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)

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-) product is the 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 $\vec{a}\cdot\vec{b} = \vec{b}\cdot\vec{a}$

satisfies Schwarz inequality $\|\vec{a}\cdot\vec{b}\| \leq \|\vec{b}\|\|\vec{a}\|$

ratifies triangle inequality $\|\vec{a}+\vec{b}\| \leq \|\vec{b}\| + \|\vec{a}\|$

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:

$$\cos\alpha = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \hat{n_a}\cdot\hat{n_b} \tag{11-2}$$

3.016 Home

Full Screen

Close

Quit

where $\hat{n}_i$ is the unit vector that shares a direction with i. *<u>Caution:</u> 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 $\times$) 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, but which normal should be picked? By convention, the "right-hand-rule" defines which of the two normals 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, so 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).

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 the 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.

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Cross Product Example

This is a simple demonstration of the vector product of two spatial vectors.

Here is the built-in cross product between two vectors

```
crossab = Cross[{a1, a2, a3} , {b1, b2, b3}]
```
1

And, here is the standard visual way to do it by hand with the determinant.

```
detab = Det[( i  j  k
              a1 a2 a3
              b1 b2 b3 )]
```
2

Pick out each of the compenents to create the vector

```
testcrossab = {Coefficient[detab, i],
  Coefficient[detab, j],
  Coefficient[detab, k]}
```
3

Check for equality between the old-fashioned way and Mathematica's built-in function

```
testcrossab == crossab
```
4

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:

$$\vec{a} \times \vec{b} = \det \begin{pmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{pmatrix}$$

3–4: `Coefficient` is used to extract each vector component and create a vector result, and then equality test the two vectors to show the equivalence.

3.016 Home

Full Screen

Close

Quit

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Visualizing Space-Curves as Time-Dependent Vectors

Examples of $\vec{x}(t)$ and $d\vec{x}/dt$ are illustrated as curves and as animations.

Create a trajectory of a point or a particle

```
XVector[t_] :=
 {Cos[6 t], Sin[4 t], Sin[t] + Cos[t]}
```
1

ParametricPlot3D allows us to visualize the entire trajectory at once.

```
ParametricPlot3D[XVector[t],
 {t, 0, 2 π}, PlotStyle → {Thick, Blue},
 AxesLabel → {"x", "y", "z"}]
```
2

Here is a function to create a graphic with a variable end - point. We will have the function remember when it has already computed a graphic, trading memory for a possible speed-up.

```
paraplot[time_] := paraplot[time] =
  ParametricPlot3D[XVector[t],
   {t, 0, time}, PlotStyle → {Thick, Blue},
   AxesLabel → {"x", "y", "z"}]
```
3

Use manipulate on the graphics function to visualize how the curve develops with its parameter

```
Manipulate[paraplot[time],
 {{time, 0.05}, 0.01, 2 π}]
```
4

However, we need to fix the length scale between frames, so we use the last graphic to infer what PlotRange should be.

```
Manipulate[Show[paraplot[time],
  PlotRange → {{-1, 1}, {-1, 1}, {-1.5, 1.5}}],
 {{time, 0.05}, 0.01, 2 π}]
```
5

Next, we add a graphic element to show the vector, drawn from the origin, for each end-point.

```
Manipulate[
 Show[{paraplot[time], Graphics3D[{Cylinder[
      {{0, 0, 0}, XVector[time]}, 0.03]}]},
  PlotRange → {{-1, 1}, {-1, 1}, {-1.5, 1.5}}],
 {{time, 0.05}, 0.01, 2 π}]
```
6

1: A list of three time-dependent components for (x, y, z) is constructed as the function $XVector$.

2: `ParametricPlot3D` takes a three-component vector as an argument and then will plot the evolution of the vector as a function of a parameter.

3: To visualize the evolution of the curve, it is useful to plot the resulting trajectory. We create a function that constructs a plot up to a variable end-point, `time`, which appears as the upper-bound to `ParametricPlot3D`.

4: `Manipulate` provides an interactive animation of the curve's development.

5: However, it is better if the length scale is fixed. We set `PlotRange` from visual inspection of the last frame of the previous example.

6: To improve the visualization, we add a `Cylinder` graphics primitive to illustrate the vector drawn from the origin.

3.016

3.016 Home

Full Screen

Close

Quit

Derivatives of 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 then reappear an instant later, $t + dt$, some finite distance distance away from $\vec{p}(t)$.

If $\vec{p}(t)$ is continuous, then the limit is:

$$\frac{d\vec{p}(t)}{dt} = \lim_{\Delta t \to 0} \frac{\vec{p}(t + \Delta t) - \vec{p}(t)}{\Delta t} \tag{11-5}$$

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.

