigenvalues.

Skew-Hermitian matrices—-the complex analogs of skew-symmetric matrices ($a_{ij} = -\bar{a_{ji}}$)—have all imaginary eigenvalues.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Unitary Matrices: unit determinant Real matrices that satisfy $\underline{A}^T = \underline{A}^{-1}$ have the property that product of all the eigenvalues is ± 1. These are called *orthogonal* matrices and they have *orthonormal* rows. Their determinants are also ± 1.

This is generalized by complex matrices that satisfy $\bar{\underline{A}}^T = \underline{A}^{-1}$. These are called *unitary* matrices and their (complex) determinants have magnitude 1. Orthogonal matrices, $\underline{A}$, have the important physical property that they preserve the inner product: $\vec{x} \cdot \vec{y} = (\underline{A}\vec{x}) \cdot (\underline{A}\vec{y})$. When the orthogonal matrix is a rotation, the interpretation is that the vectors maintain their relationship to each other if they are both rotated.

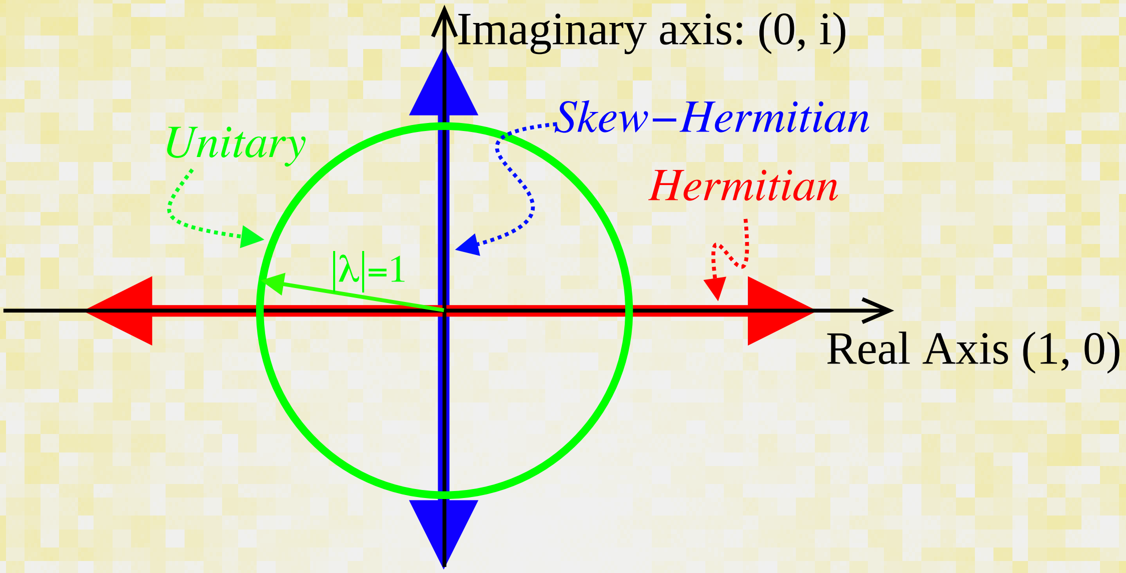


Figure 9-5: The Symmetric (complex Hermitic), Skew-Symmetric (complex Skew-Hermitian), Orthogonal, and Unitary Matrix sets characterized by the position of their eigenvalues in the complex plane. *(Hermits live alone on the real axis; SkewHermits live alone on the imaginary axis)*

Orthogonal Transformations

Multiplication of a vector by an orthogonal matrix is equivalent to an orthogonal geometric transformation on that vector.

For othogonal transformation, the inner product between any two vectors is *invariant*. That is, the inner product of two vectors is always the same as the inner product of their images under an orthogonal transformation. Geometrically, the projection (or the angular relationship) is unchanged.

This is characteristic of a rotation, or a reflection, or an inversion.

Rotations, reflections, and inversions are orthogonal transformations. The product of orthogonal matrices is also an orthogonal matrix.

Lecture 09 Mathematica® Example 2

Coordinate Transformations to The Eigenbasis

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Demonstrates that the matrix composed of columns constructed eigenvectors of a matrix can be used to *diagonalize a matrix* and the resulting diagonal entries are the matrix eigenvalues.

1: `simtrans` is constructed by assigning rows defined by the eigenvectors from Example 09-1 and then transposing so that the eigenvectors are the columns.

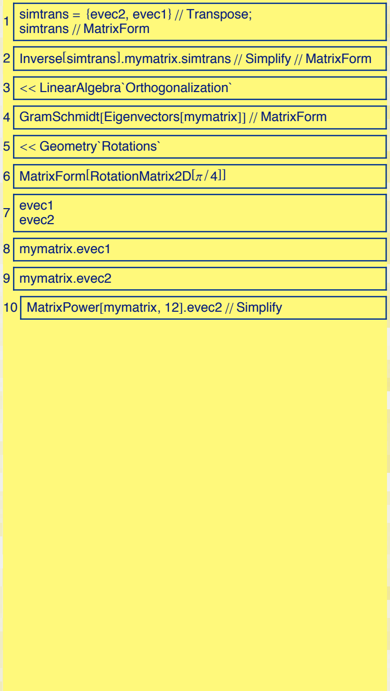
2: The original matrix is left-multiplied by the *inverse* of `simtrans` and right-multiplied by `simtrans`; the result will be a diagonal matrix with the original matrix's eigenvalues as diagonal entries.

2-4 The eigenvectors are orthogonal. There is a process called *Gram-Schmidt* orthogonalization used to define a set of vectors that are normal to each other. Here, `GramSchmidt` produces vectors that are also *normalized to unit vectors*. This, and other useful vector functions such as `Normalize` are avaiable in the `LinearAlgebra‘Orthogonalization` package.

5-6 The geometrical interpretation of the matrix that performs the diagonalization can be obtained by loading the `Geometry‘Rotations` package. The eigenvector-matrix can be compared to the $\pi/4$ rotation matrix.

8-9 Demonstrates that Eq. 9-2 is true.

10: Demonstrates that $\underline{A}^n\vec{x} = \lambda^n\vec{x}$.



```
1  simtrans = {evec2, evec1} // Transpose;
   simtrans // MatrixForm
2  Inverse[simtrans].mymatrix.simtrans // Simplify // MatrixForm
3  << LinearAlgebra`Orthogonalization`
4  GramSchmidt[Eigenvectors[mymatrix]] // MatrixForm
5  << Geometry`Rotations`
6  MatrixForm[RotationMatrix2D[π / 4]]
7  evec1
   evec2
8  mymatrix.evec1
9  mymatrix.evec2
10 MatrixPower[mymatrix, 12].evec2 // Simplify
```

3.016 Home

Full Screen

Close

Quit

Lecture 10: Real Eigenvalue Systems; Transformations to Eigenbasis

Reading:
Kreyszig Sections: 8.4, 8.5 (pages349–354, 356–361)

Similarity Transformations

A matrix has been discussed as a linear operation on vectors. The matrix itself is defined in terms of the coordinate system of the vectors that it operates on—and that of the vectors it returns.

In many applications, the coordinate system (or laboratory) frame of the vector that gets operated on is the same as the vector gets returned. This is the case for almost all physical properties. For example:

- In an electronical conductor, local current density, $\vec{J}$, is linearly related to the local electric field $\vec{E}$:
$$\underline{\rho}\vec{J} = \vec{E} \tag{10-1}$$
- In a thermal conductor, local heat current density is linearly related to the gradient in temperature:
$$\underline{k}\nabla T = \vec{j_Q} \tag{10-2}$$
- In diamagnetic and paramagnetic materials, the local magnetization, $\vec{B}$ is related to the applied field, $\vec{H}$:
$$\underline{\mu}\vec{H} = \vec{B} \tag{10-3}$$
- In dielectric materials, the local total polarization, $\vec{D}$, is related to the applied electric field:
$$\underline{\kappa}\vec{E} = \vec{D} = \kappa_o\vec{E} + \vec{P} \tag{10-4}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

When $\vec{x}$ and $\vec{y}$ are vectors representing a physical quantity in Cartesian space (such as force $\vec{F}$, electric field $\vec{E}$, orientation of a plane $\hat{n}$, current $\vec{j}$, etc.) they represent something *physical*. They don't change if we decide to use a different space in which to represent them (such as, exchanging x for y, y for z, z for x; or, if we decide to represent length in nanometers instead of inches, or if we simply decide to rotate the system that describes the vectors. The representation of the vectors themselves may change, but they stand for the same thing.

One interpretation of the operation $\underline{A}\vec{x}$ has been described as geometric transformation on the vector $\vec{x}$. For the case of orthogonal matrices $\underline{A_{orth}}$, geometrical transformations take the forms of rotation, reflection, and/or inversion.

Suppose we have some *physical* relation between two physical vectors in some coordinate system, for instance, the general form of Ohm's law is:

$$\begin{aligned}\vec{J} &= \underline{\sigma}\vec{E} \\ \begin{pmatrix} J_x \\ J_y \\ J_z \end{pmatrix} &= \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix} \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix}\end{aligned} \tag{10-5}$$

The matrix (actually it is better to call it a *rank-2 tensor*) $\underline{\sigma}$ is a physical quantity relating the amount of current that flows (in a direction) proportional to the applied electric field (perhaps in a different direction). $\underline{\sigma}$ is the "conductivity tensor" for a particular material.

The physical law in Eq. 10-5 can be expressed as an inverse relationship:

$$\begin{aligned}\vec{E} &= \underline{\rho}\vec{j} \\ \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} &= \begin{pmatrix} \rho_{xx} & \rho_{xy} & \rho_{xz} \\ \rho_{xy} & \rho_{yy} & \rho_{yz} \\ \rho_{xz} & \rho_{yz} & \rho_{zz} \end{pmatrix} \begin{pmatrix} j_x \\ j_y \\ j_z \end{pmatrix}\end{aligned} \tag{10-6}$$

where the resistivity tensor $\underline{\rho} = \underline{\sigma}^{-1}$.

What happens if we decide to use a new coordinate system (i.e., one that is rotated, reflected, or inverted) to describe the relationship expressed by Ohm's law?

The two vectors must transform from the "old" to the "new" coordinates by:

$$\begin{aligned} \underline{A_{orth}^{old\rightarrow new}}\vec{E^{old}} &= \vec{E^{new}} & \qquad \underline{A_{orth}^{old\rightarrow new}}\vec{j^{old}} &= \vec{j^{new}} \\ \underline{A_{orth}^{new\rightarrow old}}\vec{E^{new}} &= \vec{E^{old}} & \qquad \underline{A_{orth}^{new\rightarrow old}}\vec{j^{new}} &= \vec{j^{old}} \end{aligned} \tag{10-7}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Where is simple proof will show that:

$$\begin{aligned}\underline{A^{old\to new}_{orth}} &= \underline{A^{new\to old}_{orth}}^{-1}\\ \underline{A^{new\to old}_{orth}} &= \underline{A^{old\to new}_{orth}}^{-1}\\ \underline{A^{new\to old}_{orth}} &= \underline{A^{old\to new}_{orth}}^{T}\\ \underline{A^{new\to old}_{orth}} &= \underline{A^{old\to new}_{orth}}^{T}\end{aligned} \tag{10-8}$$

where the last two relations follow from the special properties of orthogonal matrices.

How does the physical law expressed by Eq. 10-5 change in a new coordinate system?

$$\begin{aligned}\text{in old coordinate system: } \vec{j}^{old} &= \underline{\chi^{old}}\vec{E}^{old}\\ \text{in new coordinate system: } \vec{j}^{new} &= \underline{\chi^{new}}\vec{E}^{new}\end{aligned} \tag{10-9}$$

To find the relationship between $\underline{\chi^{old}}$ and $\underline{\chi^{new}}$: For the first equation in 10-9, using the transformations in Eqs. 10-7:

$$\underline{A^{new\to old}_{orth}}\vec{j}^{new} = \underline{\chi^{old}}\,\underline{A^{new\to old}_{orth}}\vec{E}^{new} \tag{10-10}$$

and for the second equation in 10-9:

$$\underline{A^{old\to new}_{orth}}\vec{j}^{old} = \underline{\chi^{new}}\,\underline{A^{old\to new}_{orth}}\vec{E}^{old} \tag{10-11}$$

Left-multiplying by the inverse orthogonal transformations:

$$\begin{aligned}\underline{A^{old\to new}_{orth}}\,\underline{A^{new\to old}_{orth}}\vec{j}^{new} &= \underline{A^{old\to new}_{orth}}\,\underline{\chi^{old}}\,\underline{A^{new\to old}_{orth}}\vec{E}^{new}\\ \underline{A^{new\to old}_{orth}}\,\underline{A^{old\to new}_{orth}}\vec{j}^{old} &= \underline{A^{new\to old}_{orth}}\,\underline{\chi^{new}}\,\underline{A^{old\to new}_{orth}}\vec{E}^{old}\end{aligned} \tag{10-12}$$

Because the transformation matrices are inverses, the following relationship between *similar* matrices in the old and new coordinate systems is:

$$\begin{aligned}\underline{\chi^{old}} &= \underline{A^{old\to new}_{orth}}\,\underline{\chi^{new}}\,\underline{A^{new\to old}_{orth}}\\ \underline{\chi^{new}} &= \underline{A^{new\to old}_{orth}}\,\underline{\chi^{old}}\,\underline{A^{old\to new}_{orth}}\end{aligned} \tag{10-13}$$

The $\underline{\chi^{old}}$ is said to be *similar* to $\underline{\chi^{new}}$ and the double multiplication operation in Eq. 10-13 is called *a similarity transformation.*

Stresses and Strains

Stresses and strains are rank-2 tensors that characterize the mechanical state of a material.

A spring is an example of a one-dimensional material—it resists or exerts force in one direction only. A volume of material can exert forces in all three directions simultaneously—and the forces need not be the same in all directions. A volume of material can also be "squeezed" in many different ways: it can be squeezed along any one of the axis or it can be subjected to squeezing (*or smeared*) around any of the axes[3]

All the ways that a force can be applied to small element of material are now described. A force divided by an area is a *stress*—think of it the areal density of force.

$$\sigma_{ij} = \frac{F_i}{A_j} \qquad (\text{i.e., } \sigma_{xz} = \frac{F_x}{A_z} = \sigma_{xz} = \frac{\vec{F} \cdot \hat{\imath}}{\vec{A} \cdot \hat{k}}) \tag{10-14}$$

A_j is a plane with its normal in the $\hat{\jmath}$-direction (or the projection of the area of a plane $\vec{A}$ in the direction parallel to $\hat{\jmath}$)

[3] Consider a blob of modeling clay—you can deform it by placing between your thumbs and one opposed finger; you can deform it by simultaneously squeezing with two sets of opposable digits; you can "smear" it by pushing and pulling in opposite directions. These are examples of uniaxial, biaxial, and shear stress.

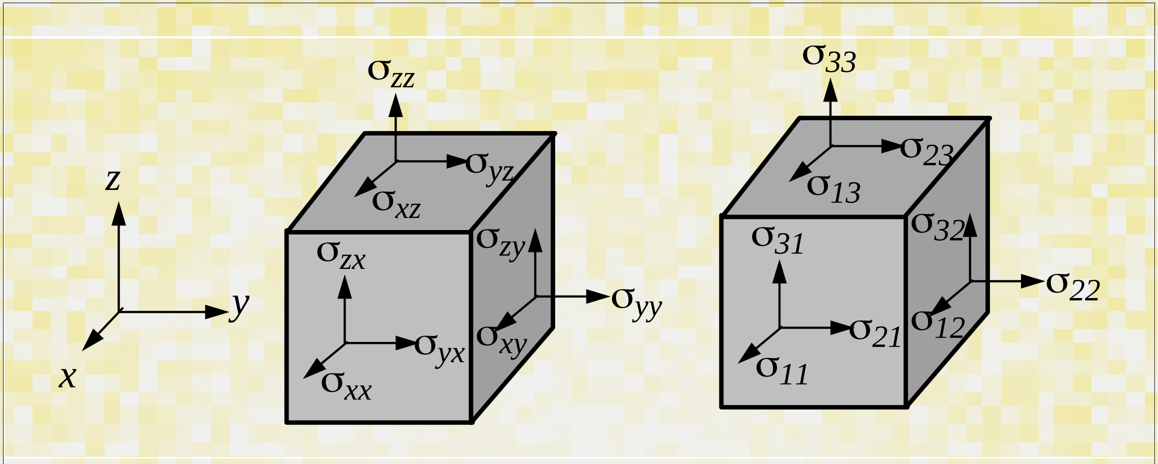


Figure 10-6: Illustration of stress on an oriented volume element.

$$\sigma_{ij} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{10-15}$$

There is one special and very simple case of elastic stress, and that is called the hydrostatic stress. It is the case of pure pressure and there are no shear (off-diagonal) stresses (i.e., all $\sigma_{ij} = 0$ for $i \neq j$, and $\sigma_{11} = \sigma_{22} = \sigma_{33}$). An equilibrium system composed of a body in a fluid environment is always in hydrostatic stress:

$$\sigma_{ij} = \begin{bmatrix} -P & 0 & 0 \\ 0 & -P & 0 \\ 0 & 0 & -P \end{bmatrix} \tag{10-16}$$

where the pure hydrostatic pressure is given by P.

Strain is also a rank-2 tensor and it is a physical measure of a how much a material changes its shape.[4]

Why should strain be a rank-2 tensor?

[4] It is unfortunate that the words of these two related physical quantities, stress and strain, sound so similar. Strain

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

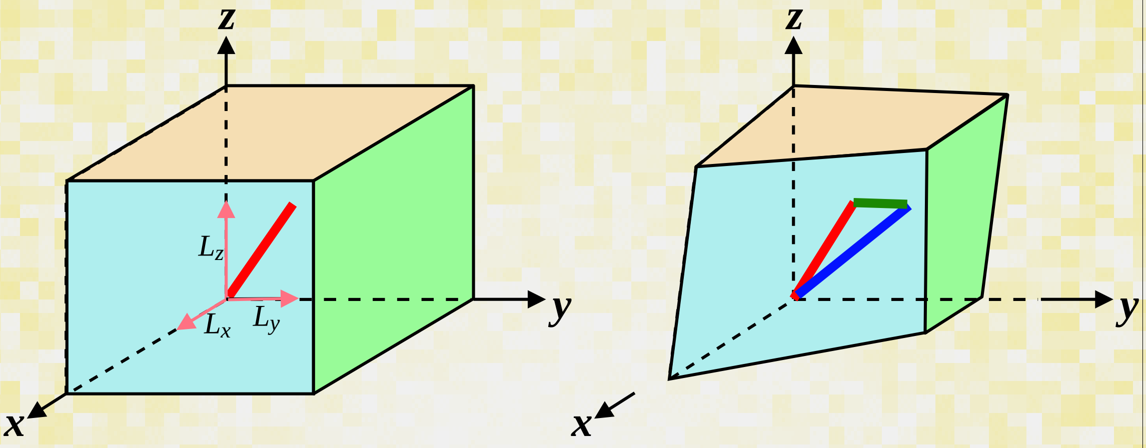


Figure 10-7: Illustration of how strain is defined: imagine a small line-segment that is aligned with a particular direction (one set of indices for the direction of the line-segment); after deformation the end-points of the line segment define a new line-segment in the deformed state. The difference in these two vectors is a vector representing how the line segment has changed from the initial state into the deformed state. The difference vector can be oriented in any direction (the second set of indices)—the strain is a representation of "a difference vectors for all the oriented line-segments" divided by the length of the original line.

Or, using the same idea as that for stress:

$$\epsilon_{ij} = \frac{\Delta L_i}{L_j} \qquad (\text{i.e., } \epsilon_{xz} = \frac{\Delta L_x}{L_z} = \epsilon_{xz} = \frac{\Delta\vec{L}\cdot\hat{i}}{\vec{L}\cdot\hat{k}}) \tag{10-17}$$

If a body that is being stressed hydro-statically is isotropic, then its response is pure dilation (in other words, it expands or shrinks uniformly and without shear):

measures the change in geometry of a body and stress measures the forces that squeeze or pull on a body. Stress is the press; Strain is the gain.

$$\epsilon_{ij} = \begin{bmatrix} \Delta/3 & 0 & 0 \\ 0 & \Delta/3 & 0 \\ 0 & 0 & \Delta/3 \end{bmatrix} \tag{10-18}$$

$$\Delta = \frac{dV}{V} \tag{10-19}$$

So, for the case of hydrostatic stress, the work term has a particularly simple form:

$$\begin{aligned} V\sum_{i=1}^{3}\sum_{j=1}^{3}\sigma_{ij}d\epsilon_{ij} &= -PdV \\ V\sigma_{ij}d\epsilon_{ij} &= -PdV \qquad \text{(summation convention)} \end{aligned} \tag{10-20}$$

This expression is the same as the rate of work performed on a compressible fluid, such as an ideal gas.

EigenStrains and EigenStresses

For any strain matrix, there is a choice of an coordinate system where line-segments that lie along the coordinate axes always deform parallel to themselves (i.e., they only stretch or shrink, they do not twist).

For any stress matrix, there is a choice of an coordinate system where all shear stresses (the off-diagonal terms) vanish and the matrix is diagonal.

These coordinate systems define the eigenstrain and eigenstress. The matrix transformation that takes a coordinate system into its eigenstate is of great interest because it simplifies the mathematical representation of the physical system.

Lecture 10 MATHEMATICA® Example 1

Principal Axes: Mohr's Circle of Stress

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Diagonalizing a quasi-two-dimensional stress tensor, the equations for Mohr's circle of stress (Fig. 10-8) are derived

1: The problem is done in reverse by finding the backwards rotation of a diagonal matrix. This is the stress in the principle coordinate system.

2: This is the rotation operator by θ about the z-axis.

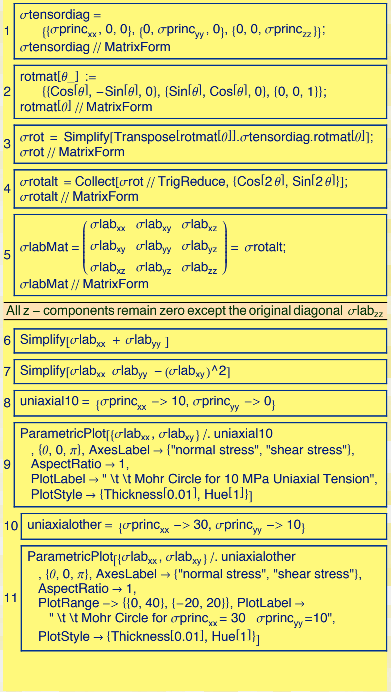
4: The rotation matrix factors well using the double angle formulas.

5: This can be compared with the general form of any quasi-two-dimensional (x–y plane) stress that has the same principle stresses identified above. In the rotated (i.e., laboratory) frame, the stresses are geometrically related to the circle plotted in 10-8.

6: The trace of the stress tensor is independent of rotation—this is a general property for any unitary transformation.

7: Like the trace, the determinant is also an *matrix invariant.*

11: Using `ParametricPlot` for $\sigma_{xx}(\theta)$ and $\sigma_{xy}(\theta)$, an example of Mohr's circle is plotted for principle stresss of 30 and 10.



```
1   σtensordiag =
        {{σprinc_xx, 0, 0}, {0, σprinc_yy, 0}, {0, 0, σprinc_zz}};
    σtensordiag // MatrixForm

2   rotmat[θ_] :=
        {{Cos[θ], -Sin[θ], 0}, {Sin[θ], Cos[θ], 0}, {0, 0, 1}};
    rotmat[θ] // MatrixForm

3   σrot = Simplify[Transpose[rotmat[θ]].σtensordiag.rotmat[θ]];
    σrot // MatrixForm

4   σrotalt = Collect[σrot // TrigReduce, {Cos[2 θ], Sin[2 θ]}];
    σrotalt // MatrixForm

5                ( σlab_xx  σlab_xy  σlab_xz )
    σlabMat =    ( σlab_xy  σlab_yy  σlab_yz ) = σrotalt;
                 ( σlab_xz  σlab_yz  σlab_zz )
    σlabMat // MatrixForm

All z - components remain zero except the original diagonal σlab_zz

6   Simplify[σlab_xx + σlab_yy ]

7   Simplify[σlab_xx σlab_yy - (σlab_xy)^2]

8   uniaxial10 = {σprinc_xx -> 10, σprinc_yy -> 0}

9   ParametricPlot[{σlab_xx, σlab_xy} /. uniaxial10
      , {θ, 0, π}, AxesLabel → {"normal stress", "shear stress"},
      AspectRatio → 1,
      PlotLabel → " \t \t Mohr Circle for 10 MPa Uniaxial Tension",
      PlotStyle → {Thickness[0.01], Hue[1]}]

10  uniaxialother = {σprinc_xx -> 30, σprinc_yy -> 10}

11  ParametricPlot[{σlab_xx, σlab_xy} /. uniaxialother
      , {θ, 0, π}, AxesLabel → {"normal stress", "shear stress"},
      AspectRatio → 1,
      PlotRange -> {{0, 40}, {-20, 20}}, PlotLabel →
        " \t \t Mohr Circle for σprinc_xx = 30  σprinc_yy =10",
      PlotStyle → {Thickness[0.01], Hue[1]}]
```

3.016

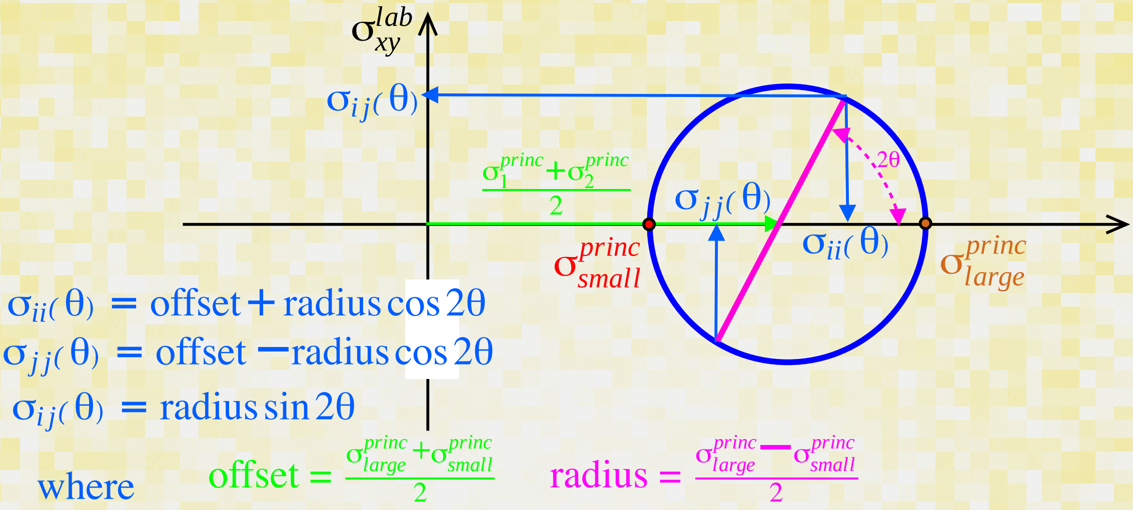
3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit



σ_{xy}^{lab}
$\sigma_{ij}(\theta)$
$\frac{\sigma_1^{princ}+\sigma_2^{princ}}{2}$
$\sigma_{jj}(\theta)$
2θ
$\sigma_{ii}(\theta)$
σ_{small}^{princ}
σ_{large}^{princ}
$\sigma_{ii}(\theta) = \text{offset} + \text{radius} \cos 2\theta$
$\sigma_{jj}(\theta) = \text{offset} - \text{radius} \cos 2\theta$
$\sigma_{ij}(\theta) = \text{radius} \sin 2\theta$
where $\text{offset} = \frac{\sigma_{large}^{princ}+\sigma_{small}^{princ}}{2}$ $\text{radius} = \frac{\sigma_{large}^{princ}-\sigma_{small}^{princ}}{2}$

Figure 10-8: Mohr's circle of stress is a way of graphically representing the two-dimensional stresses of identical stress states, but in rotated laboratory frames.
The center of the circle is displaced from the origin by a distance equal to the average of the principal stresses (or average of the eigenvalues of the stress tensor).
The maximum and minimum stresses are the eigenvalues—and they define the diameter in the principal $\theta = 0$ frame.
Any other point on the circle gives the stress tensor in a frame rotated by 2θ from the principal axis using the construction illustrated by the blue lines (and equations).

Quadratic Forms

The example above, where a matrix (rank-2 tensor) represents a material property, can be understood with a useful geometrical interpretation.

For the case of the conductivity tensor $\underline{\sigma}$, the dot product $\vec{E} \cdot \vec{j}$ is a scalar related to the local energy dissipation:

$$e = \vec{E}^T \underline{\sigma} \vec{E} \tag{10-21}$$

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

The term on the right-hand-side is called a quadratic form, as it can be written as:

$$\begin{aligned} e =&\sigma_{11}x_1^2 + \sigma_{12}x_1x_2 + \sigma_{13}x_1x_3 + \\ &\sigma_{21}x_1x_2 + \sigma_{22}x_2^2 + \sigma_{23}x_2x_3 + \\ &\sigma_{31}x_1x_3 + \sigma_{32}x_2x_3 + \sigma_{33}x_3^2 \end{aligned} \tag{10-22}$$

or, because $\underline{\sigma}$ is symmetric:

$$\begin{aligned} e =&\sigma_{11}x_1^2 + 2\sigma_{12}x_1x_2 + 2\sigma_{13}x_1x_3 + \\ &\sigma_{22}x_2^2 + 2\sigma_{23}x_2x_3 + \\ &\sigma_{33}x_3^2 \end{aligned} \tag{10-23}$$

It is not unusual for such quadratic forms to represent energy quantities. For the case of paramagnetic and diamagnetic materials with magnetic permeability tensor $\underline{\mu}$, the energy per unit volume due to an applied magnetic field $\vec{H}$ is:

$$\frac{E}{V} = \frac{1}{2}\vec{H}^T\underline{\mu}\vec{H} \tag{10-24}$$

for a dielectric (i.e., polarizable) material with electric electric permittivity tensor $\underline{\kappa}$ with an applied electric field $\vec{E}$:

$$\frac{E}{V} = \frac{1}{2}\vec{E}^T\underline{\kappa}\vec{E} \tag{10-25}$$

The geometric interpretation of the quadratic forms is obtained by turning the above equations around and asking—what are the general vectors $\vec{x}$ for which the quadratic form (usually an energy or power density) has a particular value? Picking that particular value as unity, the question becomes what are the directions and magnitudes of $\vec{x}$ for which

$$1 = \vec{x}^T\underline{A}\vec{x} \tag{10-26}$$

This equation expresses a quadratic relationship between one component of $\vec{x}$ and the others. This is a surface—known as the *quadric surface* or *representation quadric*—which is an ellipsoid or hyperboloid sheet on which the quadratic form takes on the particular value 1.

In the principal axes (or, equivalently, the eigenbasis) the quadratic form takes the quadratic form takes the simple form:

$$e = \vec{x_{\mathrm{eb}}}^T \underline{A_{\mathrm{eb}}} \vec{x_{\mathrm{eb}}} = A_{11}x_1^2 + A_{22}x_2^2 + A_{33}x_3^2 \qquad (10\text{-}27)$$

and the representation quadric

$$A_{11}x_1^2 + A_{22}x_2^2 + A_{33}x_3^2 = 1 \qquad (10\text{-}28)$$

which is easily characterized by the signs of the coefficients.

In other words, in the principal axis system (or the eigenbasis) the quadratic form has a particularly simple, in fact the most simple, form.

Eigenvector Basis

Among all similar matrices (defined by the similarity transformation defined by Eq. 10-13), the simplest matrix is the diagonal one. In the coordinate system where the similar matrix is diagonal, its diagonal entries are the eigenvalues. The question remains, "what is the coordinate transformation that takes the matrix into its diagonal form?"

The coordinate system is called the eigenbasis or principal axis system, and the transformation that takes it there is particularly simple.

The matrix that transforms from a general (old) coordinate system to a diagonalized matrix (in the new coordinate system) is the matrix of columns of the eigenvectors. The first column corresponds to the first eigenvalue on the diagonal matrix, and the n^{th} column is the eigenvector corresponding the n^{th} eigenvalue.

$$\begin{pmatrix} \text{The} \\ \text{Diagonalized} \\ \text{Matrix} \end{pmatrix} = \begin{pmatrix} \text{Eigenvector} \\ \text{Column} \\ \text{Matrix} \end{pmatrix}^{-1} \begin{pmatrix} \text{The} \\ \text{General} \\ \text{Matrix} \end{pmatrix} \begin{pmatrix} \text{Eigenvector} \\ \text{Column} \\ \text{Matrix} \end{pmatrix} \qquad (10\text{-}29)$$

This method provides a method for finding the simplest quadratic form.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 11: Geometry and Calculus of Vectors

Reading:
Kreyszig Sections: 9.1, 9.2, 9.3, 9.4 (pages364–369, 371–374, 377–383, 384–388)

Graphical Animation: Using Time as a Dimension in Visualization

Animations can be very effective tools to illustrate time-dependent phenomena in scientific presentations. Animations are sequences of multiple images—called frames—that are written to the screen interatively at a constant rate: if one second of real time is represented by N frames, then a real-time animation would display a new image every $1/N$ seconds.

There are two important practical considerations for computer animation:

frame size An image is a an array of pixels, each of which is represented as a color. The amount of memory each color requires depends on the current image depth, but this number is typically 2-5 bytes. Typical video frames contain 1024×768 pixel images which corresponds to about 2.5 MBytes/image and shown at 30 frames per second corresponding to about 4.5 GBytes/minute. Storage and editing of video is probably done at higher spatial and temporal resolution. Each frame must be read from a source—such as a hard disk—and transfered to the graphical memory (VRAM) before the screen can be redrawn with a new image. Therefore, along with storage space the rate of memory transfer becomes a practical issue when constructing an animation.

animation rate Humans are fairly good at extrapolating action between sequential images. It depends on the difference between sequential images, but animation rates below about 10 frames per second begin to appear jerky. Older Disney-type cartoons were typically displayed at about 15 frames per second, video is displayed at 30 frames per second. Animation rates above about 75 frames per second yield no addition perceptable "smoothness." The upper bound on computer displays is typically 60 hertz.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 11 Mathematica® Example 1

3.016

Introductory Animation Examples

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Several introductory examples of animated results from equation-generated graphics are presented as model starting points. In Mathematica® , the goal is produce a list of `Graphics`-objects and then display them all. The animation can be played by grouping all the individual displayed graphics objects into one super-group (i.e., a super-bracket); closing the brackets and then using the `Cell`-menu to animated the closed-and-selected group.

Two simple methods to produce animations are illustrated: 1) using `Table` with a graphics-producing argument; 2) generating a list of undisplayed graphics objects, and then displaying them using a loop structure.

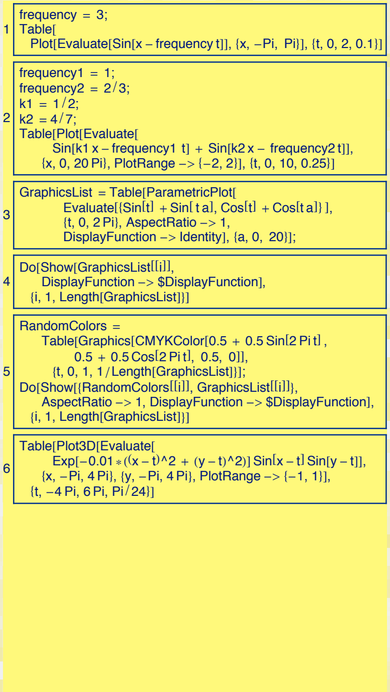
1: A traveling wave is produced by iteratively plotting $\sin(kx - \omega t)$ a discrete times using `Table` together with `Plot`. `Evaluate` must be wrapped around the function, otherwise the time-variable will not be computed automatically.

2: Beats are illustrated. However, unless the `PlotRange` of each image is the same, the resulting animation would be shoddy.

3: In this case, the animation is computed first and then displayed later. The `Graphics`-objects are stored in a list using `Table` and `ParametricPlot`, but graphical display is suppressed by using `DisplayFunction→Identity`. Any list-generating function, such as `AppendTo`, can be used to produce such *Graphics-lists*.

4: Such Graphics-lists can be combined with other computed graphics objects and shown together. In this example, a list of colors is produced with `Graphics` and `CMYKColor` is computed and then used to colorize the previously computed graphics list. `Do` is used with `Show` to produce the graphics; in this case, the previously-suppressed display is forced with `DisplayFunction→ $DisplayFunction`.

5: An animation of a sequence of 3D graphics objects is produced similarly; note that `PlotRange` must be used to ensure that the z-axis remains constant between frames.



```
1
frequency = 3;
Table[
  Plot[Evaluate[Sin[x – frequency t]], {x, –Pi, Pi}], {t, 0, 2, 0.1}]
```

```
2
frequency1 = 1;
frequency2 = 2/3;
k1 = 1/2;
k2 = 4/7;
Table[Plot[Evaluate[
      Sin[k1 x – frequency1 t] + Sin[k2 x – frequency2 t]],
    {x, 0, 20 Pi}, PlotRange –> {–2, 2}], {t, 0, 10, 0.25}]
```

```
3
GraphicsList = Table[ParametricPlot[
          Evaluate[{Sin[t] + Sin[t a], Cos[t] + Cos[t a]} ],
          {t, 0, 2 Pi}, AspectRatio –> 1,
          DisplayFunction –> Identity], {a, 0, 20}];
```

```
4
Do[Show[GraphicsList[[i]],
    DisplayFunction –> $DisplayFunction],
  {i, 1, Length[GraphicsList]}]
```

```
5
RandomColors =
    Table[Graphics[CMYKColor[0.5 + 0.5 Sin[2 Pi t] ,
          0.5 + 0.5 Cos[2 Pi t], 0.5, 0]],
       {t, 0, 1, 1/Length[GraphicsList]}];
Do[Show[{RandomColors[[i]], GraphicsList[[i]]},
    AspectRatio –> 1, DisplayFunction –> $DisplayFunction],
  {i, 1, Length[GraphicsList]}]
```

```
6
Table[Plot3D[Evaluate[
       Exp[–0.01 * ((x – t)^2 + (y – t)^2)] Sin[x – t] Sin[y – t]],
    {x, –Pi, 4 Pi}, {y, –Pi, 4 Pi}, PlotRange –> {–1, 1}],
  {t, –4 Pi, 6 Pi, Pi/24}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Vector Products

The concept of vectors as abstract objects representing a collection of data has already been presented. Every student at this point has already encountered vectors as representation of points, forces, and accelerations in two and three dimensions.

Review: The Inner (dot) product of two vectors and relation to projection

An inner (or dot-) project is multiplication of two vectors that produces a scalar.

$$
\begin{aligned}
\vec{a} \cdot \vec{b} \equiv & \\
&\equiv a_i b_i \\
&\equiv a_i b_j \delta_{ij} \text{ where } \delta_{ij} \equiv \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases} \\
&\equiv (a_1, a_2, \ldots a_N) \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix} \\
&\equiv (b_1, b_2, \ldots b_N) \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}
\end{aligned} \tag{11-1}
$$

The inner product is:

linear, distributive $(k_1\vec{a} + k_2\vec{b}) \cdot \vec{c} = k_1\vec{a} \cdot \vec{c} + k_2\vec{b} \cdot \vec{c}$

symmetric $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$

satifies Schwarz inequality $\|\vec{a} \cdot \vec{b}\| \leq \|\vec{b}\|\|\vec{a}\|$

satifies triangle inequality $\|\vec{a} + \vec{b}\| \leq \|\vec{b}\| + \|\vec{a}\|$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

If the vector components are in a cartesian (i.e., cubic lattice) space, then there is a useful equation for the angle between two vectors:

$$\cos\alpha = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \hat{n_a}\cdot\hat{n_b} \tag{11-2}$$

where $\hat{n_i}$ is the unit vector that shares a direction with i. *Caution: when working with vectors in non-cubic crystal lattices (e.g, tetragonal, hexagonal, etc.) the angle relationship above does not hold. One must convert to a cubic system first to calculate the angles.*

The projection of a vector onto a direction $\hat{n_b}$ is a scalar:

$$p = \vec{a}\cdot\hat{n_b} \tag{11-3}$$

Review: Vector (or cross-) products

The vector product (or cross ×) differs from the dot (or inner) product in that multiplication produces a vector from two vectors. One might have quite a few choices about how to define such a product, but the following idea proves to be useful (and standard).

normal Which way should the product vector point? Because two vectors (usually) define a plane, the product vector might as well point away from it.

The exception is when the two vectors are linearly dependent; in this case the product vector will have zero magnitude.

The product vector is normal to the plane defined by the two vectors that make up the product. A plane has two normals, which normal should be picked? By convention, the "right-hand-rule" defines which of the two normal should be picked.

magnitude Given that the product vector points away from the two vectors that make up the product, what should be its magnitude? We already have a rule that gives us the cosine of the angle between two vectors, a rule that gives the sine of the angle between the two vectors would be useful. Therefore, the cross product is defined so that its magnitude for two unit vectors is the sine of the angle between them.

This has the extra utility that the cross product is zero when two vectors are linearly-dependent (i.e., they do not define a plane).

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

This also has the utility, discussed below, that the triple product will be a scalar quantity equal to the volume of the parallelepiped defined by three vectors.

The triple product,

$$\begin{aligned}\vec{a} \cdot (\vec{b} \times \vec{c}) = (\vec{a} \times \vec{b}) \cdot \vec{c} =\\ \|\vec{a}\|\|\vec{b}\|\|\vec{c}\| \sin \gamma_{b-c} \cos \gamma_{a-bc} =\\ \|\vec{a}\|\|\vec{b}\|\|\vec{c}\| \sin \gamma_{a-b} \cos \gamma_{ab-c}\end{aligned} \tag{11-4}$$

where γ_{i-j} is the angle between two vectors i and j and γ_{ij-k} is the angle between the vector k and plane spanned by i and j. is equal to the parallelepiped that has $\vec{a}$, $\vec{b}$, and $\vec{c}$, emanating from its bottom-back corner.

If the triple product is zero, the volume between three vectors is zero and therefore they must be linearly dependent.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 11 Mathematica® Example 2

Cross Product Example

notebook (non-evaluated) pdf (evaluated) html (evaluated)

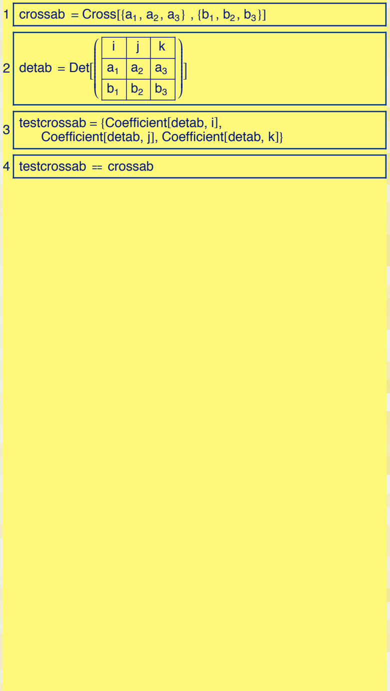
This is a simple demonstration of the vector product of two spatial vectors and comparison to the the memorization device:

$$\vec{a} \times \vec{b} = \det \begin{pmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{pmatrix}$$

1: `Cross` produces the vector product of two symbolic vectors $\vec{a}$ and $\vec{b}$ of length 3.

2: `Det` produces the same result using the memorization device.

3: `Coefficient` is used to extract each vector component.



```
1  crossab = Cross[{a1, a2, a3}, {b1, b2, b3}]

                    ⎛ i    j    k  ⎞
2  detab = Det[⎜ a1   a2   a3 ⎟]
                    ⎝ b1   b2   b3 ⎠

3  testcrossab = {Coefficient[detab, i],
        Coefficient[detab, j], Coefficient[detab, k]}

4  testcrossab == crossab
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Derivatives Vectors

Consider a vector, $\vec{p}$, as a point in space. If that vector is a function of a real continuous parameter for instance, t, then $\vec{p}(t)$ represents the loci as a function of a parameter.

If $\vec{p}(t)$ is continuous, then it sweeps out a continuous curve as t changes continuously. It is very natural to think of t as time and $\vec{p}(t)$ as the trajectory of a particle—such a trajectory would be continuous if the particle does not disappear at one instant, t, and reappear an instant later, $t + dt$, some finite distance distance away from $\vec{p}(t)$.

If $\vec{p}(t)$ is continuous, then the limit:

$$\frac{d\vec{p}(t)}{dt} = \lim_{\Delta t \to 0} \frac{\vec{p}(t + \Delta t) - \vec{p}(t)}{\Delta t} \tag{11-5}$$

Notice that the numerator inside the limit is a vector and the denominator is a scalar; so, the derivative is also a vector. Think about the equation geometrically—it should be apparent that the vector represented by the derivative is locally tangent to the curve that is traced out by the points $\vec{p}(t - dt)$, $\vec{p}(t)$ $\vec{p}(t + dt)$, etc.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 11 MATHEMATICA® Example 3

Visualizing Time-Dependent Vectors and their Derivatives

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of $\vec{x}(t)$ and $d\vec{x}/dt$ are illustrated as curves and as animations.

1: A list of three time-dependent components for (x, y, z) is constructed and. . .

2: Displayed with `ParametricPlot3D`.

3: This is a second example of a curve, `DeltoidSpiral`, for which the derivative will be calculated.

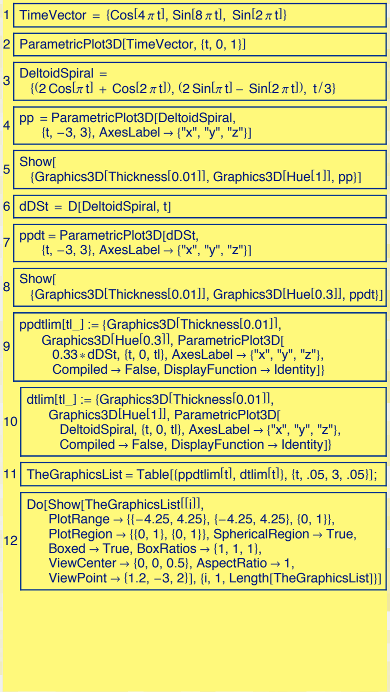
6: The derivative operator `D` is a *threadable function* so it will operate on each component of its vector argument.

7: This will display the curve that is tangent to `DeltoidSpiral` at each time t. Because the z-component is linear in t, the resulting tangent curve has a constant value of z.

9: This function will produce a graphical object which is the image superposition of *all* the results of `DeltoidSpiral` for $t = 0$ up to some specified value $t = t_{lim}$—i.e., a visible curve.

11: A graphics list of the development of the curve and its derivative is constructed and. . .

12: subsequently animated with a `Do`.



```
1   TimeVector = {Cos[4 π t], Sin[8 π t], Sin[2 π t]}

2   ParametricPlot3D[TimeVector, {t, 0, 1}]

3   DeltoidSpiral =
      {(2 Cos[π t] + Cos[2 π t]), (2 Sin[π t] − Sin[2 π t]), t/3}

4   pp = ParametricPlot3D[DeltoidSpiral,
        {t, −3, 3}, AxesLabel → {"x", "y", "z"}]

5   Show[
      {Graphics3D[Thickness[0.01]], Graphics3D[Hue[1]], pp}]

6   dDSt = D[DeltoidSpiral, t]

7   ppdt = ParametricPlot3D[dDSt,
        {t, −3, 3}, AxesLabel → {"x", "y", "z"}]

8   Show[
      {Graphics3D[Thickness[0.01]], Graphics3D[Hue[0.3]], ppdt}]

9   ppdtlim[tl_] := {Graphics3D[Thickness[0.01]],
        Graphics3D[Hue[0.3]], ParametricPlot3D[
          0.33 ∗ dDSt, {t, 0, tl}, AxesLabel → {"x", "y", "z"},
          Compiled → False, DisplayFunction → Identity]}

10  dtlim[tl_] := {Graphics3D[Thickness[0.01]],
        Graphics3D[Hue[1]], ParametricPlot3D[
          DeltoidSpiral, {t, 0, tl}, AxesLabel → {"x", "y", "z"},
          Compiled → False, DisplayFunction → Identity]}

11  TheGraphicsList = Table[{ppdtlim[t], dtlim[t]}, {t, .05, 3, .05}];

12  Do[Show[TheGraphicsList[[i]],
        PlotRange → {{−4.25, 4.25}, {−4.25, 4.25}, {0, 1}},
        PlotRegion → {{0, 1}, {0, 1}}, SphericalRegion → True,
        Boxed → True, BoxRatios → {1, 1, 1},
        ViewCenter → {0, 0, 0.5}, AspectRatio → 1,
        ViewPoint → {1.2, −3, 2}], {i, 1, Length[TheGraphicsList]}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Review: Partial and total derivatives

One might also consider a time- and space-dependent vector field, for instance $\vec{E}(x, y, z, t) = \vec{E}(\vec{x}, t)$ could be the force on a unit charge located at $\vec{x}$ at time t.

Here, there are many different things which might be varied and give rise to a derivative. Such questions might be:

1. How does the force on a unit charge differ for two nearby unit-charge particles, say at (x, y, z) and at $(x, y + \Delta y, z)$?
2. How does the force on a unit charge located at (x, y, z) vary with time?
3. How does the the force on a particle change as the particle traverses some path $(x(t), y(t), z(t))$ in space?

Each question has the "flavor" of a derivative, but each is asking a different question. So a different kind of derivative should exist for each type of question.

The first two questions are of the nature, "How does a quantity change if only one of its variables changes and the others are held fixed?" The kind of derivative that applies is the partial derivative.

The last question is of the nature, "How does a quantity change when all of its variables depend on a single variable?" The kind of derivative that applies is the total derivative. The answers are:

1.

$$\frac{\partial \vec{E}(x, y, z, t)}{\partial y} = \left(\frac{\partial \vec{E}}{\partial y}\right)_{\text{constant}x,z,t} \tag{11-6}$$

2.

$$\frac{\partial \vec{E}(x, y, z, t)}{\partial t} = \left(\frac{\partial \vec{E}}{\partial t}\right)_{\text{constant}x,y,z} \tag{11-7}$$

3.

$$\frac{d\vec{E}(x(t), y(t), z(t), t)}{dt} = \frac{\partial \vec{E}}{\partial x}\frac{dx}{dt} + \frac{\partial \vec{E}}{\partial y}\frac{dy}{dt} + \frac{\partial \vec{E}}{\partial z}\frac{dz}{dt} + \frac{\partial \vec{E}}{\partial t}\frac{dt}{dt} = \nabla \vec{E}(\vec{x}(t), t) \cdot \frac{d\vec{x}}{dt} + \frac{\partial \vec{E}}{\partial t} \tag{11-8}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Time-Dependent Scalar and Vector Fields

A physical quantity that is spatially variable is often called a *spatial field*. It is a particular case of a field quantity.

Such fields can be simple scalars, such as the altitude as a function of east and west in a topographical map. Vectors can also be field quantities, such as the direction uphill and steepness on a topographical map— this is an example of how each scalar field is naturally associated with to its *gradient field*. Higher dimensional objects, such as stress and strain, can also be field quantities.

Fields that evolve in time are *time-dependent fields* and appear frequently in physical models. Because time-dependent 3D spatial fields are four-dimensional objects, animation is frequently used to visulize them.

For a working example, consider the time-evolution of "ink concentration" $c(x, y, t)$ of a very small spot of ink spilled on absorbant paper at $x = y = 0$ at time $t = 0$. This example could be modeled with Fick's first law:

$$\vec{J} = -D\nabla c(x, y, t) = -D\left(\frac{\partial c}{\partial x} + \frac{\partial c}{\partial y}\right) \tag{11-9}$$

where D is the diffusivity that determines "how fast" the ink moves for a given gradient ∇c, and $\vec{J}$ is a time-dependent vector that represents "rate of ink flow past a unit-length line segment oriented perpendicular to $\vec{J}$. which leads to the two-dimensional diffusion equation

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right) \tag{11-10}$$

For this example, the solution, $c(x, y, t)$ is given by

$$c(x, y, t) = \frac{c_o}{4\pi Dt} e^{-\frac{x^2+y^2}{4Dt}} \tag{11-11}$$

where c_o is the initial concentration of ink.

Lecture 11 MATHEMATICA® Example 4

Visualizing a Solution to the Diffusion Equation

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The solution to 2D diffusion equation for point source initial conditions and its resulting flux, $\vec{J}$ is illustrated with several types of animation.

1: The diffusivity it set to one, this effectively sets the length and time scales for the subsequent simulation.

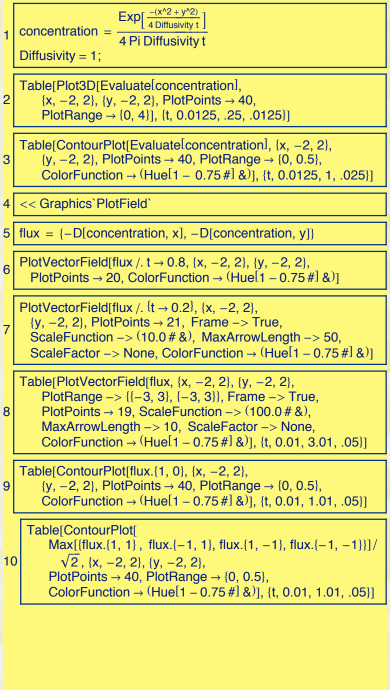
2: The concentration plotted with `Plot3D` as a two-dimensional surface embedded in three-dimensions and animated as a function of time.

3: A simpler graphical representation is obtained with `ContourPlot` by plotting contours of constant concentration. The resulting animation is of a two-dimensional object.

4: To plot the flux which is vector field, the package `Graphics‘PlotField‘` is loaded for its `PlotVectorField` function.

8: An animation of `PlotVectorField` for the flux can be obtained. However, getting the sequential frames to be consistent and the size of the arrows representing the vectors requires a somewhat complicated use of `PlotVectorField`'s options such `ScaleFunction`. This may serve as a working first example for beginning users.

9: Animations of vector fields can be simplified, by converting them into a set of scalar representations. In this example, the dot-product exracts the J_x-component only which is animated with contourplots.



```
                      Exp[ -(x^2 + y^2) / (4 Diffusivity t) ]
1  concentration  =  -----------------------------------------
                             4 Pi Diffusivity t
   Diffusivity = 1;

2  Table[Plot3D[Evaluate[concentration],
       {x, −2, 2}, {y, −2, 2}, PlotPoints → 40,
       PlotRange → {0, 4}], {t, 0.0125, .25, .0125}]

3  Table[ContourPlot[Evaluate[concentration], {x, −2, 2},
       {y, −2, 2}, PlotPoints → 40, PlotRange → {0, 0.5},
       ColorFunction → (Hue[1 − 0.75 #] &)], {t, 0.0125, 1, .025}]

4  << Graphics`PlotField`

5  flux  =  {−D[concentration, x], −D[concentration, y]}

6  PlotVectorField[flux /. t → 0.8, {x, −2, 2}, {y, −2, 2},
     PlotPoints → 20, ColorFunction → (Hue[1 − 0.75 #] &)]

7  PlotVectorField[flux /. {t → 0.2}, {x, −2, 2},
     {y, −2, 2}, PlotPoints → 21,  Frame −> True,
     ScaleFunction −> (10.0 # &),  MaxArrowLength −> 50,
     ScaleFactor −> None, ColorFunction → (Hue[1 − 0.75 #] &)]

8  Table[PlotVectorField[flux, {x, −2, 2}, {y, −2, 2},
       PlotRange −> {{−3, 3}, {−3, 3}}, Frame −> True,
       PlotPoints → 19, ScaleFunction −> (100.0 # &),
       MaxArrowLength −> 10,  ScaleFactor −> None,
       ColorFunction → (Hue[1 − 0.75 #] &)], {t, 0.01, 3.01, .05}]

9  Table[ContourPlot[flux.{1, 0}, {x, −2, 2},
       {y, −2, 2}, PlotPoints → 40, PlotRange → {0, 0.5},
       ColorFunction → (Hue[1 − 0.75 #] &)], {t, 0.01, 1.01, .05}]

10  Table[ContourPlot[
        Max[{flux.{1, 1} ,  flux.{−1, 1}, flux.{1, −1}, flux.{−1, −1}}] /
          √2 , {x, −2, 2}, {y, −2, 2},
        PlotPoints → 40, PlotRange → {0, 0.5},
        ColorFunction → (Hue[1 − 0.75 #] &)], {t, 0.01, 1.01, .05}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

All vectors are not spatial

It is useful to think of vectors as spatial objects when learning about them—but one shouldn't get stuck with the idea that all vectors are points in two- or three-dimensional space. The spatial vectors serve as a good analogy to generalize an idea.

For example, consider the following chemical reaction:

Reaction:	H_2	$\frac{1}{2}O_2$	$\rightleftharpoons$	H_2O
Initial:	1	1	$\rightleftharpoons$	0
During Rx.:	$1-\xi$	$1-\frac{1}{2}\xi$	$\rightleftharpoons$	ξ

The composition could be written as a vector:

$$\vec{N} = \begin{pmatrix} \text{moles } H_2 \\ \text{moles } O_2 \\ \text{moles } H_2O \end{pmatrix} = \begin{pmatrix} 1-\xi \\ 1-\frac{1}{2}\xi \\ \xi \end{pmatrix} \tag{11-12}$$

and the variable ξ plays the role of the "extent" of the reaction—so the composition variable $\vec{N}$ lives in a reaction-extent (ξ) space of chemical species.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 12: Multivariable Calculus

Reading:
Kreyszig Sections: 9.5, 9.6, 9.7 (pages389–398, 400–403, 403–409)

The Calculus of Curves

In the last lecture, the derivatives of a vector that varied continuously with a parameter, $\vec{r}(t)$, were considered. It is natural to think of $\vec{r}(t)$ as a curve in whatever space the vector $\vec{r}$ is defined. The most familiar example is a curve in the plane: the two values $(x(t), y(t))$ are mapped onto the plane through values as t sweeps through its range $t_{\text{initial}} \leq t \leq t_{\text{final}}$. A curve in three-dimensional cartesian space is the mapping of three values $(x(t), y(t), z(t))$; in cylindrical coordinates: $(r(t), \theta(t), z(t))$. In general, a curve is represented by N coordinates as a *single parameter* (i.e., t) takes on a range of numbers—the N coordinates form the embedding space.

Objects that have more dimensions than curves need more parameters. The number of parameters is the dimensionality of the object and the number of coordinates is the dimensionality of the embedding space. What we naturally call a *surface* is a two dimensional object embedded in a three-dimensional space—for example, in cartesian coordinates $(x(u, v), y(u, v), z(u, v))$ is a surface.

The two-dimensional surface $(x(u, v), y(u, v), z(u, v))$ can itself become an embedding space for lower dimensional objects; for example, the curve $(u(t), v(t))$ is embedded in the surface (u, v) which itself embedded in (x, y, z). In other words, the curve $(x(u(t), v(t)), y(u(t), v(t)), z(u(t), v(t)))$ can be considered to be embedded in (u, v), or embedded in (x, y, z) and constrained to the surface $(x(u, v), y(u, v), z(u, v))$.

In higher dimensions, there are many more possibilities and we can make a few introductory remarks about the language that is used to describe them. For application to physical problems, these considerations indicate the number of degrees-of-freedom that are available and the conditions that a system is overconstrained. An N-dimensional surface (sometimes called a *hyper-surface*) embedded in an M-dimensional space is said to have *codimension* $M - N$. Some objects cannot be embedded in

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

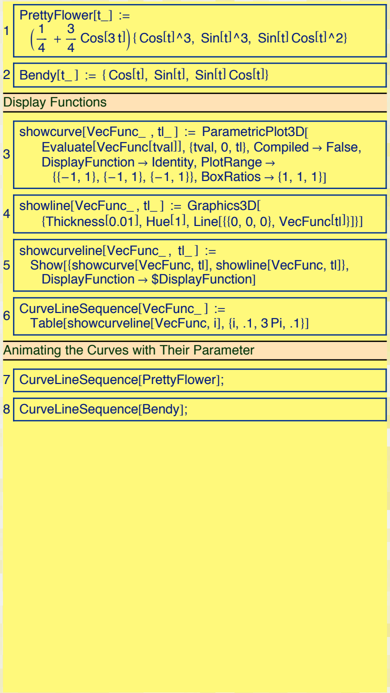
a higher dimensional space; these are called *non-embeddable*, examples include the Klein bottle which cannot be embedded in our three-dimensional space.

Lecture 12 Mathematica® Example 1

Curves in Three Dimensions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Two examples of parametric curves are presented with a visualization technique that animates the vector as it sweeps out a curve.



```
1  PrettyFlower[t_] :=
     (1/4 + 3/4 Cos[3 t]) {Cos[t]^3, Sin[t]^3, Sin[t] Cos[t]^2}
2  Bendy[t_] := {Cos[t], Sin[t], Sin[t] Cos[t]}
Display Functions
3  showcurve[VecFunc_ , tl_ ] := ParametricPlot3D[
       Evaluate[VecFunc[tval]], {tval, 0, tl}, Compiled → False,
       DisplayFunction → Identity, PlotRange →
         {{−1, 1}, {−1, 1}, {−1, 1}}, BoxRatios → {1, 1, 1}]
4  showline[VecFunc_ , tl_ ] := Graphics3D[
       {Thickness[0.01], Hue[1], Line[{{0, 0, 0}, VecFunc[tl]}]}]
5  showcurveline[VecFunc_ , tl_ ] :=
     Show[{showcurve[VecFunc, tl], showline[VecFunc, tl]},
       DisplayFunction → $DisplayFunction]
6  CurveLineSequence[VecFunc_ ] :=
     Table[showcurveline[VecFunc, i], {i, .1, 3 Pi, .1}]
Animating the Curves with Their Parameter
7  CurveLineSequence[PrettyFlower];
8  CurveLineSequence[Bendy];
```

2: This is the second of two example functions that take an argument and return a vector at the argument. It is a vector function that takes a scalar argument.

3: The function *showcurve* takes two arguments. The first argument is the name of a vector function that takes a single argument, as in the above examples. The second argument is the upper limit of time to be plotted; i.e., it will sweep out a curve from $t = 0$ to $t =$upper-limit. The option `DisplayFunction` is a replacement to `Identity`, so the graphics will not be displayed.

4: The function *showline* takes the same two arguments, and it creates a graphics-object for a line drawn from the origin to the point indicated by the second argument.

5: The function *showcurveline* takes the same two arguments and calls the previously defined functions *showcurve* and *showline* with the graphics displayed by using `DisplayFunction` → `$DisplayFunction`. The result should be a line pointing to a curve that is swept out from the time $t = 0$.

6: *CurveLineSequence* just calls *showcurveline* for a fixed interval.

8: This is the second example of an animation showing the vector as it sweeps out a curve.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 12 MATHEMATICA® Example 2

Embedding Curves in Surfaces

notebook (non-evaluated) pdf (evaluated) html (evaluated)

An example is constructed that visualizes a two-dimensional surface in three dimensions and then visualizes a one-dimensional curve constrained to the surface.

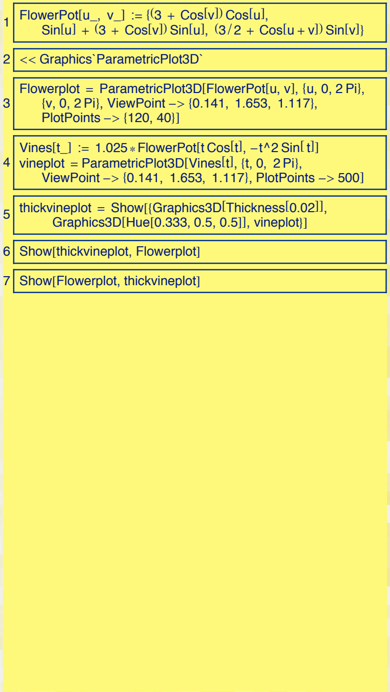
1: *FlowerPot* takes two arguments and returns a vector. As the arguments sweep through domains, the vector will trace out a surface.

3: Using the `ParametricPlot3D` that is in the `Graphics‘ParametricPlot‘` package, the surface defined by `FlowerPlot` can be visualized.

4: *Vines* takes a single argument and then calls `FlowerPot` with arguments that are functions of that single argument—the result must be a curve embedded in the surface. In this case, the function is scaled a little, so the curves will be visible.

5: This is an example that makes the curve fat and green colored.

6: Here, both the embedded curve and the surface are shown together.



```
1 FlowerPot[u_, v_] := {(3 + Cos[v]) Cos[u],
     Sin[u] + (3 + Cos[v]) Sin[u], (3/2 + Cos[u + v]) Sin[v]}

2 << Graphics`ParametricPlot3D`

3 Flowerplot = ParametricPlot3D[FlowerPot[u, v], {u, 0, 2 Pi},
     {v, 0, 2 Pi}, ViewPoint -> {0.141, 1.653, 1.117},
     PlotPoints -> {120, 40}]

4 Vines[t_] := 1.025 * FlowerPot[t Cos[t], -t^2 Sin[ t]]
  vineplot = ParametricPlot3D[Vines[t], {t, 0, 2 Pi},
     ViewPoint -> {0.141, 1.653, 1.117}, PlotPoints -> 500]

5 thickvineplot = Show[{Graphics3D[Thickness[0.02]],
       Graphics3D[Hue[0.333, 0.5, 0.5]], vineplot}]

6 Show[thickvineplot, Flowerplot]

7 Show[Flowerplot, thickvineplot]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Because the derivative of a curve with respect to its parameter is a tangent vector, the unit tangent can be defined immediately:

$$\hat{u} = \frac{\frac{d\vec{r}}{dt}}{\|\frac{d\vec{r}}{dt}\|} \tag{12-1}$$

It is convenient to find a new parameter, $s(t)$, that would make the denominator in Eq. 12-1 equal to one. This parameter, $s(t)$, is the arc-length:

$$\begin{aligned} s(t) &= \int_{t_o}^{t} ds \\ &= \int_{t_o}^{t} \sqrt{dx^2 + dy^2 + dz^2} \\ &= \int_{t_o}^{t} \sqrt{(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2 + (\frac{dz}{dt})^2 dt} \\ &= \int_{t_o}^{t} \sqrt{(\frac{d\vec{r}}{dt}) \cdot (\frac{d\vec{r}}{dt})} dt \end{aligned} \tag{12-2}$$

and with s instead of t,

$$\hat{u}(s) = \frac{d\vec{r}}{ds} \tag{12-3}$$

This is natural because $\|\vec{r}\|$ and s have the same units (i.e., meters and meters, foots and feet, etc) instead of, for instance, time, t, that makes $d\vec{r}/dt$ a velocity and involving two different kinds of units (e.g., furlongs and hours).

With the arc-length s, the magnitude of the curvature is particularly simple,

$$\kappa(s) = \|\frac{d\hat{u}}{ds}\| = \|\frac{d^2\vec{r}}{ds^2}\| \tag{12-4}$$

as is its interpretation—the curvature is a measure of how rapidly the unit tangent is changing direction.

Furthermore, the rate at which the unit tangent changes direction is a vector that must be normal to the tangent (because $d(\hat{u} \cdot \hat{u} = 1) = 0$) and therefore the unit normal is defined by:

$$\hat{p}(s) = \frac{1}{\kappa(s)} \frac{d\hat{u}}{ds} \tag{12-5}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

There two unit vectors that are locally normal to the unit tangent vector $\hat{u'}(s)$ and the curve unit normal $\hat{p}(s) \times \hat{u}$ and $\hat{u}(s) \times \hat{p}$. This last choice is called the unit binormal, $\hat{b} \equiv \hat{u}(s) \times \hat{p}$ and the three vectors together form a nice little moving orthogonal axis pinned to the curve. Furthermore, there are convenient relations between the vectors and differential geometric quantities called the Frenet equations.

Using Arc-Length as a Curve's Parameter

However, it should be pointed out that—although re-parameterizing a curve in terms of its arc-length makes for simple analysis of a curve—achieving this re-parameterization is not necessarily simple.

Consider the steps required to represent a curve $\vec{r}(t)$ in terms of its arc-length:

integration The integral in Eq. 12-2 may or may not have a closed form for $s(t)$.

If it does then we can perform the following operation:

inversion $s(t)$ is typically a complicated function that is not easy to invert, i.e., solve for t in terms of s to get a $t(s)$ that can be substituted into $\vec{r}(t(s)) = \vec{r}(s)$.

These difficulties usually result in treating the problem symbolically and the resorting to numerical methods of achieving the integration and inversion steps.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

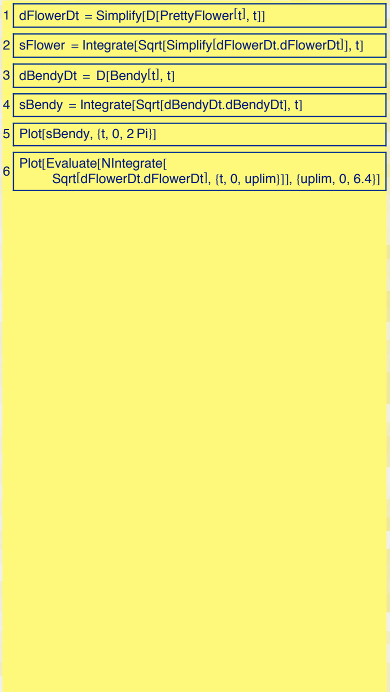
Quit

Lecture 12 MATHEMATICA® Example 3

Calculating arclength

notebook (non-evaluated)   pdf (evaluated)   html (evaluated)

Examples of computing a curve's arc-length s



```
1 dFlowerDt = Simplify[D[PrettyFlower[t], t]]
2 sFlower = Integrate[Sqrt[Simplify[dFlowerDt.dFlowerDt]], t]
3 dBendyDt = D[Bendy[t], t]
4 sBendy = Integrate[Sqrt[dBendyDt.dBendyDt], t]
5 Plot[sBendy, {t, 0, 2 Pi}]
6 Plot[Evaluate[NIntegrate[
        Sqrt[dFlowerDt.dFlowerDt], {t, 0, uplim}]], {uplim, 0, 6.4}]
```

1: Here, the tangent vector for the function, *PrettyFlower* defined above, is computed.

2: This is an attempt to find a closed-form solution for arclength $s(\tau)-s(0) = \int_0^\tau \sqrt{\left(\frac{dc}{dt}\right)^2} dt$. A closed-form doesn't exist.

4: However, a closed-form solution does exist for the *Bendy* -function defined earlier. If the closed-form $s(t)$ could be inverted (i.e., $t(s)$) then the curve $c(t)$ could be expressed in terms of its natural variable $c(s) = c(t(s))$.

5: The plot, $s(t)$ is monotonically increasing and therefore, the function could always be inverted numerically.

6: Even for the arc-length that could not be evaluated in closed-form (i.e., *PrettyFlower*), a numerical integration could be used to perform the inversion.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Scalar Functions with Vector Argument

In materials science and engineering, the concept of a spatially varying function arises frequently:
For example:

Concentration $c_i(x, y, z) = c_i(\vec{x})$ is the number (or moles) of chemical species of type i *per unit volume* located at the point $\vec{x}$.

Density $\rho(x, y, z) = \rho(\vec{x})$ mass *per unit volume* located at the point $\vec{x}$. a point $\rho(x, y, z) = \rho(\vec{x})$.

Energy Density $u(x, y, z) = u(\vec{x})$ energy *per unit volume* located at the point $\vec{x}$.

The examples above are spatially dependent densities of “extensive quantities.”
There are also spatially variable intensive quantities:

Temperature $T(x, y, z) = T(\vec{x})$ is the temperature which would be measured at the point $\vec{x}$.

Pressure $P(x, y, z) = P(\vec{x})$ is the pressure which would be measured at the point $\vec{x}$.

Chemical Potential $\mu_i(x, y, z) = \mu_i(\vec{x})$ is the chemical potential of the species i which would be measured at the point $\vec{x}$.

Each example is a scalar function of space—that is, the function associates a scalar with each point in space.

A topographical map is a familiar example of a graphical illustration of a scalar function (altitude) as a function of location (latitude and longitude).

How Confusion Can Develop in Thermodynamics

However, there are many other types of scalar functions of several arguments, such as the state function: $U = U(S, V, N_i)$ or $P = P(V, T, N_i)$. It is sometimes useful to think of these types of functions a scalar functions of a “point” in a thermodynamics space.

However, this is often a source of confusion: notice that the internal energy is used in two different contexts above. One context is the value of the energy, say 128.2 Joules. The other context is the function $U(S, V, N_i)$. While the two symbols are identical, their meanings are quite different.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The root of the confusion lurks in the question, "What are the variables of U?" Suppose that there is only one (independent) chemical species, then $U(\cdot)$ has three variables, such as $U(S, V, N)$. "But what if $S(T, P, \mu)$, $V(T, P, \mu)$, and $N(T, P, \mu)$ are known functions, what are the variables of U?" The answer is, they are any three independent variables, one could write $U(T, P, \mu) = U(S(T, P, \mu), V(T, P, \mu), N(T, P, \mu))$ and there are six other natural choices.

The trouble arises when variations of a function like U are queried—then the variables that are varying must be specified.

For this reason, it is either a good idea to keep the functional form explicit in thermodynamics, i.e.,

$$\begin{aligned} dU(S,V,N) &= \frac{\partial U(S,V,N)}{\partial S} dS + \frac{\partial U(S,V,N)}{\partial V} dV + \frac{\partial U(S,V,N)}{\partial N} dN \\ dU(T,P,\mu) &= \frac{\partial U(T,P,\mu)}{\partial T} dT + \frac{\partial U(T,P,\mu)}{\partial V} dV + \frac{\partial U(T,P,\mu)}{\partial \mu} d\mu \end{aligned} \tag{12-6}$$

or use, the common thermodynamic notation,

$$\begin{aligned} dU &= \left(\frac{\partial U}{\partial S}\right)_{V,N} dS + \left(\frac{\partial U}{\partial V}\right)_{S,N} dV + \left(\frac{\partial U}{\partial N}\right)_{S,V} dN \\ dU &= \left(\frac{\partial U}{\partial T}\right)_{P,\mu} dT + \left(\frac{\partial U}{\partial P}\right)_{T,\mu} dP + \left(\frac{\partial U}{\partial \mu}\right)_{T,P} d\mu \end{aligned} \tag{12-7}$$

Total and Partial Derivatives, Chain Rule

There is no doubt that a great deal confusion arises from the following question, "What are the variables of my function?"

For example, suppose we have a three-dimensional space (x, y, z), in which there is an embedded surface $(x(w, v), y(w, v), z(w, v))$ $\vec{x}(w, v) = \vec{x}(\vec{u})$ where $\vec{u} = (v, w)$ is a vector that lies in the surface, and an embedded curve $(x(s), y(s), z(s)) = \vec{x}(s)$. Furthermore, suppose there is a curve that lies within the surface $(w(t), v(t)) = \vec{u}(t)$.

Suppose that $\mathcal{E} = f(x, y, z)$ is a scalar function of (x, y, z).

Here are some questions that arise in different applications:

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

1. How does $\mathcal{E}$ vary as a function of position?

2. How does $\mathcal{E}$ vary along the surface?

3. How does $\mathcal{E}$ vary along the curve?

4. How does $\mathcal{E}$ vary along the curve embedded in the surface?

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 12 Mathematica® Example 4

Total Derivatives and Partial Derivatives: A Mathematica Review

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Demonstrations of 1) the three spatial derivatives of $F(x,y,z)$; 2) the two independent derivatives on a two-dimensional surface embedded in x–y–z; 3) the complete derivative of $F(x,y,z)$ along a curve $(x(t),y(t),z(t))$.

1: *AScalarFunction* is a symbolic representation of a function—it will be a place-holder for examples of partial derivatives.

3: This will print Mathematica's representation of derivatives with respect to one of several arguments—e.g., $\partial F(x,y,z)/\partial y$ is written as $\mathtt{F}^{(0,1,0)}$[x,y,z].

4: `AScalarFunction` becomes a function of two-variables when x, y, and z are restricted to a surface parameterized by (u,v): $(x(w,v),y(w,v),z(w,v))$

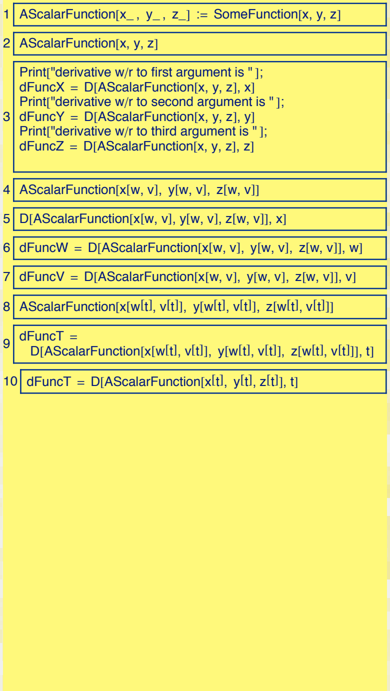
5: Caution: the distinction between the symbol `x` and the symbol `x[w,v]` is important; the following two examples show how the derivatives should appear.

7: This and the previous example show how the chain rule is computed, these two terms are the components of the gradient in the surface.

8: In this case, the previous `AScalarFunction` becomes a function of a single variable by specifying a curve in the surface with $(w(t),v(t))$.

9: Now, a total derivative can be calculated with the chain rule. This is equivalent to...

10: The total derivative along a specific curve $(x(t),y(t),z(t))$.



```
1  AScalarFunction[x_ , y_ , z_] := SomeFunction[x, y, z]

2  AScalarFunction[x, y, z]

   Print["derivative w/r to first argument is " ];
   dFuncX = D[AScalarFunction[x, y, z], x]
   Print["derivative w/r to second argument is " ];
3  dFuncY = D[AScalarFunction[x, y, z], y]
   Print["derivative w/r to third argument is " ];
   dFuncZ = D[AScalarFunction[x, y, z], z]

4  AScalarFunction[x[w, v], y[w, v], z[w, v]]

5  D[AScalarFunction[x[w, v], y[w, v], z[w, v]], x]

6  dFuncW = D[AScalarFunction[x[w, v], y[w, v], z[w, v]], w]

7  dFuncV = D[AScalarFunction[x[w, v], y[w, v], z[w, v]], v]

8  AScalarFunction[x[w[t], v[t]], y[w[t], v[t]], z[w[t], v[t]]]

   dFuncT =
9    D[AScalarFunction[x[w[t], v[t]], y[w[t], v[t]], z[w[t], v[t]]], t]

10 dFuncT = D[AScalarFunction[x[t], y[t], z[t]], t]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Taylor Series

The behavior of a function near a point is something that arises frequently in physical models. When the function has lower-order continuous partial derivatives (generally, a “smooth” function near the point in question), the stock method to model local behavior is Taylor’s series expansions around a fixed point.

Taylor’s expansion for a scalar function of n variables, $f(x_1, x_2, \ldots, x_n)$ which has continuous first and second partial derivatives near the point $\vec{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$ is:

$$
\begin{aligned}
&f(\xi_1, \xi_2, \ldots, \xi_n) = f(x_1, x_2, \ldots, x_n) \\
&+ \left.\frac{\partial f}{\partial x_1}\right|_{\vec{\xi}} (\xi_1 - x_1) + \left.\frac{\partial f}{\partial x_2}\right|_{\vec{\xi}} (\xi_2 - x_2) + \ldots + \left.\frac{\partial f}{\partial x_n}\right|_{\vec{\xi}} (\xi_n - x_n) \\
+\frac{1}{2}\,[\\
&\left.\frac{\partial^2 f}{\partial {x_1}^2}\right|_{\vec{\xi}} (\xi_1 - x_1)^2) + \left.\frac{\partial^2 f}{\partial x_1 \partial x_2}\right|_{\vec{\xi}} (\xi_1 - x_1)(\xi_2 - x_2) + \ldots + \left.\frac{\partial^2 f}{\partial x_1 \partial x_n}\right|_{\vec{\xi}} (\xi_1 - x_1)(\xi_n - x_n) \\
&+ \left.\frac{\partial^2 f}{\partial x_2 \partial x_1}\right|_{\vec{\xi}} (\xi_2 - x_2)(\xi_1 - x_1) + \left.\frac{\partial^2 f}{\partial {x_2}^2}\right|_{\vec{\xi}} (\xi_2 - x_2)^2 + \ldots + \left.\frac{\partial^2 f}{\partial x_2 \partial x_n}\right|_{\vec{\xi}} (\xi_2 - x_2)(\xi_n - x_n) \qquad (12\text{-}8) \\
&\vdots \qquad\qquad\qquad \ldots \qquad\qquad\qquad \vdots \\
&+ \left.\frac{\partial^2 f}{\partial x_n \partial x_1}\right|_{\vec{\xi}} (\xi_n - x_n)(\xi_1 - x_1) + \left.\frac{\partial^2 f}{\partial x_n \partial x_2}\right|_{\vec{\xi}} (\xi_n - x_n)(\xi_2 - x_2) + \ldots + \left.\frac{\partial^2 f}{\partial {x_n}^2}\right|_{\vec{\xi}} (\xi_n - x_n)^2 \\
&] \\
&+ \mathcal{O}\left[(\xi_1 - x_1)^3\right] + \mathcal{O}\left[(\xi_1 - x_1)^2(\xi_2 - x_2)\right] + \mathcal{O}\left[(\xi_1 - x_1)(\xi_2 - x_2)^2\right] + \mathcal{O}\left[(\xi_2 - x_2)^3\right] \\
&+ \ldots + \mathcal{O}\left[(\xi_1 - x_1)^2(\xi_n - x_n)\right] + \mathcal{O}\left[(\xi_1 - x_1)(\xi_2 - x_2)(\xi_n - x_n)\right] + \ldots + \mathcal{O}\left[(\xi_n - x_n)^3\right]
\end{aligned}
$$

or in a vector shorthand:

$$
f(\vec{x}) = f(\vec{\xi}) + \nabla_{\vec{x}} f|_{\vec{\xi}} \cdot (\vec{\xi} - \vec{x}) + (\vec{\xi} - \vec{x}) \cdot (\nabla_{\vec{x}} \nabla_{\vec{x}} f)|_{\vec{\xi}} \cdot (\vec{xi} - \vec{x}) + \mathcal{O}\left[\|\vec{\xi} - \vec{x}\|^3\right] \qquad (12\text{-}9)
$$

In the following example, visualization of local approximations will be obtained for a scalar function of two variables, $f(x, y)$. This will be extended into a an approximating function of four variables by

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

expanding it about a point (ξ, η) to second order. The expansion is now a function of four variables—the first two variables are the point the function is expanded around (x and y), and the second two are the variable of the parabolic approximation at that point (ξ and η): $f_{\text{appx}}(\xi, \eta; x, y) = f(x, y) + \left.\frac{\partial f}{\partial x}\right|_{x,y} (\xi - x) + \left.\frac{\partial f}{\partial y}\right|_{x,y} (\eta - y) + \mathcal{Q}$ where $\mathcal{Q} \equiv \frac{1}{2} \left.\frac{\partial^2 f}{\partial x^2}\right|_{x,y} (\xi - x)(\xi - x) + \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{x,y} (\xi - x)(\eta - y) + \frac{1}{2} \left.\frac{\partial^2 f}{\partial y^2}\right|_{x,y} (\eta - y)(\eta - y)$ or $f_{\text{appx}}(\xi, \eta, x, y) = f(x, y) + \nabla f \cdot \begin{pmatrix} \xi - x \\ \eta - y \end{pmatrix} + \frac{1}{2} \mathcal{Q}_{\text{form}}$ where $\mathcal{Q}_{\text{form}} \equiv (\xi - x, \eta - y) \begin{pmatrix} \left.\frac{\partial^2 f}{\partial x^2}\right|_{x,y} & \left.\frac{\partial^2 f}{\partial x \partial y}\right|_{x,y} \\ \left.\frac{\partial^2 f}{\partial y \partial x}\right|_{x,y} & \left.\frac{\partial^2 f}{\partial y^2}\right|_{x,y} \end{pmatrix} \begin{pmatrix} \xi - x \\ \eta - y \end{pmatrix}$

The function $f_{\text{appx}}(\xi, \eta, x, y)$ will be plotted as a function of ξ and η for $|\xi - x| < \delta$ and $|\eta - y| < \delta$ for a selected number of points (x, y).

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 12 MATHEMATICA® Example 5

Approximating Surfaces at Points

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Visualization of a quadratic approximations to a surface at points on that surface

1: *CrazyFun* is an example function of two variables.

3: Using `Normal` to convert the Taylor Expansion obtained by `Series` at an point x_o, y_o produces a function *Approxfunction* of four variables.

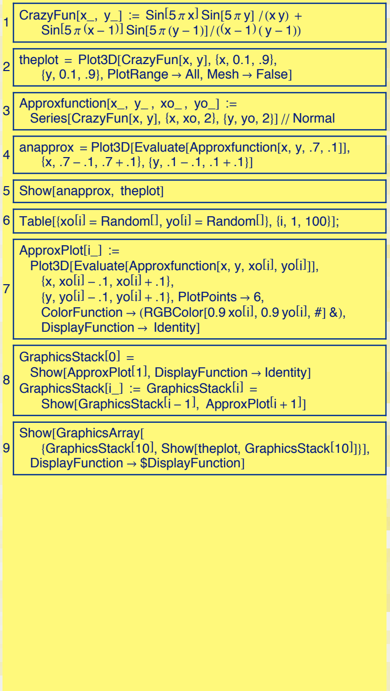
5: This illustrates how the local quadratic approximation fits the surface locally at a *particular* point.

6: Generate a list of random points at which to visualize the local approximation.

7: *ApproxPlot* is an example that will plot the local approximation for any indexed random point. The surface is colored by using the value of the point as an indicator. Visualization is delayed: only a graphics object is produced.

8: This is an example of producing a stack of graphics with a *recursive graphics function.* It iteravely adds a new approximating surface graphics object to the set of the previous ones.

9: `GraphicsArray` allows plots to be drawn in rows and columns. Here, intermediate output is produced and displayed, and then the approximating surfaces are plotted on the left of the surfaces with the original surface.



```
1  CrazyFun[x_, y_] := Sin[5 π x] Sin[5 π y] / (x y) +
       Sin[5 π (x − 1)] Sin[5 π (y − 1)] / ((x − 1) ( y − 1))

2  theplot = Plot3D[CrazyFun[x, y], {x, 0.1, .9},
       {y, 0.1, .9}, PlotRange → All, Mesh → False]

3  Approxfunction[x_, y_ , xo_ , yo_] :=
     Series[CrazyFun[x, y], {x, xo, 2}, {y, yo, 2}] // Normal

4  anapprox = Plot3D[Evaluate[Approxfunction[x, y, .7, .1]],
       {x, .7 − .1, .7 + .1}, {y, .1 − .1, .1 + .1}]

5  Show[anapprox, theplot]

6  Table[{xo[i] = Random[], yo[i] = Random[]}, {i, 1, 100}];

7  ApproxPlot[i_] :=
     Plot3D[Evaluate[Approxfunction[x, y, xo[i], yo[i]]],
       {x, xo[i] − .1, xo[i] + .1},
       {y, yo[i] − .1, yo[i] + .1}, PlotPoints → 6,
       ColorFunction → (RGBColor[0.9 xo[i], 0.9 yo[i], #] &),
       DisplayFunction → Identity]

8  GraphicsStack[0] =
     Show[ApproxPlot[1], DisplayFunction → Identity]
   GraphicsStack[i_] := GraphicsStack[i] =
       Show[GraphicsStack[i − 1], ApproxPlot[i + 1]]

9  Show[GraphicsArray[
       {GraphicsStack[10], Show[theplot, GraphicsStack[10]]}],
     DisplayFunction → $DisplayFunction]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Just a few of many examples of instances where Taylor's expansions are used are:

linearization Examining the behavior of a model near a point where the model is understood. Even if the model is wildly non-linear, a useful approximation is to make it linear by evaluating near a fixed point.

approximation If a model has a complicated representation in terms of unfamiliar functions, a Taylor expansion can be used to characterize the 'local' model in terms of a simple polynomials.

asymptotics Even when a system has singular behavior (e.g, the value of a function becomes infinite as some variable approaches a particular value), *how* the system becomes singular is very important. At singular points, the Taylor expansion will have leading order terms that are singular, for example near $x = 0$,

$$\frac{\sin(x)}{x^2} = \frac{1}{x} - \frac{x}{6} + \mathcal{O}(x^3) \tag{12-10}$$

The singularity can be subtracted off and it can be said that this function approaches ∞ "linearly" from below with slope -1/6. Comparing this to the behavior of another function that is singular near zero:

$$\frac{\exp(x)}{x} = \frac{1}{x} + 1 + \frac{x}{2} + \frac{x^2}{6} + \mathcal{O}(x^3) \tag{12-11}$$

shows that the e^x/x behavior is "more singular."

3.016 Home

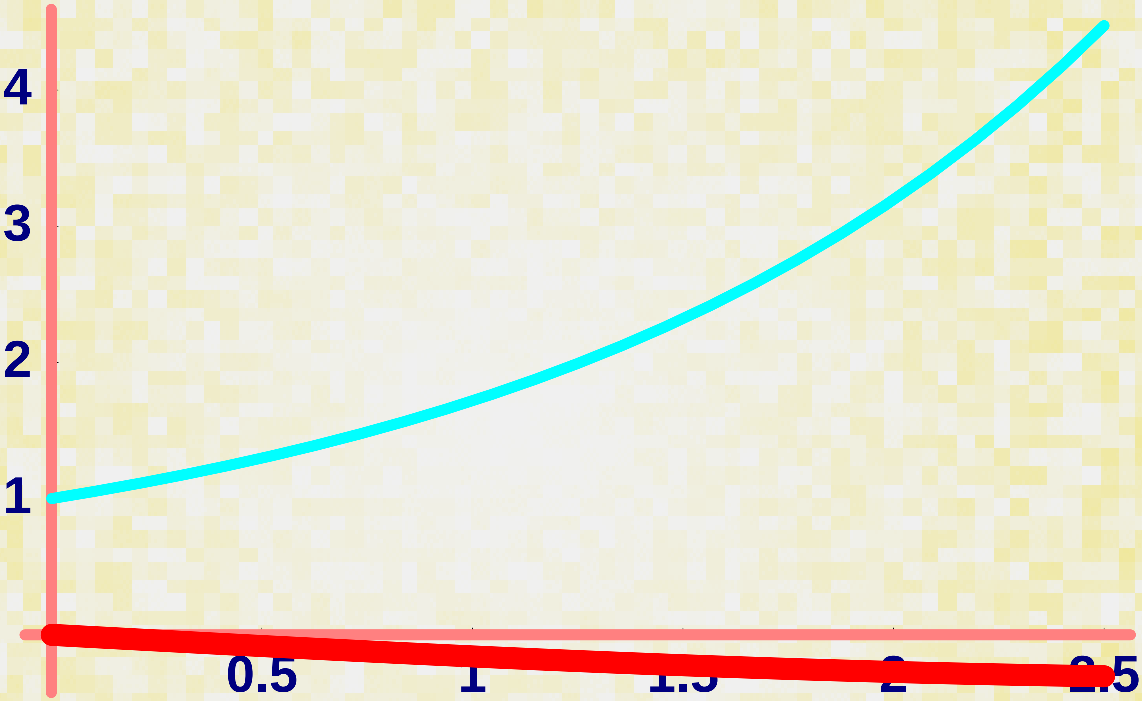
◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

```
Plot[{Sin[x]/x^2 - 1/x, Exp[x]/x - 1/x}, {x, .001, 2.5},
  PlotStyle→{{Thickness[0.02], Hue[1]}, {Thickness[0.01], Hue[0.5]}}]
```



```
- Graphics -
```

Figure 12-9: Behavior of two singular functions near their singular points.

stability In the expansion of energy about a point is obtained, then the various orders of expansion can be interpreted:

zero-order The zeroth-order term in a local expansion is the energy of the system at the point

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

evaluated. Unless this term is to be compared to another point, it has no particular meaning (if it is not infinite) as energy is always arbitrarily defined up to a constant.

first-order The first-order is related to the driving force to change the state of the system. Consider:

$$\Delta E = \nabla E \cdot \delta\vec{x} = -\vec{F} \cdot \delta\vec{x} \tag{12-12}$$

If force exists, the system can decrease it energy linearly by picking a particular change $\delta\vec{x}$ that is anti-parallel to the force.

For a system to be stable, it **is a necessary first condition that the forces (or first order expansion coefficients) vanish**.

second order If a system has no forces on it (therefore satisfying the necessary condition of stability), then where the system is stable or unstable depends on whether a small $\delta\vec{x}$ can be found that deceases the energy:

$$\begin{aligned}\Delta E &= \frac{1}{2}\delta\vec{x}\cdot\nabla\overrightarrow{-F}\cdot\delta\vec{x}\\ &= \frac{1}{2}\cdot\nabla\nabla E\cdot\delta\vec{x}\\ &= \frac{1}{2}\left.\frac{\partial^2 E}{\partial x_i \partial x_j}\right|_{x_1,x_2,\ldots x_n}\delta x_i \delta x_j\end{aligned} \tag{12-13}$$

where the summation convention is used above and the point $(x_1, x_2, \ldots, x_n)$ is one for which ∇E is zero.

numerics Derivatives are often obtained numerically in numerical simulations. The Taylor expansion provides a formula to approximate numerical derivatives—and provides an estimate of the numerical error as a function of quantities like numerical mesh size.

Gradients and Directional Derivatives

Scalar functions $F(x, y, z) = F(\vec{x})$ have a natural vector field associated with them—at each point $\vec{x}$ there is a direction $\hat{n}(\vec{x})$ pointing in the direction of the most rapid increase of F. Associating the

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

magnitude of a vector in the direction of steepest increase with the *rate of increase* of F defines the gradient.

The gradient is therefore a vector function with a vector argument ($\vec{x}$ in this case) and it is commonly written as ∇F.

Here are some natural examples:

Meteorology The “high pressure regions” are commonly displayed with weather reports—as are the ”isobars” or curves of constant barometric pressure. Thus displayed, pressure is a scalar function of latitude and longitude.

At any point on the map, there is a direction that points to local high pressure center—this is the direction of the gradient. The rate at which the pressure is increasing gives the magnitude of the gradient.

The gradient of pressure should be a vector that is related to the direction and the speed of wind.

Mosquitoes It is known that hungry mosquitoes tend to fly towards sources (or local maxima) of carbon dioxide. Therefore, it can be hypothesized that mosquitoes are able to determine the gradient in the concentration of carbon dioxide.

Heat In an isolated system, heat flows from high-temperature ($T(\vec{x})$) regions to low-temperature regions.

The Fourier empirical law of heat flow states that the rate of heat flows is proportional to the local *decrease* in temperature.

Therefore, the local rate of heat flow should be a proportional to the vector which is *minus* the gradient of $T(\vec{x})$: $-\nabla T$

Finding the Gradient

Potentials and Force Fields

Force is a vector. Force projected onto a displacement vector $\vec{dx}$ is the rate at which work, dW, is done *on* an object $dW = -\vec{F} \cdot \vec{dx}$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

If the work is being supplied by an external agent (e.g., a charged sphere, a black hole, a magnet, etc.), then it may be possible to ascribe a potential energy ($E(\vec{x})$, a scalar function with vector argument) to the agent associated with the position at which the force is being applied.[5] This $E(\vec{x})$ is the potential for the agent and the force field due to the agent is $\vec{F}(\vec{x}) = -\nabla E(\vec{x})$.

Sometimes the force (and therefore the energy) scale with the “size” of the object (i.e., the object’s total charge in an electric potential due to a fixed set of charges, the mass of an object in the gravitational potential of a black hole, the magnetization of the object in a magnetic potential, etc.). In these cases, the potential field can be defined in terms of a unit size (per unit charge, per unit mass, etc.). One can determine whether such a scaling is applied by checking the units.

[5] As with any energy, there is always an arbitrary constant associated with the position (or state) at which the energy is taken to be zero. There is no such ambiguity with force. Forces are, in a sense, more fundamental than energies. Energy appears to be fundamental because all observations of the first law of thermodynamics demonstrate that there is a conserved quantity which is a state function and is called energy.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 13: Differential Operations on Vectors

Reading:
Kreyszig Sections: 9.8, 9.9 (pages410–413, 414–416)

Generalizing the Derivative

The number of different ideas, whether from physical science or other disciplines, that can be understood with reference to the "meaning" of a derivative from the calculus of scalar functions is very very large. Our ideas about many topics, such as price elasticity, strain, stability, and optimization, are connected to our understanding of a derivative.

In vector calculus, there are generalizations to the derivative from basic calculus that acts on a scalar and gives another scalar back:

gradient (∇): A derivative on a scalar that gives a vector.

curl ($\nabla\times$): A derivative on a vector that gives another vector.

divergence ($\nabla\cdot$): A derivative on a vector that gives scalar.

Each of these have "meanings" that can be applied to a broad class of problems.

The gradient operation on $f(\vec{x}) = f(x, y, z) = f(x_1, x_2, x_3)$,

$$\text{grad} f = \nabla f \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) f \tag{13-1}$$

has been discussed previously. The curl and divergence will be discussed below.

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

Lecture 13 MATHEMATICA® Example 1

Gradients and Laplacians on Scalar Potentials

notebook (non-evaluated) pdf (evaluated) html (evaluated)

An example of a scalar potential due three point charges in the plane is visualized. Methods for computing a gradient and the divergence of a gradient (Laplacian) are presented.

1: This is the 2D $1/r$-potential; here *potential* takes four arguments: two for the location of the charge and two for the position where the "test" charge "feels" the potential.

4: This is the third of three fixed charge potentials, arranged at the vertices of an equilateral triangle.

5: *gradfield* is an example of a function that takes a scalar function of x and y and returns a vector with component derivatives...

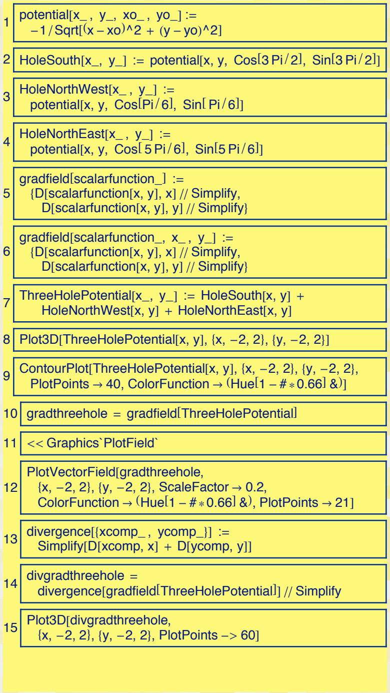
6: However, the previous example only works for functions of x and y explicitly. This expands *gradfield* to other cartesian coordinates other than x and y.

8: `Plot3D` is used to visualize the superposition of the three charge potentials defined as *ThreeHolePotential* .

9: `ContourPlot` is an alternative method to visualize this scalar field. The option `ColorFunction` points to an example of a *Pure Function*—a method of making functions that do not operate with the usual "square brackets." Pure functions are indicated with the & at the end; the # is a place-holder for the pure function's argument.

12: `PlotVectorField` is in the `Graphics‘PlotField‘` package. Because a gradient produces a vector field from a scalar potential, arrows are used at discrete points to visualize it.

14: The divergence operates on a vector and produces a scalar. Therefore, taking the divergence of the gradient of a scalar field returns a scalar field that is naturally associated with the original—its physical interpretation is (minus) the rate at which gradient vectors "diverge" from a point.



```
1   potential[x_ , y_, xo_ , yo_] :=
      -1 / Sqrt[(x - xo)^2 + (y - yo)^2]

2   HoleSouth[x_, y_] := potential[x, y, Cos[3 Pi / 2], Sin[3 Pi / 2]]

3   HoleNorthWest[x_ , y_] :=
      potential[x, y, Cos[Pi / 6], Sin[ Pi / 6]]

4   HoleNorthEast[x_ , y_] :=
      potential[x, y, Cos[ 5 Pi / 6], Sin[5 Pi / 6]]

5   gradfield[scalarfunction_] :=
      {D[scalarfunction[x, y], x] // Simplify,
        D[scalarfunction[x, y], y] // Simplify}

6   gradfield[scalarfunction_, x_ , y_] :=
      {D[scalarfunction[x, y], x] // Simplify,
        D[scalarfunction[x, y], y] // Simplify}

7   ThreeHolePotential[x_, y_] := HoleSouth[x, y] +
        HoleNorthWest[x, y] + HoleNorthEast[x, y]

8   Plot3D[ThreeHolePotential[x, y], {x, -2, 2}, {y, -2, 2}]

9   ContourPlot[ThreeHolePotential[x, y], {x, -2, 2}, {y, -2, 2},
      PlotPoints → 40, ColorFunction → (Hue[1 - # * 0.66] &)]

10  gradthreehole = gradfield[ThreeHolePotential]

11  << Graphics`PlotField`

12  PlotVectorField[gradthreehole,
      {x, -2, 2}, {y, -2, 2}, ScaleFactor → 0.2,
      ColorFunction → (Hue[1 - # * 0.66] &), PlotPoints → 21]

13  divergence[{xcomp_ , ycomp_}] :=
      Simplify[D[xcomp, x] + D[ycomp, y]]

14  divgradthreehole =
      divergence[gradfield[ThreeHolePotential]] // Simplify

15  Plot3D[divgradthreehole,
      {x, -2, 2}, {y, -2, 2}, PlotPoints -> 60]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Divergence and Its Interpretation

The divergence operates on a vector field that is a function of position, $\vec{v}(x, y, z) = \vec{v}(\vec{x}) = (v_1(\vec{x}), v_2(\vec{x}), v_3(\vec{x}))$, and returns a scalar that is a function of position. The scalar field is often called the divergence field of $\vec{v}$ or simply the divergence of $\vec{v}$.

$$\text{div } \vec{v}(\vec{x}) = \nabla \cdot \vec{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y} + \frac{\partial v_3}{\partial z} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \cdot (v_1, v_2, v_3) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \cdot \vec{v} \qquad (13\text{-}2)$$

Think about what the divergence means,

Coordinate Systems

The above definitions are for a Cartesian (x, y, z) system. Sometimes it is more convenient to work in other (spherical, cylindrical, etc) coordinate systems. In other coordinate systems, the derivative operations ∇, $\nabla\cdot$, and $\nabla\times$ have different forms. These other forms can be derived, or looked up in a mathematical handbook, or specified by using the MATHEMATICA® package "VectorAnalysis."

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 13 MATHEMATICA® Example 2

Coordinate Transformations

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of *Coordinate Transformations* obtained from the `Calculus‘VectorAnalysis‘` package. An frivolous example of computing distances from Boston to Paris along different routes using data from the `Miscellaneous‘CityData‘` package.

2: `CoordinatesFromCartesian` from the `Calculus‘VectorAnalysis‘` package transforms three cartesian coordinates, named in the first argument-list into one of many coordinate systems named by the second argument.

3: `CoordinatesFromCartesian` transforms one of many different coordinate systems, named in the second argument into three cartesian coordinates, named in the first argument-list.

7: `CityData` in the `Calculus‘VectorAnalysis‘` package can give the latitude and longitude of cities in the database—in this case Boston and Paris.

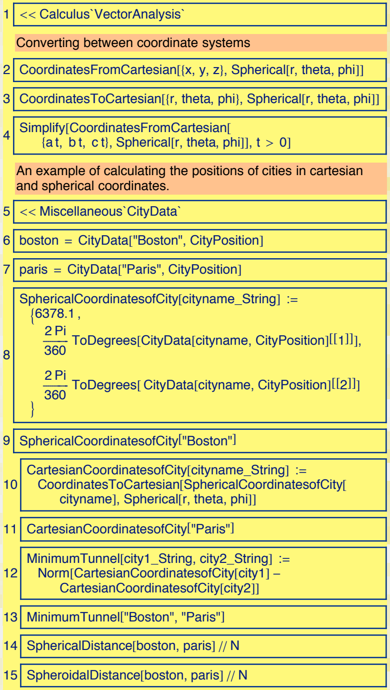
8: *SphericalCoordinatesofCity* takes the string-argument of a city name and uses `CityData` to compute its spherical coordinates (i.e., $(r_{\text{earth}}, \theta, \phi)$ are same as (average earth radius = 6378.1 km, latitude, longitude)). `ToDegrees` is from the `Miscellaneous‘Geodesy‘` package and converts a (degree, minutes, seconds)-structure to degrees.

10: *CartesianCoordinatesofCity* uses a coordinate transform and *SphericalCoordinatesofCity* to compute cartesian coordinates.

12: Imagining that a tunnel could be constructed between two cities, this function would calculate the minimum distance between cities.

14: Comparing the great circle route using `SphericalDistance` to the euclidian distance is a result that suprises me. It would save only about 55 kilometers to dig a tunnel to Paris—sigh.

15: `SpheroidalDistance` accounts for the earth's extra waistline for computing minimum distances.



```
1  << Calculus`VectorAnalysis`

Converting between coordinate systems

2  CoordinatesFromCartesian[{x, y, z}, Spherical[r, theta, phi]]

3  CoordinatesToCartesian[{r, theta, phi}, Spherical[r, theta, phi]]

4  Simplify[CoordinatesFromCartesian[
        {a t, b t, c t}, Spherical[r, theta, phi]], t > 0]

An example of calculating the positions of cities in cartesian
and spherical coordinates.

5  << Miscellaneous`CityData`

6  boston = CityData["Boston", CityPosition]

7  paris = CityData["Paris", CityPosition]

8  SphericalCoordinatesofCity[cityname_String] :=
     {6378.1 ,
       2 Pi
       ---- ToDegrees[CityData[cityname, CityPosition][[1]]],
       360

       2 Pi
       ---- ToDegrees[ CityData[cityname, CityPosition][[2]]]
       360
     }

9  SphericalCoordinatesofCity["Boston"]

10 CartesianCoordinatesofCity[cityname_String] :=
     CoordinatesToCartesian[SphericalCoordinatesofCity[
          cityname], Spherical[r, theta, phi]]

11 CartesianCoordinatesofCity["Paris"]

12 MinimumTunnel[city1_String, city2_String] :=
     Norm[CartesianCoordinatesofCity[city1] –
          CartesianCoordinatesofCity[city2]]

13 MinimumTunnel["Boston", "Paris"]

14 SphericalDistance[boston, paris] // N

15 SpheroidalDistance[boston, paris] // N
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 13 MATHEMATICA® Example 3

Gradient and Divergence Operations in Other Coordinate Systems

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A $1/r^n$-potential is used to demonstrate how to obtain gradients and divergences in other coordinate systems.

1: *SimplePot* is an example function—a $1/r^n$ potential in cartesian coordinates.

2: `Grad` is defined in the `Calculus‘VectorAnalysis‘`: in this form it takes a scalar function and returns its gradient in the coordinate system defined by the second argument.

3: An alternate form of *SimplePot* is defined here in spherical coordinates.

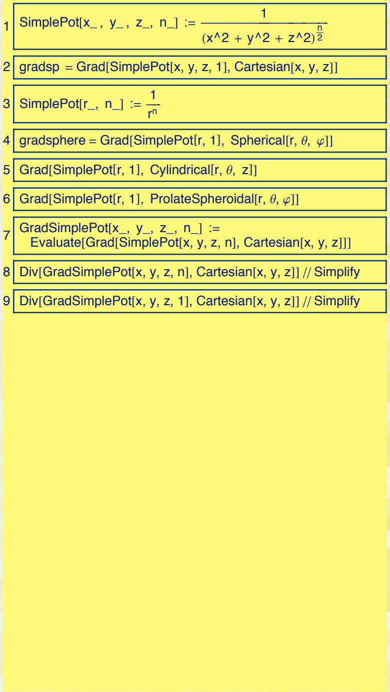
4: Here, the gradient of $1/r$ is obtained in spherical coordinates.

5: Here, the gradient of $1/r$ is obtained in cylindrical coordinates.

6: Here, the gradient of $1/r$ is obtained in prolate spheriodal coordinates.

8: The laplacian ($\nabla^2(1/r^n)$) has a particularly simple form...

9: By inspection of $\nabla^2(1/r^n)$ or by direct calculation, it follows that $\nabla^2(1/r)$ vanishes identically.



```
1  SimplePot[x_ , y_ , z_, n_] := 1/(x^2 + y^2 + z^2)^(n/2)
2  gradsp = Grad[SimplePot[x, y, z, 1], Cartesian[x, y, z]]
3  SimplePot[r_, n_] := 1/r^n
4  gradsphere = Grad[SimplePot[r, 1], Spherical[r, θ, φ]]
5  Grad[SimplePot[r, 1], Cylindrical[r, θ, z]]
6  Grad[SimplePot[r, 1], ProlateSpheroidal[r, θ, φ]]
7  GradSimplePot[x_, y_, z_, n_] :=
     Evaluate[Grad[SimplePot[x, y, z, n], Cartesian[x, y, z]]]
8  Div[GradSimplePot[x, y, z, n], Cartesian[x, y, z]] // Simplify
9  Div[GradSimplePot[x, y, z, 1], Cartesian[x, y, z]] // Simplify
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Curl and Its Interpretation

The curl is the vector valued derivative of a vector function. As illustrated below, its operation can be geometrically interpreted as the rotation of a field about a point.

For a vector-valued function of (x, y, z):

$$\vec{v}(x, y, z) = \vec{v}(\vec{x}) = (v_1(\vec{x}), v_2(\vec{x}), v_3(\vec{x})) = v_1(x, y, z)\hat{\imath} + v_2(x, y, z)\hat{\jmath} + v_3(x, y, z)\hat{k} \tag{13-3}$$

the curl derivative operation is another vector defined by:

$$\text{curl } \vec{v} = \nabla \times \vec{v} = \left(\left(\frac{\partial v_3}{\partial y} - \frac{\partial v_2}{\partial z} \right), \left(\frac{\partial v_1}{\partial z} - \frac{\partial v_3}{\partial x} \right), \left(\frac{\partial v_2}{\partial x} - \frac{\partial v_1}{\partial y} \right) \right) \tag{13-4}$$

or with the memory-device:

$$\text{curl } \vec{v} = \nabla \times \vec{v} = \det \begin{pmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ v_1 & v_2 & v_3 \end{pmatrix} \tag{13-5}$$

For an example, consider the vector function that is often used in Brakke's Surface Evolver program:

$$\vec{w} = \frac{z^n}{(x^2 + y^2)(x^2 + y^2 + z^2)^{\frac{n}{2}}} (y\hat{\imath} - x\hat{\jmath}) \tag{13-6}$$

This will be shown below, in a MATHEMATICA® example, to have the property:

$$\nabla \times \vec{w} = \frac{n z^{n-1}}{(x^2 + y^2 + z^2)^{1+\frac{n}{2}}} (x\hat{\imath} + y\hat{\jmath} + z\hat{k}) \tag{13-7}$$

which is spherically symmetric for $n = 1$ and convenient for turning surface integrals over a portion of a sphere into a path-integral over a curve on a sphere.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 13 Mathematica® Example 4

Computing and Visualizing Curl Fields

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of curls are computing for a particular family of vector fields. Visualization is produced with the `PlotVectorField3D` function from the `Graphics‘PlotField3D‘`.

1: *LeavingKansas* is the family of vector fields indicated by 13-6.

4: The function will be singular for $n > 1$ along the $z-axis$, this singularity will be reported during the numerical evaluations for visualization.

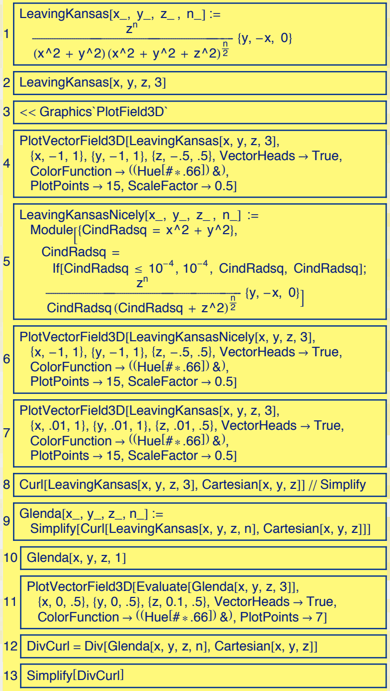
5: Here, the singularity is removed by testing the value of the argument and returning a fixed value along the singular axis.

7: Alternatively, the singular axis can be avoided by explicitly removing it from the domain of plotting.

9: This demonstrates the assertion (13-7) about the cylindrical symmetry of this curl for $n = 1$.

10: Visualizing the curl for $n = 3$: note that the field is points up with large magnitude near the vortex at the origin.