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Visualizing Time-Dependent Vectors and their Derivatives

Examples of $\vec{x}(t)$ and $d\vec{x}/dt$ are illustrated as curves and as animations.

The local derivative of the vector that we visualized above: $\vec{v} = \frac{d\vec{x}}{dt}$

```
Simplify[D[XVector[t], t]]
```
1

Write out a function for the derivative:

```
dxdt[s_] :=
 {-6 Sin[6 s], 4 Cos[4 s], Cos[s] - Sin[s]}
```
2

```
dxdtplot[time_] :=
 ParametricPlot3D[dxdt[t], {t, 0, time},
  PlotStyle → {Thick, Darker[Red]},
  AxesLabel → {"x", "y", "z"}]
dxdtplot[2 π]
```
3

```
Manipulate[
 Show[paraplot[time], dxdtplot[time],
  Graphics3D[{
    {Lighter[Blue], Cylinder[
      {{0, 0, 0}, XVector[time]}, 0.1]},
    {Lighter[Red], Cylinder[
      {{0, 0, 0}, dxdt[time]}, 0.1]}
   }], PlotRange → {{-6, 6}, {-4, 4},
    {-1.5, 1.5}}], {{time, 0.05}, 0.01, 2 π}]
```
4

To visualize the "tangency property," we translate the derivative-vector to the end of the space curve

```
Manipulate[
 Show[paraplot[time], dxdtplot[time],
  Graphics3D[{
    {Lighter[Blue], Cylinder[
      {{0, 0, 0}, XVector[time]}, 0.1]},
    {Lighter[Red], Translate[
      Cylinder[{{0, 0, 0}, dxdt[time]}, 0.1],
      XVector[time]]}
   }], PlotRange → {{-6, 6}, {-6, 6},
    {-4, 4}}], {{time, 0.05}, 0.01, 2 π}]
```
5

1: The derivative operator `D` is a *threadable function* so it will operate on each component of its vector argument; thus, we can obtain the vector derivative by operating on the entire vector.

2: The derivative-vector $d\vec{x}/dt$ is encoded as a function.

3: The derivative of the space-curve (from the above example) is visualized with a function that calls `ParameticPlot3D`. Because the space curve is differentiable and periodic, its derivative should be periodic as well, but it *appears to not be periodic*.

4: Using `Manipulate` reveals that the function is periodic.

5: Here, we repeat the interactive example, but use `Translate` to move the visualized derivative-vector to the end of the curve. This will give a visual demonstration of the tangent behavior of the derivative.

3.016 Home

Full Screen

Close

Quit

Review: Partial and total derivatives

One might also consider that 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 $\vec{x}$ and at time t.

Here, there are many different things which might be varied and which 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 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. $$\frac{\partial \vec{E}(x, y, z, t)}{\partial t} = \left(\frac{\partial \vec{E}}{\partial t}\right)_{\text{constant}x,y,z} \tag{11-7}$$

3.016 Home

Full Screen

Close

Quit

3.
$$\frac{d\vec{E}(x(t),y(t),z(t),t)}{dt} = \frac{\partial \vec{E}}{\partial x}\frac{dx}{dt} + \frac{\partial \vec{E}}{\partial y}\frac{dy}{dt} + \frac{\partial \vec{E}}{\partial z}\frac{dz}{dt} + \frac{\partial \vec{E}}{\partial t}\frac{dt}{dt} = \nabla\vec{E}(\vec{x}(t),t)\cdot\frac{d\vec{x}}{dt} + \frac{\partial \vec{E}}{\partial t} \tag{11-8}$$

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 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 visualize 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 absorbent paper at $x=y=0$ and 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 $\vec{J}$. This 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.

notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

Visualizing a Solution to the Diffusion Equation

The solution to a 2D diffusion equation in the infinite plane with rectangular initial conditions $c(-a/2 < x < a/2, -b/2 < y < b/2, t = 0) = 1$ and $c(|x| > a/2, |y| > b/2, t = 0) = 0$ is visualized and will serve as an example of the flux (or time-dependent gradient) field in the following example.

```
concentration =
 Integrate[Exp[-((xsource-x)^2 + (ysource-y)^2)/(4 Diffusivity t)]/(4 Pi Diffusivity t),
  {xsource, -a/2, a/2},
  {ysource, -b/2, b/2},
  Assumptions → Diffusivity > 0 && t > 0 &&
    a > 0 && b > 0 && x ∈ Reals && y ∈ Reals]
```
1

```
aspectRatio = 3;
b = aspectRatio a;
length = a;
time = length^2/Diffusivity;
ScaleRules =
  {t -> τ time, x -> ξ length, y -> η length};
```
2

```
scaledconc =
 Simplify[concentration /. ScaleRules,
  Assumptions → a > 0]
```
3

```
Plot3D[scaledconc /. τ → 0.003,
 {ξ, -3, 3}, {η, -3, 3}, PlotRange → {0, 1},
 MeshFunctions → {#3 &}, PlotPoints → 30,
 Mesh → 5, MeshStyle → {Thick}]
```
4

```
Manipulate[Plot3D[scaledconc /. τ → timevar,
  {ξ, -3, 3}, {η, -3, 3},
  PlotRange → {0, 1}, MaxRecursion → 4],
 {{timevar, 0.05}, 0.001, 0.1}]
```
5

```
cplots =
  Table[ContourPlot[scaledconc /. τ → timevar,
    {ξ, -3, 3}, {η, -3, 3},
    PlotRange → {0, 1}, ColorFunction →
     ColorData["TemperatureMap"]],
   {timevar, .001, .2, .005}];
```
6

```
ListAnimate[cplots]
```
7

1: This is the solution to the stated problem. It is obtained by integrating the appropriate *Green's function* which will be discussed in a later lecture. We use physical `Assumptions` in the call to `Integrate`.

2: We set a model parameter (`aspectRatio`) for the shape of the initial rectangle, and then define a characteristic length and time in terms of quantities that appear in the model. A set of rules are defined, *ScaleRules* , that can be applied to the solution to create a *non-dimensional model* the problem.

3: The re-scaled solution, *scaledconc* , depends only on non-dimensional quantities, ξ, η, and τ.

4: Here is an example of the solution at $\tau = 0.003$. We use the `MeshFunctions` option to `Plot3D` to draw the five isoconcentration lines on the surface.

5: This will produce an interactive animation of the solution. However, because the evaluation of each animation-frame is likely to be slow, this visualization will be sluggish on many computers.

6: A simpler graphical representation is obtained with `ContourPlot` by plotting contours of constant concentration. We pre-compute a table of plots and store the result.

7: The resulting animation is created from two-dimensional objects using `ListAnimate` on the pre-computed frames.

3.016 Home

Full Screen

Close

Quit

Visualizing the Diffusion Flux: The Time-Dependent Gradient Field

The time-dependent gradient field for the 2D diffusion equation treated above is visualized.

```
flux = {-D[scaledconc, ξ], -D[scaledconc, η]}
```
1

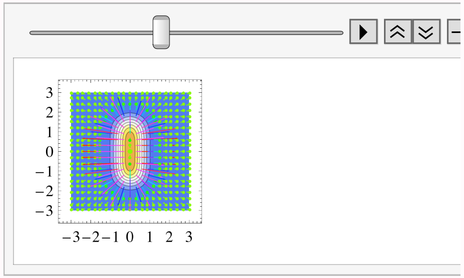
```
Needs["VectorFieldPlots`"];
```
2

```
VectorFieldPlot[flux /. τ → .08,
 {ξ, -3, 3}, {η, -3, 3}, PlotPoints → 21,
 ColorFunction → ColorData["DarkRainbow"]]
```
3

```
vplots =
  Table[VectorFieldPlot[flux /. τ → time,
    {ξ, -3, 3}, {η, -3, 3}, PlotPoints → 21,
    PlotRange → {{-4, 4}, {-4, 4}},
    Frame → True, MaxArrowLength → 3,
    ScaleFactor → None,
    ScaleFunction → (2 # &),
    ColorFunction → (Hue[0.25 + 0.75 #] &)],
   {time, .001, .2, .005}];
```
4

```
ListAnimate[vplots]
```
5

```
ListAnimate[
 Table[Show[cplots[[i]], vplots[[i]],
   PlotRange → 3.5 {{-1, 1}, {-1, 1}},
   ImageSize → {72, 72}],
  {i, 1, Length[cplots]}]]
```
6



1: The flux, $\vec{J}$, is the rate of matter-flow through an oriented plane per unit area. Fick's first law related the flux to the gradient of the concentration field, $\vec{J} = -D\nabla c$. Here, we compute a non-dimensionalized form of flux.

2: To plot the flux which is a vector field, the package `VectorFieldPlots` is loaded so that we have access to its `PlotVectorField` function. For example, here is an example of (minus) the gradient field at $\tau = 0.08$. The vectors are plotted on a square mesh and we color them according to their magnitude with `ColorFunction` and `ColorData`.

5–6: We precompute a series of frames and animate them. To ensure that the vector length scales are consistent between frames, the option `ScaleFactor` is used.

7: Here, we visualize time-dependent vector flux and the concentration together by using `Show` on pre-computed graphics.

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.

Index