11: Demonstrate that the divergence of the curl of $\vec{w}$ vanishes for any n—this is true for any differentiable vector field.



```
1   LeavingKansas[x_, y_, z_ , n_] :=
                          z^n
      ─────────────────────────────────── {y, −x, 0}
      (x^2 + y^2) (x^2 + y^2 + z^2)^(n/2)

2   LeavingKansas[x, y, z, 3]

3   << Graphics`PlotField3D`

4   PlotVectorField3D[LeavingKansas[x, y, z, 3],
      {x, −1, 1}, {y, −1, 1}, {z, −.5, .5}, VectorHeads → True,
      ColorFunction → ((Hue[# * .66]) &),
      PlotPoints → 15, ScaleFactor → 0.5]

5   LeavingKansasNicely[x_, y_, z_ , n_] :=
      Module[{CindRadsq = x^2 + y^2},
        CindRadsq =
          If[CindRadsq ≤ 10^−4, 10^−4, CindRadsq, CindRadsq];
                         z^n
         ──────────────────────────────── {y, −x, 0}]
         CindRadsq (CindRadsq + z^2)^(n/2)

6   PlotVectorField3D[LeavingKansasNicely[x, y, z, 3],
      {x, −1, 1}, {y, −1, 1}, {z, −.5, .5}, VectorHeads → True,
      ColorFunction → ((Hue[# * .66]) &),
      PlotPoints → 15, ScaleFactor → 0.5]

7   PlotVectorField3D[LeavingKansas[x, y, z, 3],
      {x, .01, 1}, {y, .01, 1}, {z, .01, .5}, VectorHeads → True,
      ColorFunction → ((Hue[# * .66]) &),
      PlotPoints → 15, ScaleFactor → 0.5]

8   Curl[LeavingKansas[x, y, z, 3], Cartesian[x, y, z]] // Simplify

9   Glenda[x_, y_, z_, n_] :=
      Simplify[Curl[LeavingKansas[x, y, z, n], Cartesian[x, y, z]]]

10  Glenda[x, y, z, 1]

11  PlotVectorField3D[Evaluate[Glenda[x, y, z, 3]],
      {x, 0, .5}, {y, 0, .5}, {z, 0.1, .5}, VectorHeads → True,
      ColorFunction → ((Hue[# * .66]) &), PlotPoints → 7]

12  DivCurl = Div[Glenda[x, y, z, n], Cartesian[x, y, z]]

13  Simplify[DivCurl]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

One important result that has physical implications is that a the curl of a gradient is always zero: $f(\vec{x}) = f(x,y,z)$:

$$\nabla \times (\nabla f) = 0 \tag{13-8}$$

Therefore *if some vector function* $\vec{F}(x,y,z) = (F_x, F_y, F_z)$ *can be derived from a scalar potential,* $\nabla f = \vec{F}$, *then the curl of* $\vec{F}$ *must be zero.* This is the property of an *exact* differential $df = (\nabla f) \cdot (dx, dy, dz) = \vec{F} \cdot (dx, dy, dz)$. Maxwell's relations follow from equation 13-8:

$$\begin{aligned}
0 &= \frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z} = \frac{\partial \frac{\partial f}{\partial z}}{\partial y} - \frac{\partial \frac{\partial f}{\partial y}}{\partial z} = \frac{\partial^2 f}{\partial z \partial y} - \frac{\partial^2 f}{\partial y \partial z} \\
0 &= \frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x} = \frac{\partial \frac{\partial f}{\partial x}}{\partial z} - \frac{\partial \frac{\partial f}{\partial z}}{\partial x} = \frac{\partial^2 f}{\partial x \partial z} - \frac{\partial^2 f}{\partial z \partial x} \\
0 &= \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} = \frac{\partial \frac{\partial f}{\partial y}}{\partial x} - \frac{\partial \frac{\partial f}{\partial x}}{\partial y} = \frac{\partial^2 f}{\partial y \partial x} - \frac{\partial^2 f}{\partial x \partial y}
\end{aligned} \tag{13-9}$$

Another interpretation is that gradient fields are *curl free, irrotational, or conservative.*

The notion of conservative means that, if a vector function can be derived as the gradient of a scalar potential, then integrals of the vector function over any path is zero for a closed curve—meaning that there is no change in "state;" energy is a common state function.

Here is a picture that helps visualize why the curl invokes names associated with spinning, rotation, etc.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

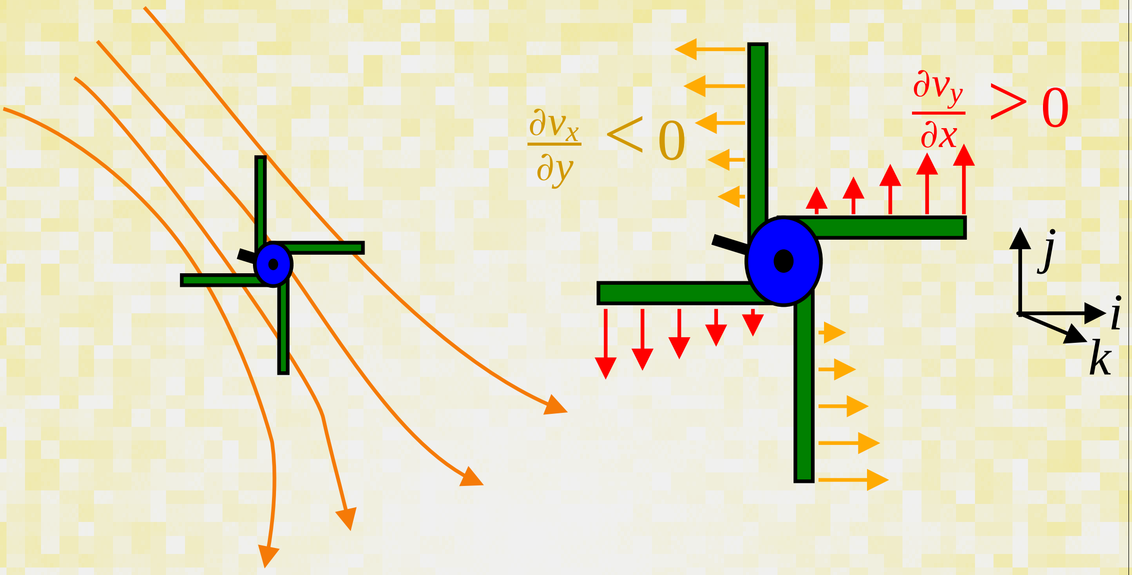

$\frac{\partial v_x}{\partial y} < 0$
$\frac{\partial v_y}{\partial x} > 0$
j
i
k

Figure 13-10: Consider a small paddle wheel placed in a set of stream lines defined by a vector field of position. If the v_y component is an increasing function of x, this tends to make the paddle wheel want to spin (positive, counter-clockwise) about the $\hat{k}$-axis. If the v_x component is a decreasing function of y, this tends to make the paddle wheel want to spin (positive, counter-clockwise) about the $\hat{k}$-axis. The net impulse to spin around the $\hat{k}$-axis is the sum of the two. Note that this is independent of the reference frame because a constant velocity $\vec{v} =$ const. and the local acceleration $\vec{v} = \nabla f$ can subtracted because of Eq. 13-10.

Another important result is that divergence of any curl is also zero, for $\vec{v}(\vec{x}) = \vec{v}(x, y, z)$:

$$\nabla \cdot (\nabla \times \vec{v}) = 0 \tag{13-10}$$

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

Lecture 14: Integrals along a Path

Reading:
Kreyszig Sections: 10.1, 10.2, 10.3 (pages420–425, 426–432, 433–439)

Integrals along a Curve

Consider the type of integral that everyone learns initially:

$$E(b) - E(a) = \int_a^b f(x)dx \tag{14-1}$$

The equation implies that f is integrable and

$$dE = f dx = \frac{dE}{dx}dx \tag{14-2}$$

so that the integral can be written in the following way:

$$E(b) - E(a) = \int_a^b dE \tag{14-3}$$

where a and b represent "points" on some *line* where E is to be evaluated.

Of course, there is no reason to restrict integration to a straight line—the generalization is the integration along a curve (or a path) $\vec{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$.

$$E(b) - E(a) = \int_{\vec{x}(a)}^{\vec{x}(b)} \vec{f}(\vec{x}) \cdot d\vec{x} = \int_a^b g(x(\vec{t}))dt = \int_a^b \nabla E \cdot \frac{d\vec{x}}{dt}dt = \int_a^b dE \tag{14-4}$$

This last set of equations assumes that the gradient exists–i.e., there is some function E that has the gradient $\nabla E = \vec{f}$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Path-Independence and Path-Integration

If the function being integrated along a simply-connected path (Eq. 14-4) is a gradient of some scalar potential, then the path between two integration points does not need to be specified: the integral is independent of path. It also follows that for closed paths, the integral of the gradient of a scalar potential is zero.[6] A simply-connected path is one that does not self-intersect or can be shrunk to a point without leaving its domain.

There are familiar examples from classical thermodynamics of simple one-component fluids that satisfy this property:

$$\oint dU = \oint \nabla_{\vec{\mathcal{S}}} U \cdot d\vec{\mathcal{S}} = 0 \qquad \oint dS = \oint \nabla_{\vec{\mathcal{S}}} S \cdot d\vec{\mathcal{S}} = 0 \qquad \oint dG = \oint \nabla_{\vec{\mathcal{S}}} G \cdot d\vec{\mathcal{S}} = 0 \tag{14-5}$$

$$\oint dP = \oint \nabla_{\vec{\mathcal{S}}} P \cdot d\vec{\mathcal{S}} = 0 \qquad \oint dT = \oint \nabla_{\vec{\mathcal{S}}} T \cdot d\vec{\mathcal{S}} = 0 \qquad \oint dV = \oint \nabla_{\vec{\mathcal{S}}} V \cdot d\vec{\mathcal{S}} = 0 \tag{14-6}$$

Where $\vec{\mathcal{S}}$ is any other set of variables that sufficiently describe the equilibrium state of the system (i.e, $U(S,V)$, $U(S,P)$, $U(T,V)$, $U(T,P)$ for U describing a simple one-component fluid).

The relation curl grad $f = \nabla \times \nabla f = 0$ provides method for testing whether some *general* $\vec{F}(\vec{x})$ is independent of path. If

$$\vec{0} = \nabla \times \vec{F} \tag{14-7}$$

or equivalently,

$$0 = \frac{\partial F_j}{\partial x_i} - \frac{\partial F_i}{\partial x_j} \tag{14-8}$$

for all variable pairs x_i, x_j, then $\vec{F}(\vec{x})$ is independent of path. These are the Maxwell relations of classical thermodynamics.

[6] In fact, there are some extra requirements on the domain (i.e., the space of all paths that are supposed to be path-independent) where such paths are defined: the scalar potential must have continuous second partial derivatives everywhere in the domain.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 14 Mathematica® Example 1

Path Dependence of Integration of Vector Function: Non-Conservative Example

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The path dependence of a vector field with a non-vanishing curl ($\vec{v}(\vec{x}) = xyz(\hat{i} + \hat{k} + \hat{z})$) is demonstrated with a family of closed curves.

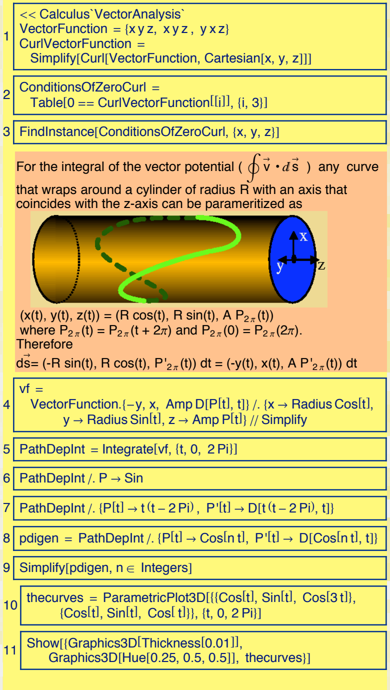
1: *VectorFunction* (xyz, xyz, xyz) is an example vector field that has a *non-vanishing curl*. The curl is computed with `Curl` which is in the `Calculus‘VectorAnalysis‘` package. Here, the particular coordinate system is specified with `Cartesian` argument to `Curl`.

3: The curl vanishes only at the origin—this is shown with `FindInstance` called with a list of equations corresponding to the vanishing curl.

4: This is the integrand $\vec{v} \cdot d\vec{s}$ computed as indicated in the figure. $P(\theta)$ represents any periodic function, but $(x, y) = R(\cos\theta, \sin\theta)$ representing paths that wrap around cylinders.

5: *PathDepInt* is an integral for $\vec{v}$ represented by *VectorFunction* an arbitrary path wrapping around the cylinder.

7: This is the second example of a computation by using a replacement for a periodic $P(\theta)$ (i.e., each of the $P(\theta)$ begin and end at the same point, but the path between differs). That the two results differ shows that $\vec{v}$ is path-dependent—this is a general result for non-vanishing curl vector functions.



```
1  << Calculus`VectorAnalysis`
   VectorFunction = {x y z, x y z , y x z}
   CurlVectorFunction =
     Simplify[Curl[VectorFunction, Cartesian[x, y, z]]]
```

```
2  ConditionsOfZeroCurl =
     Table[0 == CurlVectorFunction[[i]], {i, 3}]
```

```
3  FindInstance[ConditionsOfZeroCurl, {x, y, z}]
```

For the integral of the vector potential ($\oint \vec{v} \cdot d\vec{s}$) any curve that wraps around a cylinder of radius R with an axis that coincides with the z-axis can be parameritized as

(x(t), y(t), z(t)) = (R cos(t), R sin(t), A $P_{2\pi}$(t))
where $P_{2\pi}(t) = P_{2\pi}(t + 2\pi)$ and $P_{2\pi}(0) = P_{2\pi}(2\pi)$.
Therefore
$d\vec{s}$= (-R sin(t), R cos(t), $P'_{2\pi}$(t)) dt = (-y(t), x(t), A $P'_{2\pi}$(t)) dt

```
4  vf =
     VectorFunction.{-y, x, Amp D[P[t], t]} /. {x → Radius Cos[t],
         y → Radius Sin[t], z → Amp P[t]} // Simplify
```

```
5  PathDepInt = Integrate[vf, {t, 0, 2 Pi}]
```

```
6  PathDepInt /. P → Sin
```

```
7  PathDepInt /. {P[t] → t (t - 2 Pi), P'[t] → D[t (t - 2 Pi), t]}
```

```
8  pdigen = PathDepInt /. {P[t] → Cos[n t], P'[t] → D[Cos[n t], t]}
```

```
9  Simplify[pdigen, n ∈ Integers]
```

```
10  thecurves = ParametricPlot3D[{{Cos[t], Sin[t], Cos[3 t]},
          {Cos[t], Sin[t], Cos[ t]}}, {t, 0, 2 Pi}]
```

```
11  Show[{Graphics3D[Thickness[0.01]],
       Graphics3D[Hue[0.25, 0.5, 0.5]], thecurves}]
```

3.016 Home

Full Screen

Close

Quit

Lecture 14 Mathematica® Example 2

Examples of Path-Independence of Curl-Free Vector Fields and Curl-Free Subspaces

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A curl-free vector field can be generated from any scalar potential, in this case $\vec{w} = \nabla e^{xyz} = \vec{w}(\vec{x}) = e^{xyz}(yz\hat{i} + zx\hat{k} + xy\hat{z})$. To find a function that is curl-free on a restricted subpace (for example, the vector function $\vec{v}(\vec{x}) = (x^2 + y^2 - R^2)\hat{z}$ vanishes on the surface of a cylinder) one needs to find a $\vec{m}$ such that $\nabla \times \vec{m} = \vec{v}$ (for this case

$$\vec{m} = \frac{1}{2}\left(yR^2 \left[1 - x^2 - \frac{y^2}{3} \right] \hat{x} + -xR^2 \left[1 - y^2 - \frac{x^2}{3} \right] \hat{y} \right)$$

is one of an infinite number of such vector functions.)

1: To ensure that we will have a zero-curl, a vector field is generated from a gradient of a scalar potential. The curl vanishes because $\nabla \times \nabla f = 0$.

2: This is a demonstration that the curl does indeed vanish.

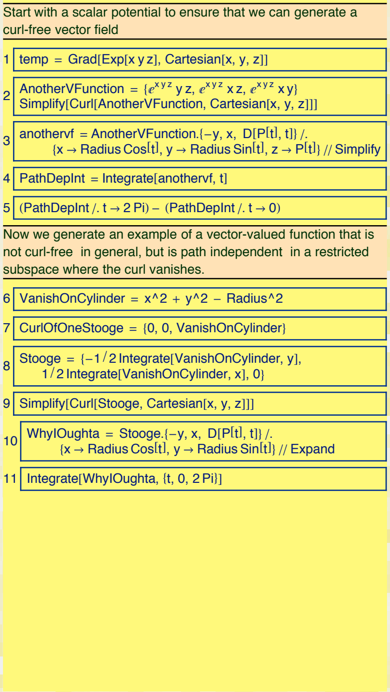
3: Here is the integrand for $\oint \vec{v} \cdot d\vec{s}$ for the family of paths that wrap around a cylinder for the particular case of this conservative fields.

4: This is the general result for the family of curves indicated by $P(\theta)\ldots$

5: This demonstrates that the path integral closes for any perioidic $P(\theta)$—which is the same as the condition that the curve is closed.

8: This demonstrates the method used to find the vector function which has a curl that vanishes on a cylinder.

11: This will demonstrate that the integral of the generally non-zero curl vector function is path independent *as long as the path lies on a surface where the curl of the vector function vanishes.*



Start with a scalar potential to ensure that we can generate a curl-free vector field

```
1   temp = Grad[Exp[x y z], Cartesian[x, y, z]]

2   AnotherVFunction = {e^(x y z) y z, e^(x y z) x z, e^(x y z) x y}
    Simplify[Curl[AnotherVFunction, Cartesian[x, y, z]]]

3   anothervf = AnotherVFunction.{-y, x, D[P[t], t]} /.
          {x → Radius Cos[t], y → Radius Sin[t], z → P[t]} // Simplify

4   PathDepInt = Integrate[anothervf, t]

5   (PathDepInt /. t → 2 Pi) - (PathDepInt /. t → 0)
```

Now we generate an example of a vector-valued function that is not curl-free in general, but is path independent in a restricted subspace where the curl vanishes.

```
6   VanishOnCylinder = x^2 + y^2 - Radius^2

7   CurlOfOneStooge = {0, 0, VanishOnCylinder}

8   Stooge = {-1/2 Integrate[VanishOnCylinder, y],
        1/2 Integrate[VanishOnCylinder, x], 0}

9   Simplify[Curl[Stooge, Cartesian[x, y, z]]]

10  WhyIOughta = Stooge.{-y, x, D[P[t], t]} /.
          {x → Radius Cos[t], y → Radius Sin[t]} // Expand

11  Integrate[WhyIOughta, {t, 0, 2 Pi}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Multidimensional Integrals

Perhaps the most straightforward of the higher-dimensional integrations (e.g., vector function along a curve, vector function on a surface) is a scalar function over a domain such as, a rectangular block in two dimensions, or a block in three dimensions. In each case, the integration over a dimension is uncoupled from the others and the problem reduces to pedestrian integration along a coordinate axis.

Sometimes difficulty arises when the domain of integration is not so easily described; in these cases, the limits of integration become functions of another integration variable. While specifying the limits of integration requires a bit of attention, the only thing that makes these cases difficult is that the integrals become tedious and lengthy. MATHEMATICA® removes some of this burden.

A short review of various ways in which a function's variable can appear in an integral follows:

	The Integral	Its Derivative
Function of limits	$p(x) = \int_{\alpha(x)}^{\beta(x)} f(\xi) d\xi$	$\frac{dp}{dx} = f(\beta(x))\frac{d\beta}{dx} - f(\alpha(x))\frac{d\alpha}{dx}$
Function of integrand	$q(x) = \int_a^b g(\xi, x) d\xi$	$\frac{dq}{dx} = \int_a^b \frac{\partial g(\xi, x)}{\partial x} d\xi$
Function of both	$r(x) = \int_{\alpha(x)}^{\beta(x)} g(\xi, x) d\xi$	$\frac{dr}{dx} = f(\beta(x))\frac{d\beta}{dx} - f(\alpha(x))\frac{d\alpha}{dx}$ $+ \int_{\alpha(x)}^{\beta(x)} \frac{\partial g(\xi, x)}{\partial x} d\xi$

Using Jacobians to Change Variables in Thermodynamic Calculations

Changing of variables is a topic in multivariable calculus that often causes difficulty in classical thermodynamics.

This is an extract of my notes on thermodynamics: http://pruffle.mit.edu/3.00/

Alternative forms of differential relations can be derived by changing variables.

To change variables, a useful scheme using Jacobians can be employed:

$$\begin{aligned}\frac{\partial(u,v)}{\partial(x,y)} &\equiv \det\begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y}\end{vmatrix} \\ &= \frac{\partial u}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial u}{\partial y}\frac{\partial v}{\partial x} \\ &= \left(\frac{\partial u}{\partial x}\right)_y \left(\frac{\partial v}{\partial y}\right)_x - \left(\frac{\partial u}{\partial y}\right)_x \left(\frac{\partial v}{\partial x}\right)_y \\ &= \frac{\partial u(x,y)}{\partial x}\frac{\partial v(x,y)}{\partial y} - \frac{\partial u(x,y)}{\partial y}\frac{\partial v(x,y)}{\partial x}\end{aligned} \tag{14-9}$$

$$\begin{aligned}\frac{\partial(u,v)}{\partial(x,y)} = -\frac{\partial(v,u)}{\partial(x,y)} &= \frac{\partial(v,u)}{\partial(y,x)} \\ \frac{\partial(u,v)}{\partial(x,v)} &= \left(\frac{\partial u}{\partial x}\right)_v \\ \frac{\partial(u,v)}{\partial(x,y)} &= \frac{\partial(u,v)}{\partial(r,s)}\frac{\partial(r,s)}{\partial(x,y)}\end{aligned} \tag{14-10}$$

For example, the heat capacity at constant volume is:

$$\begin{aligned}C_V &= T\left(\frac{\partial S}{\partial T}\right)_V = T\frac{\partial(S,V)}{\partial(T,V)} \\ &= T\frac{\partial(S,V)}{\partial(T,P)}\frac{\partial(T,P)}{\partial(T,V)} = T\left[\left(\frac{\partial S}{\partial T}\right)_P\left(\frac{\partial V}{\partial P}\right)_T - \left(\frac{\partial S}{\partial P}\right)_T\left(\frac{\partial V}{\partial T}\right)_P\right]\left(\frac{\partial P}{\partial V}\right)_T \\ &= T\frac{C_P}{T} - T\left(\frac{\partial P}{\partial V}\right)_T\left(\frac{\partial V}{\partial T}\right)_P\left(\frac{\partial S}{\partial P}\right)_T\end{aligned} \tag{14-11}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Using the Maxwell relation, $\left(\frac{\partial S}{\partial P}\right)_T = -\left(\frac{\partial V}{\partial T}\right)_P$,

$$C_P - C_V = -T\frac{[\left(\frac{\partial V}{\partial T}\right)_P]^2}{\left(\frac{\partial V}{\partial P}\right)_T} \tag{14-12}$$

which demonstrates that $C_P > C_V$ because, for any stable substance, the volume is a decreasing function of pressure at constant temperature.

. Example of a Multiple Integral: Electrostatic Potential above a Charged Region

This will be an example calculation of the spatially-dependent energy of a unit point charge in the vicinity of a charged planar region having the shape of an equilateral triangle. The calculation superimposes the charges from each infinitessimal area by integrating a $1/r$ potential from each point in space to each infinitessimal patch in the equilateral triangle The energy of a point charge $|e|$ due to a surface patch on the plane $z = 0$ of size $d\xi d\eta$ with surface charge density $\sigma(x, y)$ is:

$$dE(x, y, z, \xi, \eta) = \frac{|e|\sigma(\xi, \eta)d\xi d\eta}{\vec{r}(x, y, z, \xi, \eta)} \tag{14-13}$$

for a patch with uniform charge,

$$dE(x, y, z, \xi, \eta) = \frac{|e|\sigma d\xi d\eta}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + z^2}} \tag{14-14}$$

For an equilateral triangle with sides of length one and center at the origin, the vertices can be located at $(0, \sqrt{3}/2)$ and $(\pm 1/2, -\sqrt{3}/6)$.
The integration becomes

$$E(x, y, z) \propto \int_{-\sqrt{3}/6}^{\sqrt{3}/2} \left(\int_{\eta-\sqrt{3}/2}^{\sqrt{3}/2-\eta} \frac{d\xi}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + z^2}} \right) d\eta \tag{14-15}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 14 MATHEMATICA® Example 3

Potential near a Charged and Shaped Surface Patch: Brute Force

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A example of a multiple integral and its numerical evaluation for the triangular charged patch.

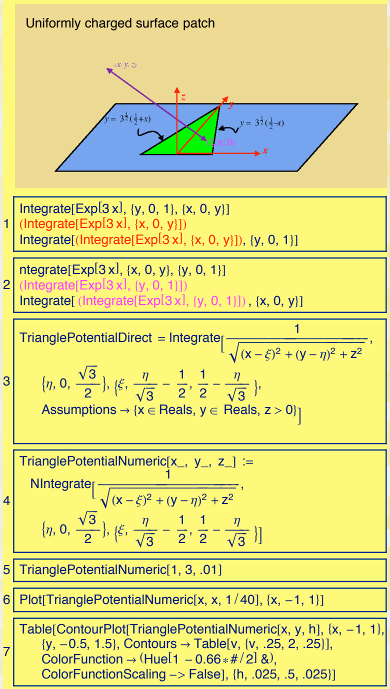
2: `Integrate`'s syntax is to integrate over the last integration iterator first, and the first iterator last.

3: This will show that the closed form of the above integral appears to be unknown to MATHEMATICA® . . .

4: However, the energy can be integrated numerically. Here is a function that calls `NIntegrate` for a location given by its arguments.

6: This will be a very slow calculation on most computers, but it will show how the potential changes along a line segment of length 2 that runs through the origin at 45°.

7: Even slower, `ContourPlot` is used at sequential heights for use as an animation.


Uniformly charged surface patch
(x; y; z)
z
y
$y = 3^{\frac{1}{2}}(\frac{1}{2}+x)$
$y = 3^{\frac{1}{2}}(\frac{1}{2}-x)$
(ξ;η;0)
x

1
```
Integrate[Exp[3 x], {y, 0, 1}, {x, 0, y}]
(Integrate[Exp[3 x], {x, 0, y}])
Integrate[(Integrate[Exp[3 x], {x, 0, y}]), {y, 0, 1}]
```

2
```
ntegrate[Exp[3 x], {x, 0, y}, {y, 0, 1}]
(Integrate[Exp[3 x], {y, 0, 1}])
Integrate[ (Integrate[Exp[3 x], {y, 0, 1}]) , {x, 0, y}]
```

3

$$\text{TrianglePotentialDirect} = \text{Integrate}\left[\frac{1}{\sqrt{(x-\xi)^2+(y-\eta)^2+z^2}},\ \left\{\eta, 0, \frac{\sqrt{3}}{2}\right\}, \left\{\xi, \frac{\eta}{\sqrt{3}}-\frac{1}{2}, \frac{1}{2}-\frac{\eta}{\sqrt{3}}\right\},\ \text{Assumptions} \to \{x \in \text{Reals}, y \in \text{Reals}, z > 0\}\right]$$

4

$$\text{TrianglePotentialNumeric}[x\_,\ y\_,\ z\_] := \text{NIntegrate}\left[\frac{1}{\sqrt{(x-\xi)^2+(y-\eta)^2+z^2}},\ \left\{\eta, 0, \frac{\sqrt{3}}{2}\right\}, \left\{\xi, \frac{\eta}{\sqrt{3}}-\frac{1}{2}, \frac{1}{2}-\frac{\eta}{\sqrt{3}}\right\}\right]$$

5
```
TrianglePotentialNumeric[1, 3, .01]
```

6
```
Plot[TrianglePotentialNumeric[x, x, 1/40], {x, −1, 1}]
```

7
```
Table[ContourPlot[TrianglePotentialNumeric[x, y, h], {x, −1, 1},
    {y, −0.5, 1.5}, Contours → Table[v, {v, .25, 2, .25}],
    ColorFunction → (Hue[1 − 0.66 ∗ # / 2] &),
    ColorFunctionScaling −> False], {h, .025, .5, .025}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 15: Surface Integrals and Some Related Theorems

Reading:
Kreyszig Sections: 10.4, 10.5, 10.6, 10.7 (pages439–444, 445–448, 449–458, 459–462)

Green's Theorem for Area in Plane Relating to its Bounding Curve

Reappraise the simplest integration operation, $g(x) = \int f(x)dx$. Temporarily ignore all the tedious mechanical rules of finding and integral and concentrate on what integration *does*.

Integration replaces a fairly complex process—adding up all the contributions of a function $f(x)$—with a clever new function $g(x)$ that only needs end-points to return the result of a complicated summation.

It is perhaps initially astonishing that this complex operation on the interior of the integration domain can be incorporated merely by the domain's endpoints. However, careful reflection provides a counterpoint to this marvel. How could it be otherwise? The function $f(x)$ is specified and there are no surprises lurking along the x-axis that will trip up dx as it marches merrily along between the endpoints. All the facts are laid out and they willingly submit to the process their preordination by $g(x)$ by virtue of the endpoints.[7]

The idea naturally translates to higher dimensional integrals and these are the basis for Green's theorem in the plane, Stoke's theorem, and Gauss (divergence) theorem. Here is the idea:

[7] I do hope you are amused by the evangelistic tone. I am a bit punchy from working non-stop on these lectures and wondering if anyone is really reading these notes. Sigh.

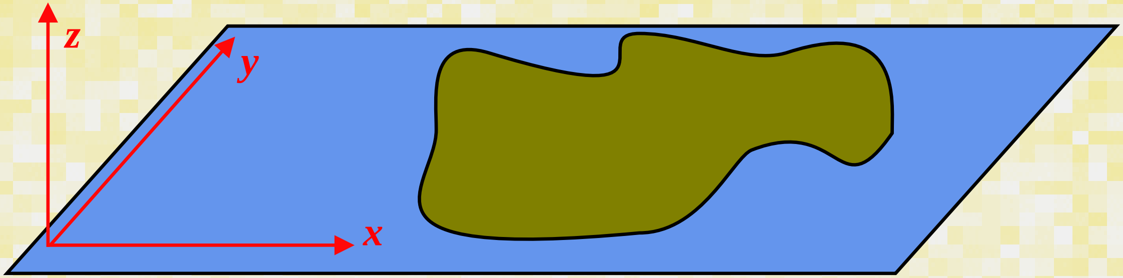


Figure 15-11: An irregular region on a plane surrounded by a closed curve. Once the closed curve (the edge of region) is specified, the area inside it is already determined. This is the simplest case as the area is the integral of the function $f = 1$ over $dxdy$. If some other function, $f(x, y)$, were specified on the plane, then its integral is also determined by summing the contributions along the boundary. This is a generalization $g(x) = \int f(x)dx$ and the basis behind Green's theorem in the plane.

The analog of the "*Fundamental Theorem of Differential and Integral Calculus*"[8] for a region $\mathcal{R}$ bounded in a plane with normal $\hat{k}$ that is bounded by a curve $\partial\mathcal{R}$ is:

$$\int\int_{\mathcal{R}} (\nabla \times \vec{F}) \cdot \hat{k} dxdy = \oint_{\partial\mathcal{R}} \vec{F} \cdot d\vec{r} \tag{15-1}$$

The following figure motivates Green's theorem in the plane:

[8] This is the theorem that implies the integral of a derivative of a function is the function itself (up to a constant).

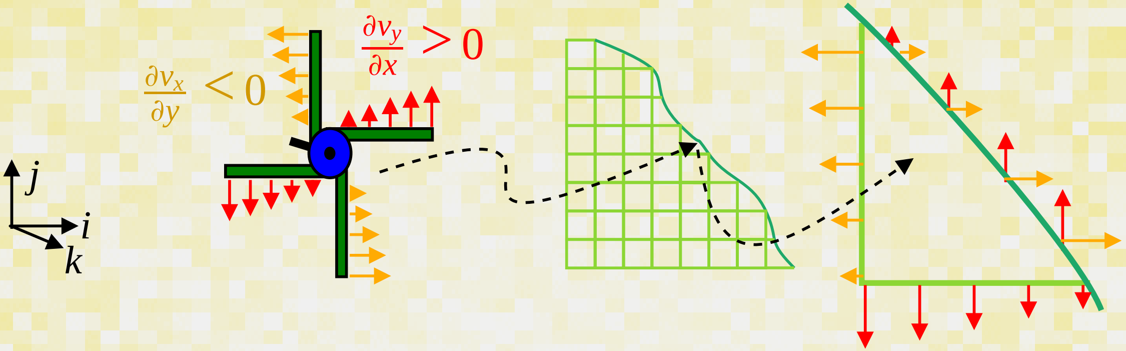

$\frac{\partial v_y}{\partial x} > 0$
$\frac{\partial v_x}{\partial y} < 0$
j
i
k

Figure 15-12: Illustration of how a vector valued function in a planar domain "spills out" of domain by evaluating the curl everywhere in the domain. Within the domain, the rotational flow $(\nabla \times F)$ from one cell moves into its neighbors; however, at the edges the local rotation is a net loss or gain. The local net loss or gain is $\vec{F} \cdot (dx, dy)$.

The generalization of this idea to a surface $\partial\mathcal{B}$ bounding a domain $\mathcal{B}$ results in Stokes' theorem, which will be discussed later.

In the following example, Green's theorem in the plane is used to simplify the integration to find the potential above a triangular path that was evaluated in a previous example. The result will be a considerable increase of efficiency of the numerical integration because the two-dimensional area integral over the interior of a triangle is reduced to a path integral over its sides.

The objective is to turn the integral for the potential

$$E(x, y, z) = \int\int_R \frac{d\xi d\eta}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + z^2}} \qquad (15\text{-}2)$$

into a path integral using Green's theorem in the x–y plane:

$$\int\int_R \left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right) dxdy = \int_{\partial R} (F_1 dx + F_2 dy) \qquad (15\text{-}3)$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

To find the vector function $\vec{F} = (F_1, F_2)$ which matches the integral in question, set $F_2 = 0$ and integrate to find F_1 via

$$\int \frac{d\eta}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + z^2}} \tag{15-4}$$

Lecture 15 MATHEMATICA® Example 1

Turning an integral over a domain into an integral over its boundary

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Here we turn the two dimensional numerical integration which requires $\mathcal{O}(N^2)$ calculations into an integration around the boundary which requires $\mathcal{O}(N)$ evaluations for the same accuracy. The path of integration must be determined (i.e., $(x(t), y(t))$) and then the integration is obtained via $(dx, dy) = (x'dt, y'dt)$.

1: Finding F_1 as indicated above is obtained easily with `Integrate`.

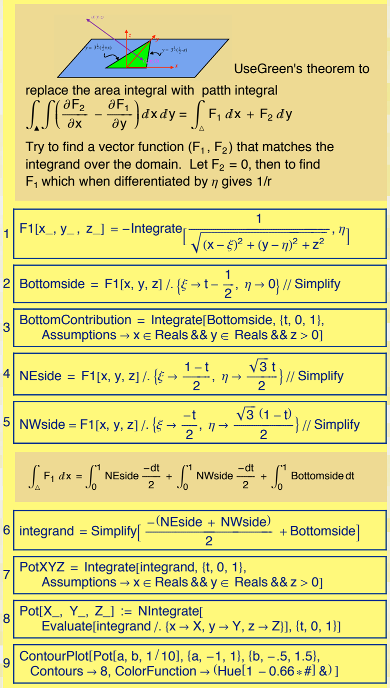
2: The bottom part of the triangle can be written as the curve: $(\zeta(t), \eta(t)) = (t - \frac{1}{2}, 0)$ for $0 < t < 1$; the integrand over that side is obtained by suitable replacement.

5: The remaining two legs of the triangle can be written similarly as: $((1-t)/2, \sqrt{3}t/2)$ and $(-t/2, \sqrt{3}(1-t)/2)$.

6: This is the integrand for the entire triangle to be integrated over $0 < t < 1$.

7: There is no free lunch—the closed form of the integral is either unknown or takes too long to compute.

8: However, `NIntegrate` is much more efficient because the problem has been reduced to a single integral instead of the double integral in the previous example.


(x, y, z)
z
y
x
$y = 3^{\frac{1}{2}}(\frac{1}{2}+x)$
$y = 3^{\frac{1}{2}}(\frac{1}{2}-x)$
(ξ, η, 0)

UseGreen's theorem to replace the area integral with patth integral

$$\int_{\blacktriangle}\int\left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y}\right) dx\, dy = \int_{\triangle} F_1\, dx + F_2\, dy$$

Try to find a vector function (F_1, F_2) that matches the integrand over the domain. Let $F_2 = 0$, then to find F_1 which when differentiated by η gives 1/r

```
1  F1[x_, y_ , z_] = -Integrate[1/Sqrt[(x - ξ)^2 + (y - η)^2 + z^2], η]

2  Bottomside = F1[x, y, z] /. {ξ → t - 1/2, η → 0} // Simplify

3  BottomContribution = Integrate[Bottomside, {t, 0, 1},
      Assumptions → x ∈ Reals && y ∈ Reals && z > 0]

4  NEside = F1[x, y, z] /. {ξ → (1 - t)/2, η → (Sqrt[3] t)/2} // Simplify

5  NWside = F1[x, y, z] /. {ξ → -t/2, η → (Sqrt[3] (1 - t))/2} // Simplify
```

$$\int_{\triangle} F_1\, dx = \int_0^1 \text{NEside}\,\frac{-dt}{2} + \int_0^1 \text{NWside}\,\frac{-dt}{2} + \int_0^1 \text{Bottomside}\,dt$$

```
6  integrand = Simplify[-(NEside + NWside)/2 + Bottomside]

7  PotXYZ = Integrate[integrand, {t, 0, 1},
      Assumptions → x ∈ Reals && y ∈ Reals && z > 0]

8  Pot[X_, Y_, Z_] := NIntegrate[
      Evaluate[integrand /. {x → X, y → Y, z → Z}], {t, 0, 1}]

9  ContourPlot[Pot[a, b, 1/10], {a, -1, 1}, {b, -.5, 1.5},
     Contours → 8, ColorFunction → (Hue[1 - 0.66*#] &) ]
```

Representations of Surfaces

Integration over the plane $z = 0$ in the form of $\int f(x, y)dxdy$ introduces surface integration—over a planar surface—as a straightforward extension to integration along a line. Just as integration over a line was generalized to integration over a curve by introducing two or three variables that depend on a *single* variable (e.g., $(x(t), y(t), z(t))$), a surface integral can be conceived as introducing three (or more) variables that depend on two parameters (i.e., $(x(u, v), y(u, v), z(u, v))$).

However, there are different ways to formulate representations of surfaces:

Surfaces and interfaces play fundamental roles in materials science and engineering. Unfortunately, the mathematics of surfaces and interfaces frequently presents a hurdle to materials scientists and engineering. The concepts in surface analysis can be mastered with a little effort, but there is no escaping the fact that the algebra is tedious and the resulting equations are onerous. Symbolic algebra and numerical analysis of surface alleviates much of the burden.

Most of the practical concepts derive from a second-order Taylor expansion of a surface near a point. The first-order terms define a tangent plane; the tangent plane determines the surface normal. The second-order terms in the Taylor expansion form a matrix and a quadratic form that can be used to formulate an expression for curvature. The eigenvalues of the second-order matrix are of fundamental importance.

The Taylor expansion about a particular point on the surface takes a particularly simple form if the origin of the coordinate system is located at the point and the z-axis is taken along the surface normal as illustrated in the following figure.

3.016

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

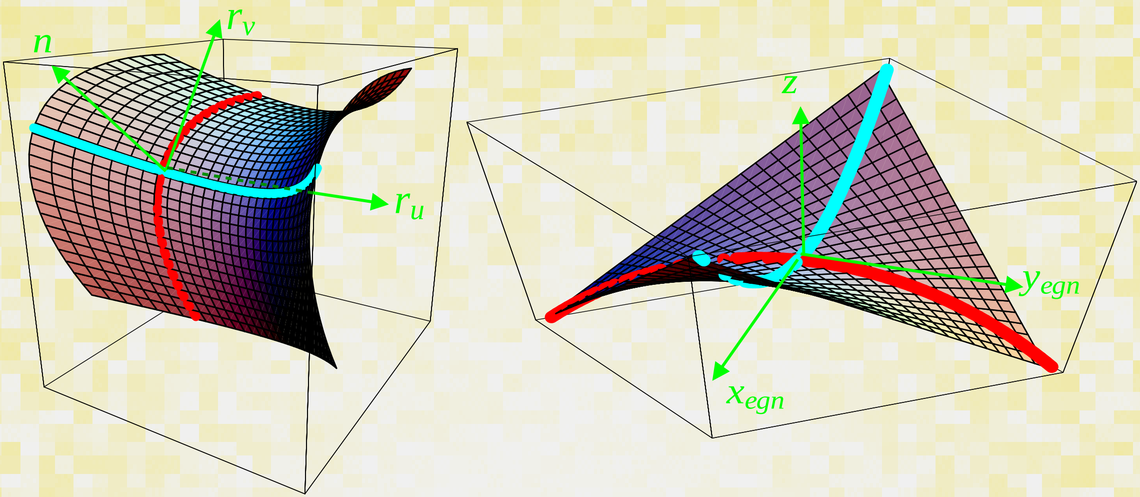


Figure 15-13: **Parabolic approximation to a surface and local eigenframe.** The surface on the left is a second-order approximation of a surface at the point where the coordinate axes are drawn. The surface has a local normal at that point which is related to the cross product of the two tangents of the coordinate curves that cross at the that point. The three directions define a coordinate system. The coordinate system can be translated so that the origin lies at the point where the surface is expanded and rotated so that the normal $\hat{n}$ coincides with the z-axis as in the right hand curve.

In this coordinate system, the Taylor expansion of $z = f(x, y)$ must be of the form

$$\Delta z = 0dx + 0dy + \frac{1}{2}(dx, dy)\begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}\begin{pmatrix} dx \\ dy \end{pmatrix}$$

If this coordinate system is rotated about the z-axis into its eigenframe where the off-diagonal components vanish, then the two eigenvalues represent the maximum and minimum curvatures. The sum of the eigenvalues is invariant to transformations and the sum is known as the *mean curvature* of the surface. The product of the eigenvalues is also invariant—this quantity is known as the *Gaussian curvature*.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The method in the figure suggests a method to calculate the normals and curvatures for a surface. Those results are tabulated below.

Level Set Surfaces: Tangent Plane, Surface Normal, and Curvature

$$F(x,y,z) = \text{const}$$

Tangent Plane ($\vec{x} = (x,y,z)$, $\vec{\xi} = (\xi,\eta,\zeta)$)

$$\nabla F \cdot (\vec{\xi} - \vec{x}) \text{ or } \frac{\partial F}{\partial x}(\xi - x) + \frac{\partial F}{\partial y}(\eta - y) + \frac{\partial F}{\partial z}(\zeta - z)$$

Normal

$$\frac{\xi - x}{\frac{\partial F}{\partial x}} = \frac{\eta - y}{\frac{\partial F}{\partial y}} = \frac{\zeta - z}{\frac{\partial F}{\partial z}}$$

Mean Curvature

$$\nabla \cdot \left(\frac{\nabla F}{\|\nabla F\|}\right) \text{ or}$$

$$\frac{\left[\begin{array}{c}\left(\frac{\partial^2 F}{\partial y^2} + \frac{\partial^2 F}{\partial z^2}\right)\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial^2 F}{\partial z^2} + \frac{\partial^2 F}{\partial x^2}\right)\left(\frac{\partial F}{\partial y}\right)^2 + \left(\frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2}\right)\left(\frac{\partial F}{\partial z}\right)^2 \\ -2\left(\frac{\partial F}{\partial x}\frac{\partial F}{\partial y}\frac{\partial^2 F}{\partial x \partial y} + \frac{\partial F}{\partial y}\frac{\partial F}{\partial z}\frac{\partial^2 F}{\partial y \partial z} + \frac{\partial F}{\partial z}\frac{\partial F}{\partial x}\frac{\partial^2 F}{\partial z \partial x}\right)\end{array}\right]}{\left(\frac{\partial F}{\partial x}^2 + \frac{\partial F}{\partial y}^2 + \frac{\partial F}{\partial z}^2\right)^{3/2}}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Parametric Surfaces: Tangent Plane, Surface Normal, and Curvature

$$\vec{x} = (p(u,v), q(u,v), s(u,v)) \text{ or } x = p(u,v) y = q(u,v) z = s(u,v)$$

Tangent Plane ($\vec{x} = (x,y,z)$, $\vec{\xi} = (\xi,\eta,\zeta)$)

$$(\vec{\xi} - \vec{x}) \cdot (\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv}) \det \begin{pmatrix} \xi - x & \eta - y & \zeta - z \\ \frac{\partial p}{\partial u} & \frac{\partial q}{\partial u} & \frac{\partial s}{\partial u} \\ \frac{\partial p}{\partial v} & \frac{\partial q}{\partial v} & \frac{\partial s}{\partial v} \end{pmatrix} = 0$$

Normal

$$\frac{\xi - x}{\frac{\partial(q,s)}{\partial(u,v)}} = \frac{\eta - y}{\frac{\partial(s,p)}{\partial(u,v)}} = \frac{\zeta - z}{\frac{\partial(p,q)}{\partial(u,v)}}$$

Mean Curvature

$$\frac{\left(\frac{d\vec{x}}{du} \cdot \frac{d\vec{x}}{du}\right)\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d^2\vec{x}}{dv^2}\right) - 2\left(\frac{d\vec{x}}{du} \cdot \frac{d\vec{x}}{dv}\right)\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d^2\vec{x}}{dudv}\right) + \left(\frac{d\vec{x}}{dv} \cdot \frac{d\vec{x}}{dv}\right)\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d^2\vec{x}}{du^2}\right)}{\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv}\right)^{3/2}}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Graph Surfaces: Tangent Plane, Surface Normal, and Curvature

$$z = f(x, y)$$

Tangent Plane ($\vec{x} = (x, y, z)$, $\vec{\xi} = (\xi, \eta, \zeta)$)

$$\frac{\partial f}{\partial x}(\xi - x) + \frac{\partial f}{\partial y}(\eta - y) = (\zeta - z)$$

Normal

$$\frac{\xi - x}{\frac{\partial f}{\partial x}} = \frac{\eta - y}{\frac{\partial f}{\partial y}} = \frac{\zeta - z}{-1}$$

Mean Curvature

$$\frac{(1 + {\frac{\partial f}{\partial x}}^2)\frac{\partial^2 f}{\partial y^2} - 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\frac{\partial^2 f}{\partial x \partial y} + (1 + {\frac{\partial f}{\partial y}}^2)\frac{\partial^2 f}{\partial x^2}}{\sqrt{1 + {\frac{\partial f}{\partial x}}^2 + {\frac{\partial f}{\partial y}}^2}}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 15 MATHEMATICA® Example 2

Representations of Surfaces

notebook (non-evaluated) pdf (evaluated) html (evaluated)

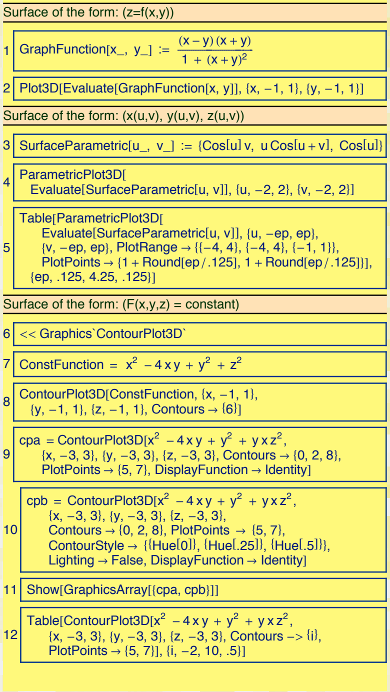
Visualization examples of surfaces represented in three ways: 1) graph $z = f(x, y)$; 2) parametric $(x(u, v), y(u, v), z(u, v))$; 3) level set constant $= F(x, y, z)$.

2: Using `Plot3D` to plot *GraphFunction* .

4: Using `ParametricPlot3D` to visualize a surface of the form $(x(u, v), y(u, v), z(u, v))$ given by *SurfaceParametric* . The lines of constant u and v generate the "square mesh" of the approximation to the surface. Each line on the surface is of the form: $\vec{r_1}(u) = (x(u, v = \text{const}), y(u, v = \text{const}), z(u, v = \text{const}))$ and $\vec{r_2}(v) = (x(u = \text{const}, v), y(u = \text{const}, v), z(u = \text{const}, v))$. The set of all crossing lines $\vec{r_1}(u)$ and $\vec{r_2}(v)$ is the surface. Each little "square" surface patch provides a convenient way to define the local surface normal—because both the vectors $d\vec{r_1}/du$ and $d\vec{r_2}/dv$ are tangent to the surface, their cross-product is either an inward-pointing normal or outward-pointing normal.

8: Using `ContourPlot3D` in the `Graphics‘ContourPlot3D‘` package to visualize the level set formulation of a surface constant $= F(x, y, z)$ given by *ConstFunction* .

12: Animation is produced by using `Table` to generate the level sets for different constants.



Surface of the form: (z=f(x,y))

```
1   GraphFunction[x_, y_] := (x - y) (x + y) / (1 + (x + y)^2)

2   Plot3D[Evaluate[GraphFunction[x, y]], {x, -1, 1}, {y, -1, 1}]
```

Surface of the form: (x(u,v), y(u,v), z(u,v))

```
3   SurfaceParametric[u_, v_] := {Cos[u] v, u Cos[u + v], Cos[u]}

4   ParametricPlot3D[
      Evaluate[SurfaceParametric[u, v]], {u, -2, 2}, {v, -2, 2}]

5   Table[ParametricPlot3D[
         Evaluate[SurfaceParametric[u, v]], {u, -ep, ep},
         {v, -ep, ep}, PlotRange → {{-4, 4}, {-4, 4}, {-1, 1}},
         PlotPoints → {1 + Round[ep / .125], 1 + Round[ep / .125]}],
      {ep, .125, 4.25, .125}]
```

Surface of the form: (F(x,y,z) = constant)

```
6   << Graphics`ContourPlot3D`

7   ConstFunction = x^2 - 4 x y + y^2 + z^2

8   ContourPlot3D[ConstFunction, {x, -1, 1},
      {y, -1, 1}, {z, -1, 1}, Contours → {6}]

9   cpa = ContourPlot3D[x^2 - 4 x y + y^2 + y x z^2,
         {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Contours → {0, 2, 8},
         PlotPoints → {5, 7}, DisplayFunction → Identity]

10   cpb = ContourPlot3D[x^2 - 4 x y + y^2 + y x z^2,
          {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
          Contours → {0, 2, 8}, PlotPoints → {5, 7},
          ContourStyle → {{Hue[0]}, {Hue[.25]}, {Hue[.5]}},
          Lighting → False, DisplayFunction → Identity]

11   Show[GraphicsArray[{cpa, cpb}]]

12   Table[ContourPlot3D[x^2 - 4 x y + y^2 + y x z^2,
          {x, -3, 3}, {y, -3, 3}, {z, -3, 3}, Contours -> {i},
          PlotPoints → {5, 7}], {i, -2, 10, .5}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Integration over Surfaces

Integration of a function over a surface is a straightforward generalization of $\int\int f(x,y)dxdy = \int f(x,y)dA$. The set of all little rectangles $dxdy$ defines a planar surface. A non-planar surface $\vec{x}(u,v)$ is composed of a set of little parallelogram patches with sides given by the infinitesimal vectors

$$\begin{aligned} \vec{r_u} du =& \frac{\partial \vec{x}}{\partial u} du \\ \vec{r_v} du =& \frac{\partial \vec{x}}{\partial v} dv \end{aligned} \tag{15-5}$$

Because the two vectors $\vec{r_u}$ and $\vec{r_v}$ are not necessarily perpendicular, their cross-product is needed to determine the magnitude of the area in the parallelogram:

$$dA = \|\vec{r_u} \times \vec{r_v}\| dudv \tag{15-6}$$

and the integral of some scalar function, $g(u,v) = g(x(u,v), y(u,v)) = g(\vec{x}(u,v))$, on the surface is

$$\int g(u,v) dA = \int\int g(u,v) \|\vec{r_u} \times \vec{r_v}\| dudv \tag{15-7}$$

However, the operation of taking the norm in the definition of the surface patch dA indicates that some information is getting lost—this is the local normal orientation of the surface. There are two choices for a normal (inward or outward).

When calculating some quantity that does not have vector nature, only the magnitude of the function over the area matters (as in Eq. 15-7). However, when calculating a vector quantity, such as the flow through a surface, or the total force applied to a surface, the surface orientation matters and it makes sense to consider the surface patch as a vector quantity:

$$\begin{aligned} \vec{A}(u,v) = \|\vec{A}\| \hat{n}(u,v) = A\hat{n}(u,v) \\ d\vec{A} = \vec{r_u} \times \vec{r_v} \end{aligned} \tag{15-8}$$

where $\hat{n}(u,v)$ is the local surface unit normal at $\vec{x}(u,v)$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 15 MATHEMATICA® Example 3

Example of an Integral over a Parametric Surface

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The surface energy of single crystals often depends on the surface orientation. This is especially the case for materials that have covalent and/or ionic bonds. To find the total surface energy of such a single crystal, one has to integrate an *orientation-dependent* surface energy, $\gamma(\hat{n})$, over the surface of a body. This example compares the total energy of such an anisotropic surface energy integrated over a sphere and a cube that enclose the same volume.

1: This is the parametric equation of the sphere in terms of longitude $v \in (0, 2\pi)$ and latitude $u \in (-\pi/2, \pi/2)$.

2: Calculate the tangent plane vectors $\vec{r_u}$ and $\vec{r_v}$

3: Using `CrossProduct` from the `Calculus‘VectorAnalysis‘` package to calculate $\vec{r_u} \times \vec{r_v}$ for subsequent use in the surface integral.

4: Using `DotProduct` to find the magnitude of the local normal.

5: This is the local unit normal $\hat{n}$.

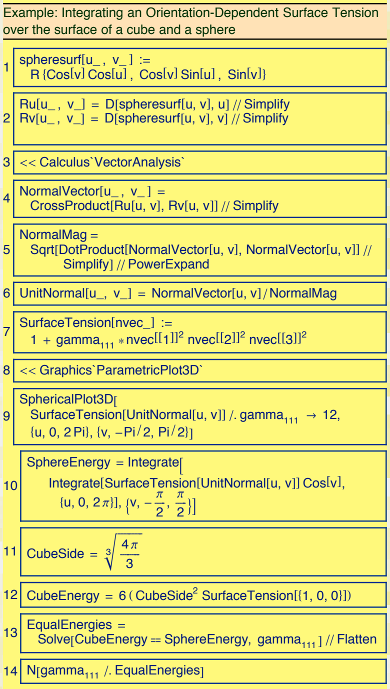
6: This is just an example of a $\gamma(\hat{n})$ that depends on direction that will be used for purposes of illustration.

9: Using `SphericalPlot3D` from the `Graphics‘ParametricPlot3D‘` package to illustrate the form of *SurfaceTension* for the particular choice of $\gamma_{111} = 12$.

10: Using the result from $|\vec{r_u} \times \vec{r_v}|$, the energy of a spherical body of radius $R = 1$ is computed by integrating $\gamma\hat{n}$ over the entire surface.

12: This would be the energy of a cubical body with the same volume.

14: This calculation is not very meaningful, but it is the value of γ_{111} such that the cube and sphere have the same total surface energy. The minimizing shape for a fixed volume is calculated using the *Wulff theorem.*



Example: Integrating an Orientation-Dependent Surface Tension over the surface of a cube and a sphere

```
1   spheresurf[u_, v_] :=
      R {Cos[v] Cos[u], Cos[v] Sin[u], Sin[v]}

2   Ru[u_, v_] = D[spheresurf[u, v], u] // Simplify
    Rv[u_, v_] = D[spheresurf[u, v], v] // Simplify

3   << Calculus`VectorAnalysis`

4   NormalVector[u_, v_] =
      CrossProduct[Ru[u, v], Rv[u, v]] // Simplify

5   NormalMag =
      Sqrt[DotProduct[NormalVector[u, v], NormalVector[u, v]] //
          Simplify] // PowerExpand

6   UnitNormal[u_, v_] = NormalVector[u, v] / NormalMag

7   SurfaceTension[nvec_] :=
      1 + gamma_111 * nvec[[1]]^2 nvec[[2]]^2 nvec[[3]]^2

8   << Graphics`ParametricPlot3D`

9   SphericalPlot3D[
      SurfaceTension[UnitNormal[u, v]] /. gamma_111 → 12,
      {u, 0, 2 Pi}, {v, -Pi/2, Pi/2}]

10  SphereEnergy = Integrate[
        Integrate[SurfaceTension[UnitNormal[u, v]] Cos[v],
          {u, 0, 2 π}], {v, -π/2, π/2}]

11  CubeSide = ∛(4 π / 3)

12  CubeEnergy = 6 (CubeSide^2 SurfaceTension[{1, 0, 0}])

13  EqualEnergies =
      Solve[CubeEnergy == SphereEnergy, gamma_111] // Flatten

14  N[gamma_111 /. EqualEnergies]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 16: Integral Theorems

Reading:
Kreyszig Sections: 10.8, 10.9 (pages463–467, 468–473)

Higher-dimensional Integrals

The fundamental theorem of calculus was generalized in a previous lecture from an integral over a single variable to an integration over a region in the plane. Specifically, for generalizing to Green's theorem in the plane, a vector derivative of a function integrated over a line and evaluated at its endpoints was generalized to a vector derivative of a function integrated over the plane.

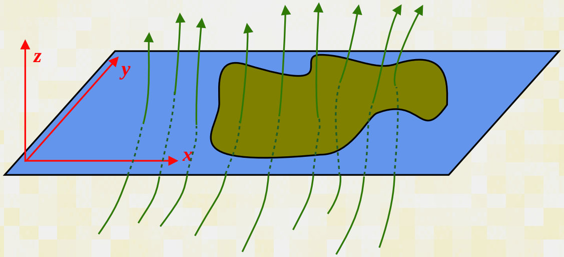


Figure 16-14: Illustrating how Green's theorem in the plane works. If a *known* vector function is integrated over a region in the plane then that integral should only depend on the bounding curve of that region.

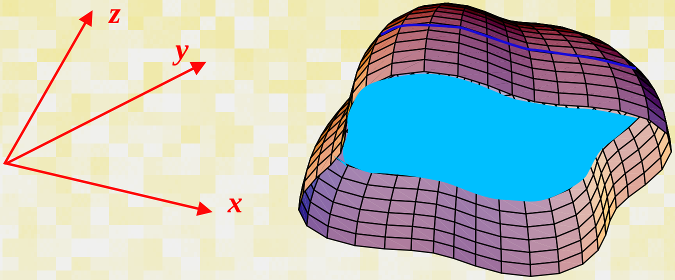


Figure 16-15: Illustration of a generalization to the Green's theorem in the plane: Suppose there is a bowl of a known shape submerged in a fluid with a trapped bubble. The bubble is bounded by two different surfaces, the bowl down to $z = 0$ and the planar liquid surface at that height. Integrating the function $\int_{V_B} dV$ over the bubble gives its volume. The volume must also be equal to an integral $\int\int_{\partial V_B} z dx dy$ over the (oriented) surface of the liquid. However, the volume of bubble can be determined from only the curve defined by the intersection of the bowl and the planar liquid surface; so the volume must also be equal to $\oint_C (\text{some function}) ds$.

The Divergence Theorem

Suppose there is "stuff" flowing from place to place in three dimensions.

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

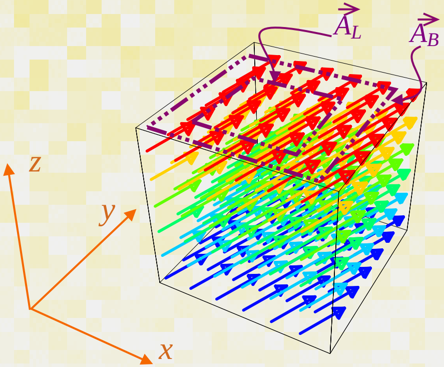


Figure 16-16: Illustration of a vector "flow field" $\vec{J}$ near a point in three dimensional space. If each vector represents the rate of "stuff" flowing per unit area of a plane that is normal to the direction of flow, then the dot product of the flow field integrated over a planar oriented area $\vec{A}$ is the rate of "stuff" flowing through that plane. For example, consider the two areas indicated with purple (or dashed) lines. The rate of "stuff" flowing through those regions is $\vec{J} \cdot \vec{A_B} = \vec{J} \cdot \hat{k} A_B$ and $\vec{J} \cdot \vec{A_L} = \vec{J} \cdot \hat{k} A_L$.

If there are no sources or sinks that create or destroy stuff inside a small box surrounding a point, then the change in the amount of stuff in the volume of the box must be related to some integral over the box's surface:

$$\begin{aligned}
\frac{d}{dt}(\text{amount of stuff in box}) &= \frac{d}{dt}\int_{\text{box}} (\frac{\text{amount of stuff}}{\text{volume}})dV \\
&= \int_{\text{box}} \frac{d}{dt}(\frac{\text{amount of stuff}}{\text{volume}})dV \\
&= \int_{\text{box}} (\text{some scalar function related to } \vec{J})dV \\
&= \int_{\substack{\text{box}\\ \text{surface}}} \vec{J} \cdot d\vec{A}
\end{aligned} \qquad (16\text{-}1)$$

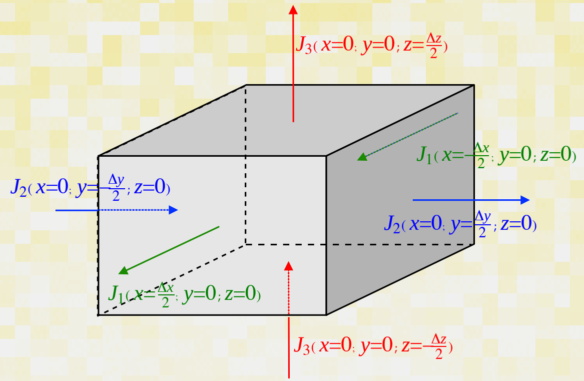

$J_3(x=0; y=0; z=\frac{\Delta z}{2})$
$J_1(x=-\frac{\Delta x}{2}; y=0; z=0)$
$J_2(x=0; y=-\frac{\Delta y}{2}; z=0)$
$J_2(x=0; y=\frac{\Delta y}{2}; z=0)$
$J_1(x=\frac{\Delta x}{2}; y=0; z=0)$
$J_3(x=0; y=0; z=-\frac{\Delta z}{2})$

Figure 16-17: Integration of a vector function near a point and its relation to the change in that vector function. The rate of change of stuff is the integral of flux over the outside—and in the limit as the box size goes to zero, the rate of change of the amount of stuff is related to the sum of derivatives of the flux components at that point.

To relate the rate at which "stuff M" is flowing into a small box of volume $\delta V = dxdydz$ located at (x, y, z) due to a flux $\vec{J}$, note that the amount that M changes in a time Δt is:

$$\begin{aligned}
\Delta M(\delta V) &= (M \text{ flowing out of } \delta V) - (M \text{ flowing in } \delta V) \\
&= \begin{array}{lll} & \vec{J}(x - \frac{dx}{2})\hat{i}dydz- & \vec{J}(x + \frac{dx}{2}) \cdot \hat{i}dydz \\ + & \vec{J}(y - \frac{dy}{2})\hat{j}dzdx- & \vec{J}(y + \frac{dy}{2}) \cdot \hat{j}dzdx \;\; \Delta t \\ + & \vec{J}(z - \frac{dz}{2})\hat{k}dxdy- & \vec{J}(z + \frac{dz}{2}) \cdot \hat{k}dxdy \end{array} \\
&= -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z})\delta V \Delta t + \mathcal{O}(dx^4)
\end{aligned} \tag{16-2}$$

If $C(x, y, z) = M(\delta V)/\delta V$ is the concentration (i.e., stuff per volume) at (x, y, z), then in the limit of small volumes and short times:

$$\frac{\partial C}{\partial t} = -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z}) = -\nabla \cdot \vec{J} = -\text{div}\vec{J} \tag{16-3}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

For an arbitrary closed volume V bounded by an oriented surface ∂V:

$$\frac{dM}{dt} = \frac{d}{dt}\int_V C dV = \int_V \frac{\partial C}{\partial t} dV = -\int_V \nabla \cdot \vec{J} dV = -\int_{\partial V} \vec{J} \cdot d\vec{A} \tag{16-4}$$

The last equality

$$\int_V \nabla \cdot \vec{J} dV = \int_{\partial V} \vec{J} \cdot d\vec{A} \tag{16-5}$$

is called the Gauss or the divergence theorem.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 16 MATHEMATICA® Example 1

London Dispersion Interaction between a point and Closed Volume

notebook (non-evaluated) pdf (evaluated) html (evaluated)

If the London interaction (i.e., energy between two induced dipoles) can be treated as a $1/r^6$ potential, then the potential due to a volume is an integration over each point in the volume and and arbitrary point in space. This calculation will be made much more efficient by turning the volume integral into a surface integral by using the divergence theorem.

1: To find a vector potential, $\vec{F}$ which has a divergence that is equal to $\nabla \cdot \vec{F} = -1/\|\vec{r} - \vec{x}\|^6$, *FVecLondon* is a 'guess.'

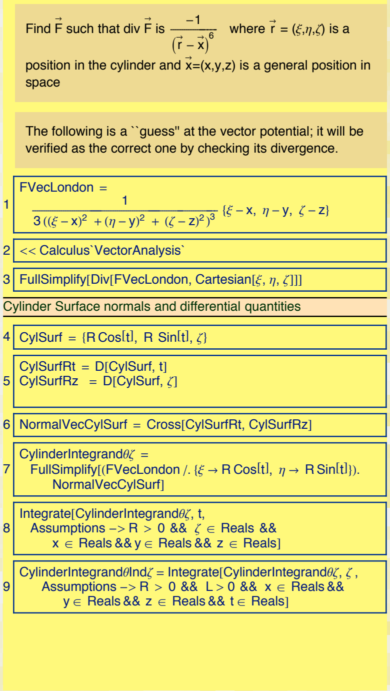
3: Using `Div` in the `Calculus‘VectorAnalysis‘` package, this will show that the guess *FVecLondon* is a correct vector function for the $1/r^6$ potential.

6: This will be the multiplier elemental area for a parameterized cylindrical surface $|d\vec{r}/d\theta \times d\vec{r}/dz|$.

7: *CylinderIntegrandθζ* is the integrand which would apply on a cylindrical surface.

8: This attempt to integrate *CylinderIntegrandθζ* over θ does not result in a closed form.

9: However, integrating *CylinderIntegrandθζ* over z does produce a closed form that could be subsequently integrated over θ numerically.



Find $\vec{F}$ such that div $\vec{F}$ is $\frac{-1}{(\vec{r}-\vec{x})^6}$ where $\vec{r} = (\xi,\eta,\zeta)$ is a position in the cylinder and $\vec{x}$=(x,y,z) is a general position in space

The following is a ``guess'' at the vector potential; it will be verified as the correct one by checking its divergence.

```
1  FVecLondon =
       1/(3 ((ξ - x)^2 + (η - y)^2 + (ζ - z)^2)^3) {ξ - x, η - y, ζ - z}

2  << Calculus`VectorAnalysis`

3  FullSimplify[Div[FVecLondon, Cartesian[ξ, η, ζ]]]
```

Cylinder Surface normals and differential quantities

```
4  CylSurf = {R Cos[t], R Sin[t], ζ}

5  CylSurfRt = D[CylSurf, t]
   CylSurfRz  = D[CylSurf, ζ]

6  NormalVecCylSurf = Cross[CylSurfRt, CylSurfRz]

7  CylinderIntegrandθζ =
     FullSimplify[(FVecLondon /. {ξ → R Cos[t], η → R Sin[t]}).
         NormalVecCylSurf]

8  Integrate[CylinderIntegrandθζ, t,
     Assumptions -> R > 0 && ζ ∈ Reals &&
         x ∈ Reals && y ∈ Reals && z ∈ Reals]

9  CylinderIntegrandθIndζ = Integrate[CylinderIntegrandθζ, ζ ,
       Assumptions -> R > 0 && L > 0 && x ∈ Reals &&
          y ∈ Reals && z ∈ Reals && t ∈ Reals]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Efficiency and Speed Issues: When to Evaluate the Right-Hand-Side of a Function in MATHEMATICA® .

The standard practice is to define functions in mathematica with `:=`. However, sometimes it makes sense to evaluate the right-hand-side when the function definition is made. These are the cases where the right hand side would take a long time to evaluate—each time the function is called, the evaluation would be needed again and again. The following example illustrates a case where it makes sense to use `Evaluate` in a function definition (or, equivalently defining the function with immediate assignment `=`).

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

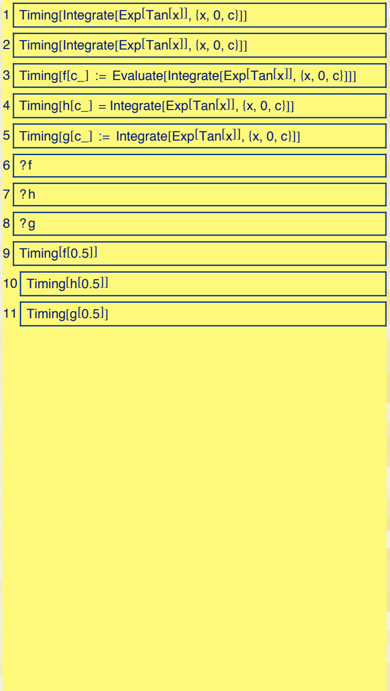
Quit

Lecture 16 MATHEMATICA® Example 2

To Evaluate or Not to Evaluate when Defining Functions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

This example illustrates a case in which immediate evaluation `=` would be preferable to delayed evaluation `:=`



```
1  Timing[Integrate[Exp[Tan[x]], {x, 0, c}]]
2  Timing[Integrate[Exp[Tan[x]], {x, 0, c}]]
3  Timing[f[c_] := Evaluate[Integrate[Exp[Tan[x]], {x, 0, c}]]]
4  Timing[h[c_] = Integrate[Exp[Tan[x]], {x, 0, c}]]
5  Timing[g[c_] := Integrate[Exp[Tan[x]], {x, 0, c}]]
6  ?f
7  ?h
8  ?g
9  Timing[f[0.5]]
10 Timing[h[0.5]]
11 Timing[g[0.5]]
```

1: When a non-trivial integral is done for the first time, Mathematica loads various libraries. Notice the difference in timing between this first computation of $\int \exp[\tan(x)]dx$ and the following one.

2: The second evaluation is faster. Now, a baseline time has been established for evaluating this integral symbolically.

3: Here, to make a function definition for the integral, the symbolic integral is obtained and so the function definition *takes longer*.

4: Using an `=` is roughly equivalent to using `Evaluate` above and the time to make the function assignment should be approximately the same.

5: Here, the symbolic integration is delayed until the function is called (later). Therefore, the function assignment is very rapid.

6: The functions, where the right-hand-side was immediately evaluated, contain the symbolic information. Therefore, when the function is called later, the symbolic integration will not be needed.

8: The function with the completely delayed assignment does not have the symbolic information.

10: The speed of the function is much faster in the case where the symbolic integration is not needed.

11: The relatively slow speed of this function indicates that it would be a poor choice when numerical efficiency is an issue.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 16 MATHEMATICA® Example 3

London Dispersion Potential of a Finite Cylinder

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The example of using the divergence theorem to compute a $1/r^6$ potential by pushing a volume integral onto its bounding surface is continued for the particular case of a cylinder.

1: Here, `Evaluate` is used to store the result of the `Simplify` function after the bounds of the integrated function are evaluated. The result is the integrand for the cylindrical part.

2: `NIntegrate` is required to do the remaining calculation, but instead integrating over three variables, the cylinder's contribution is reduced to a single integration. Because of polar symmetry, the problem is simplified by setting x to the total distance r and setting $y = 0$.

3: These are the surface differential quantities for the top surface of the cylinder.

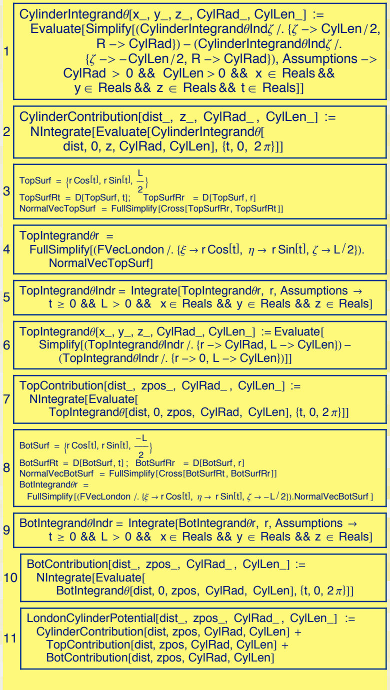
4: This is the integrand for the top surface.

5: The integral over θ does not return a closed form; so here the r-integral is performed explicitly.

6: `NIntegrate` is used to do the θ-integral and here a function is defined to give the contribution due to the top surface.

10: By direct analogy to the top surface, the contribution from the bottom surface is defined as a function.

11: The total potential is obtained by adding the contribution from the cylindrical side to the top and bottom surfaces' contributions.



```
1   CylinderIntegrandθ[x_, y_, z_, CylRad_, CylLen_] :=
      Evaluate[Simplify[(CylinderIntegrandθIndζ /. {ζ -> CylLen/2,
              R -> CylRad}) - (CylinderIntegrandθIndζ /.
              {ζ -> -CylLen/2, R -> CylRad}), Assumptions ->
          CylRad > 0 && CylLen > 0 && x ∈ Reals &&
          y ∈ Reals && z ∈ Reals && t ∈ Reals]]

2   CylinderContribution[dist_, z_, CylRad_, CylLen_] :=
      NIntegrate[Evaluate[CylinderIntegrandθ[
          dist, 0, z, CylRad, CylLen], {t, 0, 2π}]]

3   TopSurf = {r Cos[t], r Sin[t], L/2}
    TopSurfRt = D[TopSurf, t];   TopSurfRr = D[TopSurf, r]
    NormalVecTopSurf = FullSimplify[Cross[TopSurfRr, TopSurfRt]]

4   TopIntegrandθr =
      FullSimplify[(FVecLondon /. {ξ → r Cos[t], η → r Sin[t], ζ → L/2}).
        NormalVecTopSurf]

5   TopIntegrandθIndr = Integrate[TopIntegrandθr, r, Assumptions →
        t ≥ 0 && L > 0 && x ∈ Reals && y ∈ Reals && z ∈ Reals]

6   TopIntegrandθ[x_, y_, z_, CylRad_, CylLen_] := Evaluate[
       Simplify[(TopIntegrandθIndr /. {r -> CylRad, L -> CylLen}) -
         (TopIntegrandθIndr /. {r -> 0, L -> CylLen})]]

7   TopContribution[dist_, zpos_, CylRad_, CylLen_] :=
      NIntegrate[Evaluate[
        TopIntegrandθ[dist, 0, zpos, CylRad, CylLen], {t, 0, 2π}]]

8   BotSurf = {r Cos[t], r Sin[t], -L/2}
    BotSurfRt = D[BotSurf, t];  BotSurfRr = D[BotSurf, r]
    NormalVecBotSurf = FullSimplify[Cross[BotSurfRt, BotSurfRr]]
    BotIntegrandθr =
      FullSimplify[(FVecLondon /. {ξ → r Cos[t], η → r Sin[t], ζ → -L/2}).NormalVecBotSurf]

9   BotIntegrandθIndr = Integrate[BotIntegrandθr, r, Assumptions →
        t ≥ 0 && L > 0 && x ∈ Reals && y ∈ Reals && z ∈ Reals]

10  BotContribution[dist_, zpos_, CylRad_, CylLen_] :=
      NIntegrate[Evaluate[
        BotIntegrandθ[dist, 0, zpos, CylRad, CylLen], {t, 0, 2π}]]

11  LondonCylinderPotential[dist_, zpos_, CylRad_, CylLen_] :=
      CylinderContribution[dist, zpos, CylRad, CylLen] +
        TopContribution[dist, zpos, CylRad, CylLen] +
        BotContribution[dist, zpos, CylRad, CylLen]
```

3.016

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 16 MATHEMATICA® Example 4

Visualizing the London Potential of a Finite Cylinder

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The example is finished off by visualizing the results. Some of the numerical integrations are still poorly behaved in the vicinity of the cylinder's sharp edge.

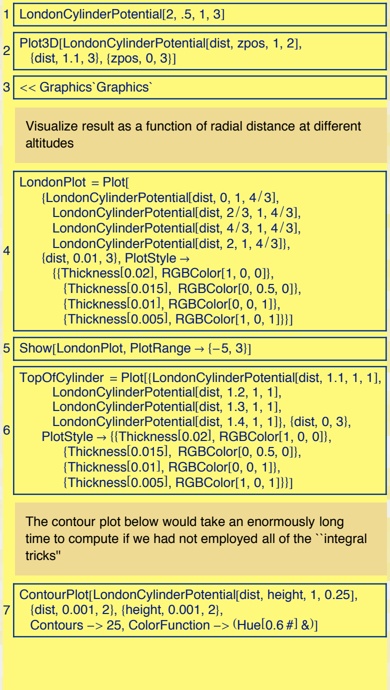
1: Demonstrating that the *LondonCylinderPotential* function, that was defined above, gives a numerical result.

2: Here, the potential is visualized with `Plot3D` outside the radius from the mid-plane to above the cylinder.

4: This is perhaps easier to interpret: the r-dependence is plotted at several different midplanes.

6: The same as the above, but for midplanes above the top of the cylinder.

7: `ContourPlot` probably gives the easiest visualization to interpret in this case.



```
1  LondonCylinderPotential[2, .5, 1, 3]

2  Plot3D[LondonCylinderPotential[dist, zpos, 1, 2],
     {dist, 1.1, 3}, {zpos, 0, 3}]

3  << Graphics`Graphics`
```

Visualize result as a function of radial distance at different altitudes

```
   LondonPlot = Plot[
       {LondonCylinderPotential[dist, 0, 1, 4/3],
         LondonCylinderPotential[dist, 2/3, 1, 4/3],
         LondonCylinderPotential[dist, 4/3, 1, 4/3],
4        LondonCylinderPotential[dist, 2, 1, 4/3]},
       {dist, 0.01, 3}, PlotStyle →
         {{Thickness[0.02], RGBColor[1, 0, 0]},
           {Thickness[0.015], RGBColor[0, 0.5, 0]},
           {Thickness[0.01], RGBColor[0, 0, 1]},
           {Thickness[0.005], RGBColor[1, 0, 1]}}]

5  Show[LondonPlot, PlotRange → {−5, 3}]

   TopOfCylinder = Plot[{LondonCylinderPotential[dist, 1.1, 1, 1],
         LondonCylinderPotential[dist, 1.2, 1, 1],
         LondonCylinderPotential[dist, 1.3, 1, 1],
6        LondonCylinderPotential[dist, 1.4, 1, 1]}, {dist, 0, 3},
       PlotStyle → {{Thickness[0.02], RGBColor[1, 0, 0]},
           {Thickness[0.015], RGBColor[0, 0.5, 0]},
           {Thickness[0.01], RGBColor[0, 0, 1]},
           {Thickness[0.005], RGBColor[1, 0, 1]}}]
```

The contour plot below would take an enormously long time to compute if we had not employed all of the ``integral tricks''

```
   ContourPlot[LondonCylinderPotential[dist, height, 1, 0.25],
7    {dist, 0.001, 2}, {height, 0.001, 2},
     Contours –> 25, ColorFunction –> (Hue[0.6 #] &)]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Stokes' Theorem

The final generalization of the fundamental theorem of calculus is the relation between a vector function integrated over an oriented surface and another vector function integrated over the closed curve that bounds the surface.

A simplified version of Stokes's theorem has already been discussed—Green's theorem in the plane can be written in full vector form:

$$\begin{aligned} \int\int_R \left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right) dxdy &= \int_R \nabla \times \vec{F} \cdot d\vec{A} \\ &= \oint_{\partial R} (F_1 dx + F_2 dy) = \oint_{\partial R} \vec{F} \cdot \frac{d\vec{r}}{ds} ds \end{aligned} \tag{16-6}$$

as long as the region R lies entirely in the z = constant plane.

In fact, Stokes's theorem is the same as the full vector form in Eq. 16-6 with R generalized to an oriented surface embedded in three-dimensional space:

$$\int_R \nabla \times \vec{F} \cdot d\vec{A} = \oint_{\partial R} \vec{F} \cdot \frac{d\vec{r}}{ds} ds \tag{16-7}$$

Plausibility for the theorem can be obtained from Figures 16-14 and 16-15. The curl of the vector field summed over a surface "spills out" from the surface by an amount equal to the vector field itself integrated over the boundary of the surface. In other words, if a vector field can be specified everywhere for a *fixed* surface, then its integral should only depend on some vector function integrated over the boundary of the surface.

Maxwell's equations

The divergence theorem and Stokes's theorem are generalizations of integration that invoke the divergence and curl operations on vectors. A familiar vector field is the electromagnetic field and Maxwell's equations depend on these vector derivatives as well:

$$\begin{aligned} \nabla \cdot \vec{B} = 0 \qquad & \qquad \nabla \times \vec{E} = \frac{\partial \vec{B}}{\partial t} \\ \nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{j} \qquad & \qquad \nabla \cdot \vec{D} = \rho \end{aligned} \tag{16-8}$$

in MKS units and the total electric displacement $\vec{D}$ is related to the total polarization $\vec{P}$ and the electric field $\vec{E}$ through:

$$\vec{D} = \vec{P} + \epsilon_o \vec{E} \tag{16-9}$$

where ϵ_o is the dielectric permittivity of vacuum. The total magnetic induction $\vec{B}$ is related to the induced magnetic field $\vec{H}$ and the material magnetization through

$$\vec{B} = \mu_o(\vec{H} + \vec{M}) \tag{16-10}$$

where μ_o is the magnetic permeability of vacuum.

Ampere's Law

Ampere's law that relates the magnetic field lines that surround a static current is a macroscopic version of the (static) Maxwell equation $\nabla \times \vec{H} = \vec{j}$:

Gauss' Law

Gauss' law relates the electric field lines that exit a closed surface to the total charge contained within the volume bounded by the surface. Gauss' law is a macroscopic version of the Maxwell equation $\nabla \cdot \vec{D} = \rho$:

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 17: Function Representation by Fourier Series

Reading:
Kreyszig Sections: 11.1, 11.2, 11.3 (pages478–485, 487–489, 490–495)

Periodic Functions

Periodic functions should be familiar to everyone. The keeping of time, the ebb and flow of tides, the patterns and textures of our buildings, decorations, and vestments invoke repetition and periodicity that seem to be inseparable from the elements of human cognition.[9] Although other species utilize music for purposes that we can only imagine—we seem to derive emotion and enjoyment from making and experience of music.

[9] I hope you enjoy the lyrical quality of the prose. While I wonder again if anyone is reading these notes, my wistfulness is taking a poetic turn:

> They repeat themselves
> What is here, will be there
> It wills, willing, to be again
> spring; neap, ebb and flow, wane; wax
> sow; reap, warp and woof, motif; melody.
> The changed changes. We remain
> Perpetually, Immutably, Endlessly.

3.016 Home

Full Screen

Close

Quit

Lecture 17 Mathematica® Example 1

Playing with Audible Periodic Phenomena

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Several example of creating sounds using mathematical functions are illustrated for education and amusement.

1: The seven musical notes around middle C indexed here with integers and then their frequencies (in hertz) are defined with a `freq`. The function *Note* takes one of the seven indexed notes and creates a wave-form for that note. The function `Play` takes the waveform and produces audio output.

2: To superimpose notes together to make a chord, it would be convenient to `Map` the function *Note* over a list...

3: The easiest way to exend a function so that it executes over a list is to use `SetAttributes` and declare the function to be `Listable`.

5: Like the function `Plot`, `Play` will frequently need `Evaluate` called on nontrivial arguments.

6: *Chord* make an ascending list of every second note and then uses `Mod` to map those notes onto the primary domain (0,1,...,6).

8: If different notes are wanted at different times, an `If` statement can be used.

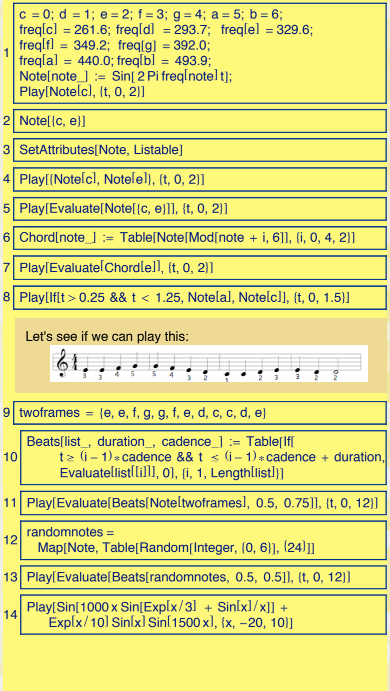
9: This is the sequence of notes associated with the displayed musical score.

10: *Beats* is a function that takes a list of notes and arranges them into a list where each member is an `If` statement stating *when* and for what *duration* it should play. In addition to the sequence of notes, the function takes two arguments, *cadence* and *duration* ¡ which specifiy how quickly and how long to sustain the notes.

11: This is musical score with notes played every 0.75 seconds and held for 0.5 second. Joy.

12: This is random "music." Oh boy.

13: This is noise generated from a function. Enjoy.



```
1  c = 0; d = 1; e = 2; f = 3; g = 4; a = 5; b = 6;
   freq[c] = 261.6; freq[d] = 293.7;  freq[e] = 329.6;
   freq[f] = 349.2;  freq[g] = 392.0;
   freq[a] = 440.0; freq[b] = 493.9;
   Note[note_] := Sin[2 Pi freq[note] t];
   Play[Note[c], {t, 0, 2}]

2  Note[{c, e}]

3  SetAttributes[Note, Listable]

4  Play[{Note[c], Note[e]}, {t, 0, 2}]

5  Play[Evaluate[Note[{c, e}]], {t, 0, 2}]

6  Chord[note_] := Table[Note[Mod[note + i, 6]], {i, 0, 4, 2}]

7  Play[Evaluate[Chord[e]], {t, 0, 2}]

8  Play[If[t > 0.25 && t < 1.25, Note[a], Note[c]], {t, 0, 1.5}]
```

Let's see if we can play this:

```
9  twoframes = {e, e, f, g, g, f, e, d, c, c, d, e}

10 Beats[list_, duration_, cadence_] := Table[If[
         t ≥ (i – 1)∗cadence && t ≤ (i – 1)∗cadence + duration,
         Evaluate[list[[i]]], 0], {i, 1, Length[list]}]

11 Play[Evaluate[Beats[Note[twoframes], 0.5, 0.75]], {t, 0, 12}]

12 randomnotes =
     Map[Note, Table[Random[Integer, {0, 6}], {24}]]

13 Play[Evaluate[Beats[randomnotes, 0.5, 0.5]], {t, 0, 12}]

14 Play[Sin[1000 x Sin[Exp[x / 3] + Sin[x] / x]] +
       Exp[x / 10] Sin[x] Sin[1500 x], {x, –20, 10}]
```

3.016 Home

Full Screen

Close

Quit

A function that is periodic in a single variable can be expressed as:

$$\begin{aligned} f(x+\lambda) &= f(x) \\ f(t+\tau) &= f(t) \end{aligned} \tag{17-1}$$

The first form is a suggestion of a spatially periodic function with wavelength λ and the second form suggests a function that is periodic in time with period τ. Of course, both forms are identical and express that the function has the same value at an infinite number of points ($x = n\lambda$ in space or $t = n\tau$ in time where n is an integer.)

Specification of a periodic function, $f(x)$, within one period $x \in (x_o, x_o + \lambda)$ defines the function everywhere. The most familiar periodic functions are the trigonometric functions:

$$\sin(x) = \sin(x + 2\pi) \quad \text{and} \quad \cos(x) = \cos(x + 2\pi) \tag{17-2}$$

However, any function can be turned into a periodic function.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 17 MATHEMATICA® Example 2

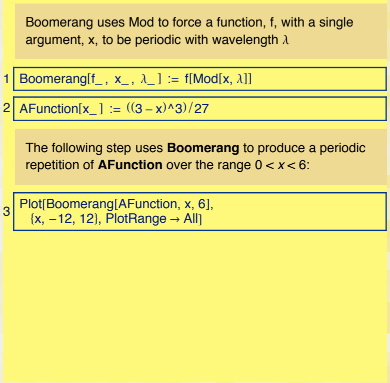
Using "Mod" to Create Periodic Functions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Periodic functions are often associated with the "modulus" operation. `Mod`$[x, \lambda]$ is the remainder of the result of *recursively* dividing x by λ until the result lies in the domain $0 \leq \mathrm{Mod}[x, \lambda] < \lambda$). Another way to think of modulus is to find the "point" where are periodic function should be evaluated if its primary domain is $x \in (0, \lambda)$.

1: *Boomerang* uses `Mod` on the argument of any function `f` of a single argument to map the argument into the domain $(0, \lambda)$. Therefore, calling *Boomerang* on any function will create a infinitely periodic repetition of the function in the domain $(0, \lambda)$.

3: `Plot` called on the periodic extension of wavelength $\lambda = 6$ of a function illustrates the effect of *Boomerang* . a periodic function with a specified period.



Boomerang uses Mod to force a function, f, with a single argument, x, to be periodic with wavelength λ

```
1 Boomerang[f_ , x_ , λ_ ] := f[Mod[x, λ]]
```

```
2 AFunction[x_ ] := ((3 – x)^3)/27
```

The following step uses **Boomerang** to produce a periodic repetition of **AFunction** over the range 0 < *x* < 6:

```
3 Plot[Boomerang[AFunction, x, 6],
    {x, –12, 12}, PlotRange → All]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Odd and Even Functions

The trigonometric functions have the additional properties of being an *odd* function about the point $x = 0$: $f_{\text{odd}} : f_{\text{odd}}(x) = -f_{\text{odd}}(-x)$ in the case of the sine, and an *even* function in the case of the cosine: $f_{\text{even}} : f_{\text{even}}(x) = f_{\text{even}}(-x)$.

This can generalized to say that a function is even or odd about a point $\lambda/2$: $f_{\text{odd}\frac{\lambda}{2}} : f_{\text{odd}\frac{\lambda}{2}}(\lambda/2+x) = -f_{\text{odd}\frac{\lambda}{2}}(\lambda/2 - x)$ and $f_{\text{even}\frac{\lambda}{2}} : f_{\text{even}\frac{\lambda}{2}}(\lambda/2 + x) = f_{\text{even}\frac{\lambda}{2}}(\lambda/2 - x)$.

Any function can be decomposed into an odd and even sum:

$$g(x) = g_{\text{even}} + g_{\text{odd}} \tag{17-3}$$

The sine and cosine functions can be considered the odd and even parts of the generalized trigonometric function:

$$e^{ix} = \cos(x) + \imath\sin(x) \tag{17-4}$$

with period 2π.

Representing a particular function with a sum of other functions

A Taylor expansion approximates the behavior of a suitably defined function, $f(x)$ in the neighborhood of a point, x_o, with a bunch of functions, $p_i(x)$, defined by the set of powers:

$$p_i \equiv \vec{p} = (x^0, x^1, \ldots, x^j, \ldots) \tag{17-5}$$

The polynomial that approximates the function is given by:

$$f(x) = \vec{A} \cdot \vec{p} \tag{17-6}$$

where the vector of coefficients is defined by:

$$A_i \equiv \vec{A} = (\frac{1}{0!}f(x_o), \frac{1}{1!}\left.\frac{df}{dx}\right|_{x_o}, \ldots, \frac{1}{j!}\left.\frac{d^jf}{dx^j}\right|_{x_o}, \ldots) \tag{17-7}$$

The idea of a vector of infinite length has not been formally introduced, but the idea that as the number of terms in the sum in Eq. 17-6 gets larger and larger, the approximation should converge to

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

the function. In the limit of an infinite number of terms in the sum (or the vectors of infinite length) the series expansion will converge to $f(x)$ if it satisfies some technical continuity constraints.

However, for periodic functions, the domain over which the approximation is required is only one period of the periodic function—the rest of the function is taken care of by the definition of periodicity in the function.

Because the function is periodic, it makes sense to use functions that have the same period to approximate it. The simplest periodic functions are the trigonometric functions. If the period is λ, any other periodic function with periods $\lambda/2$, $\lambda/3$, λ/N, will also have period λ. Using these "sub-periodic" trigonometric functions is the idea behind Fourier Series.

Fourier Series

The functions $\cos(2\pi x/\lambda)$ and $\sin(2\pi x/\lambda)$ each have period λ. That is, they each take on the same value at x and $x + \lambda$.

There are an infinite number of other simple trigonometric functions that are periodic in λ; they are $\cos[2\pi x/(\lambda/2))]$ and $\sin[2\pi x/(\lambda/2))]$ and which cycle two times within each λ, $\cos[2\pi x/(\lambda/3))]$ and $\sin[2\pi x/(\lambda/3))]$ and which cycle three times within each λ, and, in general, $\cos[2\pi x/(\lambda/n))]$ and $\sin[2\pi x/(\lambda/n))]$ and which cycle n times within each λ.

The constant function, $a_0(x) = \text{const}$, also satisfies the periodicity requirement.

The superposition of multiples of any number of periodic function must also be a periodic function, therefore any function $f(x)$ that satisfies:

$$\begin{aligned} f(x) &= \mathcal{E}_0 + \sum_{n=1}^{\infty} \mathcal{E}_n \cos\left(\frac{2\pi n}{\lambda} x\right) + \sum_{n=1}^{\infty} \mathcal{O}_n \sin\left(\frac{2\pi n}{\lambda} x\right) \\ &= \mathcal{E}_{k_0} + \sum_{n=1}^{\infty} \mathcal{E}_{k_n} \cos(k_n x) + \sum_{n=1}^{\infty} \mathcal{O}_{k_n} \sin(k_n x) \end{aligned} \tag{17-8}$$

where the k_i are the *wave-numbers* or *reciprocal wavelengths* defined by $k_j \equiv 2\pi j/\lambda$. The k's represent inverse wavelengths—large values of k represent short-period or high-frequency terms.

If any periodic function $f(x)$ could be represented by the series in in Eq. 17-8 by a suitable choice of coefficients, then an alternative representation of the periodic function could be obtained in terms of the simple trigonometric functions and their amplitudes.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The "inverse question" remains: "How are the amplitudes $\mathcal{E}_{k_n}$ (the even trigonometric terms) and $\mathcal{O}_{k_n}$ (the odd trigonometric terms) determined for a given $f(x)$?"

The method follows from what appears to be a "trick." The following three integrals have simple forms for integers M and N:

$$
\begin{aligned}
\int_{x_0}^{x_0+\lambda} \sin\left(\frac{2\pi M}{\lambda}x\right) \sin\left(\frac{2\pi N}{\lambda}x\right) dx &= \begin{cases} \frac{\lambda}{2} \text{ if } M = N \\ 0 \text{ if } M \neq N \end{cases} \\
\int_{x_0}^{x_0+\lambda} \cos\left(\frac{2\pi M}{\lambda}x\right) \cos\left(\frac{2\pi N}{\lambda}x\right) dx &= \begin{cases} \frac{\lambda}{2} \text{ if } M = N \\ 0 \text{ if } M \neq N \end{cases} \\
\int_{x_0}^{x_0+\lambda} \cos\left(\frac{2\pi M}{\lambda}x\right) \sin\left(\frac{2\pi N}{\lambda}x\right) dx &= 0 \text{ for any integers } M, N
\end{aligned}
\tag{17-9}
$$

The following shows a demonstration of this *orthogonality relation for the trignometric functions.*

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 17 MATHEMATICA® Example 3

Orthogonality of Trignometric Functions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Demonstrating that the relations in Eq. 17-9 are true.

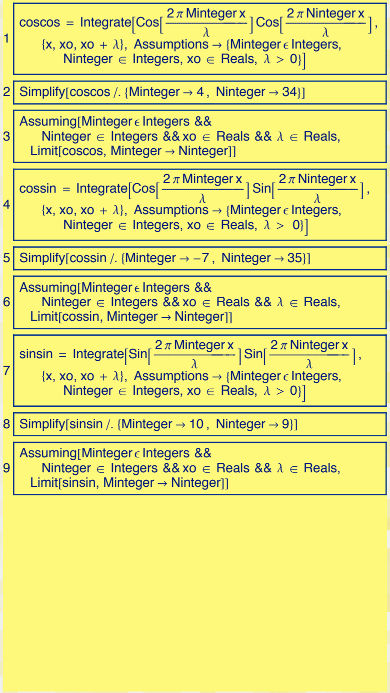
1: Using `Integrate` for $\cos(2\pi Mx/\lambda)\cos(2\pi Nx/\lambda)$ over a definite interval of a single wavelength, does not produce a result that obviously vanishes for $M \neq N$.

2: However, replacing any of the symbolic integers with actual integers results in a zero. So, one the orthogonality relation is plausible.

3: Using `Assuming` and `Limit`, one can show that the relation ship vanishes for $N = M$. Although, it is a bit odd to be thinking about continuous limits with integers.

5: Similarly for $\int \cos(2\pi Mx/\lambda)\sin(2\pi Nx/\lambda)dx$.

9: and for $\int \sin(2\pi Mx/\lambda)\sin(2\pi Nx/\lambda)dx$.



```
1  coscos = Integrate[Cos[(2 π Minteger x)/λ] Cos[(2 π Ninteger x)/λ],
       {x, xo, xo + λ}, Assumptions → {Minteger ∈ Integers,
           Ninteger ∈ Integers, xo ∈ Reals, λ > 0}]
2  Simplify[coscos /. {Minteger → 4 , Ninteger → 34}]
3  Assuming[Minteger ∈ Integers &&
       Ninteger ∈ Integers && xo ∈ Reals && λ ∈ Reals,
     Limit[coscos, Minteger → Ninteger]]
4  cossin = Integrate[Cos[(2 π Minteger x)/λ] Sin[(2 π Ninteger x)/λ],
       {x, xo, xo + λ}, Assumptions → {Minteger ∈ Integers,
           Ninteger ∈ Integers, xo ∈ Reals, λ > 0}]
5  Simplify[cossin /. {Minteger → −7 , Ninteger → 35}]
6  Assuming[Minteger ∈ Integers &&
       Ninteger ∈ Integers && xo ∈ Reals && λ ∈ Reals,
     Limit[cossin, Minteger → Ninteger]]
7  sinsin = Integrate[Sin[(2 π Minteger x)/λ] Sin[(2 π Ninteger x)/λ],
       {x, xo, xo + λ}, Assumptions → {Minteger ∈ Integers,
           Ninteger ∈ Integers, xo ∈ Reals, λ > 0}]
8  Simplify[sinsin /. {Minteger → 10 , Ninteger → 9}]
9  Assuming[Minteger ∈ Integers &&
       Ninteger ∈ Integers && xo ∈ Reals && λ ∈ Reals,
     Limit[sinsin, Minteger → Ninteger]]
```

3.016 Home

Full Screen

Close

Quit

Using this orthogonality trick, any amplitude can be determined by multiplying both sides of Eq. 17-8 by its conjugate trigonometric function and integrating over the domain. (Here we pick the domain to start at zero, $x \in (0, \lambda)$, but any other starting point would work fine.)

$$\begin{aligned}
\cos(k_M x) f(x) &= \cos(k_M x) \left(\mathcal{E}_{k_0} + \sum_{n=1}^{\infty} \mathcal{E}_{k_n} \cos(k_n x) + \sum_{n=1}^{\infty} \mathcal{O}_{k_n} \sin(k_n x) \right) \\
\int_0^{\lambda} \cos(k_M x) f(x) dx &= \int_0^{\lambda} \cos(k_M x) \left(\mathcal{E}_{k_0} + \sum_{n=1}^{\infty} \mathcal{E}_{k_n} \cos(k_n x) + \sum_{n=1}^{\infty} \mathcal{O}_{k_n} \sin(k_n x) \right) dx \qquad (17\text{-}10) \\
\int_0^{\lambda} \cos(k_M x) f(x) dx &= \frac{\lambda}{2} \mathcal{E}_{k_M}
\end{aligned}$$

This provides a formula to calculate the even coefficients (amplitudes) and multiplying by a sin function provides a way to calculate the odd coefficients (amplitudes) for $f(x)$ periodic in the fundamental domain $x \in (0, \lambda)$.

$$\begin{aligned}
\mathcal{E}_{k_0} &= \frac{1}{\lambda} \int_0^{\lambda} f(x) dx \\
\mathcal{E}_{k_N} &= \frac{2}{\lambda} \int_0^{\lambda} f(x) \cos(k_N x) dx \qquad k_N \equiv \frac{2\pi N}{\lambda} \qquad (17\text{-}11) \\
\mathcal{O}_{k_N} &= \frac{2}{\lambda} \int_0^{\lambda} f(x) \sin(k_N x) dx \qquad k_N \equiv \frac{2\pi N}{\lambda}
\end{aligned}$$

The constant term has an extra factor of two because $\int_0^{\lambda} \mathcal{E}_{k_0} dx = \lambda \mathcal{E}_{k_0}$ instead of the $\lambda/2$ found in Eq. 17-9.

Other forms of the Fourier coefficients

Sometimes the primary domain is defined with a different starting point and different symbols, for instance Kreyszig uses a centered domain by using $-L$ as the starting point and $2L$ as the period, and in these cases the forms for the Fourier coefficients look a bit different. One needs to look at the domain in order to determine which form of the formulas to use.

Extra Information and Notes

Potentially interesting but currently unnecessary

The "trick" of multiplying both sides of Eq. 17-8 by a function and integrating comes from the fact that the trigonometric functions form an orthogonal basis *for functions with inner product defined by*

$$f(x) \cdot g(x) = \int_0^{\lambda} f(x)g(x)dx$$

Considering the trigonometric functions as components of a vector:

$$\begin{aligned}
\vec{e_0}(x) =& (1, 0, 0, \ldots,) \\
\vec{e_1}(x) =& (0, \cos(k_1 x), 0, \ldots,) \\
\vec{e_2}(x) =& (0, 0, \sin(k_1 x), \ldots,) \\
\ldots =& \qquad \vdots \\
\vec{e_n}(x) =& (\ldots\ldots, \sin(k_n x), \ldots,)
\end{aligned}$$

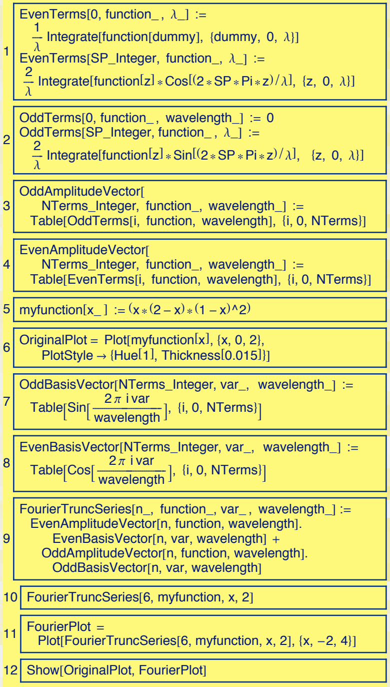
then these "basis vectors" satisfy $\vec{e_i} \cdot \vec{e_j} = (\lambda/2)\delta_{ij}$, *where* $\delta_{ij} = 0$ *unless* $i = j$. *The trick is just that, for an arbitrary function represented by the basis vectors,* $\vec{P}(x) \cdot \vec{e_j}(x) = (\lambda/2)P_j$.

Lecture 17 Mathematica® Example 4

Calculating Fourier Series Amplitudes

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Functions are developed which compute the even (cosine) amplitudes and odd (sine) amplitudes for an input function of one variable. These functions are extended to produce the first N terms of a Fouriers series.



```
1   EvenTerms[0, function_ , λ_] :=
      1/λ Integrate[function[dummy], {dummy, 0, λ}]
    EvenTerms[SP_Integer, function_, λ_] :=
     2/λ Integrate[function[z] * Cos[(2 * SP * Pi * z)/λ], {z, 0, λ}]

2   OddTerms[0, function_ , wavelength_] := 0
    OddTerms[SP_Integer, function_ , λ_] :=
      2/λ Integrate[function[z] * Sin[(2 * SP * Pi * z)/λ],  {z, 0, λ}]

3   OddAmplitudeVector[
        NTerms_Integer, function_, wavelength_] :=
      Table[OddTerms[i, function, wavelength], {i, 0, NTerms}]

4   EvenAmplitudeVector[
        NTerms_Integer, function_, wavelength_] :=
      Table[EvenTerms[i, function, wavelength], {i, 0, NTerms}]

5   myfunction[x_ ] := (x * (2 – x) * (1 – x)^2)

6   OriginalPlot = Plot[myfunction[x], {x, 0, 2},
        PlotStyle → {Hue[1], Thickness[0.015]}]

7   OddBasisVector[NTerms_Integer, var_,  wavelength_] :=
      Table[Sin[2 π i var/wavelength], {i, 0, NTerms}]

8   EvenBasisVector[NTerms_Integer, var_,  wavelength_] :=
      Table[Cos[2 π i var/wavelength], {i, 0, NTerms}]

9   FourierTruncSeries[n_, function_, var_ , wavelength_] :=
      EvenAmplitudeVector[n, function, wavelength].
          EvenBasisVector[n, var, wavelength] +
        OddAmplitudeVector[n, function, wavelength].
          OddBasisVector[n, var, wavelength]

10   FourierTruncSeries[6, myfunction, x, 2]

11   FourierPlot =
       Plot[FourierTruncSeries[6, myfunction, x, 2], {x, –2, 4}]

12   Show[OriginalPlot, FourierPlot]
```

1: *EvenTerms* computes symbolic representations of the even (cosine) coefficients using the formulas in Eq. 17-11. The $N = 0$ term is computed with a supplemental defintion because of its extra factor of 2. The domain is chosen so that it begins at $x = 0$ and ends at $x = \lambda$.

2: *OddTerms* performs a similar computation for the sine-coefficients; the $N = 0$ amplitude is set to zero explicitely. It will become convenient to include the zeroth-order coefficient for the odd (sine) series which vanishes by definition. The functions work by doing an integral for each term—this is not very efficient. It would be more efficient to calculate the integral symbolically once and then evaluate it once for each term.

4: efOddAmplitudeVector and *EvenAmplitudeVectors* create *amplitude vectors* for the cosine and sine terms with specified lengths and domains.

5: This function, $f(x) = x(1-x)^2(2-x)$, will be used for particular examples of Fourier series, note that it is an even function over $0 < x < 2\ldots$

7: The functions, *OddBasisVector* and *EvenBasisVector* , create vectors of *basis functions* of specified lengths and perioidic domains.

8: The Fourier series up to a certain order can be defined as the sum of two inner (dot) products: the inner product of the odd coefficient vector and the sine basis vector, and the inner product of the even coefficient vector and the cosine basis vector.

12: This will illustrate the approximation for a truncated ($N = 6$) Fourier series

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

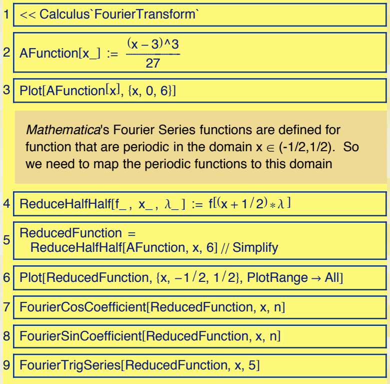
Quit

Lecture 17 MATHEMATICA® Example 5

Using the `Calculus‘FourierTransform‘` package

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Fourier series expansions are a common and useful mathematical tool, and it is not surprising that MATHEMATICA® would have a package to do this and replace the inefficient functions defined in the previous example.



```
1  << Calculus`FourierTransform`

                     (x - 3)^3
2  AFunction[x_] := ---------
                        27

3  Plot[AFunction[x], {x, 0, 6}]
```

Mathematica's Fourier Series functions are defined for function that are periodic in the domain x ∈ (-1/2,1/2). So we need to map the periodic functions to this domain

```
4  ReduceHalfHalf[f_ , x_ , λ_ ] := f[(x + 1/2) * λ ]

5  ReducedFunction =
     ReduceHalfHalf[AFunction, x, 6] // Simplify

6  Plot[ReducedFunction, {x, -1/2, 1/2}, PlotRange → All]

7  FourierCosCoefficient[ReducedFunction, x, n]

8  FourierSinCoefficient[ReducedFunction, x, n]

9  FourierTrigSeries[ReducedFunction, x, 5]
```

1: The functions in `Calculus‘FourierTransform‘` are designed to operate on the unit period located at $x \in (-1/2, 1/2)$. Therefore, the domains of functions of interest must be mapped onto this domain by a change of variables.

4: *ReduceHalfHalf* is an example of a function design to do the required mapping. First the length of original domain is mapped to unity by dividing through by λ and then the origin is shifted by mapping the x (that the MATHEMATICA® functions will see) to $(0, 1)$ with the transformation $x \rightarrow x + \frac{1}{2}$.

8: Particular amplitudes of the properly remapped function can be obtained with the functions `FourierCosCoefficient` and `FourierSinCoefficient`. In this example, a symbolic n is entered and a symbolic representation of the n^{th} amplitude is returned.

9: A truncated Fourier series can be obtained symbolically to any order with `FourierTrigSeries`.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 17 MATHEMATICA® Example 6

Visualizing Convergence of the Fourier Series: Gibbs Phenomenon

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Functions that produce animations (each frame representing a different order of truncation of the Fourier series) are developed. This example illustrates *Gibbs phenomenon* where the approximating function oscillates wildly near discontinuities in the original function.

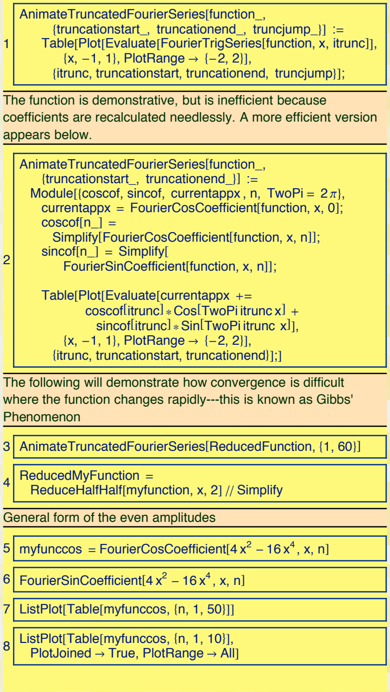
1: *AnimateTruncatedFourierSeries* is a simple example of an animation function for the truncated Fourier series. It uses the `Table` function with three arguments in the iterator for the initial truncation `truncationstart`, final truncation, and the number to skip in between. . . .

2: However, because the entire series is recomputed for each frame, the function above is not very efficient. In this second version, only two arguments are supplied to the iterator. At each frame, the two N^{th} Fourier terms are added to the sum of terms computed previously.

3: Because *ReducedFunction* has a discontinuity (its end-value and intitial value differ), this animation will show Gibbs phenomena near the edges of the domain.

5: `FourierCosCoefficient` will show a frequently observed feature in the amplitudes of Fourier coefficients. The amplitude's magnitude becomes smaller and smaller with larger n, and the sign of the terms tend to oscillate between positive and negative values.

8: Plots of the magnitudes of the amplitudes are a "signature" of the original function in a new space. Each term indicates what "amount" of each periodicity is present in the original function. The plot could be interpreted as a "frequency" or "wavelength" representation of the original function.



```
1  AnimateTruncatedFourierSeries[function_,
       {truncationstart_, truncationend_, truncjump_}] :=
     Table[Plot[Evaluate[FourierTrigSeries[function, x, itrunc]],
         {x, –1, 1}, PlotRange → {–2, 2}],
       {itrunc, truncationstart, truncationend, truncjump}];
```

The function is demonstrative, but is inefficient because coefficients are recalculated needlessly. A more efficient version appears below.

```
2  AnimateTruncatedFourierSeries[function_,
     {truncationstart_, truncationend_}] :=
    Module[{coscof, sincof, currentappx , n, TwoPi = 2π},
     currentappx = FourierCosCoefficient[function, x, 0];
     coscof[n_] =
       Simplify[FourierCosCoefficient[function, x, n]];
     sincof[n_] = Simplify[
         FourierSinCoefficient[function, x, n]];

     Table[Plot[Evaluate[currentappx +=
             coscof[itrunc] * Cos[TwoPi itrunc x] +
              sincof[itrunc] * Sin[TwoPi itrunc x]],
         {x, –1, 1}, PlotRange → {–2, 2}],
       {itrunc, truncationstart, truncationend}];]
```

The following will demonstrate how convergence is difficult where the function changes rapidly---this is known as Gibbs' Phenomenon

```
3  AnimateTruncatedFourierSeries[ReducedFunction, {1, 60}]

4  ReducedMyFunction =
     ReduceHalfHalf[myfunction, x, 2] // Simplify
```

General form of the even amplitudes

```
5  myfunccos = FourierCosCoefficient[4 x^2 – 16 x^4 , x, n]

6  FourierSinCoefficient[4 x^2 – 16 x^4 , x, n]

7  ListPlot[Table[myfunccos, {n, 1, 50}]]

8  ListPlot[Table[myfunccos, {n, 1, 10}],
     PlotJoined → True, PlotRange → All]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Complex Form of the Fourier Series

The behavior of the Fourier coefficients for both the odd (sine) and for the even (cosine) terms was illustrated above. Functions that are even about the center of the fundamental domain (reflection symmetry) will have only even terms—all the sine terms will vanish. Functions that are odd about the center of the fundamental domain (reflections across the center of the domain and then across the x-axis.) will have only odd terms—all the cosine terms will vanish.

Functions with no odd or even symmetry will have both types of terms (odd and even) in its expansion. This is a restatement of the fact that any function can be decomposed into odd and even parts (see Eq. 17-3).

This suggests a short-hand in Eq. 17-4 can be used that combines both odd and even series into one single form. However, because the odd terms will all be multiplied by the imaginary number $\imath$, the coefficients will generally be complex. Also because $\cos(nx) = (\exp(inx) + \exp(-inx))/2$, writing the sum in terms of exponential functions only will require that the sum must be over both positive and negative integers.

For a periodic domain $x \in (0, \lambda)$, $f(x) = f(x + \lambda)$, the *complex form of the fourier series is given by:*

$$
\begin{aligned}
f(x) &= \sum_{n=-\infty}^{\infty} \mathcal{C}_{k_n} e^{\imath k_n x} \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\
\mathcal{C}_{k_n} &= \frac{1}{\lambda} \int_0^{\lambda} f(x) e^{-\imath k_n x} dx
\end{aligned}
\tag{17-12}
$$

Because of the orthogonality of the basis functions $\exp(\imath k_n x)$, the domain can be moved to any wavelength, the following is also true:

$$
\begin{aligned}
f(x) &= \sum_{n=-\infty}^{\infty} \mathcal{C}_{k_n} e^{\imath k_n x} \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\
\mathcal{C}_{k_n} &= \frac{1}{\lambda} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx
\end{aligned}
\tag{17-13}
$$

although the coefficients may have a different form.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 18: The Fourier Transform and its Interpretations

Reading:
Kreyszig Sections: 11.4, 11.7, 11.8, 11.9 (pages496–498, 506–512 513–517, 518–523)

Fourier Transforms

Expansion of a function in terms of Fourier Series proved to be an effective way to represent functions that were periodic in an interval $x \in (-\lambda/2, -\lambda/2)$. Useful insights into "what makes up a function" are obtained by considering the amplitudes of the harmonics (i.e., each of the sub-periodic trigonometric or complex oscillatory functions) that compose the Fourier series. That is, the component harmonics can be quantified by inspecting their amplitudes. For instance, one could quantitatively compare the same note generated from a Stradivarius to an ordinary violin by comparing the amplitudes of the Fourier components of the notes component frequencies.

However there are many physical examples of phenomena that involve nearly, but not completely, periodic phenomena—and of course, quantum mechanics provides many examples of isolated events that are composed of wave-like functions.

It proves to be very useful to extend the Fourier analysis to functions that are not periodic. Not only are the same interpretations of contributions of the elementary functions that compose a more complicated object available, but there are many others to be obtained.

For example:

momentum/position The wavenumber $k_n = 2\pi n/\lambda$ turns out to be proportional to the momentum in quantum mechanics. The position of a function, $f(x)$, can be expanded in terms of a series of wave-like functions with amplitudes that depend on each component momentum—this is the essence of the Heisenberg uncertainty principle.

diffraction Bragg's law, which formulates the conditions of constructive and destructive interference of photons diffracting off of a set of atoms, is much easier to derive using a Fourier representation of the atom positions and photons.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

To extend Fourier series to non-periodic functions, the domain of periodicity will extended to infinity, that is the limit of $\lambda \rightarrow \infty$ will be considered. This extension will be worked out in a heuristic manner in this lecture—the formulas will be correct, but the rigorous details are left for the math textbooks.

Recall that the complex form of the Fourier series was written as:

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} \mathcal{A}_{k_n} e^{\imath k_n x} \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\ \mathcal{A}_{k_n} &= \frac{1}{\lambda} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx \end{aligned} \tag{18-1}$$

where $\mathcal{A}_{k_n}$ is the complex amplitude associated with the $k_n = 2\pi n/\lambda$ reciprocal wavelength or wavenumber.

This can be written in a more symmetric form by scaling the amplitudes with λ—let $\mathcal{A}_{k_n} = \sqrt{2\pi}\mathcal{C}_{k_n}/\lambda$, then

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} \frac{\sqrt{2\pi}\mathcal{C}_{k_n}}{\lambda} e^{\imath k_n x} \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\ \mathcal{C}_{k_n} &= \frac{1}{\sqrt{2\pi}} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx \end{aligned} \tag{18-2}$$

Considering the first sum, note that the difference in wave-numbers can be written as:

$$\Delta k = k_{n+1} - k_n = \frac{2\pi}{\lambda} \tag{18-3}$$

which will become infinitesimal in the limit as $\lambda \rightarrow \infty$. Substituting $\Delta k/(2\pi)$ for $1/\lambda$ in the sum, the more "symmetric result" appears,

$$\begin{aligned} f(x) &= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \mathcal{C}_{k_n} e^{\imath k_n x} \Delta k \qquad \text{where } k_n \equiv \frac{2\pi n}{\lambda} \\ \mathcal{C}_{k_n} &= \frac{1}{\sqrt{2\pi}} \int_{-\lambda/2}^{\lambda/2} f(x) e^{-\imath k_n x} dx \end{aligned} \tag{18-4}$$

Now, the limit $\lambda \rightarrow \infty$ can be obtained an the summation becomes an integral over a continuous spectrum of wave-numbers; the amplitudes become a continuous function of wave-numbers, $\mathcal{C}_{k_n} \rightarrow g(k)$:

$$\begin{aligned} f(x) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(k) e^{\imath k x} dk \\ g(k) =& \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-\imath k x} dx \end{aligned} \qquad (18\text{-}5)$$

The function $g(k = 2\pi/\lambda)$ represents the density of the amplitudes of the periodic functions that make up $f(x)$. The function $g(k)$ is called *the Fourier Transform* of $f(x)$. The function $f(x)$ is called *the Inverse Fourier Transform* of $g(k)$, and $f(x)$ and $g(k)$ are a *the Fourier Transform Pair.*

Higher Dimensional Fourier Transforms

Of course, many interesting periodic phenomena occur in two dimensions (e.g., two spatial dimensions, or one spatial plus one temporal), three dimensions (e.g., three spatial dimensions or two spatial plus one temporal), or more.

The Fourier transform that integrates $\frac{dx}{\sqrt{2\pi}}$ over all x can be extended straightforwardly to a two dimensional integral of a function $f(\vec{r}) = f(x, y)$ by $\frac{dxdy}{2\pi}$ over all x and y—or to a three-dimensional integral of $f(\vec{r})\frac{dxdydz}{\sqrt{(2\pi)^3}}$ over an infinite three-dimensional volume.

A wavenumber appears for each new spatial direction and they represent the periodicities in the x-, y-, and z-directions. It is natural to turn the wave-numbers into a **wave-vector**

$$\vec{k} = (k_x, k_y, k_z) = (\frac{2\pi}{\lambda_x}, \frac{2\pi}{\lambda_y}, \frac{2\pi}{\lambda_y}) \qquad (18\text{-}6)$$

where λ_i is the wavelength of the wave-function in the i^{th} direction.

The three dimensional Fourier transform pair takes the form:

$$\begin{aligned} f(\vec{x}) =& \frac{1}{\sqrt{(2\pi)^3}} \iiint_{-\infty}^{\infty} g(\vec{k}) e^{\imath \vec{k}\cdot\vec{x}} dk_x dk_y dk_z \\ g(\vec{k}) =& \frac{1}{\sqrt{(2\pi)^3}} \iiint_{-\infty}^{\infty} f(\vec{x}) e^{-\imath \vec{k}\cdot\vec{x}} dxdydz \end{aligned} \qquad (18\text{-}7)$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Properties of Fourier Transforms

Dirac Delta Functions

Because the inverse transform of a transform returns the original function, this allows a definition of an interesting function called the Dirac delta function $\delta(x - x_o)$. Combining the two equations in Eq. 18-5 into a single equation, and then interchanging the order of integration:

$$
\begin{aligned}
f(x) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} f(\xi) e^{-\imath k \xi} d\xi \right\} e^{\imath k x} dk \\
f(x) &= \int_{-\infty}^{\infty} f(\xi) \left\{ \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{\imath k (x-\xi)} dk \right\} d\xi
\end{aligned}
\tag{18-8}
$$

Apparently, a function can be defined

$$\delta(x - x_o) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{\imath k (x-\xi)} dk \tag{18-9}$$

that has the property

$$f(x_o) = \int_{-\infty}^{\infty} \delta(x - x_o) f(x) dx \tag{18-10}$$

in other words, δ picks out the value at $x = x_o$ and returns it outside of the integration.

Parseval's Theorem

The delta function can be used to derive an important *conservation theorem.*

If $f(x)$ represents the density of some function (i.e., a wave-function like $\psi(x)$), the square-magnitude of f integrated over all of space should be the total amount of material in space.

$$\int_{-\infty}^{\infty} f(x) \bar{f}(x) dx = \int_{-\infty}^{\infty} \left\{ \left(\frac{1}{\sqrt{2\pi}} g(k) e^{-\imath k x} dk \right) \left(\frac{1}{\sqrt{2\pi}} \bar{g}(\kappa) e^{-\imath \kappa x} d\kappa \right) \right\} dx \tag{18-11}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

where the complex-conjugate is indicated by the over-bar. This exponentials can be collected together and the definition of the δ-function can be applied and the following simple result can is obtained

$$\int_{-\infty}^{\infty} f(x)\bar{f}(x)dx = \int_{-\infty}^{\infty} g(k)\bar{g}(k)dk = \tag{18-12}$$

which is Parseval's theorem. It says, that the magnitude of the wave-function, whether it is summed over real space or over momentum space must be the same.

Convolution Theorem

The *convolution* of two functions is given by

$$F(x) = p_1(x) \star p_2(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p_1(\eta)p_2(x-\eta)d\eta \tag{18-13}$$

If p_1 and p_2 can be interpreted as densities in probability, then this convolution quantity can be interpreted as "the total joint probability due to two probability distributions whose arguments add up to x."[10]

The proof is straightforward that the convolution of two functions, $p_1(x)$ and $p_2(x)$, is a Fourier integral over the product of their Fourier transforms, $\psi_1(k)$ and $\psi_2(k)$:

$$p_1(x) \star p_2(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p_1(\eta)p_2(x-\eta)d\eta = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \psi_1(k)\psi_2(k)e^{\imath kx}dk \tag{18-14}$$

This implies that Fourier transform of a convolution is a direct product of the Fourier transforms $\psi_1(k)\psi_2(k)$.

Another way to think of this is that "the net effect on the spatial function due two interfering waves is contained by product the fourier transforms." Practically, if the effect of an aperture (i.e., a sample of only a finite part of real space) on a wave-function is desired, then it can be obtained by multiplying the Fourier transform of the aperture and the Fourier transform of the entire wave-function.

[10] To think this through with a simple example, consider the probability that two dice sum up 10. It is the sum of $p_1(n)p_2(10-n)$ over all possible values of n.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 1

Creating Images of Lattices for Subsequent Fourier Transform

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A very large matrix of ones (white) and zeroes (black) is created as a set of "pixels" for imaging. The white regions arranged as 8 × 8 squares in a rectangular patterns. A diffraction pattern from a group of scattering centers such atoms is related to the Fourier transform of the "atom" positions.

1: `Table` is used to created "submatrices" of 8×8 ones or zeroes. `Join` will combine the rows of matrices and creates a "tall skinny" matrix from of three square ones. "pixel images" of lattices by placing ones (white) and zeroes (black) in a rectangular grid.

2: *latcell* will be a 32×32 black region with an 16×8 white rectangle near the center. The `Transpose` of four `Join`-ed squares will be a short-fat matrix. Joining four of the resulting `Transpose` operations produces the square matrix.

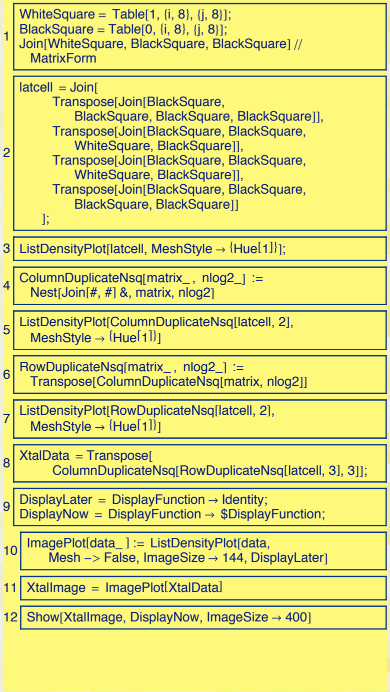
3: `ListDensityPlot` produces a grayscale image from an array of "pixel values" between 0 (black) and 1 (white).

4: *ColumnDuplicateNsq* takes a matrix as an argument and then recursively duplicates its *rows* into a matrix that has the *same number of columns* as the original. It makes at copy of all the rows at the first iteration, doubling the number of rows—at the second iteration it copies all the rows of the previous result quadrupling the number of rows, and so on. *ColumnDuplicateNsq* uses `Nest` with a *pure function.*

8: The result of calling *RowDuplicateNsq* and *ColumnDuplicateNsq* with "recursion" arguments of 3, creates an $8^3 \times 8^3$ matrix with a square lattice of white rectangles.

9: *DisplayLater* and *DisplayNow* are examples of rule definitions that can be passed to `Show` to delay display or to show a delayed display.

12: *XtalImage* will be used for the Fourier transfrom "diffraction" simulations in the following example.



```
1  WhiteSquare = Table[1, {i, 8}, {j, 8}];
   BlackSquare = Table[0, {i, 8}, {j, 8}];
   Join[WhiteSquare, BlackSquare, BlackSquare] //
     MatrixForm

2  latcell = Join[
           Transpose[Join[BlackSquare,
               BlackSquare, BlackSquare, BlackSquare]],
           Transpose[Join[BlackSquare, BlackSquare,
               WhiteSquare, BlackSquare]],
           Transpose[Join[BlackSquare, BlackSquare,
               WhiteSquare, BlackSquare]],
           Transpose[Join[BlackSquare, BlackSquare,
               BlackSquare, BlackSquare]]
       ];

3  ListDensityPlot[latcell, MeshStyle → {Hue[1]}];

4  ColumnDuplicateNsq[matrix_ , nlog2_] :=
     Nest[Join[#, #] &, matrix, nlog2]

5  ListDensityPlot[ColumnDuplicateNsq[latcell, 2],
     MeshStyle → {Hue[1]}]

6  RowDuplicateNsq[matrix_ , nlog2_] :=
     Transpose[ColumnDuplicateNsq[matrix, nlog2]]

7  ListDensityPlot[RowDuplicateNsq[latcell, 2],
     MeshStyle → {Hue[1]}]

8  XtalData = Transpose[
           ColumnDuplicateNsq[RowDuplicateNsq[latcell, 3], 3]];

9  DisplayLater = DisplayFunction → Identity;
   DisplayNow = DisplayFunction → $DisplayFunction;

10 ImagePlot[data_ ] := ListDensityPlot[data,
       Mesh –> False, ImageSize → 144, DisplayLater]

11 XtalImage = ImagePlot[XtalData]

12 Show[XtalImage, DisplayNow, ImageSize → 400]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016

Fast Fourier Transforms and Simulated Diffraction

The fast fourier transform (FFT) is a very fast algorithim for compute discrete Fourier transforms (DFT) (i.e., the Fourier transform of a data set) and is widely used in the physical sciences. For image data, the Fourier transform is the diffraction pattern (i.e., the intensity of reflected waves from a set of objects, the pattern results from positive or negative reinforcement or coherence).

However, for FFT simulations of the diffraction pattern from an image, the question arises on what to do with the rest of space *which is not the original image*. In other words, the Fourier transform is taken over all space, but the image is finite. In the examples that follow, the rest of space is occupied by *periodic duplications* of the original image. Thus, because the original image is rectangular, there will always be an additional rectangular symmetry in the diffraction pattern due to scattering from the duplicate features in the neighboring images.

The result of a discrete Fourier transform is a also a discrete set. There are a finite number of pixels in the data, the same finite number of subperiodic wavenumbers. In other words, the Discrete Fourier Transform of a $N \times M$ image will be a data set of $N \times M$ wavenumbers:

$$\begin{aligned}\text{Discrete FT Data} = {} & 2\pi\left(\frac{1}{N\text{pixels}}, \frac{2}{N\text{pixels}}, \ldots, \frac{N}{N\text{pixels}}\right) \\ & \times 2\pi\left(\frac{1}{M\text{pixels}}, \frac{2}{M\text{pixels}}, \ldots, \frac{M}{M\text{pixels}}\right)\end{aligned} \tag{18-15}$$

representing the amplitudes of the indicated periodicities.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 2

Discrete Fourier Transforms

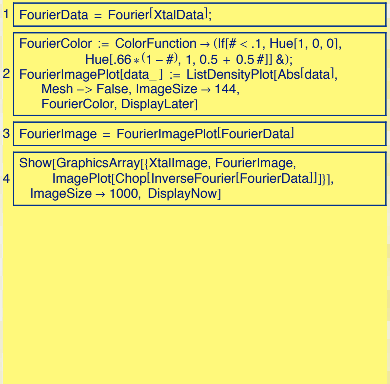
notebook (non-evaluated) pdf (evaluated) html (evaluated)

Example of taking the DFT of the perfect lattice created above and visualizing the diffraction pattern.

1: *FourierData* is the DFT (obtained with `Fourier`) of *XtalImage* .

2: *FourierColor* is a special `ColorFunction` for visualizing diffraction patterns. If the intensity is very low (¡0.1), the result will be black; otherwise it will scale from blue at low intensities to red at the highest intensity. *FourierImage* is a function to display the result of a DFT, it uses `Abs` to get the magnitudes of the imaginary data set created by `Fourier`.

4: Notice that the DFT has very sharp features, this is because the underlying lattice is perfect. Each feature represents a different periodic function in the direction $\vec{k} = (k_x, k_y)$.



```
1 FourierData = Fourier[XtalData];

  FourierColor := ColorFunction → (If[# < .1, Hue[1, 0, 0],
            Hue[.66 * (1 – #), 1, 0.5 + 0.5 #]] &);
2 FourierImagePlot[data_ ] := ListDensityPlot[Abs[data],
     Mesh –> False, ImageSize → 144,
     FourierColor, DisplayLater]

3 FourierImage = FourierImagePlot[FourierData]

  Show[GraphicsArray[{XtalImage, FourierImage,
4        ImagePlot[Chop[InverseFourier[FourierData]]]}],
    ImageSize → 1000, DisplayNow]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 3

Visualizing Diffraction Patterns

notebook (non-evaluated) pdf (evaluated) html (evaluated)

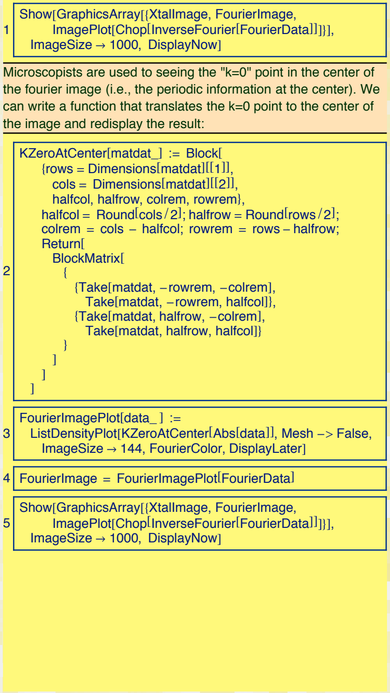
Visualization examples are created and a function is constructed to move the *longest wavelength (i.e.,* $\vec{k} \approx 0$ *periodicities* to the center of the resulting pattern.

1: Using `GraphicsArray`, the original image, its diffraction image, and the inverse Fourier transform of the diffraction image are viewed side-by-side.

2: Diffraction images are usually observed with the long wavelength features at the center of the image, instead of at the corners. *KZeroAtCenter* divides the original matrix data into four approximately equal-sized parts, and then exchanges the data from the northwest and southeast parts of the original matrix and exchanges the northeast and southwest data. The result is an image with $\vec{k} \approx 0$ at the center.

3: *FourierImagePlot* takes input Fourier-transformed data, rearranges the diffraction image and produces an image with $\vec{k} \approx 0$ at the center.

5: This will be a row similar to (1) above, but with the diffraction pattern adjusted.



```
1  Show[GraphicsArray[{XtalImage, FourierImage,
         ImagePlot[Chop[InverseFourier[FourierData]]]}],
     ImageSize → 1000,  DisplayNow]
```

Microscopists are used to seeing the "k=0" point in the center of the fourier image (i.e., the periodic information at the center). We can write a function that translates the k=0 point to the center of the image and redisplay the result:

```
2  KZeroAtCenter[matdat_] := Block[
       {rows = Dimensions[matdat][[1]],
         cols = Dimensions[matdat][[2]],
         halfcol, halfrow, colrem, rowrem},
       halfcol = Round[cols / 2]; halfrow = Round[rows / 2];
       colrem = cols − halfcol; rowrem = rows − halfrow;
       Return[
         BlockMatrix[
           {
             {Take[matdat, −rowrem, −colrem],
               Take[matdat, −rowrem, halfcol]},
             {Take[matdat, halfrow, −colrem],
               Take[matdat, halfrow, halfcol]}
           }
         ]
       ]
     ]
```

```
3  FourierImagePlot[data_] :=
     ListDensityPlot[KZeroAtCenter[Abs[data]], Mesh −> False,
       ImageSize → 144, FourierColor, DisplayLater]
```

```
4  FourierImage = FourierImagePlot[FourierData]
```

```
5  Show[GraphicsArray[{XtalImage, FourierImage,
         ImagePlot[Chop[InverseFourier[FourierData]]]}],
     ImageSize → 1000,  DisplayNow]
```

3.016 Home

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 4

Diffraction Patterns of Defective Lattices

notebook (non-evaluated) pdf (evaluated) html (evaluated)

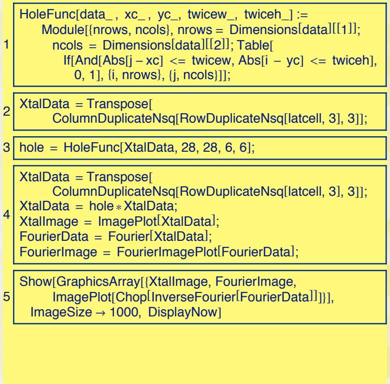
Data from the perfect lattice is used to create a defect by *scalar multiplying* with another matrix with a *small hole* created with zeroes.

1: *HoleFunc* uses the size of the data to create a matrix of ones with a rectangular region of zeroes at a specified position and size.

3: A 12×12 hole is created at position $28, 28$.

4: The hole is multiplied by the original perfect crystal to create a defect and the diffraction pattern is obtained.

5: The defect gives rises to "diffuse" scattering near $\vec{k} = 0$.



```
1  HoleFunc[data_ , xc_ , yc_, twicew_, twiceh_] :=
       Module[{nrows, ncols}, nrows = Dimensions[data][[1]];
         ncols = Dimensions[data][[2]]; Table[
           If[And[Abs[j – xc] <= twicew, Abs[i – yc] <= twiceh],
             0, 1], {i, nrows}, {j, ncols}]];

2  XtalData = Transpose[
         ColumnDuplicateNsq[RowDuplicateNsq[latcell, 3], 3]];

3  hole = HoleFunc[XtalData, 28, 28, 6, 6];

4  XtalData = Transpose[
         ColumnDuplicateNsq[RowDuplicateNsq[latcell, 3], 3]];
   XtalData = hole∗XtalData;
   XtalImage = ImagePlot[XtalData];
   FourierData = Fourier[XtalData];
   FourierImage = FourierImagePlot[FourierData];

5  Show[GraphicsArray[{XtalImage, FourierImage,
         ImagePlot[Chop[InverseFourier[FourierData]]]}],
     ImageSize → 1000, DisplayNow]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 5

Diffraction Patterns from Lattices with Thermal 'Noise'

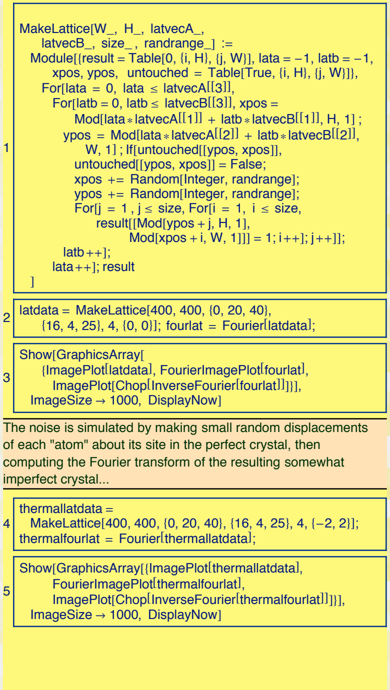
notebook (non-evaluated) pdf (evaluated) html (evaluated)

Functions to create a larger family of two-dimensional lattices are developed with a variable that simulates random deviation from a perfect position.

1: *MakeLattice* takes input for the width and height of the resulting lattice image, structures for the lattice vectors and the number of repeats to produce, a size for the 'atom,' and a random amplitude from which to simulate noise. This function is not very well-constructed and doesn't always work perfectly. I'll improve it someday.

3: This will display the original 'perfect' lattice, its resulting diffraction pattern, and the inverse fourier image of the diffraction pattern.

5: This will illustrate the effect of adding thermal noise: a diffuse ring will be superimposed on the original diffraction pattern.



```
1  MakeLattice[W_, H_, latvecA_,
       latvecB_, size_, randrange_] :=
     Module[{result = Table[0, {i, H}, {j, W}], lata = −1, latb = −1,
         xpos, ypos, untouched = Table[True, {i, H}, {j, W}]},
       For[lata = 0, lata ≤ latvecA[[3]],
         For[latb = 0, latb ≤ latvecB[[3]], xpos =
             Mod[lata∗latvecA[[1]] + latb∗latvecB[[1]], H, 1] ;
           ypos = Mod[lata∗latvecA[[2]] + latb∗latvecB[[2]],
               W, 1] ; If[untouched[[ypos, xpos]],
             untouched[[ypos, xpos]] = False;
             xpos += Random[Integer, randrange];
             ypos += Random[Integer, randrange];
             For[j = 1 , j ≤ size, For[i = 1, i ≤ size,
                 result[[Mod[ypos + j, H, 1],
                       Mod[xpos + i, W, 1]]] = 1; i++]; j++]];
           latb++];
         lata++]; result
     ]
```

```
2  latdata = MakeLattice[400, 400, {0, 20, 40},
       {16, 4, 25}, 4, {0, 0}]; fourlat = Fourier[latdata];
```

```
3  Show[GraphicsArray[
       {ImagePlot[latdata], FourierImagePlot[fourlat],
         ImagePlot[Chop[InverseFourier[fourlat]]]}],
     ImageSize → 1000, DisplayNow]
```

The noise is simulated by making small random displacements of each "atom" about its site in the perfect crystal, then computing the Fourier transform of the resulting somewhat imperfect crystal...

```
4  thermallatdata =
     MakeLattice[400, 400, {0, 20, 40}, {16, 4, 25}, 4, {−2, 2}];
   thermalfourlat = Fourier[thermallatdata];
```

```
5  Show[GraphicsArray[{ImagePlot[thermallatdata],
         FourierImagePlot[thermalfourlat],
         ImagePlot[Chop[InverseFourier[thermalfourlat]]]}],
     ImageSize → 1000, DisplayNow]
```

3.016 Home

Full Screen

Close

Quit

Lecture 18 Mathematica® Example 6

Using an Aperature to Select Particular Perioidicities in a Diffraction Pattern

notebook (non-evaluated) pdf (evaluated) html (evaluated)

This is a function designed to select particular regions from fourier transforms of a perfect lattice and the perturbation of a perfect lattice, and then display eight images in two columns. The left column of graphics illustrates (from top to bottom) the ”clean” input image, the entire fourier transform with the rectangular aperature illustrated, the ”reconstructed image” that derives from the fourier transform of the aperature region, and finally a magnified image of the fourier transform within the aperature only.

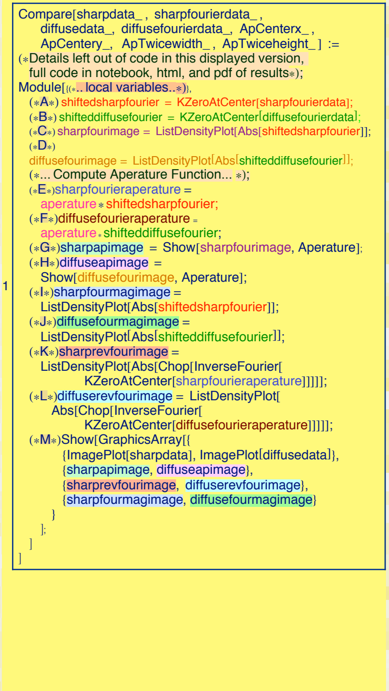
A,B The adjusted ($\vec{k} = 0$ at center) input data from the Fourier transforms of the reference lattice and one that will create a ‘diffuse’ pattern.

B,C The diffraction images of the data.

E,F Data from only a selected portion of values $\Delta\vec{k}_x, \Delta\vec{k}_y$ of the input data. This data should only have the periodicities of the original lattice for these selected values.

G–L The images associated with all the data and their reconstructions.

M Producing the array of graphics.



```
Compare[sharpdata_ , sharpfourierdata_ ,
    diffusedata_, diffusefourierdata_, ApCenterx_ ,
    ApCentery_, ApTwicewidth_ , ApTwiceheight_ ] :=
(*Details left out of code in this displayed version,
  full code in notebook, html, and pdf of results*);
Module[{(*.. local variables..*)},
  (*A*) shiftedsharpfourier = KZeroAtCenter[sharpfourierdata];
  (*B*) shifteddiffusefourier = KZeroAtCenter[diffusefourierdata];
  (*C*) sharpfourimage = ListDensityPlot[Abs[shiftedsharpfourier]];
  (*D*)
  diffusefourimage = ListDensityPlot[Abs[shifteddiffusefourier]];
  (*... Compute Aperature Function... *);
  (*E*)sharpfourieraperature =
    aperature * shiftedsharpfourier;
  (*F*)diffusefourieraperature =
    aperature * shifteddiffusefourier;
  (*G*)sharpapimage = Show[sharpfourimage, Aperature];
  (*H*)diffuseapimage =
    Show[diffusefourimage, Aperature];
  (*I*)sharpfourmagimage =
    ListDensityPlot[Abs[shiftedsharpfourier]];
  (*J*)diffusefourmagimage =
    ListDensityPlot[Abs[shifteddiffusefourier]];
  (*K*)sharprevfourimage =
    ListDensityPlot[Abs[Chop[InverseFourier[
            KZeroAtCenter[sharpfourieraperature]]]]];
  (*L*)diffuserevfourimage = ListDensityPlot[
      Abs[Chop[InverseFourier[
            KZeroAtCenter[diffusefourieraperature]]]]];
  (*M*)Show[GraphicsArray[{
          {ImagePlot[sharpdata], ImagePlot[diffusedata]},
          {sharpapimage, diffuseapimage},
          {sharprevfourimage, diffuserevfourimage},
          {sharpfourmagimage, diffusefourmagimage}
      }
    ];
  ]
]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 7

Visualizing Simulated Selected Area Diffraction

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples of selecting particular periodicities from ideal input and ‘noisy’ input data.

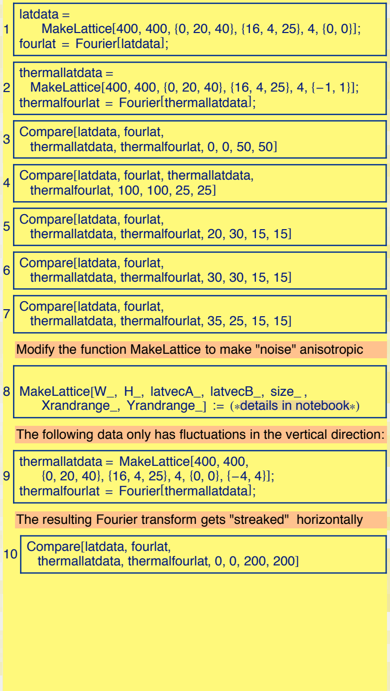
1: An ideal lattice and its Fourier transform are computed.

2: A ‘thermal’ perturbation of the ideal lattice and its Fourier transform are computed.

3: An example of selecting only those periodicities within 50 increments of $\vec{k} = 0$.

9: Creating input data where the ‘phonon modes’ have anisotropic amplitudes.

10: If the ‘thermal vibration’ appears only in the vertical direction, the resulting diffraction pattern gets ‘streaked’ in the horizontal direction.



```
1  latdata =
       MakeLattice[400, 400, {0, 20, 40}, {16, 4, 25}, 4, {0, 0}];
   fourlat = Fourier[latdata];

2  thermallatdata =
     MakeLattice[400, 400, {0, 20, 40}, {16, 4, 25}, 4, {-1, 1}];
   thermalfourlat = Fourier[thermallatdata];

3  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 0, 0, 50, 50]

4  Compare[latdata, fourlat, thermallatdata,
     thermalfourlat, 100, 100, 25, 25]

5  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 20, 30, 15, 15]

6  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 30, 30, 15, 15]

7  Compare[latdata, fourlat,
     thermallatdata, thermalfourlat, 35, 25, 15, 15]
```

Modify the function MakeLattice to make "noise" anisotropic

```
8  MakeLattice[W_, H_, latvecA_, latvecB_, size_,
       Xrandrange_, Yrandrange_] := (*details in notebook*)
```

The following data only has fluctuations in the vertical direction:

```
9  thermallatdata = MakeLattice[400, 400,
       {0, 20, 40}, {16, 4, 25}, 4, {0, 0}, {-4, 4}];
   thermalfourlat = Fourier[thermallatdata];
```

The resulting Fourier transform gets "streaked" horizontally

```
10  Compare[latdata, fourlat,
      thermallatdata, thermalfourlat, 0, 0, 200, 200]
```

3.016 Home

Full Screen

Close

Quit

Lecture 18 MATHEMATICA® Example 8

Discrete Fourier Transforms of Real Images

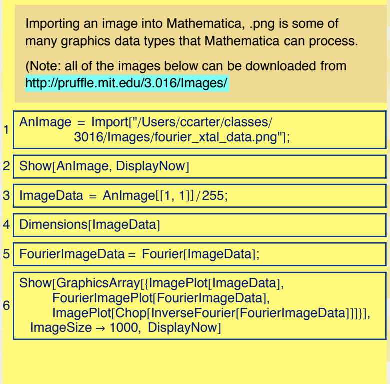
notebook (non-evaluated) pdf (evaluated) html (evaluated)

A image in graphics format, such as a .png, contains intensity as a function of position. If the function is gray-scale data, then each pixel typically takes on 2^8 discrete gray values between 0 and 255. This data can be input into MATHEMATICA® and then Fourier transformed. Images used here and below can be obtained from http://pruffle.mit.edu/3.016/Images.

1: An image in a number of different graphics formats can be imported into MATHEMATICA® with `Import`.

3: The image data is stored in a complicated format, but the gray values (indexed as integers between 0 and 255) are stored as the first item in the first list.

6: This illustrates the original image, its diffraction pattern, and reconstructed image.



Importing an image into Mathematica, .png is some of many graphics data types that Mathematica can process.

(Note: all of the images below can be downloaded from http://pruffle.mit.edu/3.016/Images/

```
1 AnImage = Import["/Users/ccarter/classes/
            3016/Images/fourier_xtal_data.png"];
2 Show[AnImage, DisplayNow]
3 ImageData = AnImage[[1, 1]] / 255;
4 Dimensions[ImageData]
5 FourierImageData = Fourier[ImageData];
6 Show[GraphicsArray[{ImagePlot[ImageData],
        FourierImagePlot[FourierImageData],
        ImagePlot[Chop[InverseFourier[FourierImageData]]]}],
   ImageSize → 1000, DisplayNow]
```

3.016 Home

Full Screen

Close

Quit

Lecture 18 Mathematica® Example 9

Selected Area Diffraction on Image Data

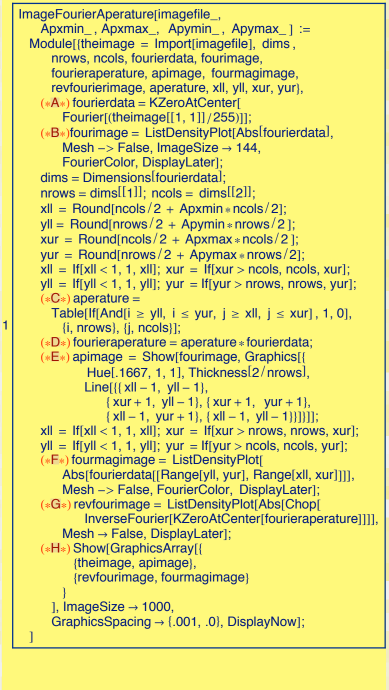
notebook (non-evaluated) pdf (evaluated) html (evaluated)

An function, *ImageFourierAperature* , that reads in the name of an image file and information about which regions of $\vec{k}$-space to select is developed.

A *fourierdata* is the shifted fourier transform of the image-file's data.

C *aperature* is a matrix of zeroes and ones representing the region in $\vec{k}$-space to be retained.

H This will display four images. To the right of the orginal image will be the diffraction pattern with an indication of where the aperature is located. Below the diffraction image will be a magnification of the aperature region. Below the original image will be the reconstructed image obtained from only those periodicities available in the aperature.



```
ImageFourierAperature[imagefile_,
    Apxmin_ , Apxmax_,  Apymin_ ,  Apymax_ ] :=
  Module[{theimage = Import[imagefile], dims ,
      nrows, ncols, fourierdata, fourimage,
      fourieraperature, apimage, fourmagimage,
      revfourierimage, aperature, xll, yll, xur, yur},
    (*A*) fourierdata = KZeroAtCenter[
        Fourier[(theimage[[1, 1]] / 255)]];
    (*B*)fourimage = ListDensityPlot[Abs[fourierdata],
        Mesh –> False, ImageSize → 144,
        FourierColor, DisplayLater];
    dims = Dimensions[fourierdata];
    nrows = dims[[1]]; ncols = dims[[2]];
    xll = Round[ncols / 2 + Apxmin * ncols / 2];
    yll = Round[nrows / 2 + Apymin * nrows / 2 ];
    xur = Round[ncols / 2 + Apxmax * ncols / 2 ];
    yur = Round[nrows / 2 + Apymax * nrows / 2];
    xll = If[xll < 1, 1, xll]; xur = If[xur > ncols, ncols, xur];
    yll = If[yll < 1, 1, yll]; yur = If[yur > nrows, nrows, yur];
    (*C*) aperature =
      Table[If[And[i ≥ yll, i ≤ yur, j ≥ xll, j ≤ xur] , 1, 0],
        {i, nrows}, {j, ncols}];
    (*D*) fourieraperature = aperature * fourierdata;
    (*E*) apimage = Show[fourimage, Graphics[{
              Hue[.1667, 1, 1], Thickness[2 / nrows],
             Line[{{ xll – 1, yll – 1},
                  { xur + 1, yll – 1}, { xur + 1, yur + 1},
                  { xll – 1, yur + 1}, { xll – 1, yll – 1}}]}]];
    xll = If[xll < 1, 1, xll]; xur = If[xur > nrows, nrows, xur];
    yll = If[yll < 1, 1, yll]; yur = If[yur > ncols, ncols, yur];
    (*F*) fourmagimage = ListDensityPlot[
        Abs[fourierdata[[Range[yll, yur], Range[xll, xur]]]],
        Mesh –> False, FourierColor, DisplayLater];
    (*G*) revfourimage = ListDensityPlot[Abs[Chop[
            InverseFourier[KZeroAtCenter[fourieraperature]]]],
        Mesh → False, DisplayLater];
    (*H*) Show[GraphicsArray[{
          {theimage, apimage},
          {revfourimage, fourmagimage}
        }
      ], ImageSize → 1000,
      GraphicsSpacing → {.001, .0}, DisplayNow];
  ]
```

Lecture 18 Mathematica® Example 10

Visualizing Selected Area Diffraction on Image Data

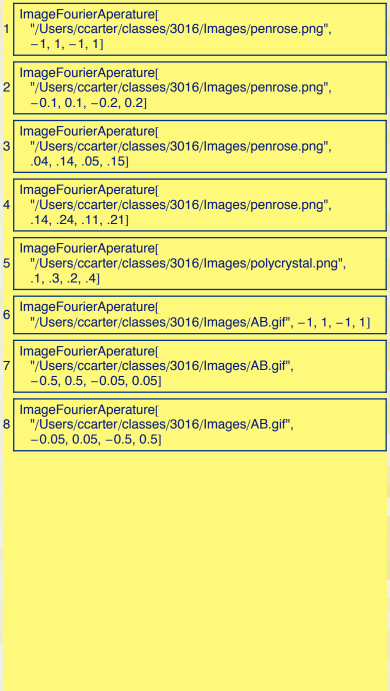
notebook (non-evaluated) pdf (evaluated) html (evaluated)

Examples and interpretations of using the function *ImageFourierAperature* on several input files. Images used here and below can be obtained from http://pruffle.mit.edu/3.016/Images.

1-4 Examples of selecting diffraction spots from an image of a penrose tiling. The ‘streaks’ come from the lines between tiles. Picking out particular $\vec{k}$ image lines of similar tilt in the reconstructed image.

5: Picking out particular periodicities allows one to image a selected set of oriented grains in a polycrystal.

6: Of course, one is not limited to playing with images of crystals and tilings. . .



```
1  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/penrose.png",
     −1, 1, −1, 1]

2  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/penrose.png",
     −0.1, 0.1, −0.2, 0.2]

3  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/penrose.png",
     .04, .14, .05, .15]

4  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/penrose.png",
     .14, .24, .11, .21]

5  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/polycrystal.png",
     .1, .3, .2, .4]

6  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/AB.gif", −1, 1, −1, 1]

7  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/AB.gif",
     −0.5, 0.5, −0.05, 0.05]

8  ImageFourierAperature[
     "/Users/ccarter/classes/3016/Images/AB.gif",
     −0.05, 0.05, −0.5, 0.5]
```

3.016 Home

Full Screen

Close

Quit

Lecture 19: Ordinary Differential Equations: Introduction

Reading:
Kreyszig Sections: 1.1, 1.2, 1.3 (pages2–8, 9–11, 12–17)

Differential Equations: Introduction

Ordinary differential equations are relations between a function of a single variable, its derivatives, and the variable:

$$F\left(\frac{d^n y(x)}{dx^n}, \frac{d^{n-1} f(x)}{dx^{n-1}}, \ldots, \frac{d^2 y(x)}{dx^2}, \frac{dy(x)}{dx}, y(x), x\right) = 0 \tag{19-1}$$

A *first-order* Ordinary Differential Equation (ODE) has only first derivatives of a function.

$$F(\frac{dy(x)}{dx}, y(x), x) = 0 \tag{19-2}$$

A *second-order* ODE has second and possibly first derivatives.

$$F\left(\frac{d^2 y(x)}{dx^2}, \frac{dy(x)}{dx}, y(x), x\right) = 0 \tag{19-3}$$

For example, the one-dimensional time-independent Shrödinger equation,

$$-\frac{\hbar}{2m}\frac{d^2\psi(x)}{dx^2} + U(x)\psi(x) = E\psi(x)$$

or

$$-\frac{\hbar}{2m}\frac{d^2\psi(x)}{dx^2} + U(x)\psi(x) - E\psi(x) = 0$$

is a second-order ordinary differential equation that specifies a relation between the wave function, $\psi(x)$, its derivatives, and a spatially dependent function $U(x)$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Differential equations result from physical models of anything that varies—whether in space, in time, in value, in cost, in color, etc. For example, differential equations exist for modeling quantities such as: volume, pressure, temperature, density, composition, charge density, magnetization, fracture strength, dislocation density, chemical potential, ionic concentration, refractive index, entropy, stress, etc. That is, almost all models for physical quantities are formulated with a differential equation.

The following example illustrates how some first-order equations arise:

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 19 MATHEMATICA® Example 1

Iteration: First-Order Sequences

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Sequences are developed in which the next iteration only depends on the current value; in this most simple case simulate exponential growth and decay.

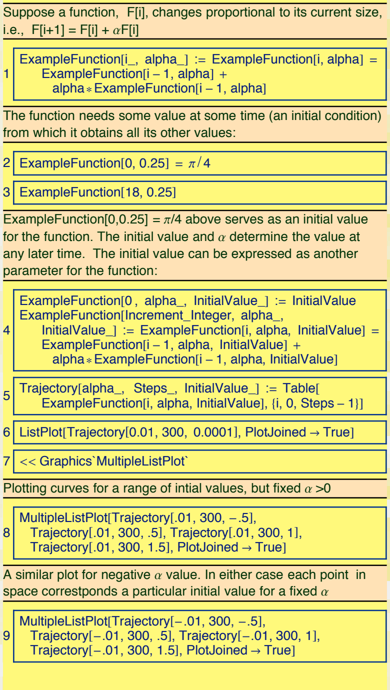
1: *ExampleFunction* taking two arguments is defined: the first argument represents the iteration and the second represents a single parameter expressing how the current iteration grows. The value at the $i+1^{\text{th}}$ iteration is the sum of the value of the i^{th} plus α times value of the i^{th}iteration. If this is a bank account and interest is compounded yearly, then the i^{th}iteration is the value of an account after i years at a compounded annual interest rate of α. This function has improved performance (but consumes more memory) by storing its intermediate values. Of course, the function would iterate for ever if an initial value is not specified. . .

4: Because the initial value and the 'growth factor' α deterimine all subsequent iterations, it is sensible to 'overload' *ExampleFunction* to take an extra argument for the intial value: here, if *ExampleFunction* is called with three arguments *and the first argument is zero*, then the initial value is set; otherwise it is a recursive definition with intermediate value storage.

5: *Trajectory* is an example of a function that builds a list by first-order iteration; its resulting list structure is plotted with `ListPlot`.

8: To visualize the behavior as a function of its initial value, several plots can be superposed with `MultipleListPlot` from the `MultipleListPlot` package. If $\alpha > 0$, the function goes to $\pm\infty$ depending on the sign of the initial value. For a fixed α every point in the plane belongs to one and only one trajectory associated with an initial value and that α.

9: If $\alpha < 0$, the function asymptotically goes to zero, independent of the initial value. In this case as well, the plane is completely covered by non-intersecting trajectories.



Suppose a function, F[i], changes proportional to its current size, i.e., F[i+1] = F[i] + αF[i]

```
1  ExampleFunction[i_, alpha_] := ExampleFunction[i, alpha] =
       ExampleFunction[i – 1, alpha] +
         alpha∗ExampleFunction[i – 1, alpha]
```

The function needs some value at some time (an initial condition) from which it obtains all its other values:

```
2  ExampleFunction[0, 0.25] = π/4
3  ExampleFunction[18, 0.25]
```

ExampleFunction[0,0.25] = π/4 above serves as an initial value for the function. The initial value and α determine the value at any later time. The initial value can be expressed as another parameter for the function:

```
4  ExampleFunction[0 , alpha_, InitialValue_] := InitialValue
   ExampleFunction[Increment_Integer, alpha_,
       InitialValue_] := ExampleFunction[i, alpha, InitialValue] =
       ExampleFunction[i – 1, alpha, InitialValue] +
         alpha∗ExampleFunction[i – 1, alpha, InitialValue]
5  Trajectory[alpha_, Steps_, InitialValue_] := Table[
       ExampleFunction[i, alpha, InitialValue], {i, 0, Steps – 1}]
6  ListPlot[Trajectory[0.01, 300, 0.0001], PlotJoined → True]
7  << Graphics`MultipleListPlot`
```

Plotting curves for a range of intial values, but fixed α >0

```
8  MultipleListPlot[Trajectory[.01, 300, –.5],
     Trajectory[.01, 300, .5], Trajectory[.01, 300, 1],
     Trajectory[.01, 300, 1.5], PlotJoined → True]
```

A similar plot for negative α value. In either case each point in space correstponds a particular initial value for a fixed α

```
9  MultipleListPlot[Trajectory[–.01, 300, –.5],
     Trajectory[–.01, 300, .5], Trajectory[–.01, 300, 1],
     Trajectory[–.01, 300, 1.5], PlotJoined → True]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The previous example is generalized to a discrete change Δt of a continous (i.e., time-like) parameter t. The following example demonstrates *first-order Euler finite differencing* or *Euler integration*. It is an integration approximation because the method uses a finite time step Δt to approximate $f(t) = \int_{t_0} tA(t)dt$ for a known first-order differential equation $df/dt = A(t)$ where $f(t_0)$ is an *initial condition*. In this example, the iteration sequence approximates

$$\begin{aligned}
(f(t=0),& f(\Delta t), f(2\Delta t), \ldots) \approx \\
&\left(f(t=0), f_{\text{aprx}}(\Delta t) = f(t=0) + \left.\frac{df}{dt}\right|_{t=0} \Delta t, f_{\text{aprx}}(2\Delta t) = f_{\text{aprx}}(2\Delta t) + \left.\frac{df}{dt}\right|_{t=\Delta t} \Delta t, \ldots \right) \\
&= (f(t=0), A(t=0)\Delta t, A(t=\Delta t)\Delta t, \ldots)
\end{aligned} \tag{19-4}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 19 MATHEMATICA® Example 2

First-Order Finite Differences

notebook (non-evaluated) pdf (evaluated) html (evaluated)

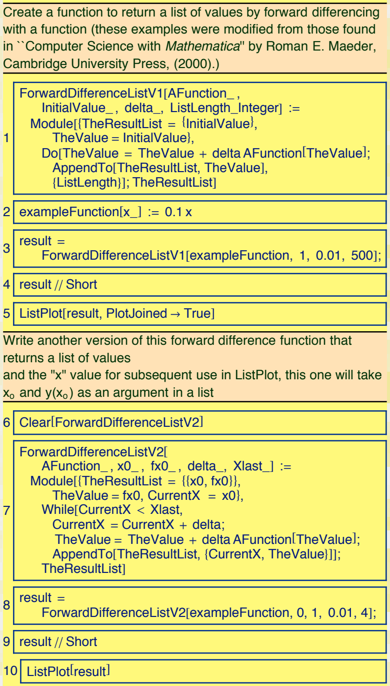
Two functions that 'grow' lists by using simple forward Euler finite differencing are constructed.

1: The function *ForwardDifferenceV1* is defined with four arguments: argument 1 is a placeholder for *another function* that determines how each increment changes (i.e., the function df/ft); argument 2 is the initial value; argument 3 is the distrete forward difference (i.e., Δt); argument 4 indicates the size of the list that will be returned. The function uses `Module` to hide an internal variable representing the current value of the list, and `AppendTo` to incrementally grow the list. In a following example, `NestList` and `NestWhileList` will be used to generate more efficient functions than the ones generate by `AppendTo` here.

2: *exampleFunction* is defined to to pass to sequence-generating functions—it plays the role of df/dt in Eq. 19-4.

5: `ListPlot` will produce a plot of the exemplary result which is a list of length 500.

7: *ForwardDifferenceV1* is unsatisfactory because the x-axis of the plot is the iteration and not the time-like variable that is more physical; so it is generalized with *ForwardDifferenceV2* which also takes arguments for the initial value and the final value of the continuous parameter. This function returns a list containing lists $(x_i, f(x_i))$ also suitable an argument to `ListPLot`.



Create a function to return a list of values by forward differencing with a function (these examples were modified from those found in ``Computer Science with *Mathematica*" by Roman E. Maeder, Cambridge University Press, (2000).)

1
```
ForwardDifferenceListV1[AFunction_ ,
    InitialValue_ , delta_, ListLength_Integer] :=
  Module[{TheResultList = {InitialValue},
      TheValue = InitialValue},
    Do[TheValue = TheValue + delta AFunction[TheValue];
      AppendTo[TheResultList, TheValue],
      {ListLength}]; TheResultList]
```

2
```
exampleFunction[x_] := 0.1 x
```

3
```
result =
    ForwardDifferenceListV1[exampleFunction, 1, 0.01, 500];
```

4
```
result // Short
```

5
```
ListPlot[result, PlotJoined → True]
```

Write another version of this forward difference function that returns a list of values
and the "x" value for subsequent use in ListPlot, this one will take x_o and $y(x_o)$ as an argument in a list

6
```
Clear[ForwardDifferenceListV2]
```

7
```
ForwardDifferenceListV2[
    AFunction_ , x0_ , fx0_ , delta_, Xlast_] :=
  Module[{TheResultList = {{x0, fx0}},
      TheValue = fx0, CurrentX  = x0},
    While[CurrentX < Xlast,
      CurrentX = CurrentX + delta;
       TheValue = TheValue + delta AFunction[TheValue];
      AppendTo[TheResultList, {CurrentX, TheValue}]];
    TheResultList]
```

8
```
result =
    ForwardDifferenceListV2[exampleFunction, 0, 1, 0.01, 4];
```

9
```
result // Short
```

10
```
ListPlot[result]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 19 MATHEMATICA® Example 3

Nested Operations

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The concept of nesting operations to produce finite differences is demonstrated. The MATHEMATICA® notion of a *pure function* is utilized with an example.

1: Construct a function, *StepOnce* , that operates on its first argument (the pair $\{x_i, y_i(x_i)\}$) with an input function dy/dx and a discrete increment δ. The function returns a list (the pair $\{x_{i+1}, y(x_{i+1})\}$) representing the finite difference approximation at "time" $x_i + \delta$.

2: Here, a specific case is developed by defining a function that explicitely defines both the input function (here *exampleFunction*) and the fixed increment (here $\Delta t = 0.01$). The result is a function that takes a single x-y pair.

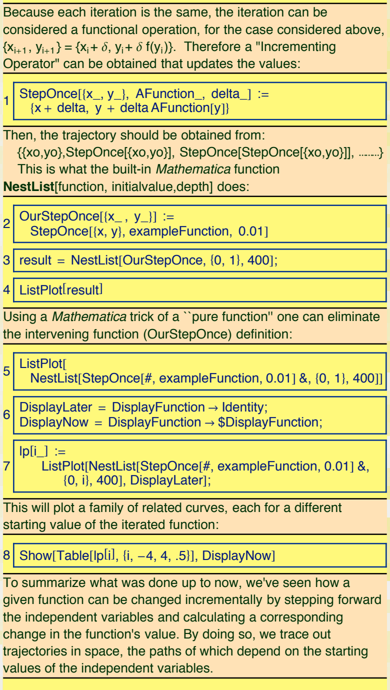
3: Instead of the forward difference techniques that used `AppendTo` to grow a list of x-y pairs—here, `NestList` is used to build a list (`result`) by repeatedly (400 times) applying a function to the result at the previous iteration.

5: Here, `NestList` is called with a *pure function* which is indicated by the & that appears the `StepOnce[#,exampleFunction,0.01]&` definition; the # is a placeholder for the functions argument. `NestList` calls this pure function repeatedly starting with the first argument (here $\{0, 1\}$) and stores intermediate values in a list.

6: In the next few steps, the goal is to generate such trajectories for a variety of initial conditions. This is achieved by creating and saving plots as *Graphics Objects*. In this step, rules (*DisplayLater* and *DisplayNow*) are defined that can be passed to plotting functions that control whether the created Graphics Object is displayed or not.

7: A function is defined that creates a Graphics object for a trajectory as a function of its sole argument representing the initial value. dd

8: A list of trajectories for initial values $(-4, -3.5, \ldots, 3.5, 4.0)$ are plotted together.



Because each iteration is the same, the iteration can be considered a functional operation, for the case considered above, $\{x_{i+1}, y_{i+1}\} = \{x_i + \delta, y_i + \delta f(y_i)\}$. Therefore a "Incrementing Operator" can be obtained that updates the values:

```
1  StepOnce[{x_, y_}, AFunction_, delta_] :=
     {x + delta, y + delta AFunction[y]}
```

Then, the trajectory should be obtained from:
{{xo,yo},StepOnce[{xo,yo}], StepOnce[StepOnce[{xo,yo}]], ...}
This is what the built-in *Mathematica* function **NestList**[function, initialvalue,depth] does:

```
2  OurStepOnce[{x_ , y_}] :=
     StepOnce[{x, y}, exampleFunction, 0.01]

3  result = NestList[OurStepOnce, {0, 1}, 400];

4  ListPlot[result]
```

Using a *Mathematica* trick of a ``pure function'' one can eliminate the intervening function (OurStepOnce) definition:

```
5  ListPlot[
     NestList[StepOnce[#, exampleFunction, 0.01] &, {0, 1}, 400]]

6  DisplayLater = DisplayFunction → Identity;
   DisplayNow = DisplayFunction → $DisplayFunction;

7  lp[i_] :=
       ListPlot[NestList[StepOnce[#, exampleFunction, 0.01] &,
         {0, i}, 400], DisplayLater];
```

This will plot a family of related curves, each for a different starting value of the iterated function:

```
8  Show[Table[lp[i], {i, −4, 4, .5}], DisplayNow]
```

To summarize what was done up to now, we've seen how a given function can be changed incrementally by stepping forward the independent variables and calculating a corresponding change in the function's value. By doing so, we trace out trajectories in space, the paths of which depend on the starting values of the independent variables.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Geometrical Interpretation of Solutions

The relationship between a function and its derivatives for a first-order ODE,

$$F(\frac{dy(x)}{dx}, y(x), x) = 0 \tag{19-5}$$

can be interpreted as a level set formulation for a two-dimensional surface embedded in a three-dimensional space with coordinates (y', y, x). The surface specifies a relationship that must be satisfied between the three coordinates.

If $y'(x)$ can be solved for exactly,

$$\frac{dy(x)}{dx} = f(x, y) \tag{19-6}$$

then $y'(x)$ can be thought of as a height above the x-y plane.

For a very simple example, consider Newton's law of cooling which relates the change in temperature, dT/dt, of a body to the temperature of its environment and a *kinetic coefficient* k:

$$\frac{dT(t)}{dt} = -k(T - T_o) \tag{19-7}$$

It is very useful to "non-dimensionalize" variables by scaling via the physical parameters. In this way, a single ODE represents *all* physical situations and provides a way to describe universal behavior in terms of the single ODE. For Newton's law of cooling, this can be done by defining non-dimensional temperatures and time with $\Theta = T/T_o$ and $\tau = kt$, then if T_o and k are constants:

$$\frac{d\Theta(\tau)}{d\tau} = (1 - \Theta)$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 19 MATHEMATICA® Example 4

The Geometry of First-Order ODES

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The surface representation provides a useful way to think about differential equations—much can be inferred about a solution's behavior without computing the solution exactly. This is shown for a simple case of Newton's law of cooling Equation 19 and an artificial case.

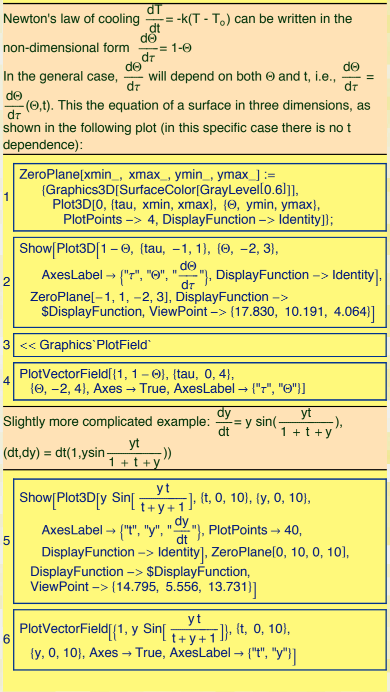
1: For first-order ODEs, behavior is dominated by whether the derivative term is positive or negative. Here, a 3D graphics object is created for a gray-colored horizontal plane at $z = 0$. This is achieved by combining (in a list) the `SurfaceColor` directive in a `Graphics3D` object, and then using `Plot3D` to create the plane with delayed display.

2: This will create the surface associated with Newton's law of cooling with the zero plane. This case is very simple. The sign of the change of Θ depends only the sign of $1 - \Theta$ and therefore $d\Theta/dt = 0$ is the parametric curve (a line in this case) $(d\Theta/dt = 0, \Theta = 1, \tau)$. That is, if $\Theta = 1$ at any time τ it will stay there at all subsequent times (also, at all previous times as well). Because $\Theta(\tau)$ will always increase when $\Theta < 1$ and will always decrease when $\Theta > 1$, the solutions will asymptotically approach $\Theta = 1$.

4: The asymptotic behavior can be further visualized by plotting a first-order difference representation of how the solution is changing in time, i.e, $(d\tau, d\Theta) = d\tau\left(1, \frac{d\Theta}{d\tau}\right)$ This can be obtained with `PlotVectorField` from the `PlotField` package. Here the magnitude of the arrows is scaled by setting $d\tau = 1$.

5: A more complex case $\frac{dy}{dt} = y \sin\left(\frac{yt}{1+y+t}\right)$ can be visualized as well and the behavior can be inferred whether the derivative lies above or below the zero-plane (i.e., the sign of the derivative).

6: `PlotVectorField` provides another method to follow a solution trajectories.



Newton's law of cooling $\frac{dT}{dt}$= -k(T - T_o) can be written in the non-dimensional form $\frac{d\Theta}{d\tau}$= 1-Θ

In the general case, $\frac{d\Theta}{d\tau}$ will depend on both Θ and t, i.e., $\frac{d\Theta}{d\tau}$ = $\frac{d\Theta}{d\tau}$(Θ,t). This the equation of a surface in three dimensions, as shown in the following plot (in this specific case there is no t dependence):

```
1  ZeroPlane[xmin_, xmax_, ymin_, ymax_] :=
     {Graphics3D[SurfaceColor[GrayLevel[0.6]]],
      Plot3D[0, {tau, xmin, xmax}, {Θ, ymin, ymax},
        PlotPoints -> 4, DisplayFunction -> Identity]};
```

```
2  Show[Plot3D[1 - Θ, {tau, -1, 1}, {Θ, -2, 3},
       AxesLabel → {"τ", "Θ", "dΘ/dτ"}, DisplayFunction -> Identity],
     ZeroPlane[-1, 1, -2, 3], DisplayFunction ->
       $DisplayFunction, ViewPoint -> {17.830, 10.191, 4.064}]
```

```
3  << Graphics`PlotField`
```

```
4  PlotVectorField[{1, 1 - Θ}, {tau, 0, 4},
     {Θ, -2, 4}, Axes → True, AxesLabel → {"τ", "Θ"}]
```

Slightly more complicated example: $\frac{dy}{dt}$= y sin($\frac{yt}{1 + t + y}$), (dt,dy) = dt(1,ysin $\frac{yt}{1 + t + y}$))

```
5  Show[Plot3D[y Sin[yt/(t + y + 1)], {t, 0, 10}, {y, 0, 10},
       AxesLabel → {"t", "y", "dy/dt"}, PlotPoints → 40,
       DisplayFunction -> Identity], ZeroPlane[0, 10, 0, 10],
     DisplayFunction -> $DisplayFunction,
     ViewPoint -> {14.795, 5.556, 13.731}]
```

```
6  PlotVectorField[{1, y Sin[yt/(t + y + 1)]}, {t, 0, 10},
     {y, 0, 10}, Axes → True, AxesLabel → {"t", "y"}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Separable Equations

If a first-order ordinary differential equation $F(y', y, x) = 0$ can be rearranged so that only one variable, for instance y, appears on the left-hand-side multiplying its derivative and the other, x, appears only on the right-hand-side, then the equation is said to be 'separated."

$$\begin{aligned} g(y)\frac{dy}{dx} &= f(x) \\ g(y)dy &= f(x)dx \end{aligned} \tag{19-8}$$

Each side of such an equation can be integrated with respect to the variable that appears on that side:

$$\int_{y(x_o)}^{y} g(\eta)d\eta = \int_{x_o}^{x} f(\xi)d\xi \tag{19-9}$$

if the initial value, $y(x_o)$ is known. If not, the equation can be solved with an integration constant C_0,

$$\int g(y)dy = \int f(x)dx + C_0 \tag{19-10}$$

where C_0 is determined from initial conditions. or

$$\int_{y_{\text{init}}}^{y} g(\eta)d\eta = \int_{x_{\text{init}}}^{x} f(\zeta)d\zeta \tag{19-11}$$

where the initial conditions appear explicitely.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 19 MATHEMATICA® Example 5

Using MATHEMATICA® 's Built-in Ordinary Differential Equation Solver

notebook (non-evaluated) pdf (evaluated) html (evaluated)

MATHEMATICA® has built-in exact and numerical differential equations solvers. DSolve takes a representation of a differential equation with initial and boundary conditions and returns a solution if it can find one. If insufficient initial or boundary conditions are specified, then "integration constants" are added to the solution.

1: `DSolve` operates like `Solve` . It takes a list of equations containing symbolic derivatives, the function to be solved for, and the dependent variable. In this case, the general solution of $\frac{dy(x)}{dx} = -xy(x)$ is returned as a list of rules. The solutions are be obtained by applying the rules (i.e., `y[x]/.dsol`).

2: The solution will depend on an integration constant(s) in general. If additional If more constraints (i.e., equations) are provided, then (provided a solution exists) the integration constant is determined as well.

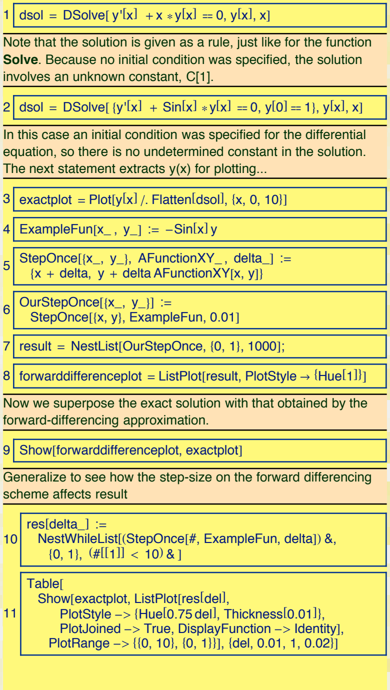
3: The solution is plotted by turning the "solution rule" into a single list with `Flatten`. The plot is stored as a graphics object *exactplot* .

4: To see how finite differencing compares to the exact solution, the method from an earlier example is used. Here, a the forward differencing function is defined for the solution that was just obtained.

9: The previously defined forward differencing method is compared to the exact solution.

10: Here, the method is generalized to take an argument for the size Δt. `NestWhileList` is used with a pure function (where one arguments is fixed by passing through *res*). A pure function is also defined for the test of when to stop building the list—in this case, it stops when the first element in the list (accumulated time) exceeds 10.

11: An animation which will visualize the effect of time-step on accuracy of the Euler method is created. `Show` repeatedly called on *exactplot* (the exact solution) and a graphics object created from calling `ListPlot` on *res* with different time-steps.



```
1  dsol = DSolve[ y'[x] + x * y[x] == 0, y[x], x]
```

Note that the solution is given as a rule, just like for the function **Solve**. Because no initial condition was specified, the solution involves an unknown constant, C[1].

```
2  dsol = DSolve[ {y'[x] + Sin[x] * y[x] == 0, y[0] == 1}, y[x], x]
```

In this case an initial condition was specified for the differential equation, so there is no undetermined constant in the solution. The next statement extracts y(x) for plotting...

```
3  exactplot = Plot[y[x] /. Flatten[dsol], {x, 0, 10}]

4  ExampleFun[x_ , y_] := -Sin[x] y

5  StepOnce[{x_, y_}, AFunctionXY_ , delta_] :=
     {x + delta, y + delta AFunctionXY[x, y]}

6  OurStepOnce[{x_, y_}] :=
     StepOnce[{x, y}, ExampleFun, 0.01]

7  result = NestList[OurStepOnce, {0, 1}, 1000];

8  forwarddifferenceplot = ListPlot[result, PlotStyle → {Hue[1]}]
```

Now we superpose the exact solution with that obtained by the forward-differencing approximation.

```
9  Show[forwarddifferenceplot, exactplot]
```

Generalize to see how the step-size on the forward differencing scheme affects result

```
10  res[delta_] :=
      NestWhileList[(StepOnce[#, ExampleFun, delta]) &,
        {0, 1}, (#[[1]] < 10) & ]

11  Table[
      Show[exactplot, ListPlot[res[del],
          PlotStyle -> {Hue[0.75 del], Thickness[0.01]},
          PlotJoined -> True, DisplayFunction -> Identity],
        PlotRange -> {{0, 10}, {0, 1}}], {del, 0.01, 1, 0.02}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

While the accuracy of the first-order differencing scheme can be determined by comparison to an exact solution, the question remains of how to establish accuracy and convergence with the step-size δ for an arbitrary ODE. This is a question of primary importance and studied by Numerical Analysis.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 20: Linear Homogeneous and Heterogeneous ODEs

Reading:
Kreyszig Sections: 1.4, 1.5 (pages19–25, 26–32)

Ordinary Differential Equations from Physical Models

In engineering and physics, modeling physical phenomena is the means by which technological and natural phenomena are understood and predicted. A model is an abstraction of a physical system, often with simplifying assumptions, into a mathematical framework. Every model should be verifiable by an experiment that, to the greatest extent possible, satisfies the approximations that were used to obtain the model.

In the context of modeling, differential equations appear frequently. Learning how to model new and interesting systems is a learned skill—it is best to learn by following a few examples. Grain growth provides some interesting modeling examples that result in first-order ODES.

Grain Growth

In materials science and engineering, a grain usually refers a single element in an ensemble that comprises a polycrystal. In a single phase polycrystal, a grain is a contiguous region of material with the same crystallographic orientation. It is separated from other grains by *grain boundaries* where the crystallographic orientation changes abruptly.

A grain boundary contributes extra free energy to the entire system that is proportional to the grain boundary area. Thus, if the boundary can move to reduce the free energy it will.

Consider simple, uniformly curved, isolated two- and three-dimensional grains.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

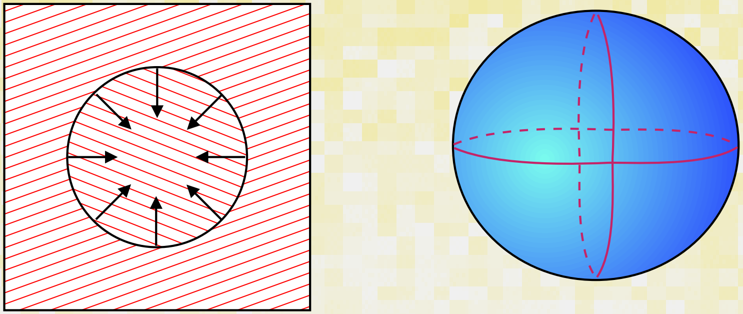


Figure 20-18: Illustration of a two-dimensional isolated circular grain and a three-dimensional isolated spherical grain. Because there is an extra energy in the system $\Delta G_{2D} = 2\pi R \gamma_{gb}$ and $\Delta G_{3D} = 4\pi R^2 \gamma_{gb}$, there is a driving force to reduce the radius of the grain. A simple model for grain growth is that the velocity (normal to itself) of the grain boundary is $v_{gb} = M_{gb}\gamma_{gb}\kappa$ where M_{gb} is the grain boundary mobility and κ is the mean curvature of the boundary. The normal velocity v_{gb} is towards the center of curvature.

A relevant question is "how fast will a grain change its size assuming that grain boundary migration velocity is proportional to curvature?"

For the two-dimensional case, the rate of change of area can be formulated by considering the following illustration.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

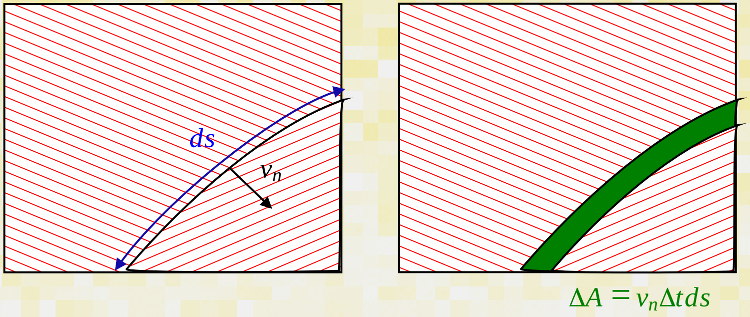


Figure 20-19: A segment of a grain boundary moving with normal velocity v_n will move a distance $v_n\Delta t$ in a short time Δt. The motion will result in a change of area $-\Delta A$ for the shrinking grain. Each segment, ds, of boundary contributes to the loss of area by $\Delta A = -v_n\Delta t ds$.

Because for a circle, the curvature is the same at each location on the grain boundary, the curvature is uniform and $v_n = M_{gb}\kappa_{gb}\gamma_{gb} = M_{gb}\gamma_{gb}/R$. Thus

$$\frac{dA}{dt} = -M_{gb}\gamma_{gb}\frac{1}{R}2\pi R = -2\pi M_{gb}\gamma_{gb} \qquad (20\text{-}1)$$

Thus, the area of a circular grain changes at a constant rate, the rate of change of radius is:

$$\frac{dA}{dt} = \frac{d\pi R^2}{dt} = 2\pi R\frac{dR}{dt} = -2\pi M_{gb}\gamma_{gb} \qquad (20\text{-}2)$$

which is a first-order, separable ODE with solution:

$$R^2(t) - R^2(t=0) = -2M_{gb}\gamma_{gb}t \qquad (20\text{-}3)$$

For a spherical grain, the change in volume ΔV due to the motion of a surface patch dS in a time Δt is $\Delta V = v_n\Delta t\, dS$. The curvature of a sphere is

$$\kappa_{sphere} = \left(\frac{1}{R} + \frac{1}{R}\right) \qquad (20\text{-}4)$$

Therefore the velocity of the interface is $v_n = 2M_{gb}\gamma_{gb}/R$. The rate of change of volume due to the contributions of each surface patch is

$$\frac{dV}{dt} = -M_{gb}\gamma_{gb}\frac{2}{R}4\pi R^2 = -8\pi M_{gb}\gamma_{gb}R == -4(6\pi^2)^{1/3}M_{gb}\gamma_{gb}V^{1/3} \tag{20-5}$$

which can be separated and integrated:

$$V^{2/3}(t) - V^{2/3}(t=0) = -\text{constant}_1 t \tag{20-6}$$

or

$$R^2(t) - R^2(t=0) = -\text{constant}_2 t \tag{20-7}$$

which is the same functional form as derived for two-dimensions.

The problem (and result) is more interesting if the grain doesn't have uniform curvature.

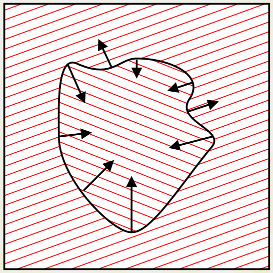


Figure 20-20: For a two-dimensional grain with non-uniform curvature, the local normal velocity (assumed to be proportional to local curvature) varies along the grain boundary. Because the motion is in the direction of the center of curvature, the velocity can be such that its motion increases the area of the interior grain for some regions of grain boundary and decreases the area in other regions.

However, it can still be shown that, even for an irregularly shaped two-dimensional grain, $A(t) - A(t=0) = -(\text{const})t$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Integrating Factors, Exact Forms

Exact Differential Forms

In classical thermodynamics for simple fluids, expressions such as

$$\begin{aligned} dU =& TdS - PdV \\ =& \left(\frac{\partial U}{\partial S}\right)_V dS + \left(\frac{\partial U}{\partial V}\right)_S dV \\ =& \delta q + \delta w \end{aligned} \tag{20-8}$$

represent the differential form of the combined first and second laws of thermodynamics. If $dU = 0$, meaning that the differential Eq. 20-8 is evaluated on a surface for which internal energy is constant, $U(S, V) = \text{const}$, then the above equation becomes a *differential form*

$$0 = \left(\frac{\partial U}{\partial S}\right)_V dS + \left(\frac{\partial U}{\partial V}\right)_S dV \tag{20-9}$$

This equation expresses a relation between changes in S and changes in V that are necessary to remain on the surface $U(S, V) = \text{const}$.

Suppose the situation is turned around and you are given the first-order ODE

$$\frac{dy}{dx} = -\frac{M(x, y)}{N(x, y)} \tag{20-10}$$

which can be written as the differential form

$$0 = M(x, y)dx + N(x, y)dy \tag{20-11}$$

Is there a function $U(x, y) = \text{const}$ or, equivalently, is it possible to find a curve represented by $U(x, y) = \text{const}$?

If such a curve exists then it depends only on one parameter, such as arc-length, and on that curve $dU(x, y) = 0$.

The answer is, "Yes, such a function $U(x, y) = \text{const}$ exists if an only if $M(x, y)$ and $N(x, y)$ satisfy the Maxwell relations"

$$\frac{\partial M(x, y)}{\partial y} = \frac{\partial N(x, y)}{\partial x} \tag{20-12}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Then if Eq. 20-12 holds, the differential form Eq. 20-11 is called *an exact differential* and a U exists such that $dU = 0 = M(x,y)dx + N(x,y)dy$.

Integrating Factors and Thermodynamics

For fixed number of moles of ideal gas, the internal energy is a function of the temperature only, $U(T) - U(T_o) = C_V(T - T_o)$. Consider the heat that is transfered to a gas that changes it temperature and volume a very small amount:

$$\begin{aligned} dU =& C_V dT = \delta q + \delta w = \delta q - PdV \\ \delta q =& C_V dT + PdV \end{aligned} \tag{20-13}$$

Can a Heat Function $q(T,V) =$ constant be found?

To answer this, apply Maxwell's relations.

Homogeneous and Heterogeneous Linear ODES

A linear differential equation is one that does not contain any powers (greater than one) of the function or its derivatives. The most general form is:

$$Q(x)\frac{dy}{dx} + P(x)y = R(x) \tag{20-14}$$

Equation 20-15 can always be reduced to a simpler form by defining $p = P/Q$ and $r = R/Q$:

$$\frac{dy}{dx} + p(x)y = r(x) \tag{20-15}$$

If $r(x) = 0$, Eq. 20-15 is said to be a *homogeneous linear first-order ODE*; otherwise Eq. 20-15 is a *heterogeneous linear first-order ODE.*

The reason that the homogeneous equation is linear is because solutions can superimposed—that is, if $y_1(x)$ and $y_2(x)$ are solutions to Eq. 20-15, then $y_1(x) + y_2(x)$ is also a solution to Eq. 20-15. This is the case if the first derivative and the function are themselves linear. The heterogeneous equation is also called *linear* in this case, but it is important to remember that sums and/or multiples of heterogeneous solutions are also solutions to the heterogeneous equation.

It will be demonstrated below (directly and with a MATHEMATICA® example) that the homogeneous equation has a solution of the form

$$y(x) = \text{const } e^{-\int p(x)dx} \tag{20-16}$$

To show this form directly, the homogenous equation can be written as

$$\frac{dy}{dx} = -p(x)y$$

Dividing each side through by through by y and integrate:

$$\int \frac{dy}{y} = \log y = -\int p(x)dx + \text{const}$$

which has solution

$$y(x) = \text{const} e^{-\int p(x)dx}$$

For the case of the heterogeneous first-order ODE, A trick (or, an integrating factor which amounts to the same thing) can be employed. Multiply both sides of the heterogeneous equation by $e^{\int p(x)}$:[11]

$$\exp\left[\int_a^x p(z)dz\right] \frac{dy(x)}{dx} + \exp\left[\int_a^x p(z)dz\right] p(x)y(x) = \exp\left[\int_a^x p(z)dz\right] r(x) \tag{20-17}$$

Notice that the left-hand-side can be written as a derivative of a simple expression

$$\exp\left[\int_a^x p(z)dz\right] \frac{dy(x)}{dx} + \exp\left[\int_a^x p(z)dz\right] p(x)y(x) = \frac{d}{dx}\left\{\exp\left[\int_a^x p(z)dz\right] y(x)\right\} \tag{20-18}$$

therefore

$$\frac{d}{dx}\left\{\exp\left[\int_a^x p(z)dz\right] y(x)\right\} = \exp\left[p(x)\right] r(x) \tag{20-19}$$

which can be integrated and then solved for $y(x)$:

$$y(x) = \exp\left[-\int_a^x p(z)dz\right] \left\{y(x=a) + \int_a^x r(z) \exp\left[\int_a^z p(\eta)d\eta\right] dz\right\} \tag{20-20}$$

[11] The statistical definition of entropy is $S(T,V) = k \log \Omega(U(T,V))$ or $\Omega(U(T,V)) = \exp(S/k)$. Entropy plays the role of integrating factor.

3.016 Home

Full Screen

Close

Quit

Lecture 20 Mathematica® Example 1

Using DSolve to solve Homogeneous and Heterogeneous ODEs

notebook (non-evaluated)   pdf (evaluated)   html (evaluated)

The solutions, Eqs. 20 and 20-20, are derived and replacement rules are used to convert the Bernoulli equation into a linear ODE.

1: DSolve solves the linear homogeneous equation first-order ODE $dy/dx + p(x)y = 0$. Two variables are introduced in the solution: one is the 'dummy-variable' of the integration in Eq. 20 which Mathematica® introduces in the form K$N and an integration constant which is given the form C[N].

2: Here, a specfic $p(x)$ is given, so the dummy variable doesn't appear...

3: Furthermore, if enough boundary conditions are given to solve for the integration constants, then the C[N] are not needed either.

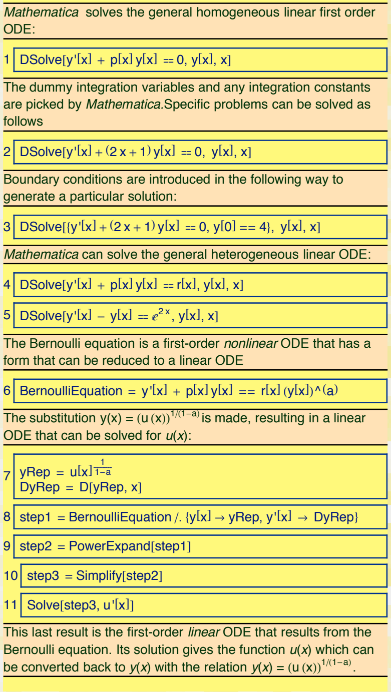
4: DSolve solves the heterogeneous equation $dy/x + p(x)y = r(x)$ to give the form Eq. 20-20. Note how the homogeneous solution is one of the terms in the sum for the heterogenous solution.

5: This is an example for a specific case: $p(x) = -1$ and $r(x) = e^{2x}$.

6: The Bernoulli equation is a non-linear first order ODE, but a series of transformations can turn it into an equivalent linear form.

7: Replacements for $y(x)$ and its derivative are defined.

11: Using the replacements, PowerExpand, Simplify, and Solve produces a linear first-order ODE for $u(x) \equiv [y(x)]^{1-a}$.



Mathematica solves the general homogeneous linear first order ODE:

```
1  DSolve[y'[x] + p[x] y[x] == 0, y[x], x]
```

The dummy integration variables and any integration constants are picked by *Mathematica.*Specific problems can be solved as follows

```
2  DSolve[y'[x] + (2 x + 1) y[x] == 0,  y[x], x]
```

Boundary conditions are introduced in the following way to generate a particular solution:

```
3  DSolve[{y'[x] + (2 x + 1) y[x] == 0, y[0] == 4},  y[x], x]
```

Mathematica can solve the general heterogeneous linear ODE:

```
4  DSolve[y'[x] + p[x] y[x] == r[x], y[x], x]
5  DSolve[y'[x] - y[x] == ⅇ^(2 x), y[x], x]
```

The Bernoulli equation is a first-order *nonlinear* ODE that has a form that can be reduced to a linear ODE

```
6  BernoulliEquation = y'[x] + p[x] y[x] == r[x] (y[x])^(a)
```

The substitution $y(x) = (u(x))^{1/(1-a)}$ is made, resulting in a linear ODE that can be solved for $u(x)$:

```
7  yRep = u[x]^(1/(1-a))
   DyRep = D[yRep, x]
8  step1 = BernoulliEquation /. {y[x] → yRep, y'[x] → DyRep}
9  step2 = PowerExpand[step1]
10 step3 = Simplify[step2]
11 Solve[step3, u'[x]]
```

This last result is the first-order *linear* ODE that results from the Bernoulli equation. Its solution gives the function $u(x)$ which can be converted back to $y(x)$ with the relation $y(x) = (u(x))^{1/(1-a)}$.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Example: The Bernoulli Equation

The linear first-order ODEs always have a closed form solution in terms of integrals. In general non-linear ODEs do not have a general expression for their solution. However, there are some non-linear equations that can be reduced to a linear form; one such case is the Bernoulli equation:

$$\frac{dy}{dx} + p(x)\, y = r(x)\, y^a \tag{20-21}$$

Reduction relies on a clever change-of-variable, let $u(x) = [y(x)]^{1-a}$, then Eq. 20-21 becomes

$$\frac{du}{dx} + (1-a)p(x)\, u = (1-a)\, r(x) \tag{20-22}$$

which is a linear heterogeneous first-order ODE and has a closed-form solution.

However, not all non-linear problems can be converted to a linear form. In these cases, numerical methods are required.

Lecture 20 MATHEMATICA® Example 2

Numerical Solutions to Non-linear First-Order ODEs

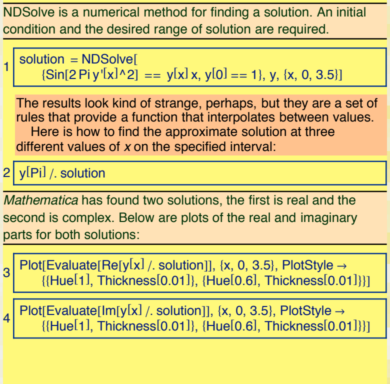
notebook (non-evaluated) pdf (evaluated) html (evaluated)

An example of computing the numerical approximation to the solution to a non-linear ODE is presented. The solutions are returned in the forms of a list of replacement rules to `InterpolatingFunction`. An `InterpolatingFunction` is a method to use numerical interpolation to extract an approximation for any point—it works just like a function and can be called on a variable like `InterpolatingFunction[0.2]`. In addition to the interpolation table, the definition specifies the domain over which the interpolation is considered valid.

1: Using NDSolve on a non-linear ODE, the solution is returned as a `InterpolatingFunction` replacement list.

2: This demonstrates how the numerical approximation is obtained at particular values.

3: In this case, two solutions are found and `Plot` called on the replacement generates two curves. Here, `Re` is used to compute the real part of the numerical approximation.



NDSolve is a numerical method for finding a solution. An initial condition and the desired range of solution are required.

```
1  solution = NDSolve[
       {Sin[2 Pi y'[x]^2] == y[x] x, y[0] == 1}, y, {x, 0, 3.5}]
```

The results look kind of strange, perhaps, but they are a set of rules that provide a function that interpolates between values.
Here is how to find the approximate solution at three different values of *x* on the specified interval:

```
2  y[Pi] /. solution
```

Mathematica has found two solutions, the first is real and the second is complex. Below are plots of the real and imaginary parts for both solutions:

```
3  Plot[Evaluate[Re[y[x] /. solution]], {x, 0, 3.5}, PlotStyle →
       {{Hue[1], Thickness[0.01]}, {Hue[0.6], Thickness[0.01]}}]

4  Plot[Evaluate[Im[y[x] /. solution]], {x, 0, 3.5}, PlotStyle →
       {{Hue[1], Thickness[0.01]}, {Hue[0.6], Thickness[0.01]}}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 21: Higher-Order Ordinary Differential Equations

Reading:
Kreyszig Sections: 2.1, 2.2 (pages45–52, 53–58)

Higher-Order Equations: Background

For first-order ordinary differential equations (ODEs), $F(y'(x), y(x), x)$, one value $y(x_o)$ was needed to specify a particular solution. Recall the example in Lecture 19 of a first-order differencing scheme: at each iteration the function grew proportionally to its current size. In the limit of very small forward differences, the scheme converged to exponential growth.

Now consider a situation in which function's current rate of growth increases proportionally to two terms: its current rate of growth and its size.

$$\text{Change in Value's Rate of Change} + \alpha\ (\text{the Value}) + \beta\ (\text{Value's Rate of Change}) = 0$$

To calculate a forward differencing scheme for this case, let Δ be the forward-differencing increment.

$$\left(\frac{\frac{F_{i+2}-F_{i+1}}{\Delta} - \frac{F_{i+1}-F_i}{\Delta}}{\Delta}\right) + \alpha F_i + \beta\left(\frac{F_{i+1}-F_i}{\Delta}\right) = 0$$

and then solve for the "next increment" F_{i+2} if F_{i+1} and F_i are known.

This indicates that, for second-order equations, two independent values are needed to generate the 'solution trajectory.'

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 21 MATHEMATICA® Example 1

A Second-Order Forward Differencing Example

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A second order differencing formula is developed for the case of constant growth and acceleration coefficients.

1: *CurrentChangeperDelta* is an example of a first-order finite difference.

2: Applying the first-order difference operator twice, a second-order differencing operator is obtained. Notice that, as the higher the order of difference operation goes, the number of surrounding points required to evaluate the difference gets larger and larger—i.e., for the second order difference, function values are needed at three different i compared to two different i for the first-order case.

3: For a particular case of $d^2y/dx^2 = -\alpha dy/dx - \beta y$, the two difference operators replace the derivatives and a *difference relation* can be derived as a function of parameters α and β.

4: The difference operator is derived by solving the difference relation for F_{i+2}—it will depend on the immediate last value F_{i+1} and that value's antecedent F_i. Therefore, any value—including the first one calculated—requires *two values* to be specified.

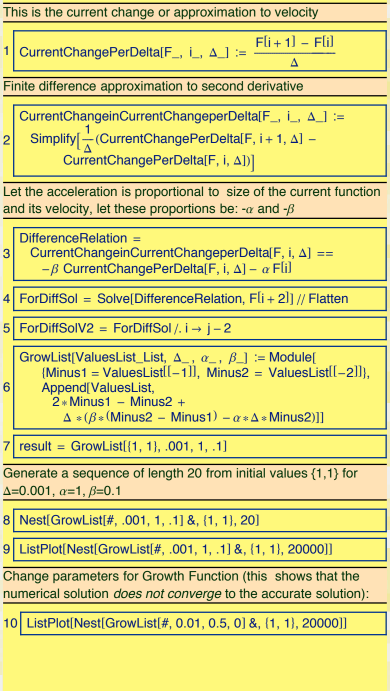
5: Typically, the current j–value is expressed in terms of the $(j-1)$ and $(j-2)$–values. This form is generated by the replacement $i \rightarrow j-2$.

6: The difference operator is incorporated in *GrowList* : a function that grows a list (input as `ValuesList`) using a difference Δ and parameters α and β. The two previous values in the list become *localized variables* in a `Module` function. The `Module` returns a new list that is created using `Append` to place the current value at end of the input list.

7: Here is an example of using *GrowList* once.

8: Using `Nest` the list can be grown iteratively to N times to generate a sequence of length $N+2$ (the first two values being specified).

10: `ListPlot` visualizes the results for different growth constants α and β.



This is the current change or approximation to velocity

```
1  CurrentChangePerDelta[F_, i_, Δ_] := (F[i + 1] - F[i])/Δ
```

Finite difference approximation to second derivative

```
2  CurrentChangeinCurrentChangeperDelta[F_, i_, Δ_] :=
     Simplify[1/Δ (CurrentChangePerDelta[F, i + 1, Δ] -
           CurrentChangePerDelta[F, i, Δ])]
```

Let the acceleration is proportional to size of the current function and its velocity, let these proportions be: -α and -β

```
3  DifferenceRelation =
     CurrentChangeinCurrentChangeperDelta[F, i, Δ] ==
       -β CurrentChangePerDelta[F, i, Δ] - α F[i]

4  ForDiffSol = Solve[DifferenceRelation, F[i + 2]] // Flatten

5  ForDiffSolV2 = ForDiffSol /. i → j - 2

6  GrowList[ValuesList_List, Δ_ , α_ , β_] := Module[
       {Minus1 = ValuesList[[-1]], Minus2 = ValuesList[[-2]]},
       Append[ValuesList,
         2 * Minus1 - Minus2 +
           Δ * (β * (Minus2 - Minus1) - α * Δ * Minus2)]]

7  result = GrowList[{1, 1}, .001, 1, .1]
```

Generate a sequence of length 20 from initial values {1,1} for Δ=0.001, α=1, β=0.1

```
8  Nest[GrowList[#, .001, 1, .1] &, {1, 1}, 20]

9  ListPlot[Nest[GrowList[#, .001, 1, .1] &, {1, 1}, 20000]]
```

Change parameters for Growth Function (this shows that the numerical solution *does not converge* to the accurate solution):

```
10  ListPlot[Nest[GrowList[#, 0.01, 0.5, 0] &, {1, 1}, 20000]]
```

Linear Differential Equations; Superposition in the Homogeneous Case

A linear differential equation is one for which the function and its derivatives are each linear—that is they appear in distinct terms and only to the first power. In the case of a homogeneous linear differential equation, the solutions are *superposable.* In other words, sums of solutions and their multiples are also solutions.

Therefore, a linear heterogeneous ordinary differential equation can be written as a product of general functions of the dependent variable and the derivatives for the n-order linear case:

$$\begin{aligned}
0 &= f_0(x) + f_1(x)\frac{dy}{dx} + f_2(x)\frac{d^2y}{dx^2} + \cdots + f_n(x)\frac{d^ny}{dx^n} \\
&= (f_0(x), f_1(x), f_2(x), \ldots, f_n(x)) \cdot \left(1, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \ldots, \frac{d^ny}{dx^n}\right) \\
&= \vec{f}(x) \cdot \vec{D_n y}
\end{aligned} \tag{21-1}$$

The homogeneous n^{th}-order linear ordinary differential equation is defined by $f_0(x) = 0$ in Eq. 21-1:

$$\begin{aligned}
0 &= f_1(x)\frac{dy}{dx} + f_2(x)\frac{d^2y}{dx^2} + \cdots + f_n(x)\frac{d^ny}{dx^n} \\
&= (0, f_1(x), f_2(x), \cdots, f_n(x)) \cdot \left(1, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \ldots, \frac{d^ny}{dx^n}\right) \\
&= \vec{f_{hom}}(x) \cdot \vec{D_n y}
\end{aligned} \tag{21-2}$$

Equation 21-1 can always be multiplied by $1/f_n(x)$ to generate the general form:

$$\begin{aligned}
0 &= F_0(x) + F_1(x)\frac{dy}{dx} + F_2(x)\frac{d^2y}{dx^2} + \cdots + \frac{d^ny}{dx^n} \\
&= (F_0(x), F_1(x), F_2(x), \ldots, 1)) \cdot (1, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \ldots, \frac{d^ny}{dx^n}) \\
&= \vec{F}(x) \cdot \vec{D_n y}
\end{aligned} \tag{21-3}$$

For the second-order linear ODE, the heterogeneous form can always be written as:

$$\frac{d^2y}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = r(x) \tag{21-4}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

and the homogeneous second-order linear ODE is:

$$\frac{d^2y}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = 0 \tag{21-5}$$

Basis Solutions for the homogeneous second-order linear ODE

Because two values must be specified for each solution to a second order equation—the solution can be broken into two basic parts, each deriving from a different constant. These two independent solutions form a *basis pair* for any other solution to the homogeneous second-order linear ODE that derives from any other pair of specified values.

The idea is the following: suppose the solution to Eq. 21-5 is found the particular case of specified parameters $y(x = x_0) = A_0$ and $y(x = x_1) = A_1$, the solution $y(x; A_0, A_1)$ can be written as the sum of solutions to two *other problems*.

$$y(x; A_0, A_1) = y(x, A_0, 0) + y(x, 0, A_1) = y_1(x) + y_2(x) \tag{21-6}$$

where

$$\begin{array}{rcl} y(x_0, A_0, 0) = A_0 & \quad\text{and}\quad & y(x_1, A_0, 0) = 0 \\ y(x_0, 0, A_1) = 0 & \quad\text{and}\quad & y(x_1, 0, A_1) = A_1 \end{array} \tag{21-7}$$

from these two solutions, any others can be generated.

The two arbitrary integration constants can be included in the definition of the general solution:

$$\begin{aligned} y(x) &= C_1 y_1(x) + C_1 y_2(x) \\ &= (C_1, C_2) \cdot (y_1, y_2) \end{aligned} \tag{21-8}$$

Second Order ODEs with Constant Coefficients

The most simple case—but one that results from models of many physical phenomena—is that functions in the homogeneous second-order linear ODE (Eq. 21-5) are constants:

$$a\frac{d^2y}{dx^2} + b\frac{dy}{dx} + cy = 0 \tag{21-9}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

If two independent solutions can be obtained, then any solution can be formed from this basis pair.

Surmising solutions seems a sensible strategy, certainly for shrewd solution seekers. Suppose the solution is of the form $y(x) = \exp(\lambda x)$ and put it into Eq. 21-9:

$$(a\lambda^2 + b\lambda + c)e^{\lambda x} = 0 \tag{21-10}$$

which has solutions when and only when the quadratic equation $a\lambda^2 + \lambda x + c = 0$ has solutions for λ.

Because two solutions are needed and because the quadratic equation yields two solutions:

$$\begin{aligned} \lambda_+ &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} \\ \lambda_- &= \frac{-b - \sqrt{b^2 - 4ac}}{2a} \end{aligned} \tag{21-11}$$

or by removing the redundant coefficient by diving through by a:

$$\begin{aligned} \lambda_+ &= \frac{-\beta}{2} + \sqrt{(\frac{\beta}{2})^2 - \gamma} \\ \lambda_- &= \frac{-\beta}{2} - \sqrt{(\frac{\beta}{2})^2 - \gamma} \end{aligned} \tag{21-12}$$

where $\beta \equiv b/a$ and $\gamma \equiv c/a$.

Therefore, any solution to Eq. 21-9 can be written as

$$y(x) = C_+ e^{\lambda_+ x} + C_- e^{\lambda_- x} \tag{21-13}$$

This solution recreated with a slightly different method in the following MATHEMATICA® example.

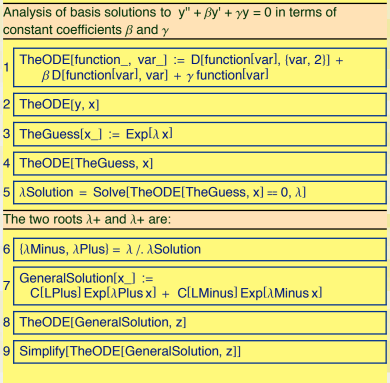
3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

Lecture 21 MATHEMATICA® Example 2

Solutions to the Homogeneous Linear Second Order ODE with Constant Coefficients

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Even though MATHEMATICA® is able to determine solutions to linear second-order ODEs with constant coefficients directly, it is still instructive to use MATHEMATICA® to derive these solutions.



Analysis of basis solutions to y'' + βy' + γy = 0 in terms of constant coefficients β and γ

```
1 TheODE[function_, var_] := D[function[var], {var, 2}] +
      β D[function[var], var] + γ function[var]
2 TheODE[y, x]
3 TheGuess[x_] := Exp[λ x]
4 TheODE[TheGuess, x]
5 λSolution = Solve[TheODE[TheGuess, x] == 0, λ]
```

The two roots λ+ and λ+ are:

```
6 {λMinus, λPlus} = λ /. λSolution
7 GeneralSolution[x_] :=
    C[LPlus] Exp[λPlus x] + C[LMinus] Exp[λMinus x]
8 TheODE[GeneralSolution, z]
9 Simplify[TheODE[GeneralSolution, z]]
```

1: *TheODE* represents the left-hand side of any second-order ODE with constant coefficients. It takes an argument for the name of the function (i.e., y) and the dependent variable (i.e., x in $y(x)$).

3: This will serve as a 'guess' of a solution—if we can find λ(s) that satisfy the ODE, then the solution(s) are determined.

5: Using `Solve` with the guess inserted into *TheODE* will determine solutiion conditions on λ—this will be a quadratic equation in λ.

6: By inspecting the solution, assignments can be made to the two possible λ.

7: This is the form of the general solution in terms of two arbitrary constants.

9: This should show that the general solution always satisfies the ODE.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 21 Mathematica® Example 3

Characterizing the Solution Behavior for the Second-Order ODE with Constant Coefficients

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Because the fundamental solution depend on only two parameters β and γ, the behavior (i.e., whether $\Re\lambda \overset{>}{<} 0$ and $\Im\lambda \overset{?}{=} 0$) of all solutions can be visualized in the γ-β plane.

1: `Reduce` is a function for determing the conditions on parameters (here β and γ assumed to be real numbers) such that an expression satisfies particular constraints. The result will create the following graphic.

2: This will create a plot that distinguishes two regions in the γ-β plane: above $\gamma = \beta^2/4$, the λ are real; below, the λ are complex and oscillatory solutions appear (because $\exp(r + \imath\theta) = \exp(r)(\cos(\theta) + \imath\sin(\theta))$).

3: `Graphics` and `Text` create annotation; `Show` combines annotation and the plot.

4: This will create a plot for the conditions that the $\Re(\lambda)$ are real and either positive or negative. The sign of the real part of λ in $\exp(\lambda x)$ determines whether the solution grows without bound ($\Re(\lambda) > 0$) or shrinks asympotically towards 0 ($\Re\lambda < 0$).

5: This creates graphics to annotate and display together with the plot. The extra function `StyleForm` allows the passage of options (such as `FontColor`, `FontFamily`, etc.) to be passed simply.

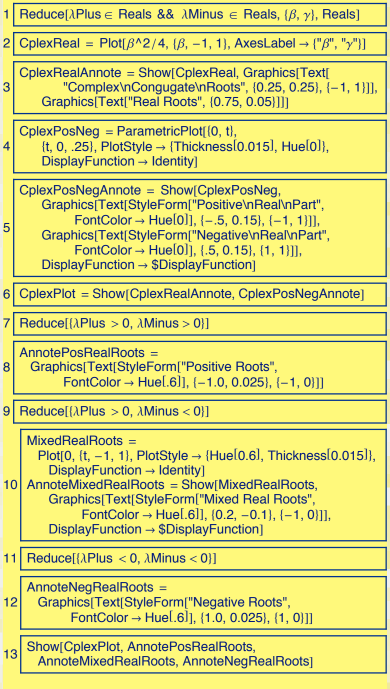
7: `Reduce` determines the conditions on β and γ so that both λ are positive and real. These will be unbounded and non-oscillating solutions.

8: The conditions will be annotated by greating a graphical object.

10: Here the curves and annotation are created for the case of mixed real roots (i.e., $\lambda_+ > 0$ and $\lambda_- < 0$—one growing and one decaying non-oscillatory solutions)

12: The final region to be determined and annotated is the one with the monotonically decaying solutions $\lambda_- < \lambda_- < 0$.

13: Collecting all the graphical objects together into one image that was used to construct Fig. 21-21.



```
1  Reduce[λPlus ∈ Reals && λMinus ∈ Reals, {β, γ}, Reals]

2  CplexReal = Plot[β^2/4, {β, −1, 1}, AxesLabel → {"β", "γ"}]

3  CplexRealAnnote = Show[CplexReal, Graphics[Text[
           "Complex\nCongugate\nRoots", {0.25, 0.25}, {−1, 1}]],
       Graphics[Text["Real Roots", {0.75, 0.05}]]]

4  CplexPosNeg = ParametricPlot[{0, t},
       {t, 0, .25}, PlotStyle → {Thickness[0.015], Hue[0]},
       DisplayFunction → Identity]

5  CplexPosNegAnnote = Show[CplexPosNeg,
       Graphics[Text[StyleForm["Positive\nReal\nPart",
           FontColor → Hue[0]], {−.5, 0.15}, {−1, 1}]],
       Graphics[Text[StyleForm["Negative\nReal\nPart",
           FontColor → Hue[0]], {.5, 0.15}, {1, 1}]],
       DisplayFunction → $DisplayFunction]

6  CplexPlot = Show[CplexRealAnnote, CplexPosNegAnnote]

7  Reduce[{λPlus > 0, λMinus > 0}]

8  AnnotePosRealRoots =
     Graphics[Text[StyleForm["Positive Roots",
         FontColor → Hue[.6]], {−1.0, 0.025}, {−1, 0}]]

9  Reduce[{λPlus > 0, λMinus < 0}]

10 MixedRealRoots =
     Plot[0, {t, −1, 1}, PlotStyle → {Hue[0.6], Thickness[0.015]},
       DisplayFunction → Identity]
   AnnoteMixedRealRoots = Show[MixedRealRoots,
       Graphics[Text[StyleForm["Mixed Real Roots",
           FontColor → Hue[.6]], {0.2, −0.1}, {−1, 0}]],
       DisplayFunction → $DisplayFunction]

11 Reduce[{λPlus < 0, λMinus < 0}]

12 AnnoteNegRealRoots =
     Graphics[Text[StyleForm["Negative Roots",
         FontColor → Hue[.6]], {1.0, 0.025}, {1, 0}]]

13 Show[CplexPlot, AnnotePosRealRoots,
     AnnoteMixedRealRoots, AnnoteNegRealRoots]
```

3.016 Home

Full Screen

Close

Quit

The behavior of all solutions can be collected into a simple picture:

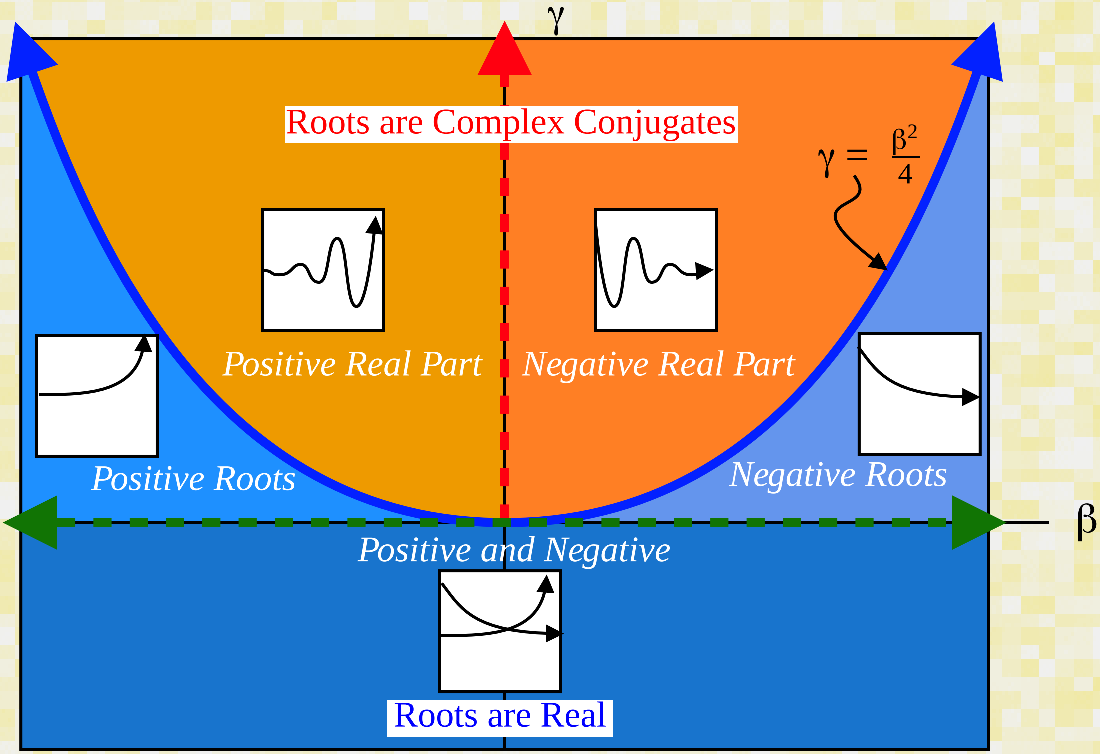


Figure 21-21: The behaviors of the linear homogeneous second-order ordinary differential equation $\frac{d^2y}{dx^2} + \beta\frac{dy}{dx} + \gamma y = 0$ plotted according the behavior of the solutions for all β and γ.

The case that separates the complex solutions from the real solutions, $\gamma = (\beta/2)^2$ must be treated separately, for the case $\gamma = (\beta/2)^2$ it can be shown that $y(x) = \exp(\beta x/2)$ and $y(x) = x\exp(\beta x/2)$ form an independent basis pair (see Kreyszig *AEM*, p. 74).

Boundary Value Problems

It has been shown that all solutions to $\frac{d^2y}{dx^2} + \beta\frac{dy}{dx} + \gamma y = 0$ can be determined from a linear combination of the basis solution. Disregard for a moment whether the solution is complex or real, and ignoring the special case $\gamma = (\beta/2)^2$. The solution to any problem is given by

$$y(x) = C_+ e^{\lambda_+ x} + C_- e^{\lambda_- x} \tag{21-14}$$

How is a solution found for a particular problem? Recall that *two values* must be specified to get a solution—these two values are just enough so that the two constants C_+ and C_- can be obtained.

In many physical problems, these two conditions appear at the boundary of the domain. A typical problem is posed like this:

Solve

$$m\frac{d^2y(x)}{dx^2} + \nu\frac{dy(x)}{dx} + ky(x) = 0 \qquad \text{on } 0 < x < L \tag{21-15}$$

subject to the boundary conditions

$$y(x=0) = 0 \qquad \text{and} \qquad y(x=L) = 1$$

or, solve

$$m\frac{d^2y(x)}{dx^2} + \nu\frac{dy(x)}{dx} + ky(x) = 0 \qquad \text{on } 0 < x < \infty \tag{21-16}$$

subject to the boundary conditions

$$y(x=0) = 1 \qquad \text{and} \qquad y'(x=L) = 0$$

When the value of the function is specified at a point, these are called *Dirichlet* conditions; when the derivative is specified, the boundary condition is called a *Neumann* condition. It is possible have boundary conditions that are mixtures of Dirichlet and Neumann.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

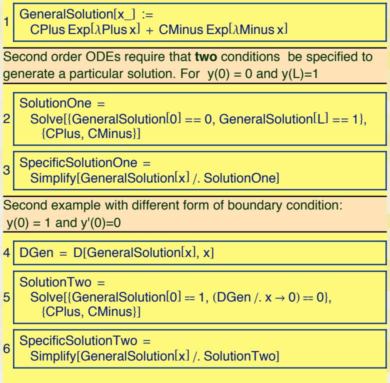
Quit

Lecture 21 Mathematica® Example 4

Determining Solution Constants from Boundary Values

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Here is an example of taking the general solution with undetermined constants and using boundary conditions to determine a specific solution.



```
1 GeneralSolution[x_] :=
    CPlus Exp[λPlus x] + CMinus Exp[λMinus x]
```

Second order ODEs require that **two** conditions be specified to generate a particular solution. For y(0) = 0 and y(L)=1

```
2 SolutionOne =
    Solve[{GeneralSolution[0] == 0, GeneralSolution[L] == 1},
      {CPlus, CMinus}]
3 SpecificSolutionOne =
    Simplify[GeneralSolution[x] /. SolutionOne]
```

Second example with different form of boundary condition: y(0) = 1 and y'(0)=0

```
4 DGen = D[GeneralSolution[x], x]
5 SolutionTwo =
    Solve[{GeneralSolution[0] == 1, (DGen /. x → 0) == 0},
      {CPlus, CMinus}]
6 SpecificSolutionTwo =
    Simplify[GeneralSolution[x] /. SolutionTwo]
```

1: *GeneralSolution* is the solution to $y'' + \beta y' + \gamma y = 0$ with undetermined constants `Cplus` and `Cminus`.

2: To find the constants for a particular solution the boundary conditions, $y(0) = 0$ and $y(L) = 1$ where $y(x)$ is the general solution, are used with `Solve` to determine the constants.

3: The form of the particular solution is obtained by back-substituting the solution for the constants into the general solution.

4: For application of a Neumann condition, the symbolic form of the derivative is required.

6: The particular solution for boundary conditions $y'(0) = y(0) = 0$ is obtained by inserting these equations into `Solve` and subsequent replacement into the general solution.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Fourth Order ODEs, Elastic Beams

Another linear ODE that has important applications in materials science is that for the deflection of a beam. The beam deflection $y(x)$ is a linear fourth-order ODE:

$$\frac{d^2}{dx^2}\left(EI\frac{d^2y(x)}{dx^2}\right) = w(x) \tag{21-17}$$

where $w(x)$ is the load density (force per unit length of beam), E is Young's modulus of elasticity for the beam, and I is the moment of inertia of the cross section of the beam:

$$I = \int_{\mathrm{A}_{\times - sect}} y^2 dA \tag{21-18}$$

is the second-moment of the distribution of heights across the area.

If the moment of inertia and the Young's modulus do not depend on the position in the beam (the case for a uniform beam of homogeneous material), then the beam equation becomes:

$$EI\frac{d^4y(x)}{dx^4} = w(x) \tag{21-19}$$

The homogeneous solution can be obtained by inspection—it is a general cubic equation $y_{homog}(x) = C_0 + C_1x + C_2x^2 + C_3x^3$ which has the four constants that are expected from a fourth-order ODE.

The particular solution can be obtained by integrating $w(x)$ four times—if the constants of integration are included then the particular solution naturally contains the homogeneous solution.

The load density can be discontinuous or it can contain Dirac-delta functions $F_o\delta(x - x_o)$ representing a point load F_o applied at $x = x_o$.

It remains to determine the constants from boundary conditions. The boundary conditions can be determined because each derivative of $y(x)$ has a specific meaning as illustrated in Fig. 21-22.

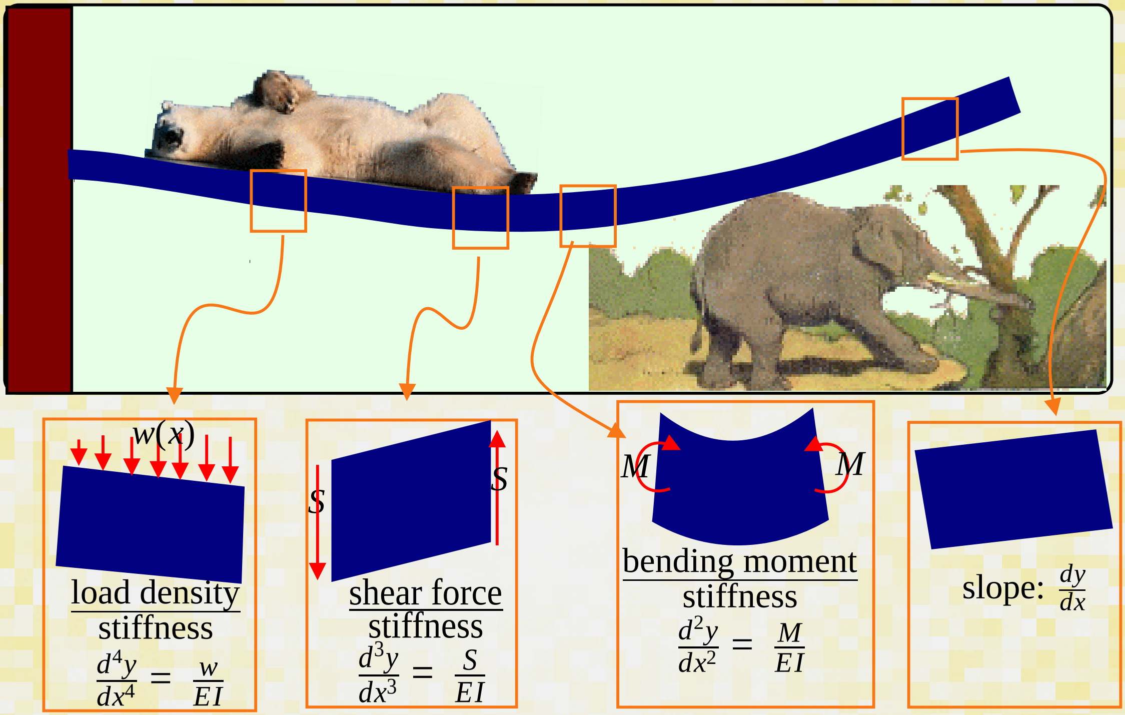
3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit



$$\frac{\text{load density}}{\text{stiffness}} \qquad \frac{d^4y}{dx^4} = \frac{w}{EI}$$

$$\frac{\text{shear force}}{\text{stiffness}} \qquad \frac{d^3y}{dx^3} = \frac{S}{EI}$$

$$\frac{\text{bending moment}}{\text{stiffness}} \qquad \frac{d^2y}{dx^2} = \frac{M}{EI}$$

$$\text{slope: } \frac{dy}{dx}$$

Figure 21-22: The shape of a loaded beam is determined by the loads applied over its length and its boundary conditions. The beam curvature is related to the local moment (imagine two handles rotated in opposite directions on a free beam) divided by the effective beam stiffness. Shear forces are related to the rate of change of moment along the beam.
(Polar Bear Photo Art Wolfe The Zone Network
http://classic.mountainzone.com/climbing/greenland/graphics/polar-bear.html)

There are common loading conditions that determine boundary conditions:

Free No applied moments or applied shearing force:

$$M = \left.\frac{d^2y}{dx^2}\right|_{boundary} = 0$$

$$S = \left.\frac{d^3y}{dx^3}\right|_{boundary} = 0$$

Point Loaded local applied moment, displacement specified.

$$M = \left.\frac{d^2y}{dx^2}\right|_{boundary} = M_o$$

$$y(x)|_{boundary} = y_o$$

Clamped Displacement specified, slope specified

$$\left.\frac{dy}{dx}\right|_{boundary} = s_o$$

$$y(x)|_{boundary} = y_o$$

Lecture 21 MATHEMATICA® Example 5

Visualizing Beam Deflections

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A method for solving and visualizing the deflection of a uniform beam is developed for typical boundary conditions and load distributions

1: *BeamEquation* takes arguments for the (unknown) deflection y and its dependent argument x, a loading density $w(x)$, and boundary condition lists `BC1` and `BC2`, and uses `DSolve` to return replacement rules for a particular solution to the beam deflection equation (i.e., $d^4y/dx^4 = w(x)$).

2: *Clamp* , *PointLoad* , and *FreeEnd* are functions that specify the typical boundary conditions: *Clamp* takes an x-value where the clamp is applied, the displacement (`position`) of the clamp and the clamps angle. *PointLoad* takes a position, deflection, and applied torque; *FreeEnd* specifies a point that is unloaded and untorqued.

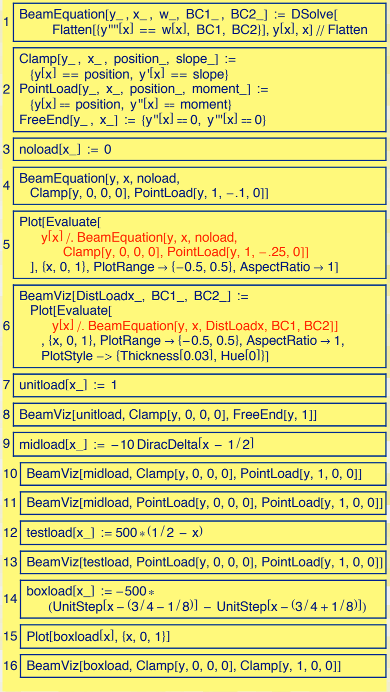
3: *noload* is an example loading distribution where there is no applied load.

4: As an example of application of *BeamEquation* , here the solution for an unloaded beam is calculated with a fixed horizontal clamp at the origin and a fixed torque-free displacent at the end.

5: To plot the beam deflection, the solution condition is applied to $y(x)$.

6: The function *BeamViz* collects the solution with the visualization for beams of unit normalized length, and uniform normalized stiffness EI.

7: etc. Several different loading conditions and boundary conditions are visualized as examples of *BeamViz*



```
1  BeamEquation[y_ , x_ , w_, BC1_ , BC2_] := DSolve[
       Flatten[{y''''[x] == w[x], BC1, BC2}], y[x], x] // Flatten

2  Clamp[y_ , x_ , position_, slope_] :=
     {y[x] == position, y'[x] == slope}
   PointLoad[y_, x_, position_, moment_] :=
     {y[x] == position, y''[x] == moment}
   FreeEnd[y_ , x_] := {y''[x] == 0, y'''[x] == 0}

3  noload[x_] := 0

4  BeamEquation[y, x, noload,
     Clamp[y, 0, 0, 0], PointLoad[y, 1, -.1, 0]]

5  Plot[Evaluate[
       y[x] /. BeamEquation[y, x, noload,
           Clamp[y, 0, 0, 0], PointLoad[y, 1, -.25, 0]]
     ], {x, 0, 1}, PlotRange → {-0.5, 0.5}, AspectRatio → 1]

6  BeamViz[DistLoadx_, BC1_, BC2_] :=
     Plot[Evaluate[
         y[x] /. BeamEquation[y, x, DistLoadx, BC1, BC2]]
       , {x, 0, 1}, PlotRange → {-0.5, 0.5}, AspectRatio → 1,
       PlotStyle -> {Thickness[0.03], Hue[0]}]

7  unitload[x_] := 1

8  BeamViz[unitload, Clamp[y, 0, 0, 0], FreeEnd[y, 1]]

9  midload[x_] := -10 DiracDelta[x - 1/2]

10 BeamViz[midload, Clamp[y, 0, 0, 0], PointLoad[y, 1, 0, 0]]

11 BeamViz[midload, PointLoad[y, 0, 0, 0], PointLoad[y, 1, 0, 0]]

12 testload[x_] := 500 * (1/2 - x)

13 BeamViz[testload, PointLoad[y, 0, 0, 0], PointLoad[y, 1, 0, 0]]

14 boxload[x_] := -500 *
       (UnitStep[x - (3/4 - 1/8)] - UnitStep[x - (3/4 + 1/8)])

15 Plot[boxload[x], {x, 0, 1}]

16 BeamViz[boxload, Clamp[y, 0, 0, 0], Clamp[y, 1, 0, 0]]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 22: Differential Operators, Harmonic Oscillators

Reading:
Kreyszig Sections: 2.3,2.4, 2.7 (pages59–60, 61–69, 78–83)

Differential Operators

The idea of a function as "something" that takes a value (real, complex, vector, etc.) as "input" and returns "something else" as "output" should be very familiar and useful.

This idea can be generalized to *operators* that take a function as an argument and return another function.

The derivative operator operates on a function and returns another function that describes how the function changes:

$$\begin{aligned}
\mathcal{D}[f(x)] &= \frac{df}{dx} \\
\mathcal{D}[\mathcal{D}[f(x)]] &= \mathcal{D}^2[f(x)] = \frac{d^2 f}{dx^2} \\
\mathcal{D}^n[f(x)] &= \frac{d^n f}{dx^n} \\
\mathcal{D}[\alpha f(x)] &= \alpha D[f(x)] \\
\mathcal{D}[f(x) + g(x)] &= D[f(x)] + D[g(x)]
\end{aligned} \tag{22-1}$$

The last two equations above indicate that the "differential operator" is a linear operator.

The integration operator is the right-inverse of $\mathcal{D}$

$$\mathcal{D}[\mathcal{I}[f(x)]] = \mathcal{D}[\int f(x)dx] \tag{22-2}$$

but is only the left-inverse up to an arbitrary constant.

3.016 Home

Full Screen

Close

Quit

Consider the differential operator that returns a constant multiplied by itself

$$\mathcal{D}f(x) = \lambda f(x) \tag{22-3}$$

which is another way to write the the homogenous linear first-order ODE and has the same form as an eigenvalue equation. In fact, $f(x) = \exp(\lambda x)$, can be considered an *eigenfunction* of Eq. 22-3.

For the homogeneous second-order equation,

$$\left(\mathcal{D}^2 + \beta\mathcal{D} - \gamma\right)[f(x)] = 0 \tag{22-4}$$

It was determined that there were two eigensolutions that can be used to span the entire solution space:

$$f(x) = C_+ e^{\lambda_+ x} + C_- e^{\lambda_- x} \tag{22-5}$$

Operators can be used algebraically, consider the inhomogeneous second-order ODE

$$\left(a\mathcal{D}^2 + b\mathcal{D} + c\right)[y(x)] = x^3 \tag{22-6}$$

By treating the operator as an algebraic quantity, a solution can be found[12]

$$\begin{aligned} y(x) &= \left(\frac{1}{a\mathcal{D}^2 + b\mathcal{D} + c}\right)[x^3] \\ &= \left(\frac{1}{c} - \frac{b}{c^2}\mathcal{D} + \frac{b^2 - ac}{c^3}\mathcal{D}^2 - \frac{b(b^2 - 2ac)}{c^3}\mathcal{D}^3 + \mathcal{O}(\mathcal{D}^4)\right)x^3 \\ &= \frac{x^3}{c} - \frac{3bx^2}{c^2} + \frac{6(b^2 - ac)x}{c^3} - \frac{6b(b^2 - 2ac)}{c^3} \end{aligned} \tag{22-7}$$

which solves Eq. 22-6.

The Fourier transform is also a linear operator:

$$\begin{aligned} \mathcal{F}[f(x)] =& g(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{\imath kx} dx \\ \mathcal{F}^{-1}[g(k)] =& f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(k) e^{-\imath kx} dk \end{aligned} \tag{22-8}$$

[12] This method can be justified by plugging back into the original equation and verifying that the result is a solution.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Combining operators is another useful way to solve differential equations. Consider the Fourier transform, $\mathcal{F}$, operating on the differential operator, $\mathcal{D}$:

$$\mathcal{F}[\mathcal{D}[f]] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{df(x)}{dx} e^{ikx} dx \tag{22-9}$$

Integrating by parts,

$$= \frac{1}{\sqrt{2\pi}} f(x) \,\Big|_{x=-\infty}^{x=\infty} - \frac{\imath k}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{ikx} dx \tag{22-10}$$

If the Fourier transform of $f(x)$ exists, then *typically*[13] $\lim_{x\rightarrow\pm\infty} f(x) = 0$. In this case,

$$\mathcal{F}[\mathcal{D}[f]] = -ik\mathcal{F}[f(x)] \tag{22-11}$$

and by extrapolation:

$$\begin{aligned} \mathcal{F}[\mathcal{D}^2[f]] &= -k^2\mathcal{F}[f(x)] \\ \mathcal{F}[\mathcal{D}^n[f]] &= (-1)^n \imath^n k^n \mathcal{F}[f(x)] \end{aligned} \tag{22-12}$$

Operational Solutions to ODEs

Consider the heterogeneous second-order linear ODE which represent a forced, damped, harmonic oscillator that will be discussed later in this lecture.

$$M\frac{d^2y(t)}{dt^2} + V\frac{dy(t)}{dt} + K_s y(t) = \cos(\omega_o t) \tag{22-13}$$

[13] It is not necessary that $\lim_{x\rightarrow\pm\infty} f(x) = 0$ for the Fourier transform to exist but it is satisfied in most every case. The condition that the Fourier transform exists is that

$$\int_{-\infty}^{\infty} |f(x)| dx$$

exists and is bounded.

Apply a Fourier transform (mapping from the time (t) domain to a frequency (ω) domain) to both sides of 22-13:

$$\begin{aligned}\mathcal{F}[M\frac{d^2y(t)}{dt^2} + V\frac{dy(t)}{dt} + K_s y(t)] &= \mathcal{F}[\cos(\omega_o t)] \\ -M\omega^2\mathcal{F}[y] - \imath\omega V\mathcal{F}[y] + K_s\mathcal{F}[y] &= \sqrt{\frac{\pi}{2}}\,[\delta(\omega-\omega_o)+\delta(\omega+\omega_o)]\end{aligned} \tag{22-14}$$

because the Dirac Delta functions result from taking the Fourier transform of $\cos(\omega_o t)$.

Equation 22-14 can be solved for the Fourier transform:

$$\mathcal{F}[y] = \sqrt{\frac{-\pi}{2}}\frac{[\delta(\omega-\omega_o)+\delta(\omega+\omega_o)]}{M\omega^2+\imath\omega V - K_s} \tag{22-15}$$

In other words, the particular solution Eq. 22-13 can be obtained by finding the function $y(t)$ that has a Fourier transform equal the the right-hand-side of Eq. 22-15–or, equivalently, operating with the inverse Fourier transform on the right-hand-side of Eq. 22-15.

MATHEMATICA® does have built-in functions to take Fourier (and other kinds of) integral transforms. However, using operational calculus to solve ODEs is a bit clumsy in MATHEMATICA® . Nevertheless, it may be instructive to force it—if only as an an example of using a good tool for the wrong purpose.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 22 Mathematica® Example 1

Use of Fourier Transform for Solution to the Damped-Forced Harmonic Oscillator

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A check is made to see if `FourierTransform` obeys the rules of a linear operator (Eq. 22-1) and define rule-patterns for those cases where it doesn't. Subsequently, an example the damped-forced linear harmonic oscillator is Fourier transformed, solved algebraically, and then inverse-transformed for a solution.

1: As of Mathematica® 5.0, `FourierTransform` automatically implements Eqs. 22-12. Here, `Table` is used to demonstrate this up to 24 derivatives.

2: However, this will demonstrate that the "sum-rule" isn't implemented automatically (n.b., although `Distribute` would implement this rule).

3: Define rules so that the FourierTransform acts as a linear functional operator. *ConstantRule* is an example of a `RuleDelayed` (`:>`) that will allow replacement with patterns that will be evaluated when the rule is applied with `ReplaceAll` (`/.`); in this case, a `Condition` (`/;`) is appended to the rule so that those cofactors which don't depend on the transformation variable, x, can be identified with `FreeQ` and those that depend on x can be identified with `MemberQ`. *DistributeRule* uses `Distribute` to replace the Fourier transform of a sum with a sum of Fourier transforms.

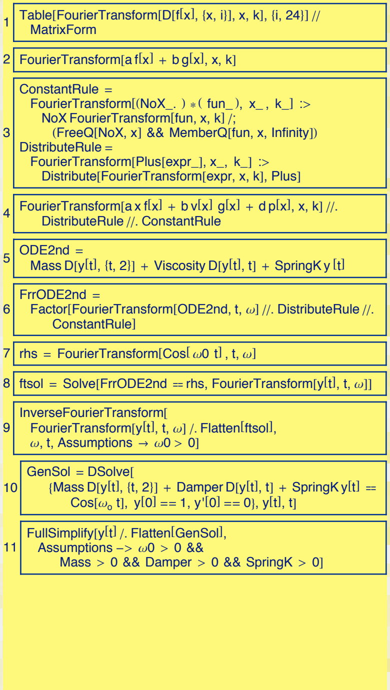
4: The linear rules are dispatched by a `ReplaceRepeated` (`//.`) that will continue to use the replacement until the result stops changing.

6: Apply the Fourier transform to the the left-hand-side damped-second-order ODE 22-13...

8: And, set the transform of the left-hand-side equal to the Fourier transform of a forcing function $\cos(\omega_o t)$. Solve for the Fourier Transform...

9: Back-transform the solution to find the particular solution to the damped forced second-order ODE.

11: This is the general solution obtained directly with `DSolve`; it is the solution to the homogeneous equation plus the particular solution that was obtained by the Fourier transform method.



```
1   Table[FourierTransform[D[f[x], {x, i}], x, k], {i, 24}] //
      MatrixForm

2   FourierTransform[a f[x] + b g[x], x, k]

3   ConstantRule =
      FourierTransform[(NoX_. ) *( fun_), x_ , k_] :>
        NoX FourierTransform[fun, x, k] /;
          (FreeQ[NoX, x] && MemberQ[fun, x, Infinity])
    DistributeRule =
      FourierTransform[Plus[expr_], x_, k_] :>
        Distribute[FourierTransform[expr, x, k], Plus]

4   FourierTransform[a x f[x] + b v[x] g[x] + d p[x], x, k] //.
        DistributeRule //. ConstantRule

5   ODE2nd =
      Mass D[y[t], {t, 2}] + Viscosity D[y[t], t] + SpringK y [t]

6   FrrODE2nd =
      Factor[FourierTransform[ODE2nd, t, ω] //. DistributeRule //.
          ConstantRule]

7   rhs = FourierTransform[Cos[ ω0 t] , t, ω]

8   ftsol = Solve[FrrODE2nd == rhs, FourierTransform[y[t], t, ω]]

9   InverseFourierTransform[
      FourierTransform[y[t], t, ω] /. Flatten[ftsol],
      ω, t, Assumptions → ω0 > 0]

10  GenSol = DSolve[
        {Mass D[y[t], {t, 2}] + Damper D[y[t], t] + SpringK y[t] ==
            Cos[ωo t], y[0] == 1, y'[0] == 0}, y[t], t]

11  FullSimplify[y[t] /. Flatten[GenSol],
      Assumptions -> ω0 > 0 &&
        Mass > 0 && Damper > 0 && SpringK > 0]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Operators to Functionals

Equally powerful is the concept of a *functional* which takes a function as an argument and returns a value. For example $\mathcal{S}[y(x)]$, defined below, operates on a function $y(x)$ and returns its surface of revolution's area for $0 < x < L$:

$$\mathcal{S}[y(x)] = 2\pi \int_0^L y\sqrt{1 + \left(\frac{dy}{dx}\right)^2}\,dx \tag{22-16}$$

This is the functional to be minimized for the question, "Of all surfaces of revolution that span from $y(x=0)$ to $y(x=L)$, which is the $y(x)$ that has the smallest surface area?"

This idea of finding "which function maximizes or minimizes something" can be very powerful and practical.

Suppose you are asked to run an "up-hill" race from some starting point $(x = 0, y = 0)$ to some ending point $(x = 1, y = 1)$ and there is a ridge $h(x, y) = x^2$. What is the most efficient running route $y(x)$?[14]

[14] An amusing variation on this problem would be to find the path that the path that a winning downhill skier should traverse.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

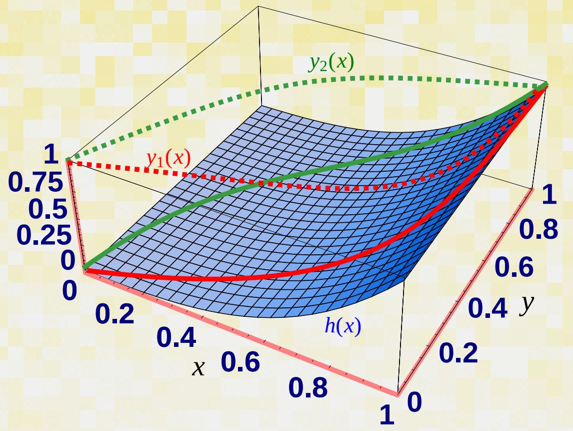


Figure 22-23: The terrain separating the starting point $(x = 0, y = 0)$ and ending point $(x = 1, y = 1)$. Assuming a model for how much running speed slows with the steepness of the path—which route would be quicker, one $(y_1(x))$ that starts going up-hill at first or another $(y_2(x))$ that initially traverses a lot of ground quickly?

A reasonable model for running speed as a function of climbing-angle α is

$$v(s) = \cos(\alpha(s)) \tag{22-17}$$

where s is the arclength along the path. The maximum speed occurs on flat ground $\alpha = 0$ and running speed monotonically falls to zero as $\alpha \rightarrow \pi/2$. To calculate the time required to traverse *any* path $y(x)$ with endpoints $y(0) = 0$ and $y(1) = 1$,

$$\begin{aligned}\frac{ds}{dt} = v(s) = \cos(\alpha(s)) = \frac{ds}{\sqrt{ds^2 + dh^2}} = \frac{1}{\sqrt{1 + \frac{dh}{ds}^2}} = \frac{1}{\sqrt{1 + \frac{dh^2}{dx^2+dy^2}}} \\ dt = \frac{ds}{v(s)} = \frac{\sqrt{dx^2 + dy^2}}{\cos(\alpha(s))} = \sqrt{dx^2 + dy^2 + dh^2} = \sqrt{1 + \frac{dy}{dx}^2 + \frac{dh}{dx}^2}\, dx\end{aligned} \tag{22-18}$$

So, with the hill $h(x) = x^2$, the time as a functional of the path is:

$$\mathcal{T}[y(x)] = \int_0^1 \sqrt{1 + \frac{dy}{dx}^2 + 4x^2}\; dx \tag{22-19}$$

There is a powerful and beautiful mathematical method for finding the extremal functions of functionals which is called *Calculus of Variations*.

By using the calculus of variations, the optimal path $y(x)$ for Eq. 22-19 can be determined:

$$y(x) = \frac{2x\sqrt{1+4x^2} + \sinh^{-1}(2x)}{2\sqrt{5} + \sinh^{-1}(2)} \tag{22-20}$$

The approximation determined in the MATHEMATICA® example above is pretty good.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 22 Mathematica® Example 2

Functionals: Introduction to Variational Calculus by Variation of Parameters

notebook (non-evaluated) pdf (evaluated) html (evaluated)

An example of *minimizing an integral of a particular form* produced where the *variational calculus* method of minimizing over all possible functions $y(x)$ is replaced by a three-parameter family of functions $y(x; a, b, c)$.

1: Here, the "hill-function" in Eq. 22-18 specified as $h(x) = x^2$ and a three-parameter family of possible trajectories $y(x; a, b, c)$ is specified: the parameters a, b, and c, that minimize the functional subject to boundary condtiions will be determined.

2: The condition cubic equation satisfies the boundary conditions is determined using `Solve` which generates a rule that fixes two of the three free parameters.

3: By integrating $y(x)$ in Eq. 22-19, the functional equation is transformed to a function of the remaining free variable. (It is faster to do the indefinite integral evaluate the limits for the definite integral in a separate step.)

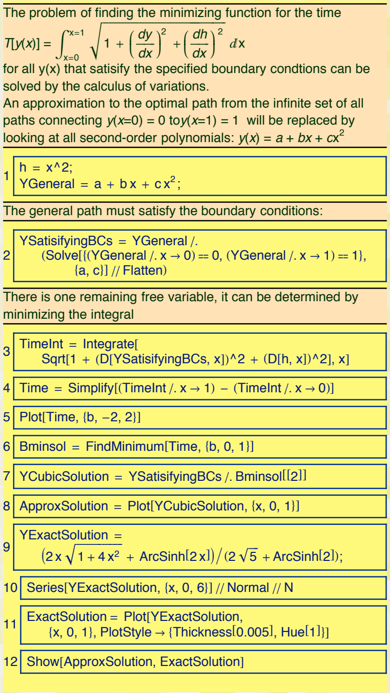
5: Plotting the time to trajectory traversal time as a function of the remaining parameter, shows there is a minimum.

6: The minimizing condition can be determined with `FindMinimum`.

7: The approximation (i.e., cubic polynomial) is fully determined by back-substitution of the mininimality condition.

9: The exact solution can be determined by a method called the *calculus of variations* and is given here.

12: The cubic polynomial is a very good approximation to the exact solution.



The problem of finding the minimizing function for the time

$$T[y(x)] = \int_{x=0}^{x=1} \sqrt{1 + \left(\frac{dy}{dx}\right)^2 + \left(\frac{dh}{dx}\right)^2}\, dx$$

for all y(x) that satisify the specified boundary condtions can be solved by the calculus of variations.

An approximation to the optimal path from the infinite set of all paths connecting $y(x{=}0) = 0$ to$y(x{=}1) = 1$ will be replaced by looking at all second-order polynomials: $y(x) = a + bx + cx^2$

```
1  h = x^2;
   YGeneral = a + b x + c x²;
```

The general path must satisfy the boundary conditions:

```
2  YSatisifyingBCs = YGeneral /.
       (Solve[{(YGeneral /. x → 0) == 0, (YGeneral /. x → 1) == 1},
            {a, c}] // Flatten)
```

There is one remaining free variable, it can be determined by minimizing the integral

```
3  TimeInt = Integrate[
       Sqrt[1 + (D[YSatisifyingBCs, x])^2 + (D[h, x])^2], x]
4  Time = Simplify[(TimeInt /. x → 1) − (TimeInt /. x → 0)]
5  Plot[Time, {b, −2, 2}]
6  Bminsol = FindMinimum[Time, {b, 0, 1}]
7  YCubicSolution = YSatisifyingBCs /. Bminsol[[2]]
8  ApproxSolution = Plot[YCubicSolution, {x, 0, 1}]
9  YExactSolution =
       (2 x Sqrt[1 + 4 x²] + ArcSinh[2 x]) / (2 Sqrt[5] + ArcSinh[2]);
10 Series[YExactSolution, {x, 0, 6}] // Normal // N
11 ExactSolution = Plot[YExactSolution,
       {x, 0, 1}, PlotStyle → {Thickness[0.005], Hue[1]}]
12 Show[ApproxSolution, ExactSolution]
```

3.016 Home

Full Screen

Close

Quit

Harmonic Oscillators

Methods for finding general solution to the linear inhomogeneous second-order ODE

$$a\frac{d^2y(t)}{dt^2} + b\frac{dy(t)}{dt} + cy(t) = F(t) \tag{22-21}$$

have been developed and worked out in MATHEMATICA® examples.

Eq. 22-21 arises frequently in physical models, among the most common are:

$$\begin{aligned} \text{Electrical circuits:} \qquad & L\frac{d^2I(t)}{dt^2} + \rho l_o\frac{dI(t)}{dt} + \frac{1}{C}I(t) = V(t) \\ \text{Mechanical oscillators:} \qquad & M\frac{d^2y(t)}{dt^2} + \eta l_o\frac{dy(t)}{dt} + K_s y(t) = F_{app}(t) \end{aligned} \tag{22-22}$$

where:

	Mechanical	Electrical
Second Order	**Mass M**: Physical measure of the ratio of momentum field to velocity	**Inductance L**: Physical measure of the ratio of stored magnetic field to current
First Order	**Drag Coefficient $c = \eta l_o$** (η is viscosity l_o is a unit displacement): Physical measure of the ratio environmental resisting forces to velocity—or proportionality constant for energy dissipation with square of velocity	**Resistance $R = \rho l_o$** (ρ is resistance per unit material length l_o is a unit length): Physical measure of the ratio of voltage drop to current—or proportionality constant for power dissipated with square of the current.
Zeroth Order	**Spring Constant K_s**: Physical measure of the ratio environmental force developed to displacement—or proportionality constant for energy stored with square of displacement	**Inverse Capacitance $1/C$**: Physical measure of the ratio of voltage storage rate to current—or proportionality constant for energy storage rate dissipated with square of the current.
Forcing Term	**Applied Voltage $V(t)$**: Voltage applied to circuit as a function of time.	**Applied Force $F(t)$**: Force applied to oscillator as a function of time.

For the homogeneous equations (i.e. no applied forces or voltages) the solutions for physically allowable values of the coefficients can either be oscillatory, oscillatory with damped amplitudes, or,

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

completely damped with no oscillations. (See Figure 21-21). The homogeneous equations are sometimes called *autonomous* equations—or *autonomous systems.*

Simple Undamped Harmonic Oscillator

The simplest version of a homogeneous Eq. 22-21 with no damping coefficient ($b = 0$, $R = 0$, or $\eta = 0$) appears in a remarkably wide variety of physical models. This simplest physical model is a simple harmonic oscillator—composed of a mass accelerating with a linear spring restoring force:

$$\begin{aligned} \text{Inertial Force} &= \text{Restoring Force} \\ M\,\text{Acceleration} &= \text{Spring Force} \\ M\frac{d^2y(t)}{dt^2} &= -K_s y(t) \\ M\frac{d^2y(t)}{dt^2} + K_s y(t) &= 0 \end{aligned} \tag{22-23}$$

Here y is the displacement from the equilibrium position–i.e., the position where the force, $F = -dU/dx = 0$. Eq. 22-23 has solutions that oscillate in time with frequency ω:

$$\begin{aligned} y(t) &= A\cos\omega t + B\sin\omega t \\ y(t) &= C\sin(\omega t + \phi) \end{aligned} \tag{22-24}$$

where $\omega = \sqrt{K_s/M}$ is the natural frequency of oscillation, A and B are integration constants written as amplitudes; or, C and ϕ are integration constants written as an amplitude and a phase shift.

The simple harmonic oscillator has an *invariant*, for the case of mass-spring system the invariant

3.016 Home

Full Screen

Close

Quit

is the total energy:

$$\begin{aligned}\text{Kinetic Energy} + \text{Potential Energy} &= \\ \frac{M}{2}v^2 + \frac{K_s}{2}y^2 &= \\ \frac{M}{2}\frac{dy}{dt}^2 + \frac{K_s}{2}y^2 &= \\ A^2\omega^2\frac{M}{2}\cos^2(\omega t + \phi) + A^2\frac{K_s}{2}\sin^2(\omega t + \phi) &= \\ A^2(\omega^2\frac{M}{2}\cos^2(\omega t + \phi) + \frac{M\omega^2}{2}\sin^2(\omega t + \phi) &= \\ A^2M\omega^2 &= \text{constant}\end{aligned} \tag{22-25}$$

There are a remarkable number of physical systems that can be reduced to a simple harmonic oscillator (i.e., the model can be reduced to Eq. 22-23). Each such system has an analog to a mass, to a spring constant, and thus to a natural frequency. Furthermore, every such system will have an invariant that is an analog to the total energy—an in many cases the invariant will, in fact, be the total energy.

The advantage of reducing a physical model to a harmonic oscillator is that *all of the physics follows from the simple harmonic oscillator.*

Here are a few examples of systems that can be reduced to simple harmonic oscillators:

Pendulum By equating the rate of change of angular momentum equal to the torque, the equation for pendulum motion can be derived:

$$MR^2\frac{d^2\theta}{dt^2} + MgR\sin\theta = 0 \tag{22-26}$$

for small-amplitude pendulum oscillations, $\sin(\theta) \approx \theta$, the equation is the same as a simple harmonic oscillator.

It is instructive to consider the invariant for the non-linear equation. Because

$$\frac{d^2\theta}{dt^2} = \frac{d\theta}{dt}\left(\frac{d\frac{d\theta}{dt}}{d\theta}\right) \tag{22-27}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Eq. 22-26 can be written as:

$$MR^2\frac{d\theta}{dt}\left(\frac{d\frac{d\theta}{dt}}{d\theta}\right) + MgR\sin(\theta) = 0 \tag{22-28}$$

$$\frac{d}{d\theta}\left[\frac{MR^2}{2}\left(\frac{d\theta}{dt}\right)^2 - MgR\cos(\theta)\right] = 0 \tag{22-29}$$

which can be integrated with respect to θ:

$$\frac{MR^2}{2}\left(\frac{d\theta}{dt}\right)^2 - MgR\cos(\theta) = \text{constant} \tag{22-30}$$

This equation will be used as a level-set equation to visualize pendulum motion.

Buoyant Object Consider a buoyant object that is slightly displaced from its equilibrium floating position. The force (downwards) due to gravity of the buoy is $\rho_{bouy} g V_{bouy}$ The force (upwards) according to Archimedes is $\rho_{water} g V_{sub}$ where V_{sub} is the volume of the buoy that is submerged. The equilibrium position must satisfy $V_{sub-eq}/V_{bouy} = \rho_{bouy}/\rho_{water}$.

If the buoy is slightly perturbed at equilibrium by an amount δx the force is:

$$\begin{aligned} F =& \rho_{water} g (V_{sub-eq} + \delta x A_o) - \rho_{buoy} g V_{buoy} \\ F =& \rho_{water} g \delta x A_o \end{aligned} \tag{22-31}$$

where A_o is the cross-sectional area at the equilibrium position. Newton's equation of motion for the buoy is:

$$M_{buoy}\frac{d^2 y}{dt^2} - \rho_{water} g A_o y = 0 \tag{22-32}$$

so the characteristic frequency of the buoy is $\omega = \sqrt{\rho_{water} g A_o / M_{bouy}}$.

Single Electron Wave-function The one-dimensional Schrödinger equation is:

$$\frac{d^2\psi}{dx^2} + \frac{2m}{\hbar^2}\left(E - U(x)\right)\psi = 0 \tag{22-33}$$

where $U(x)$ is the potential energy at a position x. If $U(x)$ is constant as in a free electron in a box, then the one-dimensional wave equation reduces to a simple harmonic oscillator.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

In summation, just about any system that oscillates about an equilibrium state can be reduced to a harmonic oscillator.

Lecture 23: Resonance Phenomena, Beam Theory

Reading:
Kreyszig Sections: 2.8, 2.9, 3.1, 3.2, 3.3 (pages84–90, 91–96, 105–111, 111–115, 116–121)

Resonance Phenomena

The physics of an isolated damped linear harmonic oscillator follows from the behavior of the homogeneous equation:[15]

$$M\frac{d^2y(t)}{dt^2} + \eta l_o \frac{dy(t)}{dt} + K_s y(t) = 0 \tag{23-1}$$

This equation is the sum of three forces:

inertial force depending on the acceleration of the object.

drag force depending on the velocity of the object.

spring force depends on the displacement of the object.

The system is *autonomous* in the sense that everything depends on the system itself; there are no outside agents changing the system.

The zero on the right-hand-side of Eq. 23-1 implies that there are no external forces applied to the system. The system oscillates with a characteristic frequency $\omega = \sqrt{K_s/M}$ with amplitude that are damped by a characteristic time $\tau = (2M)/(\eta l_o)$ (i.e., the amplitude is damped $\propto \exp(-t/\tau)$.)

[15] A concise and descriptive description of fairly general harmonic oscillator behavior appears at http://hypertextbook.com/chaos/41.shtml

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 23 MATHEMATICA® Example 1

Simulating Harmonic Oscillation with Biased and Unbiased Noise

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The second-order differencing simulation of a harmonic oscillator is modified to include white and biased stochastic nudging.

1: *GrowListGeneralNoise* is extended from a previous example for simulating $\ddot{y} + \beta\dot{y} + \alpha y = 0$ (*GrowList* in example 21-1) and adds a random uniform displacement $y+\delta$, $\delta \in (-\text{randomamp}, \text{randomamp})$ at each iteration. The *ValuesList_List* argument should be a list containing two lists: the first list is comprised of the sequence of displacements y; the second list records the corresponding stochastic displacement δ. The function uses a list's two previous values and `Append` and to grow the list iteratively.

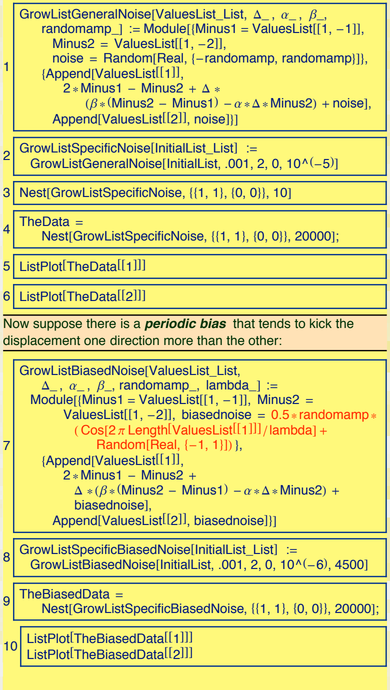
4: Exemplary data from 2×10^5 iterations (using `Nest`) is produced for the specific case of $\Delta = 0.001$, $\alpha = 2$, $\beta = 0$.

5: The displacements (i.e., first list) are plotted with `ListPlot`.

6: The random 'nudges' (i.e., second list) are also plotted.

7: Biased nudges are simulated with *GrowListBiasedNoise* . This extends the unbiased example above, by including a wavelength for a cosine-biased random amplitude. A sample, δ, from the uniform random distribution as above is selected and then multiplied by $\cos 2\pi t/\lambda$. The time-like variable is simulated with `Length` and the current data.

10: The biased data for approximately the resonance condition for the same model parameters above is plotted with the biased noise.



```
1  GrowListGeneralNoise[ValuesList_List, Δ_ , α_ , β_,
       randomamp_] := Module[{Minus1 = ValuesList[[1, −1]],
         Minus2 = ValuesList[[1, −2]],
         noise = Random[Real, {−randomamp, randomamp}]},
       {Append[ValuesList[[1]],
           2∗Minus1 − Minus2 + Δ ∗
               (β∗(Minus2 − Minus1) − α∗Δ∗Minus2) + noise],
         Append[ValuesList[[2]], noise]}]

2  GrowListSpecificNoise[InitialList_List] :=
     GrowListGeneralNoise[InitialList, .001, 2, 0, 10^(−5)]

3  Nest[GrowListSpecificNoise, {{1, 1}, {0, 0}}, 10]

4  TheData =
       Nest[GrowListSpecificNoise, {{1, 1}, {0, 0}}, 20000];

5  ListPlot[TheData[[1]]]

6  ListPlot[TheData[[2]]]
```

Now suppose there is a ***periodic bias*** that tends to kick the displacement one direction more than the other:

```
7  GrowListBiasedNoise[ValuesList_List,
       Δ_ , α_ , β_, randomamp_, lambda_] :=
     Module[{Minus1 = ValuesList[[1, −1]], Minus2 =
           ValuesList[[1, −2]], biasednoise = 0.5∗randomamp∗
             ( Cos[2π Length[ValuesList[[1]]]/lambda] +
                 Random[Real, {−1, 1}]) },
       {Append[ValuesList[[1]],
           2∗Minus1 − Minus2 +
             Δ ∗(β∗(Minus2 − Minus1) − α∗Δ∗Minus2) +
             biasednoise],
         Append[ValuesList[[2]], biasednoise]}]

8  GrowListSpecificBiasedNoise[InitialList_List] :=
     GrowListBiasedNoise[InitialList, .001, 2, 0, 10^(−6), 4500]

9  TheBiasedData =
       Nest[GrowListSpecificBiasedNoise, {{1, 1}, {0, 0}}, 20000];

10 ListPlot[TheBiasedData[[1]]]
   ListPlot[TheBiasedData[[2]]]
```

3.016 Home

Full Screen

Close

Quit

A general model for a damped and forced harmonic oscillator is

$$M\frac{d^2y(t)}{dt^2} + \eta l_o \frac{dy(t)}{dt} + K_s y(t) = F_{app}(t) \tag{23-2}$$

where F_{app} represents a time-dependent applied force to the mass M.

General Solutions to Non-homogeneous ODEs

Equation 23-2 is a non-homogeneous ODE—the functions and its derivatives appear on one side and an arbitrary function appears on the other. The general solution to Eq. 23-2 will be the sum of two parts:

$$\begin{aligned} y_{gen}(t) &= y_{part}(t) + y_{homog}(t) \\ y_{gen}(t) &= y_{F_{app}}(t) + y_{homog}(t) \end{aligned} \tag{23-3}$$

$$y_{homg}(t) = \begin{cases} C_+e^{-|\lambda_+|t} + C_-e^{-|\lambda_-|t} & (\eta l_o)^2 > 4MK_s \quad \text{Over-damped} \\ C_1e^{-|\lambda|t} + C_2te^{-|\lambda|t} & (\eta l_o)^2 = 4MK_s \quad \text{Critical Damping} \\ C_+e^{-|\mathrm{Re}\lambda|t}e^{\imath|\mathrm{Im}\lambda|t} + C_-e^{-|\mathrm{Re}\lambda|t}e^{-\imath|\mathrm{Im}\lambda|t} & (\eta l_o)^2 < 4MK_s \quad \text{Under-damped} \end{cases} \tag{23-4}$$

where $y_{part} \equiv y_{F_{app}}$ is the solution for the particular F_{app} on the right-hand-side and y_{homog} is the solution for the right-hand-side being zero. *Adding the homogeneous solution y_{homog} to the particular solution y_{part} is equivalent to adding a "zero" to the applied force F_{app}*

Interesting cases arise when the applied force is periodic $F_{app}(t) = F_{app}(t+T) = F_{app}(t+2\pi/\omega_{app})$, especially when the applied frequency, ω_{app} is close to the the characteristic frequency of the oscillator $\omega_{char} = \sqrt{K_s/M}$.

Modal Analysis

For the case of a periodic forcing function, the time-dependent force can be represented by a Fourier Series. Because the second-order ODE (Eq. 23-2) is linear, the particular solutions for each term in a Fourier series can be summed. Therefore, particular solutions can be analyzed for one trigonometric term at a time:

$$M\frac{d^2y(t)}{dt^2} + \eta l_o \frac{dy(t)}{dt} + K_s y(t) = F_{app}\cos(\omega_{app}t) \tag{23-5}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

There are three general cases for the particular solution:

	Condition	Solution for $F(t) = F_{app}\cos(\omega_{app}t)$
Undamped, Frequency-Mismatch	$\eta = 0$ $\omega_{char}^2 = \frac{K_s}{M} \neq \omega_{app}^2$	$y_{part}(t) = \frac{F_{app}\cos(\omega_{app}t)}{M(\omega_{char}+\omega_{app})(\omega_{char}-\omega_{app})}$
Undamped, Frequency-Matched	$\eta = 0$ $\omega_{char}^2 = \frac{K_s}{M} = \omega_{app}^2$	$y_{part}(t) = \frac{F_{app}t\sin(\omega_{app}t)}{2M\omega_{app}}$
Damped	$\eta > 0$	$y_{part}(t) = \frac{F_{app}\cos(\omega_{app}t+\phi_{lag})}{\sqrt{M^2(\omega_{char}^2-\omega_{app}^2)^2+\omega_{app}^2\eta^2 l_o^2}}$ $\phi_{lag} = \tan^{-1}\left(\frac{\omega_{app}\eta l_o}{M(\omega_{char}^2-\omega_{app}^2)}\right)$

The phenomenon of resonance can be observed as the driving frequency approaches the characteristic frequency.

Lecture 23 MATHEMATICA® Example 2

Resonance and Near-Resonance Behavior

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Solutions to $m\ddot{y} + \eta\dot{y} + ky = F_{app}\cos(\omega_{app}t)$ analyzed near the resonance condition $\omega_{app} \approx \omega_{char} \equiv \sqrt{k/m}$.

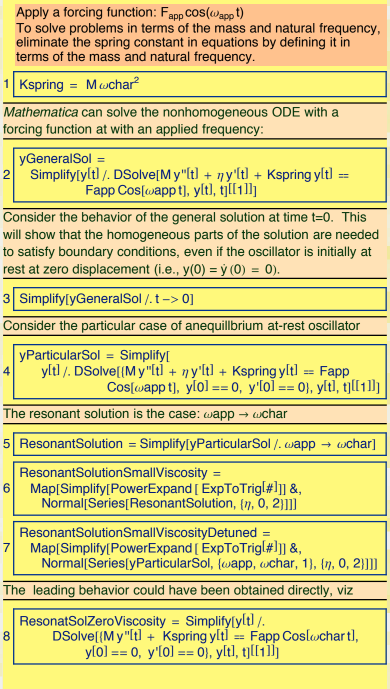
2: The general solution will include two arbitrary constants `C[1]` and `C[2]` in terms that derive from the homogeneous solution plus a part that derives from the heterogeneous (i.e., forced) part.

3: Examining the form of the general solution at $t = 0$, it will be clear that the constants from the homogeneous part will be needed to satisfy arbitrary boundary conditions—most importantly, the constants will include terms that depend on the characteristic and applied frequencies.

4: Here `DSolve` will be used *yParticularSolution* to analyze the particular case of a forced $(F(t) = F_{app}\cos(\omega_{app}t))$ and damped harmonic oscillator initially at resting equilibrium $(y(t = 0) = 1$ and $y'(t = 0) = 0)$.

5: The most interesting cases are the resonance and near resonance cases: *ResonantSolution* is obtained by setting the forcing frequency equal to the characteristic frequency.

6: To analyze the at-resonance case, the solution will be expanded to second order for small viscosity with `Series`. Some extra manipulation is required to display the results in a form that is straightforward to interpret. Here, `Map` will be used with a *pure function* to simplify each term produced by `Series`. First, the `SeriesData` object created by `Series` is transformed into a regular expression with `Normal`. The pure function will first transform any $\exp(x)$ into $\cosh(x) + \sinh(x)$, then any fractional powers will be cleaned up (e.g., $\sqrt{x^2} \rightarrow x$) assuming real parameters; finally the individual terms will be simplified.



Apply a forcing function: $F_{app}\cos(\omega_{app}t)$
To solve problems in terms of the mass and natural frequency, eliminate the spring constant in equations by defining it in terms of the mass and natural frequency.

```
1  Kspring = M ωchar^2
```

Mathematica can solve the nonhomogeneous ODE with a forcing function at with an applied frequency:

```
2  yGeneralSol =
     Simplify[y[t] /. DSolve[M y''[t] + η y'[t] + Kspring y[t] ==
          Fapp Cos[ωapp t], y[t], t][[1]]]
```

Consider the behavior of the general solution at time t=0. This will show that the homogeneous parts of the solution are needed to satisfy boundary conditions, even if the oscillator is initially at rest at zero displacement (i.e., $y(0) = \dot{y}(0) = 0$).

```
3  Simplify[yGeneralSol /. t -> 0]
```

Consider the particular case of anequillbrium at-rest oscillator

```
4  yParticularSol = Simplify[
       y[t] /. DSolve[{M y''[t] + η y'[t] + Kspring y[t] == Fapp
              Cos[ωapp t], y[0] == 0, y'[0] == 0}, y[t], t][[1]]]
```

The resonant solution is the case: ωapp → ωchar

```
5  ResonantSolution = Simplify[yParticularSol /. ωapp → ωchar]

6  ResonantSolutionSmallViscosity =
     Map[Simplify[PowerExpand [ ExpToTrig[#]]] &,
       Normal[Series[ResonantSolution, {η, 0, 2}]]]

7  ResonantSolutionSmallViscosityDetuned =
     Map[Simplify[PowerExpand [ ExpToTrig[#]]] &,
       Normal[Series[yParticularSol, {ωapp, ωchar, 1}, {η, 0, 2}]]]
```

The leading behavior could have been obtained directly, viz

```
8  ResonatSolZeroViscosity = Simplify[y[t] /.
        DSolve[{M y''[t] + Kspring y[t] == Fapp Cos[ωchar t],
          y[0] == 0, y'[0] == 0}, y[t], t][[1]]]
```

7: This illustrates how near resonance $\omega_{app} \approx \omega_{char}$ can be analyzed in the small viscosity limit. Here, `Series` first expands around $\eta = 0$ to second order and then around small $\delta\omega = \omega_{app} - \omega_{char}$.

8: Setting the viscosity to zero *a priori* is possible and returns the leading order behavior, but the *asymptotic behavior* for small parameters cannot be ascertained.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 23 Mathematica® Example 3

Visualizing Forced and Damped Harmonic Oscillation

notebook (non-evaluated) pdf (evaluated) html (evaluated)

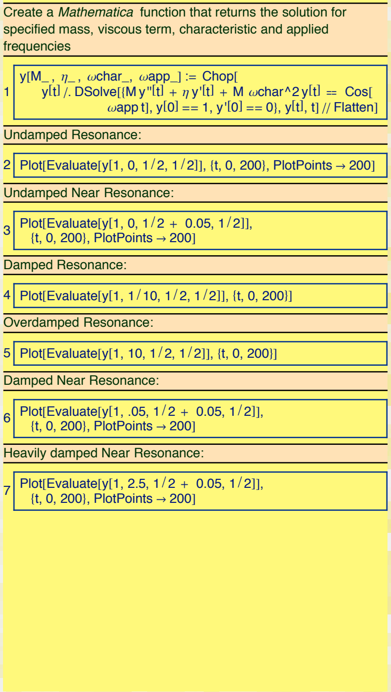
1: This function solves the heterogeneous damped harmonic oscillator ODE (where $F(t) = \cos(\omega_{app} t)$) for any input mass, damping coefficient, and spring constant M, η, $k = M\omega^2_{char}$.

2: Undamped resonance $\omega_{char} = \omega_{app} = 1/2$ should show linearly growing amplitude.

3: Near resonance will show a beat-phenomena because of "de-tuning."

4: Damped resonance will show that the amplitudes approaching to a finite asymptotic limit.

6: The beats will still be apparent for the damped near resonance condition, but the finite damping coefficient will prevent the amplitude from completely disappearing.



Create a *Mathematica* function that returns the solution for specified mass, viscous term, characteristic and applied frequencies

```
1  y[M_, η_, ωchar_, ωapp_] := Chop[
      y[t] /. DSolve[{M y''[t] + η y'[t] + M ωchar^2 y[t] == Cos[
              ωapp t], y[0] == 1, y'[0] == 0}, y[t], t] // Flatten]
```

Undamped Resonance:

```
2  Plot[Evaluate[y[1, 0, 1/2, 1/2]], {t, 0, 200}, PlotPoints → 200]
```

Undamped Near Resonance:

```
3  Plot[Evaluate[y[1, 0, 1/2 + 0.05, 1/2]],
     {t, 0, 200}, PlotPoints → 200]
```

Damped Resonance:

```
4  Plot[Evaluate[y[1, 1/10, 1/2, 1/2]], {t, 0, 200}]
```

Overdamped Resonance:

```
5  Plot[Evaluate[y[1, 10, 1/2, 1/2]], {t, 0, 200}]
```

Damped Near Resonance:

```
6  Plot[Evaluate[y[1, .05, 1/2 + 0.05, 1/2]],
     {t, 0, 200}, PlotPoints → 200]
```

Heavily damped Near Resonance:

```
7  Plot[Evaluate[y[1, 2.5, 1/2 + 0.05, 1/2]],
     {t, 0, 200}, PlotPoints → 200]
```

3.016

3.016 Home

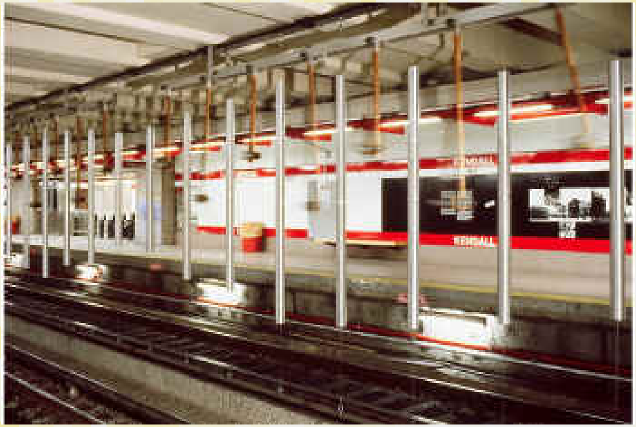
◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Resonance can have catastrophic or amusing (or both) consequences:



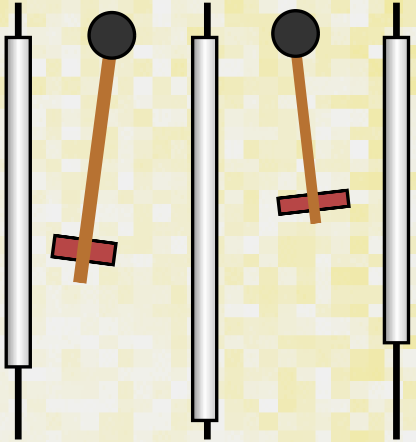


Figure 23-24: Picture and illustration of the bells at Kendall square. Many people shake the handles vigorously but with apparently no pleasant effect. The concept of resonance can be used to to operate the bells efficiently Perturb the handle slightly and observe the frequencies of the the pendulums—select one and wiggle the handle at the pendulum's characteristic frequency. The amplitude of that pendulum will increase and eventually strike the neighboring tubular bells.

From Cambridge Arts Council Website:

http://www.ci.cambridge.ma.us/~CAC/public_art_tour/map_11_kendall.html

Artist: Paul Matisse Title: The Kendall Band - Kepler, Pythagoras, Galileo Date: 1987

Materials: Aluminum, teak, steel

Handles located on the platforms allow passengers to play these mobile-like instruments, which are suspended in arches between the tracks, "Kepler" is an aluminum ring that will hum for five minutes after it is struck by the large teak hammer above it. "Pythagoras" consists of a 48-foot row of chimes made from heavy aluminum tubes interspersed with 14 teak hammers. "Galileo" is a large sheet of metal that rattles thunderously when one shakes the handle.

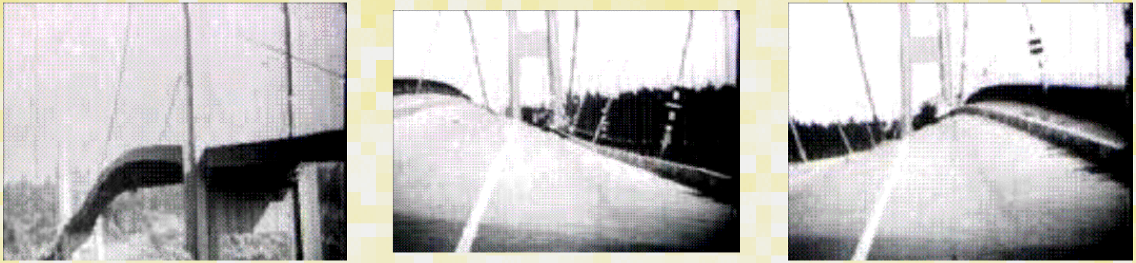


Figure 23-25: Animation Available in individual lecture, deleted here because of filesize constraints The Tacoma bridge disaster is perhaps one of the most well-knownfailures that resulted directly from resonance phenomena. It is believed that the the wind blowing across the bridge caused the bridge to vibrate like a reed in a clarinet.(Images from Promotional Video Clip from *The Camera Shop 1007 Pacific Ave., Tacoma, Washington* Full video Available http://www.camerashoptacoma.com/)

Lecture 24: Systems of Ordinary Differential Equations

Reading:
Kreyszig Sections: 4.1, 4.2 (pages131–135, 136–139)

Systems of Ordinary Differential Equations

The ordinary differential equations that have been treated thus far are relations between a single function and how it changes:

$$F(\frac{d^n y}{dx^n}, \frac{d^{n-1} y}{dx^{n-1}}, \ldots, \frac{dy}{dx}, y, x) = 0 \tag{24-1}$$

Many physical models of systems result in differential relations between *several* functions. For example, a first-order system of ordinary differential equations for the functions $(y_1(x), y_2(x), \ldots, y_n(x))$ is:

$$\begin{aligned}
\frac{dy_1}{dx} &= f_1(y_1(x), y_2(x), \ldots, y_n(x), x)\\
\frac{dy_2}{dx} &= f_2(y_1(x), y_2(x), \ldots, y_n(x), x)\\
\vdots &= \vdots\\
\frac{dy_n}{dx} &= f_n(y_1(x), y_2(x), \ldots, y_n(x), x)
\end{aligned} \tag{24-2}$$

or with a vector notation,

$$\frac{d\vec{y}(x)}{dx} = \vec{f}(\vec{y}, x) \tag{24-3}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Example: The Spread of a MIT Joke

The predator-prey model serves as the classical example of a system of differential equations. This is a (possibly humorous) variant of the predator-prey problem.

Suppose there is a fairly bad joke that circulates around the student population. Students either know the joke or they don't and thus can be divided into two populations:

Jaded, J Knows the joke, and if someone tries to tell it to them, they interrupt with, "Yeah, Yeah. I heard that one. It's pretty, like, stupid."

Naive, N Never heard the joke or has forgotten it.

As the joke spreads, or as students graduate, or students forget the joke, or as new students are admitted to MIT, the populations change.

We will try to construct a model that reflects how the populations change each day.

We will suppose that freshman enter MIT a constant daily rate; in order to keep the population of students regulated, the admissions office accepts freshman at a rate that depends on how many of 4000 slots are open. Therefore, freshman enter MIT, and thus the Naive population at a daily rate of:

$$\frac{dN_{frsh}}{dt} = \frac{4000 - (J + N)}{365} \tag{24-4}$$

Students have a lot of things on their mind (some of which is education) and so they tend to be forgetful. Students who know the joke tend to forget at rate ϕ/year. Suppose that a fraction, ϕ, of the Jaded students forget the joke each year—these students leave the J-group and enter the N-group at a daily rate:

$$\begin{aligned} \frac{dJ_{forg}}{dt} &= \frac{\alpha_A J}{365} = \alpha J \\ \frac{dN_{forg}}{dt} &= -\frac{\alpha_A J}{365} = -\alpha J \end{aligned} \tag{24-5}$$

It is closely held secret that Susan Hockfield, MIT's president, has an odd sense of humor. At each commencement ceremony, as the proud candidates for graduation approach the president to collect their hard-earned diploma, President Hockfield whispers to the student, "Have you the joke about. . . ?" If the student says, "Yes. I have heard that joke. It is *very* funny!!!" then the diploma is awarded. However, if the student says, "No. But, I am dying to hear it!!!", the president's face drops into a sad frown and the student is asked to leave without collecting the diploma.[16]

Therefore, only students in the J-group can graduate. Let's assume that at any one time, 1/3 of the jaded students have satisfied the graduation requirements, and of this group 99% will graduate:

$$\frac{dJ_{grdt}}{dt} = -\frac{0.99J/3}{365} = -\gamma J \tag{24-6}$$

The joke spreads in proportion to its "funniness coefficient" and the probability that a naive student runs into a jaded student:

$$\begin{aligned} \frac{dJ_{sprd}}{dt} &= -\frac{\phi_A JN}{365^2} = -\phi JN \\ \frac{dN_{sprd}}{dt} &= \frac{\phi_A JN}{365^2} = \phi JN \end{aligned} \tag{24-7}$$

Therefore, an iterative model for the student population that knows the joke is:

$$\begin{aligned} \text{Naive Fraction(Tomorrow)} =& \text{Naive Fraction(Today)} + \text{Change in Naive Fraction} \\ \text{Jaded Fraction(Tomorrow)} =& \text{Jaded Fraction(Today)} + \text{Change in Jaded Fraction} \end{aligned} \tag{24-8}$$

or

$$\begin{aligned} N_{i+1} =& N_i + \frac{4000 - (N_i + J_i)}{365} + \alpha J_i - \phi J_i N_i \\ J_{i+1} =& J_i + \phi J_i N_i - \gamma J_i - \alpha J_i \end{aligned} \tag{24-9}$$

[16] This event is an annual source of confusion and embarrassment for the students' proud families—and a source of sadistic amusement to the attending faculty (who have an even stranger sense of humor than Hockfield's).

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 24 MATHEMATICA® Example 1

Iterative Example of Predator-Prey Simulation

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Functions to the simulate system of iterative equations, Eq. 24-8, are developed.

1: *FreshmanEntranceRate* takes the current Jaded and Naive populations and returns how many students are admitted to the Naive popuation each day.

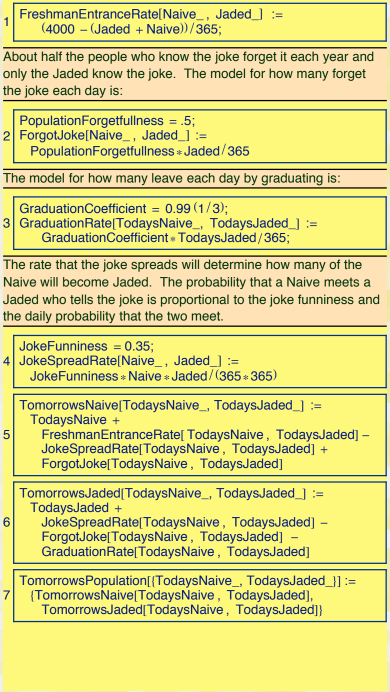
2: The number of students that forget the joke each day is proportional to the current Jaded population.

3: Because only Jaded students can graduate, *GraduationRate* is proportional to the current Jaded population.

4: *JokeSpreadRate* depends linearly on the probability that a Jaded meets a Naive student, and therefore the spread rate depends on the product of the two populations.

6: *TomorrowsNaive* and *TomorrowsJaded* advance the two populations by one day.

7: If the two populations are kept in a two-item list, then *TomorrowsPopulation* will advance the entire population list.



1
```
FreshmanEntranceRate[Naive_ , Jaded_] :=
    (4000 − (Jaded + Naive))/365;
```

About half the people who know the joke forget it each year and only the Jaded know the joke. The model for how many forget the joke each day is:

2
```
PopulationForgetfullness = .5;
ForgotJoke[Naive_ , Jaded_] :=
  PopulationForgetfullness∗Jaded/365
```

The model for how many leave each day by graduating is:

3
```
GraduationCoefficient = 0.99 (1/3);
GraduationRate[TodaysNaive_, TodaysJaded_] :=
    GraduationCoefficient∗TodaysJaded/365;
```

The rate that the joke spreads will determine how many of the Naive will become Jaded. The probability that a Naive meets a Jaded who tells the joke is proportional to the joke funniness and the daily probability that the two meet.

4
```
JokeFunniness = 0.35;
JokeSpreadRate[Naive_ , Jaded_] :=
  JokeFunniness∗Naive∗Jaded/(365∗365)
```

5
```
TomorrowsNaive[TodaysNaive_, TodaysJaded_] :=
  TodaysNaive +
    FreshmanEntranceRate[ TodaysNaive , TodaysJaded] −
    JokeSpreadRate[TodaysNaive , TodaysJaded] +
    ForgotJoke[TodaysNaive , TodaysJaded]
```

6
```
TomorrowsJaded[TodaysNaive_, TodaysJaded_] :=
  TodaysJaded +
    JokeSpreadRate[TodaysNaive , TodaysJaded] −
    ForgotJoke[TodaysNaive , TodaysJaded] −
    GraduationRate[TodaysNaive , TodaysJaded]
```

7
```
TomorrowsPopulation[{TodaysNaive_, TodaysJaded_}] :=
  {TomorrowsNaive[TodaysNaive , TodaysJaded],
    TomorrowsJaded[TodaysNaive , TodaysJaded]}
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 24 Mathematica® Example 2

Visualizing the Spread of Jokes at MIT

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The behavior of the iterative model presented in the above example are visualized.

1: Here, `NestList` is used with the *TomorrowsPopulation* to create a sequence of population-lists, starting at $\{N, J\} = \{400, 600\}$, for five consectutive days.

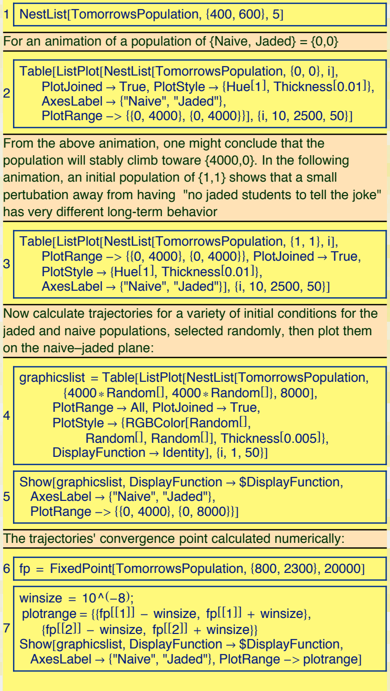
2: The results of `NestList` are repeatly plotted by using `Table` at 50 day intervals. This will create sequence of plots that can be collected into an animation. In this case, an initial population of $\{0, 0\}$ will asymptotically approach $\{4000, 0\}$; at very long times, no one would graduate, one one would enter MIT, and no one would know the joke. This is not only bad policy, but the system is unstable...

3: In a similar animation to the above, any small positive perturbation from the $\{0, 0\}$ population will initially be driven towards the $\{4000, 0\}$ solution, but will 'veer' away and then rush towards another asymptotic solution.

4: To examine the behavior for a variety of initial populations, iteration sequences are generated for 50 different initial random populations, and then represented with an (undisplayed) graphics object. Each trajectory is identified with a random color.

6: The graphics example above indicates that trajectories tend towards a stable fixed point. The numerical value of the fixed point can be determined by iterating the list until the result ceases to change numerically. This method is implemented in the function `FixedPoint` which is insensitive to the initial population.

7: Here, the fixed point is used to create a small 'window' in `PlotRange` so that the local behavior can be observed.



1
```
NestList[TomorrowsPopulation, {400, 600}, 5]
```

For an animation of a population of {Naive, Jaded} = {0,0}

2
```
Table[ListPlot[NestList[TomorrowsPopulation, {0, 0}, i],
     PlotJoined → True, PlotStyle → {Hue[1], Thickness[0.01]},
     AxesLabel → {"Naive", "Jaded"},
     PlotRange –> {{0, 4000}, {0, 4000}}], {i, 10, 2500, 50}]
```

From the above animation, one might conclude that the population will stably climb toware {4000,0}. In the following animation, an initial population of {1,1} shows that a small pertubation away from having "no jaded students to tell the joke" has very different long-term behavior

3
```
Table[ListPlot[NestList[TomorrowsPopulation, {1, 1}, i],
     PlotRange –> {{0, 4000}, {0, 4000}}, PlotJoined → True,
     PlotStyle → {Hue[1], Thickness[0.01]},
     AxesLabel → {"Naive", "Jaded"}], {i, 10, 2500, 50}]
```

Now calculate trajectories for a variety of initial conditions for the jaded and naive populations, selected randomly, then plot them on the naive–jaded plane:

4
```
graphicslist = Table[ListPlot[NestList[TomorrowsPopulation,
          {4000 * Random[], 4000 * Random[]}, 8000],
       PlotRange → All, PlotJoined → True,
       PlotStyle → {RGBColor[Random[],
             Random[], Random[]], Thickness[0.005]},
       DisplayFunction → Identity], {i, 1, 50}]
```

5
```
Show[graphicslist, DisplayFunction → $DisplayFunction,
  AxesLabel → {"Naive", "Jaded"},
  PlotRange –> {{0, 4000}, {0, 8000}}]
```

The trajectories' convergence point calculated numerically:

6
```
fp = FixedPoint[TomorrowsPopulation, {800, 2300}, 20000]
```

7
```
winsize = 10^(–8);
 plotrange = {{fp[[1]] – winsize, fp[[1]] + winsize},
     {fp[[2]] – winsize, fp[[2]] + winsize}}
Show[graphicslist, DisplayFunction → $DisplayFunction,
  AxesLabel → {"Naive", "Jaded"}, PlotRange –> plotrange]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The stability and behavior of the iterative model 24-8 can be analyzed by replacing the coupled iterative equations with coupled ODEs:

$$\begin{aligned}\frac{dN}{dt} =& \frac{4000 - (N + J)}{365} + \alpha J - \phi J N \\ \frac{dJ}{dt} =& \phi J N - \gamma J - \alpha J\end{aligned} \qquad (24\text{-}10)$$

A *critical point* is one at which the left-hand side of Equations 24-2, 24-3, or 24-10 vanish—in other words, a critical point is a special value of the vector $\vec{y}(x)$ where the system of equations does not evolve. For the system defined by Eq. 24-10, there are two critical points

$$(N_{\text{unst}} = 4000,\ J_{\text{unst}} = 0) \qquad \text{and} \qquad \left(N_{\text{stab}} = \frac{\alpha + \gamma}{\phi},\ J_{\text{stab}} = \frac{4000\phi - \gamma - \alpha}{\phi + 365\gamma\phi}\right) \qquad (24\text{-}11)$$

However, while a system that is sitting *exactly* at a critical point will not evolve, some of the critical points are not *stable*. There are three broad categories of critical points:[17]

Stable Any slight perturbation of the system away from the critical point results in an evolution back to that critical point. In other words, all points in the neighborhood of a stable critical point have a trajectory that is attracted back to that point.

Unstable Some slight perturbation of the system away from the critical point results in an evolution away from that critical point. In other words, some points in the neighborhood of an unstable critical point have trajectories that are repelled by the point.

Circles Any slight perturbation away from a critical point results in an evolution that always remains near the critical point. In other words, all points in the neighborhood of a circle critical point have trajectories that remain in the neighborhood of the point.

Reduction of Higher Order ODEs to a System of First Order ODEs

Higher-order ordinary differential equations can usually be re-written as a *system* of first-order differential equations. If the higher-order ODE can be solved for its largest derivative:

$$\frac{d^n y}{dt^n} = F\left(\frac{d^{n-1}y}{dt^{n-1}}, \frac{d^{n-2}y}{dt^{n-2}}, \ldots, \frac{dy}{dt}, t\right) \qquad (24\text{-}12)$$

[17] There are others that will be discussed later.

then $n-1$ "new" functions can be introduced via

$$\begin{aligned}
y_0(t) &\equiv y(t) \\
y_1(t) &\equiv \frac{dy}{dt} = \frac{dy_0}{dt} \\
y_2(t) &\equiv \frac{d^2y}{dt^2} = \frac{dy_1}{dt} \\
\vdots &\equiv \vdots \\
y_{n-1}(t) &\equiv \frac{d^{n-1}y}{dt^{n-1}} = \frac{dy_{n-2}}{dt} \\
y_n(t) &\equiv \frac{d^ny}{dt^n} = F(\frac{d^{n-1}y}{dt^{n-1}}, \frac{d^{n-2}y}{dt^{n-2}}, \ldots, \frac{dy}{dt}, t) = \frac{dy_{n-1}}{dt}
\end{aligned} \tag{24-13}$$

or

$$\frac{d}{dt}\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ F(y_{n-1}, y_{n-2}, \ldots, y_1, y_0, t) \end{pmatrix} \tag{24-14}$$

For example, the damped harmonic oscillator, $M\ddot{y}+\eta l_o\dot{y}+K_sy = 0$, can be re-written by introducing the momentum variable, $p = Mv = M\dot{y}$, as the system:

$$\begin{aligned}
\frac{dy}{dt} &= \frac{p}{M} \\
\frac{dp}{dt} &= -K_sy - \eta l_o p
\end{aligned} \tag{24-15}$$

which has only one critical point $y = p = 0$.

The equation for a free pendulum, $MR^2\ddot{\theta} + MgR\sin(\theta) = 0$, can be re-written by introducing the angular momentum variable, $\omega = MR\dot{\theta}$ as the system,

$$\begin{aligned}
\frac{d\theta}{dt} &= \frac{\omega}{MR} \\
\frac{d\omega}{dt} &= -Mg\sin(\theta)
\end{aligned} \tag{24-16}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

which has two *different* kinds of critical points: $(\omega = 0, \theta = n_{even}\pi)$ and $(\omega = 0, \theta = n_{odd}\pi)$.

Finally, the beam equation $EI\frac{d^4y}{dx^4} = w(x)$ can be rewritten as the system:

$$\frac{d}{dx}\begin{pmatrix} y \\ m_{slope} \\ M \\ S \end{pmatrix} = \begin{pmatrix} m_{slope} \\ \frac{M}{EI} \\ S \\ w(x) \end{pmatrix} \tag{24-17}$$

where m_{slope} is the slope of the beam, M is the local bending moment in the beam, S is the local shearing force in the beam, and $w(x)$ is the load density.

This beam equation does not have any interesting critical points.

Linearization of Systems of ODEs

The fixed point plays a very important role in understanding the behavior of non-linear ODEs.

The general autonomous non-linear ODE can be written as:

$$\frac{d\vec{y}}{dt} \equiv \frac{d}{dt}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} F_1(y_1, y_2, \ldots, y_n) \\ F_2(y_1, y_2, \ldots, y_n) \\ \vdots \\ F_n(y_1, y_2, \ldots, y_n) \end{pmatrix} \equiv \vec{F}(y_1, y_2, \ldots, y_n) \tag{24-18}$$

The fixed points are the solutions to:

$$\vec{F}(y_1^f, y_2^f, \ldots, y_n^f) = \begin{pmatrix} F_1(y_1^f, y_2^f, \ldots, y_n^f) \\ F_2(y_1^f, y_2^f, \ldots, y_n^f) \\ \vdots \\ F_n(y_1^f, y_2^f, \ldots, y_n^f) \end{pmatrix} = \vec{0} \tag{24-19}$$

If the fixed points can be found, then the behavior *near* the fixed points can be analyzed by linearization. Letting $\vec{\delta} = \vec{y} - \vec{y}^f$ be a point near a fixed point, then a linear approximation is:

$$\frac{d}{dt}\vec{\delta} = \underline{J}\delta \tag{24-20}$$

3.016

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

where

$$\underline{J} = \begin{pmatrix} \frac{\partial F_1}{\partial y_1}\big|_{\vec{y}^f} & \frac{\partial F_2}{\partial y_1}\big|_{\vec{y}^f} & \cdots & \frac{\partial F_n}{\partial y_1}\big|_{\vec{y}^f} \\ \frac{\partial F_1}{\partial y_2}\big|_{\vec{y}^f} & \frac{\partial F_2}{\partial y_2}\big|_{\vec{y}^f} & \cdots & \frac{\partial F_n}{\partial y_2}\big|_{\vec{y}^f} \\ \frac{\partial F_1}{\partial y_3}\big|_{\vec{y}^f} & \frac{\partial F_2}{\partial y_3}\big|_{\vec{y}^f} & \ddots & \vdots \\ \frac{\partial F_1}{\partial y_n}\big|_{\vec{y}^f} & \cdots & \cdots & \frac{\partial F_n}{\partial y_n}\big|_{\vec{y}^f} \end{pmatrix} \qquad (24\text{-}21)$$

Equation 24-20 looks very much like a simple linear first-order ODE. The expression

$$\vec{y}(t) = e^{\underline{J}t}\vec{y}(t=0) \qquad (24\text{-}22)$$

might solve it if the proper analog to the exponential of a matrix were known.

Rather than solve the matrix equation directly, it makes more sense to transform the system into one that is diagonalized. In other words, instead of solving Eq. 24-20 with Eq. 24-21 near the fixed point, find the eigenvalues, λ_i, of Eq. 24-21 and solve the simpler system by transforming the $\vec{\delta}$ into the eigenframe $\vec{\eta}$:

$$\begin{aligned} \frac{d\eta_1}{dt} &= \lambda_1\eta_1 \\ \frac{d\eta_2}{dt} &= \lambda_2\eta_2 \\ \vdots &= \vdots \\ \frac{d\eta_n}{dt} &= \lambda_n\eta_n \end{aligned} \qquad (24\text{-}23)$$

for which solutions can be written down immediately:

$$\begin{aligned} \eta_1(t) &= \eta_1(t=0)e^{\lambda_1 t} \\ \eta_2(t) &= \eta_2(t=0)e^{\lambda_2 t} \\ \vdots &= \vdots \\ \eta_n(t) &= \eta_n(t=0)e^{\lambda_n t} \end{aligned} \qquad (24\text{-}24)$$

If any of the eigenvalues of $\underline{J}$ have a positive real part positive, then an initial condition near that fixed point will diverge from that point—stability occurs only if all the eigenvalues are negative.

Lecture 24 Mathematica® Example 3

Analyzing the Stability for the MIT Joke

notebook (non-evaluated) pdf (evaluated) html (evaluated)

It is shown that the MIT has an unstable and a stable fixed point for the parameters utilized in Example 24-1.

1: These expressions are the right-hand-sides of Eqs. 24-10.

2: A fixed point appears whenever the right-hand-sides of a system of coupled ODEs vanish. In this case, there will be two solutions defined in the list associated with *fixedpoint* .

3: *MITModel* will be defined as a short-hand for the replacement rule associated with the parameters used in Example 24-1.

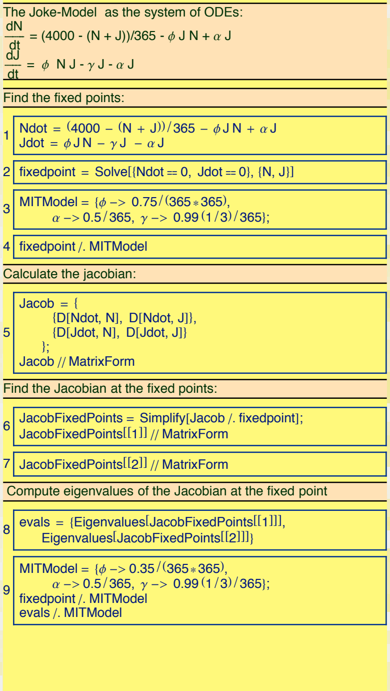
4: Therefore, for the ODE model, this will calculate fixed points.

5: This is the Jacobian (i.e., Eq. 24-21) for Eq. 24-10.

6: The Jacobian analyzed at the fixed points, *fixedpoint* , will produce the characteristic matrix for the stability of each point.

8: A list is created with the `Eigenvalues` associated with the Jacobian at each fixed point.

9: The MIT model shows that there are two fixed points. The one with negative real parts of its eigenvalues is the stable solution and it has a slow oscillatory part.



The Joke-Model as the system of ODEs:

$$\frac{dN}{dt} = (4000 - (N + J))/365 - \phi\, J\, N + \alpha\, J$$

$$\frac{dJ}{dt} = \phi\, N\, J - \gamma\, J - \alpha\, J$$

Find the fixed points:

```
1  Ndot = (4000 − (N + J))/365 − ϕ J N + α J
   Jdot = ϕ J N − γ J − α J

2  fixedpoint = Solve[{Ndot == 0, Jdot == 0}, {N, J}]

3  MITModel = {ϕ −> 0.75/(365∗365),
          α −> 0.5/365, γ −> 0.99 (1/3)/365};

4  fixedpoint /. MITModel
```

Calculate the jacobian:

```
   Jacob = {
          {D[Ndot, N], D[Ndot, J]},
5         {D[Jdot, N], D[Jdot, J]}
        };
   Jacob // MatrixForm
```

Find the Jacobian at the fixed points:

```
6  JacobFixedPoints = Simplify[Jacob /. fixedpoint];
   JacobFixedPoints[[1]] // MatrixForm

7  JacobFixedPoints[[2]] // MatrixForm
```

Compute eigenvalues of the Jacobian at the fixed point

```
8  evals = {Eigenvalues[JacobFixedPoints[[1]]],
       Eigenvalues[JacobFixedPoints[[2]]]}

9  MITModel = {ϕ −> 0.35/(365∗365),
          α −> 0.5/365, γ −> 0.99 (1/3)/365};
   fixedpoint /. MITModel
   evals /. MITModel
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 25: Phase Plane Analysis and Critical Points

Reading:
Kreyszig Sections: 4.1, 4.2 (pages131–135, 136–139)

Phase Plane and Critical Points

A few examples of physical models that can be represented by systems of first-order differential equations:

$$\frac{d\vec{y}}{dt} \equiv \frac{d}{dt}\begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_N(t) \end{pmatrix} = \begin{pmatrix} F_1(y_1, y_2, \ldots, t) \\ F_2(y_1, y_2, \ldots, t) \\ \vdots \\ F_N(y_1, y_2, \ldots, t) \end{pmatrix} = \begin{pmatrix} F_1(\vec{y}, t) \\ F_2(\vec{y}, t) \\ \vdots \\ F_N(\vec{y}, t) \end{pmatrix} \equiv \vec{F}(\vec{y}, t) \tag{25-1}$$

and, furthermore, it has been shown that many higher-order systems of ODEs can be reduced to larger systems of first-order ODEs.

The behavior of systems of first-order equations can be visually interpreted by plotting the trajectories $\vec{y}(t)$ for a variety of initial conditions $\vec{y}(t = 0)$. An illustrative example is provided by the equation for the pendulum, $MR^2\ddot{\theta} + MgR\sin\theta = 0$. can be re-written with the angular momentum ω as the system of first-order ODEs

$$\begin{aligned} \frac{d\theta}{dt} &= \frac{\omega}{MR} \\ \frac{d\omega}{dt} &= -Mg\sin\theta \end{aligned} \tag{25-2}$$

which was shown in Lecture 22 to have solutions:

$$\frac{\omega^2}{2M} - MgR\cos\theta = E_o \tag{25-3}$$

Eq. 25-3 can be used to plot the the trajectories in the phase plane.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

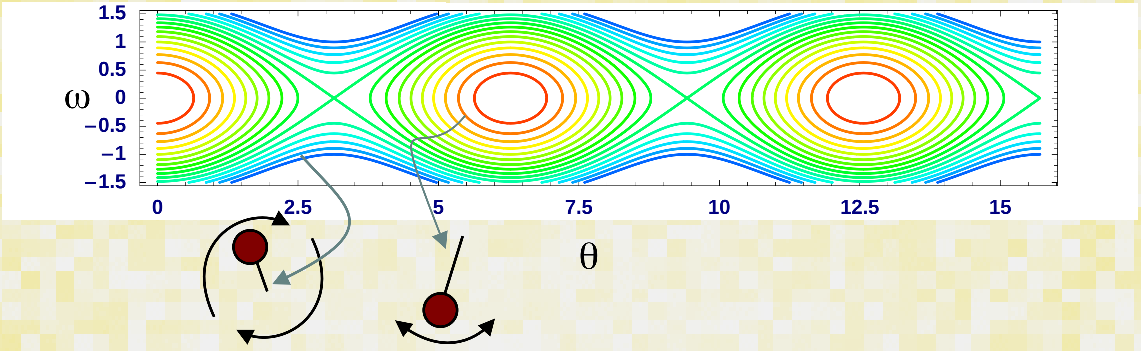


Figure 25-26: Example of the phase plane for the pendulum equation. The small closed orbits are the stable harmonic oscillations about the stable position $\overset{|}{\bullet}$. The larger orbits are those with increasing energy until the energy is just large enough that the pendulum rises to its unstable equilibrium position $\underset{|}{\bullet}$. The two kinds of fixed points (i.e., the stable and unstable points where $\dot{\omega} = \dot{\theta} = 0$) regulate the portrait of the phase plane. (Note: The word “phase” here should not be confused with the common usage of phase in materials science. In the current context for example, the phase represents the positions and momenta of all the particles in a system—this usage is important in statistical mechanics. However, the word “phase” in materials science and engineering is usually interpreted as a portion of material that lies within an identifiable interface—this usage is implied in “equilibrium phase diagrams.”)

Behavior for a wide variety of initial conditions can be comprehended by the following approach:

Identify Fixed Points If all the points in the phase plane where $d\vec{y}/dt = 0$ can be established, then these fixed points can be used as reference points around which the phase-behavior will be determined.

Linearization At each fixed point, Linearization is obtained by expanding Eq. 25-1 to first order in

$\vec{\eta} = \vec{y} - \vec{y}_{\text{fixed}}$, the zeroth-order term vanishes by construction:

$$\frac{d}{dt}\begin{pmatrix}\eta_1(t)\\ \eta_2(t)\\ \vdots\\ \eta_N(t)\end{pmatrix} = \begin{pmatrix} \left.\frac{\partial F_1}{\partial y_1}\right|_{\vec{y}_{\text{fixed}}} & \left.\frac{\partial F_1}{\partial y_2}\right|_{\vec{y}_{\text{fixed}}} & \cdots & \left.\frac{\partial F_1}{\partial y_N}\right|_{\vec{y}_{\text{fixed}}} \\ \left.\frac{\partial F_2}{\partial y_1}\right|_{\vec{y}_{\text{fixed}}} & \ddots & & \\ \vdots & & & \vdots \\ \left.\frac{\partial F_N}{\partial y_1}\right|_{\vec{y}_{\text{fixed}}} & \cdots & \cdots & \left.\frac{\partial F_N}{\partial y_N}\right|_{\vec{y}_{\text{fixed}}}\end{pmatrix}\begin{pmatrix}\eta_1\\ \eta_2\\ \vdots\\ \eta_N\end{pmatrix} \tag{25-4}$$

Eigenvalues/Eigenvectors When the system Eq. 25-4 is transformed into a coordinate frame in which the matrix is diagonal, then each component of $\vec{\eta}_{\text{eigen-frame}}$ has a trajectory that is unaffected by the others and determined by only the diagonal entry associated with that component.

The $\vec{\eta}_{\text{eigen-frame}}$ are the eigenvectors of Eq. 25-4 and the diagonal component is its associated eigenvalue.

Fixed Point Characterization If the eigenvalue is real, then any point that lies in the direction of its eigenvector will evolve along a straight path parallel to the eigenvector. If the real eigenvalue is negative, that straight path will asymptotically approach the origin; if the eigenvalue is positive the trajectory will diverge along the straight-path towards infinity.

If the eigenvalue is imaginary, then the trajectory will circulate about the fixed point with a frequency proportional the eigenvalue's magnitude.

If the eigenvalue, λ is complex, its trajectory will both circulate with a frequency proportional to its imaginary part and diverge from or converge to the fixed point according to $\eta_o \exp(\text{Re}\lambda)$.

If any one of the fixed points has an eigenvalue with a positive real part, the fixed point cannot be stable—this is because "typical" points in the neighborhood of the fixed points will possess some component of the unstable eigenvector.

Stability of Critical Points

For the two-dimensional linear system

$$\frac{d}{dt}\begin{pmatrix}\eta_1(t)\\ \eta_2(t)\end{pmatrix} = \begin{pmatrix} a & b\\ c & d\end{pmatrix}\begin{pmatrix}\eta_1\\ \eta_2\end{pmatrix} \tag{25-5}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

can be analyzed because the eigenvalues can be calculated directly from the quadratic equation.

Every two-by-two matrix has two invariants (i.e., values that do not depend on a unitary transformation of coordinates). These invariants are the *trace*, T of the matrix (the sum of all the diagonals) and the determinant D. The eigenvalue equation can be written in terms of these two invariants:

$$\lambda^2 - T\lambda + D = 0 \tag{25-6}$$

The *discriminant* $\Delta \equiv T^2 - 4D$ appears in the solutions to the eigenvalues:

$$\lambda_\pm = \frac{T \pm \sqrt{\Delta}}{2} \tag{25-7}$$

There are five regions of behavior:

$\Delta \geq 0$ The eigenvalues are real.

Eigenvalues both positive ***An Unstable Node***: All trajectories in the neighborhood of the fixed point will be directed outwards and away from the fixed point.

Eigenvalues both negative ***A Stable Node***: All trajectories in the neighborhood of the fixed point will be directed towards the fixed point.

Eigenvalues opposite sign ***An Unstable Saddle Node***: Trajectories in the general direction of the negative eigenvalue's eigenvector will initially approach the fixed point but will diverge as they approach a region dominated by the positive (unstable) eigenvalue.

$\Delta < 0$ Eigenvalues are complex conjugates—their real parts are equal and their imaginary parts have equal magnitudes but opposite sign.

Real parts positive ***An Unstable Spiral***: All trajectories in the neighborhood of the fixed point spiral away from the fixed point with ever increasing radius.

Real parts negative ***An Stable Spiral***: All trajectories in the neighborhood of the fixed point spiral into the fixed point with ever decreasing radius.

The curves separating these regions have singular behavior. For example, where $T = 0$ for positive D, the eigenvalues are purely imaginary and trajectories circulate about the fixed point in a stable orbit. This is called a ***center*** and is the case for an undamped harmonic oscillator.

The regions can be mapped with the invariants and the following diagram illustrates the behavior.

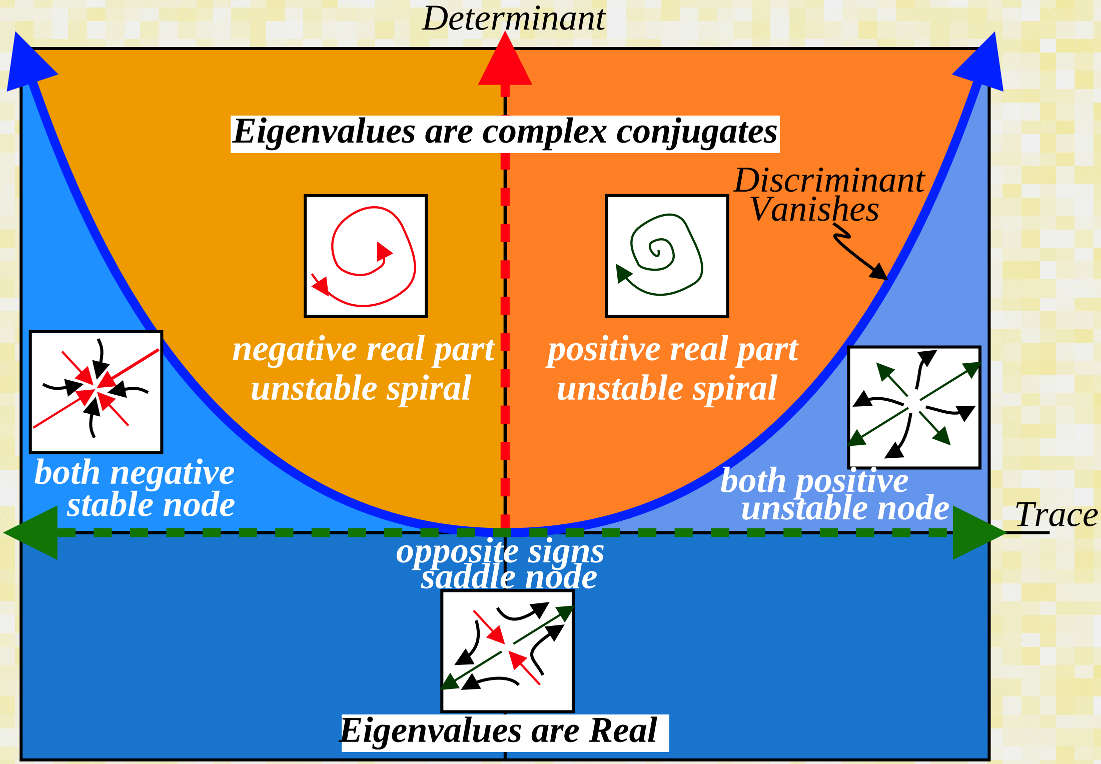


Figure 25-27: Illustration of the five regions according to their behavior near the fixed point.

At the point where the five regions come together, all the entries of the matrix of coefficients are zero and the physical behavior is then determined by expanding Eq. 25-1 to the next highest order at which the coefficients are not all zero.

Lecture 25 MATHEMATICA® Example 1

Functions to Analyze Fixed Points for Two-Dimensional Systems

notebook (non-evaluated) pdf (evaluated) html (evaluated)

Several functions are defined that identify the type of fixed point, their stability and the orientation of the stable and unstable eigenvectors.

1: *Eigenconsistency* takes two eigenvalues as arguments and checks if they derive from a real 2 × 2 matrix. If the eigenvalues are complex and the matrix is real, then the two eigenvalues must be complex conjugates.

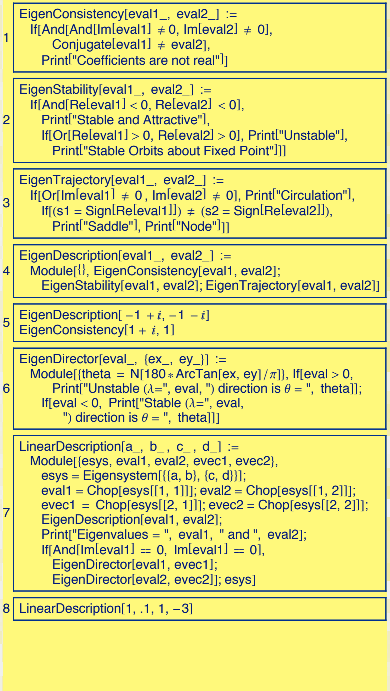
2: *Eigenstability* determines the sign of the real part of each eigenvalue and uses `Print` to display a friendly message about the solution's stability.

3: From the nature of the two eigenvalues, *EigenTrajectory* will print a friendly message about the *fixed point type.*

4: *EigenDescription* collects the previous three messaging functions into a single function.

6: *EigenDirector* takes a the stucture resulting from `Eigensystem` and uses the orientation of the eigenvectors and `ArcTan` to describe the local orientation of any stable or unstable directions.

7: *LinearDescription* takes the entries of a 2 × 2 matrix, calculates the eigensystem, removes any numerically trivial parts with `Chop` and then proceeds to call all the previous messaging functions to give a complete description of the fixed points behavior.



```
1  EigenConsistency[eval1_, eval2_] :=
     If[And[And[Im[eval1] ≠ 0, Im[eval2] ≠ 0],
         Conjugate[eval1] ≠ eval2],
       Print["Coefficients are not real"]]

2  EigenStability[eval1_, eval2_] :=
     If[And[Re[eval1] < 0, Re[eval2] < 0],
       Print["Stable and Attractive"],
       If[Or[Re[eval1] > 0, Re[eval2] > 0], Print["Unstable"],
         Print["Stable Orbits about Fixed Point"]]]

3  EigenTrajectory[eval1_, eval2_] :=
     If[Or[Im[eval1] ≠ 0 , Im[eval2] ≠ 0], Print["Circulation"],
       If[(s1 = Sign[Re[eval1]]) ≠ (s2 = Sign[Re[eval2]]),
         Print["Saddle"], Print["Node"]]]

4  EigenDescription[eval1_, eval2_] :=
     Module[{}, EigenConsistency[eval1, eval2];
       EigenStability[eval1, eval2]; EigenTrajectory[eval1, eval2]]

5  EigenDescription[ −1 + i, −1 − i]
   EigenConsistency[1 + i, 1]

6  EigenDirector[eval_, {ex_, ey_}] :=
     Module[{theta = N[180 * ArcTan[ex, ey] / π]}, If[eval > 0,
         Print["Unstable (λ=", eval, ") direction is θ = ", theta]];
       If[eval < 0, Print["Stable (λ=", eval,
           ") direction is θ = ", theta]]]

7  LinearDescription[a_, b_ , c_ , d_] :=
     Module[{esys, eval1, eval2, evec1, evec2},
       esys = Eigensystem[{{a, b}, {c, d}}];
       eval1 = Chop[esys[[1, 1]]]; eval2 = Chop[esys[[1, 2]]];
       evec1 = Chop[esys[[2, 1]]]; evec2 = Chop[esys[[2, 2]]];
       EigenDescription[eval1, eval2];
       Print["Eigenvalues = ", eval1, " and ", eval2];
       If[And[Im[eval1] == 0, Im[eval1] == 0],
         EigenDirector[eval1, evec1];
         EigenDirector[eval2, evec2]]; esys]

8  LinearDescription[1, .1, 1, −3]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

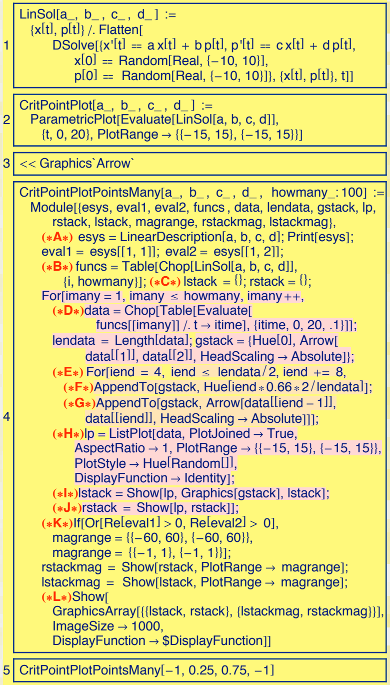
Quit

Lecture 25 Mathematica® Example 2

Visualizing the Behavior at a Fixed Point in the Plane

notebook (non-evaluated) pdf (evaluated) html (evaluated)

A function for visualizing the behavior of a fixed point with oriented arrows is constructed.



```
1  LinSol[a_, b_ , c_ , d_ ] :=
     {x[t], p[t]} /. Flatten[
         DSolve[{x'[t] == a x[t] + b p[t], p'[t] == c x[t] + d p[t],
             x[0] == Random[Real, {-10, 10}],
             p[0] == Random[Real, {-10, 10}]}, {x[t], p[t]}, t]]

2  CritPointPlot[a_, b_ , c_ , d_ ] :=
     ParametricPlot[Evaluate[LinSol[a, b, c, d]],
       {t, 0, 20}, PlotRange → {{-15, 15}, {-15, 15}}]

3  << Graphics`Arrow`

4  CritPointPlotPointsMany[a_, b_ , c_ , d_ , howmany_:100] :=
     Module[{esys, eval1, eval2, funcs , data, lendata, gstack, lp,
         rstack, lstack, magrange, rstackmag, lstackmag},
       (*A*) esys = LinearDescription[a, b, c, d]; Print[esys];
       eval1 = esys[[1, 1]]; eval2 = esys[[1, 2]];
       (*B*) funcs = Table[Chop[LinSol[a, b, c, d]],
           {i, howmany}]; (*C*) lstack = {}; rstack = {};
       For[imany = 1, imany ≤ howmany, imany++,
         (*D*)data = Chop[Table[Evaluate[
                 funcs[[imany]] /. t → itime], {itime, 0, 20, .1}]];
         lendata = Length[data]; gstack = {Hue[0], Arrow[
               data[[1]], data[[2]], HeadScaling → Absolute]};
         (*E*) For[iend = 4, iend ≤ lendata/2, iend += 8,
           (*F*)AppendTo[gstack, Hue[iend*0.66*2/lendata]];
           (*G*)AppendTo[gstack, Arrow[data[[iend - 1]],
               data[[iend]], HeadScaling → Absolute]]];
         (*H*)lp = ListPlot[data, PlotJoined → True,
             AspectRatio → 1, PlotRange → {{-15, 15}, {-15, 15}},
             PlotStyle → Hue[Random[]],
             DisplayFunction → Identity];
         (*I*)lstack = Show[lp, Graphics[gstack], lstack];
         (*J*)rstack = Show[lp, rstack]];
       (*K*)If[Or[Re[eval1] > 0, Re[eval2] > 0],
         magrange = {{-60, 60}, {-60, 60}},
         magrange = {{-1, 1}, {-1, 1}}];
       rstackmag = Show[rstack, PlotRange → magrange];
       lstackmag = Show[lstack, PlotRange → magrange];
       (*L*)Show[
         GraphicsArray[{{lstack, rstack}, {lstackmag, rstackmag}}],
         ImageSize → 1000,
         DisplayFunction → $DisplayFunction]]

5  CritPointPlotPointsMany[-1, 0.25, 0.75, -1]
```

1: *Linsol* , with four arguments representing the Jacobian entries, calculates the solution for an initial point picked randomly from the domain $(-10 \leq x \leq 10), (-10 \leq y \leq 10)$. In this case, the fixed point is assumed to have been translated to the origin. It calls `DSolve`, on a coupled pair of first-order ODEs and uses `Random` to generate the initial conditions.

2: This function takes the entries for the Jacobian and then uses `ParametricPlot` to plot the solution returns from *LinSolve* . However, the results are not very easy to interpret: one can't discern the direction of the tragjectory, nor the nature of an fixed point.

3: `Graphics‘Arrow‘` has arrow graphics objects to indicate the direction.

4: *CritPointPlotPointsMany* is an elaborate function that provides four graphical views of a critical point by computing and plotting a large number of trajectories emanating from random intial points. This function illustrates an example of the use of *optional function-arguments* in `howmany_:100` (i.e., the number of random trajectories to compute) which defaults to 100 if that argument is left off the function when it is called.

4A Compute the eigensystem and assign it to `esys`, *LinearDescription* will also print information about the critial point.

4B `funcs` is a list of solution-pairs for different initial points.

4C `Show` will be called recursively to add graphical objects to a list `lstack`. `lstack` is a placeholder for plots that contain arrows. Items **D–J** are in the body of a loop over each of the `homany` trajectories. **4D**: data is created for 20 discrete points at equal $\Delta t = 0.1$ along the current trajectory. A red arrow object is created at the beginning of the trajectory and stored in `gstack`. **4E**: This will loop over points the first half of the data. Items **F–G** are within the loop's body. **4F**: The color of the arrow will shift towards blue with time. **4G**: The arrow object is added to `gstack`. **4H**: The data is plotted with `ListPlot`. **4I**: This combines the contents of the plot and the two graphics-lists and assigns it to `lstack`—this iteratively grows `lstack`. **4J**: `rstack` will only contain plots and no arrows. **4K**: To focus on the long-time behavior, the `PlotRange` will need to be large (like a telescope) for unstable systems and small (like a microscope) for stable fixed points. The `If` will pick the appropriate `PlotRange`. **4L**: `GraphicsArray` produces an array of plots.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Unstable Manifolds

The phase portraits that were visualized in the above example help illustrate a very powerful mathematical method from non-linear mechanics.

Consider the saddle-node that has one positive (unstable) and one negative (stable) eigenvalue. Those initial points that are located in regions where the negative (stable) eigenvalue dominates are quickly swept towards the fixed point and then follow the unstable direction away from the fixed point. Roughly speaking, the stable values are 'smashed' onto the unstable direction and virtually all of the motion takes place near the unstable direction.

This idea allows a large system (i.e., one in which the vector $\vec{y}(t)$ has many components) to be reduced to a smaller system in which the stable directions have been approximated by a thin region near the trajectories associated with the unstable eigenvalues. This is sometimes called reduction of "fast variables" onto the unstable manifolds.

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Lecture 26: Separation of Variables and Solutions to Common ODEs

Reading:
Kreyszig Sections: 5.3, 5.5, 5.6 (pages177–180, 189–197, 198–202)

Special Functions: Solutions to Common ODEs

Most calculators have a button that evaluates the eigensolution to the simple first-order ODE $dy/dt = \lambda y$. Also, most calculators have buttons that evaluate the eigensolutions to the simple second-order ODE: $d^2y/dt^2 = \lambda y$.

Of course, these are also just the exponential and trigonometric functions.

However, there are many more simple differential equations that follow from physical models and these also have known solutions that are not simple combinations of sines, cosines, and exponentials. The solutions to these differential equations are called *special functions.* MATHEMATICA® has an extensive list of special functions and these are collected in its help browser.

For example, the positions of a vibrating drum head are modeled with in cylindrical coordinates by Bessel's equation:

$$\begin{aligned} r^2\frac{d^2h}{dr^2} + r\frac{dh}{dr} + (k^2r^2 - m^2)h = 0 \\ \rho^2\frac{d^2h}{d\rho^2} + \rho\frac{dh}{d\rho} + (\rho^2 - m^2)h = 0 \end{aligned} \tag{26-1}$$

where in the second equation $\rho = kr$. The displacement of the drum is $h(r)$; k is related to an inverse wavelength (e.g., the wavelength would be the radius of the drum divided by the number of maxima in the drum head shape) and m is the mode (e.g., the number of maxima traversing the drum by 2π in a circular direction).

There two solutions to Bessel's equation and the general solution is the sum the two:

$$\begin{aligned} h(r) = C_1J_m(kr) + C_2Y_m(kr) \\ h(\rho) = C_1J_m(\rho) + C_2Y_m(\rho) \end{aligned} \tag{26-2}$$

where $J_m(x)$ is called (naturally enough) an *order-m Bessel function of the first kind* and $Y_m(x)$ is called (naturally enough) an *order-m Bessel function of the second kind.* These are analogous to the sines and cosines, but for a different ODE.

Another equation that appears in models of the angular deformations of body in a central force potentials (for example, the ion distribution about a fixed charge; or, the Schrödinger equation for the electron in a hydrogen atom) in spherical coordinates is Legendre's equation:

$$\begin{aligned} &\frac{1}{\sin\theta}\frac{d}{d\theta}\left(\sin\theta\frac{d\Xi}{d\theta}\right)+\left[\ell(\ell+1)-\frac{m^2}{\sin^2\theta}\right]\Xi=0 \\ &\frac{d}{d\mu}\left[(1-\mu^2)\frac{d\Xi}{d\theta}\right]+\left[\ell(\ell+1)-\frac{m^2}{1-\mu^2}\right]\Xi=0 \end{aligned} \tag{26-3}$$

where $\mu \equiv \cos\theta$ so that $-1 \le \mu \le 1$. The value ℓ is related to the number of modes in the θ direction and m is related to the number of modes in the ϕ direction.

Legendre's equation has two solutions:

$$\Xi(\mu) = C_1 P_{lm}(\mu) + C_2 Q_{lm}(\mu) \tag{26-4}$$

The eigensolution $P_{lm}(\mu)$ is called (again, naturally enough) *order m Legendre functions of the first kind* and $Q_{lm}(\mu)$ are called *order lm Legendre functions of the second kind.*

There are many other types of special functions.

Lecture 26 MATHEMATICA® Example 1

Visualizing special functions.

notebook (non-evaluated) pdf (evaluated) html (evaluated)

The ODEs that produce Bessel, Legendre, LaGuerre, and Hypogeometric functions are solved and these special functions are visualized.

1: This is *Bessel's equation* for $y(x)$.

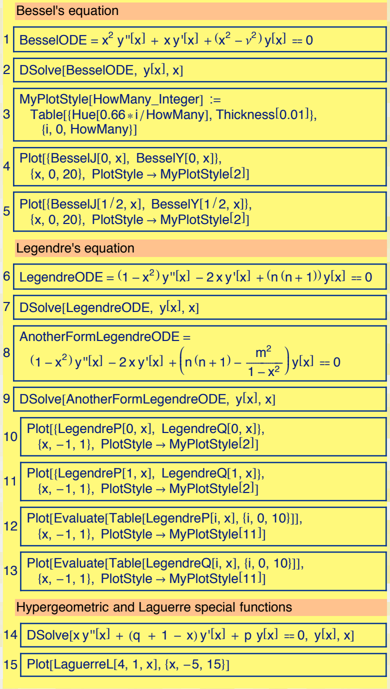
3: *MyPlotStyle* is a function to set `PlotStyle` for a given number of curves so that their colors are spread over `Hue` from red (0) to blue (0.66).

4: This will produce at plot of the zeroeth Bessel's function of the first kind (`BesselJ`) and zeroeth Bessel's function of the second kind (`BesselY`)/ solutions.

6: This is two-parameter form of *Ledendre's equation* for $y(x)$.

8: This is another form of Ledendre's equation for $y(x)$.

14: This will produce solutions to *Laguerres's equation* for $y(x)$.



Bessel's equation

```
1  BesselODE = x^2 y''[x] + x y'[x] + (x^2 - ν^2) y[x] == 0

2  DSolve[BesselODE, y[x], x]

3  MyPlotStyle[HowMany_Integer] :=
     Table[{Hue[0.66 * i / HowMany], Thickness[0.01]},
       {i, 0, HowMany}]

4  Plot[{BesselJ[0, x], BesselY[0, x]},
     {x, 0, 20}, PlotStyle → MyPlotStyle[2]]

5  Plot[{BesselJ[1 / 2, x], BesselY[1 / 2, x]},
     {x, 0, 20}, PlotStyle → MyPlotStyle[2]]
```

Legendre's equation

```
6  LegendreODE = (1 - x^2) y''[x] - 2 x y'[x] + (n (n + 1)) y[x] == 0

7  DSolve[LegendreODE, y[x], x]

8  AnotherFormLegendreODE =
     (1 - x^2) y''[x] - 2 x y'[x] + (n (n + 1) - m^2/(1 - x^2)) y[x] == 0

9  DSolve[AnotherFormLegendreODE, y[x], x]

10 Plot[{LegendreP[0, x], LegendreQ[0, x]},
     {x, -1, 1}, PlotStyle → MyPlotStyle[2]]

11 Plot[{LegendreP[1, x], LegendreQ[1, x]},
     {x, -1, 1}, PlotStyle → MyPlotStyle[2]]

12 Plot[Evaluate[Table[LegendreP[i, x], {i, 0, 10}]],
     {x, -1, 1}, PlotStyle → MyPlotStyle[11]]

13 Plot[Evaluate[Table[LegendreQ[i, x], {i, 0, 10}]],
     {x, -1, 1}, PlotStyle → MyPlotStyle[11]]
```

Hypergeometric and Laguerre special functions

```
14 DSolve[x y''[x] + (q + 1 - x) y'[x] + p y[x] == 0, y[x], x]

15 Plot[LaguerreL[4, 1, x], {x, -5, 15}]
```

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Partial Differential Equations: Separation of Variables

Many *ordinary* differential equations that arise in practice derive from methods to solve *partial differential equations.*

In other words, the solution to the partial differential equation involving $c(x, y, z, t)$ and its partial derivatives with respect to x, y, z, and t can sometimes be reduced to the solution of several ordinary differential equations.

In practice, most of the partial differential equations that can be solved analytically are solved by the *method of separation of variables.* Separation of variables works by isolating one of the variables onto one side of equality—it is best described by simple example and here the one-dimensional wave-equation is a prototype. The wave-equation (e.g., the time (t)-dependent propagation of a scalar quantity (h) such as height, density, charge, etc. in a single direction x is:

$$\frac{\partial^2 h(x,t)}{\partial t^2} = v^2 \frac{\partial^2 h(x,t)}{\partial x^2} \qquad (26\text{-}5)$$

where v is the phase-velocity $v = \omega/k$, ω is the angular frequency describing how rapidly the phase of the wave changes as it moves past a fixed position and $k = 2\pi/\lambda$ is the wave-number.

Consider a specific case in which waves are propogating in a guitar string of length L—this will give Dirichlet boundary conditions at the guitar's nut and saddle:

$$h(x = 0, t) = 0 \qquad \text{and} \qquad h(x = L, t) = 0 \qquad (26\text{-}6)$$

(two boundary conditions—one for each spatial derivative). The shape of plucked string gives the *initial condition*; for example, this could be modeled with a triangular shape:

$$h(x, t = 0) = \begin{cases} Ax/\ell & 0 < x < \ell \\ A\frac{L-x}{L-\ell} & \ell < x < L \end{cases} \qquad (26\text{-}7)$$

where the string is plucked at a position $x = \ell$ with a displacement A at time $t = 0$.

The separation of variables method begins with the assumption that *the function can be factored into independent functions of the dependent variables.* For Eq. 26-5, this assumption is written as:

$$h(x, t) = \chi(x)\tau(t) \qquad (26\text{-}8)$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

If this is inserted into Eq. 26-5, and both sides are divided by $v^2\chi(x)\tau(t)$ then

$$\frac{1}{v^2\tau(t)}\frac{d^2\tau(t)}{dt^2} = \frac{1}{\chi(x)}\frac{d^2\chi(x)}{dx^2} \tag{26-9}$$

Note that both sides of Eq. 26-9 depend on different variables and observe Fig. 26-28.

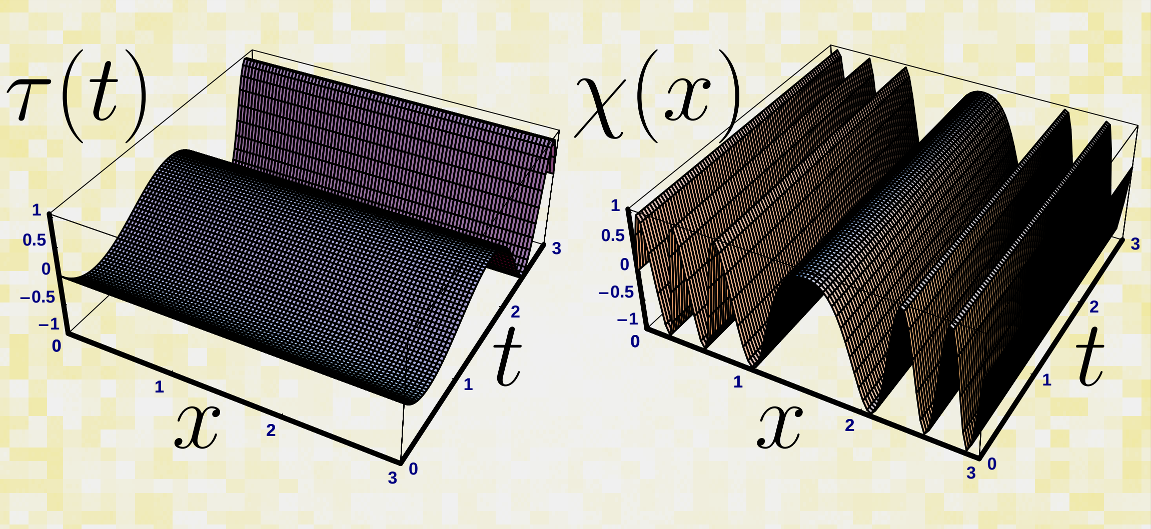


Figure 26-28: If two functions, $T(t)$ and $X(x)$, depend on different variables and are equal, then they can only be constant $\chi(x) = \tau(t) = \lambda$

Thus, both sides of Eq. 26-9 can be set equal to a *separation constant* λ:

$$\frac{1}{\chi(x)}\frac{d^2\chi}{dx^2} = \lambda \qquad \text{or} \qquad \frac{d^2\chi}{dx^2} - \lambda\chi = 0 \tag{26-10}$$

and

$$\frac{1}{v^2\tau(t)}\frac{d^2\tau}{dt^2} = \lambda \qquad \text{or} \qquad \frac{d^2\tau}{dt^2} - \lambda v^2\tau = 0 \tag{26-11}$$

For Eq. 26-10, the boundary conditions (Eq. 26-6) can also be written in terms of $\chi(x)$:

$$\chi(x=0)=0 \qquad \text{and} \qquad \chi(x=L)=0 \tag{26-12}$$

As advertised, ODEs are generated (Eqs. 26-10 and 26-11) in the process of solving the PDE 26-5. As only Eq. 26-10 has a boundary condition, it is solved first; in general its solution appears as:

$$\chi(x) = \begin{cases} A_+ \exp(\sqrt{\lambda}x) + B_+ \exp(-\sqrt{\lambda}x) \text{ 0.25in} & \text{if}\lambda > 0 \\ A_0 x + B_0 & \text{if}\lambda = 0 \\ A_- \cos(\sqrt{-\lambda}x) + B_- \sin(\sqrt{-\lambda}x) \text{ 0.25in} & \text{if}\lambda < 0 \end{cases} \tag{26-13}$$

The boundary conditions 26-12 place an initial restriction on the separation constant $\lambda > 0$ and specify that $\chi(x)$ must be a sum of sines and cosines. Furthermore, trying to solve Eqs. 26-12 at $x = 0$ shows that $A_- = 0$; at $x = L$, solutions must coincide with any of the *zeroes of the eigenfunction* $\sin(kx) = \sin(\sqrt{-\lambda}x)$, or

$$\lambda_n = \frac{-n^2\pi^2}{L^2} = -k_n^2 \quad n = 1, 2, 3, \ldots \tag{26-14}$$

The λ_n (or equivalently the k_n) become *eigenvalues* of the ODE and generate an infinity of eigenfunctions with independent amplitudes A_n: $\chi_n(x) = A_n \sin(n\pi x/L)$.

This infinity of eigenfunctions are needed to satisfy the initial conditions Eqs. 26-7, but first the solution to the second ODE 26-11 must be obtained for *the restriced set of eigenvalues* for the separation constant:

$$\frac{d^2\tau}{dt^2} + \frac{n^2\pi^2}{L^2} v^2 \tau = 0 \tag{26-15}$$

so that, in general,

$$\tau_n(t) = \mathcal{E}_n^\star \cos(\frac{n\pi v t}{L}) + \mathcal{O}_n^\star \sin(\frac{n\pi v t}{L}) \tag{26-16}$$

and therefore with Eq. 26-8, the superposition of all solutions is

$$\begin{aligned} h(x,t) &= \sum_{n=1}^{\infty} \tau_n(t)\chi_n(x) \\ &= \sum_{n=1}^{\infty} \left(\mathcal{E}_n \cos\frac{n\pi v t}{L} + \mathcal{O}_n \sin\frac{n\pi v t}{L} \right) \sin\frac{n\pi x}{L} \end{aligned} \tag{26-17}$$

And the initial conditions become

$$h(x,0) = \sum_{n=1}^{\infty} \tau_n(0)\chi_n(x) = \sum_{n=1}^{\infty} \mathcal{E}_n \sin\frac{n\pi x}{L} \tag{26-18}$$

(the sine (odd) coefficients are not needed in this case) and determination of the coefficients is reduced to the Fourier representation. For the initial conditions in Eq. 26-7, these can be computed using Eq. 17-10:

$$\mathcal{E}_n = \frac{AL^2 \sin\frac{2\pi n\ell}{L}}{(2\pi n)^2 \ell(L-\ell)} \tag{26-19}$$

giving a solution:

$$h(x,t) = \sum_{n=1}^{\infty} \frac{AL^2 \sin\frac{2\pi n\ell}{L}}{(2\pi n)^2 \ell(L-\ell)} \cos\frac{n\pi v t}{L} \sin\frac{n\pi x}{L} \tag{26-20}$$

The Shrödinger equation for a central potential serves as a more involved example for the method of separation of variables and is provided in the following section.

Special Functions in the Eigenfunctions of the Hydrogen Atom

The time-independent Shrödinger for the electron in a hydrogen atom is a partial differential equation involving three spatial variables. If the mass of the nucleus can be considered very large compared to that of an electron, then it is reasonable to fix the center of a spherical $1/r$–potential at the origin and use spherical coordinates (r, θ, ϕ) to express the Shrödinger equation:[18]

$$\begin{gathered} \frac{\hbar^2}{2m_e}\nabla^2\psi + V\psi = E\psi \\ \frac{\hbar^2}{2m_e}\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right]\psi - \frac{Ze^2\psi}{\epsilon_o r} = E\psi \end{gathered} \tag{26-21}$$

Here, ϕ wraps around like longitude and ϕ goes north and south from the equator (θ=0) like latitude.

[18] To treat the hydrogen atom more accurately, the reduced mass $\tilde{m} = 1/(1/m + 1/M)$, weighted coordinates $\tilde{x} = (mx_m + Mx_M)/M$ etc., and relative positions $\Delta x = x_m - x_M$, etc., would produce PDEs for the entire system and for the relative positions.

Because the potential depends only on r, the initial separation is between the radial and angular parts,

$$\psi(r,\theta,\phi) = \rho(r)Y(\theta,\phi) \tag{26-22}$$

which separates Eq. 26-21 into

$$\begin{aligned} \frac{1}{\rho}\frac{d}{dr}\left(r^2\frac{d\rho}{dr}\right) + \frac{2m_e r^2}{\hbar^2}\left(E + \frac{Ze^2}{\epsilon_o r}\right) \\ = \frac{1}{Y}\left[\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial Y}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 Y}{\partial\phi^2}\right] \end{aligned} \tag{26-23}$$

Therefore, the two sides must be equal to a separation constant λ. The radial part becomes

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\rho}{dr}\right) + \left[\frac{2m_e}{\hbar^2}\left(E - \frac{Ze^2}{\epsilon_o r}\right) - \frac{\lambda}{r^2}\right]\rho = 0 \tag{26-24}$$

and the angular part becomes

$$\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial Y}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 Y}{\partial\phi^2} + \lambda Y = 0 \tag{26-25}$$

Solutions to Eq. 26-25 are related to the *spherical harmonics* and can be derived through another separation of variables. Putting

$$Y(\theta,\phi) = \Theta(\theta)\Phi(\phi) \tag{26-26}$$

into Eq. 26-25 gives

$$\frac{1}{\Phi}\frac{d^2\Phi}{d\phi^2} = -m^2 = \frac{-\sin\theta}{\Theta}\frac{d}{d\theta}\left(\sin\theta\frac{d\Theta}{d\theta}\right) - \lambda\sin^2\theta \tag{26-27}$$

where the separation constant, $-m^2$, is explicitly set to a negative quantity reflecting that Φ must have periodic solutions, i.e., the two separated ODEs are:

$$\begin{aligned} 0 &= \frac{d^2\Phi}{d\phi^2} + m^2\Phi \\ 0 &= \frac{1}{\sin\theta}\frac{d}{d\theta}\left(\sin\theta\frac{d\Theta}{d\theta}\right)\left(\lambda - \frac{m^2}{\sin^2\theta}\right)\Theta \end{aligned} \tag{26-28}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

The first of these has the same form as Eq. 26-10 (solutions given by Eq. 26-13) but here the solutions will be written as

$$\Phi(\phi) = A_+ e^{\imath m \phi} + A_- e^{-\imath m \phi} \tag{26-29}$$

because the wavefunction is complex in general. Here, m, indicated how many maxima that the latitudinal part of ϕ will have, and the two different A multiply wavefunctions that are out-of-phase by π. Either solution can be obtained by changing the sign of m, therefore in general

$$\Phi_m(\phi) = \frac{1}{\sqrt{2\pi}} e^{\imath m \phi} \qquad m = 0, \pm 1, \pm 2, \ldots \tag{26-30}$$

where the prefactor normalizes Φ so that $\int_0^{2\pi} \Phi_m \Phi_m^* d\phi = 1$.

The second of Eqs. 26-28 has the same form as Eq. 26-3 for which the relevant solution (because they are bounded) if an only if

$$\lambda = \ell(\ell + 1) \quad \text{and} \quad |m| \leq \ell \quad \text{for} \quad \ell = 0, 1, 2, \ldots \tag{26-31}$$

Putting all of this together and normalizing, the spherical harmonic part of the H-atom orbital (Eq. 26-25) is:

$$Y_{\ell,m} = \sigma(m) \sqrt{\frac{2l+1}{4\pi} \frac{(\ell - |m|)!}{(\ell + |m|)!}} P_{\ell,m}(\cos\theta) e^{\imath m \phi} \tag{26-32}$$

where $\sigma(m) = 1$ if $m \leq 0$ or m even, and $\sigma(m) = -1$ for odd-positive m.

Finally, for the radial part of the H-atom orbital (Eq. 26-24) with $\lambda = \ell(\ell + 1)$ becomes an ODE in r:

$$\frac{1}{r^2} \frac{d}{dr} \left(r^2 \frac{d\rho}{dr} \right) + \left[\frac{2m_e}{\hbar^2} \left(E - \frac{Ze^2}{\epsilon_o r} \right) - \frac{\ell(\ell+1)}{r^2} \right] \rho = 0 \tag{26-33}$$

where $E < 0$ defines a bound state.

This equation can also be solved analytically, and it has integer eigenvalues $n > \ell \geq |m|$:

$$\rho_{n,\ell}(r) = -\sqrt{\left(\frac{2Z}{na_0} \right)^3 \frac{(n - \ell - 1)!}{2n[(n+\ell)!]^3}} \exp\left(\frac{-Zr}{na_0}\right) \left(\frac{-2Zr}{na_0} \right)^\ell L_{n+1,2\ell+1} \left(\frac{2Zr}{na_0} \right) \tag{26-34}$$

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

where $a_0 = \hbar^2 \epsilon_0 / m_e e^2$ is the *Bohr radius* and the $L_{q,p}$ is yet another special function—c'est *LaGuerre polynomials*.

The Hydrogen orbitals are visualized in the following example.

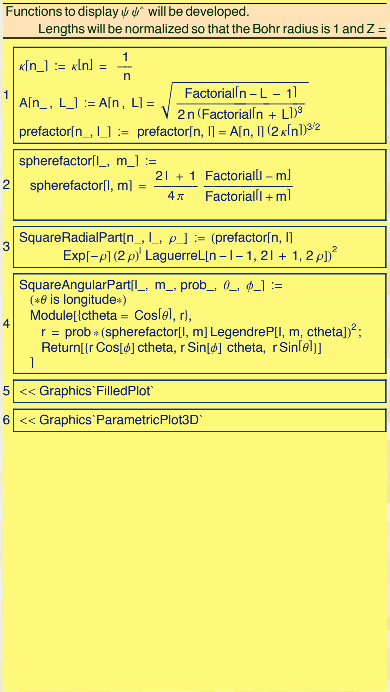
Lecture 26 MATHEMATICA® Example 2

Visualizing the Hydrogen atom eigenfunctions

notebook (non-evaluated) pdf (evaluated) html (evaluated)

This example is still in progress—it has not been check for accuracy yet.

1: This example will be completed at a later date.



Functions to display $\psi\,\psi^*$ will be developed.
Lengths will be normalized so that the Bohr radius is 1 and Z =

1 κ[n_] := κ[n] = $\frac{1}{n}$
A[n_ , L_] := A[n , L] = $\sqrt{\frac{\text{Factorial}[n-L-1]}{2\,n\,(\text{Factorial}[n+L])^3}}$
prefactor[n_, l_] := prefactor[n, l] = A[n, l] $(2\,\kappa[n])^{3/2}$

2 spherefactor[l_, m_] :=
spherefactor[l, m] = $\frac{2\,l+1}{4\pi}\ \frac{\text{Factorial}[l-m]}{\text{Factorial}[l+m]}$

3 SquareRadialPart[n_, l_, ρ_] := (prefactor[n, l]
Exp[$-\rho$] $(2\,\rho)^l$ LaguerreL[n − l − 1, 2 l + 1, 2 ρ])^2

4 SquareAngularPart[l_, m_, prob_, θ_, ϕ_] :=
(*θ is longitude*)
Module[{ctheta = Cos[θ], r},
r = prob * (spherefactor[l, m] LegendreP[l, m, ctheta])^2;
Return[{r Cos[ϕ] ctheta, r Sin[ϕ] ctheta, r Sin[θ]}]
]

5 << Graphics`FilledPlot`

6 << Graphics`ParametricPlot3D`

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

Index

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016
3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home
Full Screen
Close
Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

3.016 Home
◀◀ ◀ ▶ ▶▶
Full Screen
Close
Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit

3.016 Home

◀◀ ◀ ▶ ▶▶

Full Screen

Close

Quit