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 their 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.

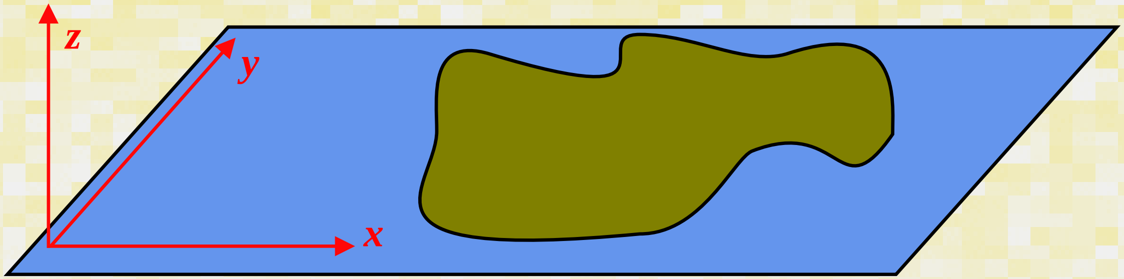


Figure 15-12: An irregular region on a plane surrounded by a closed curve. Once the closed curve (the edge of region) is specified, the area inside it is already determined. This is the simplest case as the area is the integral of the function $f = 1$ over $dxdy$. If some other function, $f(x,y)$, were 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).

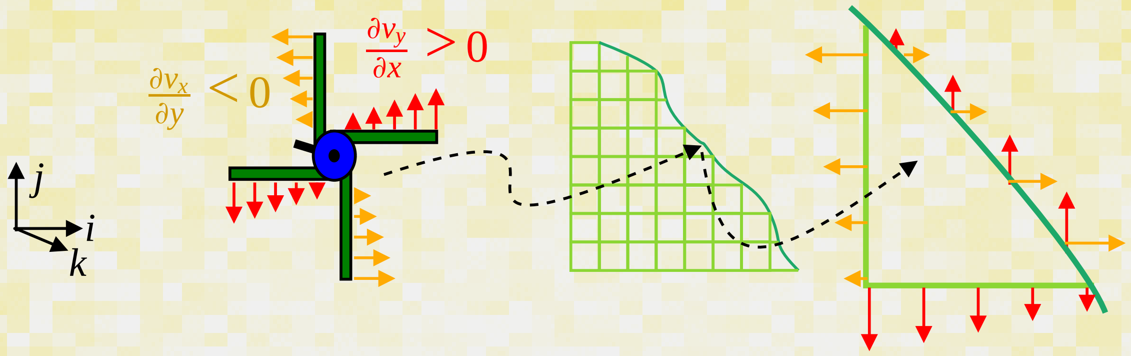


Figure 15-13: Illustration of how a vector valued function in a planar domain "spills out" of domain by evaluating the curl everywhere in the domain. Within the domain, the rotational flow ($\nabla \times \vec{v}$) from one cell moves into its neighbors; however, at the edges the local rotation is a net loss or gain. The local net loss or gain is $\vec{v} \cdot (dx, dy)$.

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

$$E(x, y, z) = \iint_R \frac{d\xi d\eta}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + z^2}} \tag{15-2}$$

into a path integral using Green's theorem in the x–y plane:

$$\int\int_R \left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right) dxdy = \int_{\partial R} (F_1 dx + F_2 dy) \tag{15-3}$$

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

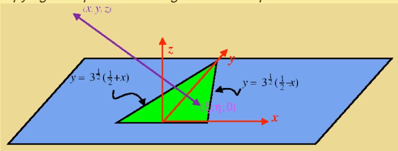
$$\int \frac{d\eta}{\sqrt{(x-\xi)^2 + (y-\eta)^2 + z^2}} \tag{15-4}$$

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

Converting an area-integral over a variable domain into a path-integral over its boundary

We reproduce the example from Lecture 14 where the potential was calculated in the vicinity of a triangular patch, but with much improved accuracy and speed. The previous example's two dimensional numerical integration which requires $\mathcal{O}(N^2)$ calculations into a path 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 $(dx, dy) = (x'dt, y'dt)$.

Suppose there is a uniformly charged surface (σ≡charge/area=1) occupying an equilaterial triangle in the z=0 plane:




```
F1[x_, y_, z_] =
 -Integrate[1/Sqrt[(x - ξ)^2 + (y - η)^2 + z^2], η]
```
1

The third (horizontal) boundary of the triangle patch looks like the easiest, let's see if an integral can be found over that patch:

```
Bottomside =
 F1[x, y, z] /. {ξ → t - 1/2, η → 0} // Simplify
```
2

```
NEside = F1[x, y, z] /.
  {ξ → (1 - t)/2, η → (Sqrt[3] t)/2} // Simplify
```
3

```
NWside = F1[x, y, z] /.
  {ξ → -t/2, η → (Sqrt[3] (1 - t))/2} // Simplify
```
4

```
integrand =
 Simplify[-(NEside + NWside)/2 + Bottomside]
```
5

1: We use Green's theorem in the plane to turn our original integral

$$\iint_{\substack{\text{triangle}\\\text{region}}} \left(\frac{\partial F_2}{\partial \eta} - \frac{\partial F_1}{\partial \xi}\right) d\xi d\eta = \phi(x, y, z)$$
$$= \iint \frac{d\eta d\xi}{r(x-\xi, y-\eta, z)} = \oint_{\substack{\text{triangle}\\\text{perimeter}}} \vec{F} \cdot d\vec{s}$$

A closed form for F_1 (as indicated in Equation 15-4) is obtained 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.

3–4: The remaining two legs of the triangle can be written similarly as: $((1-t)/2, \sqrt{3}t/2)$ and $(-t/2, \sqrt{3}(1-t)/2)$.

5: This is the integrand for the entire triangle to be integrated over $0 < t < 1$. Note, as t goes from 0 to 1, each leg of the triangle is traversed; this integrand sums all three contributions.

3.016 Home
Full Screen
Close
Quit

Faster and More Accurate Numerical Integration by Using Green's Theorem.

Continuing the example above, we are now able to find the potential over a triangular patch with uniform charge density, with a one-dimensional numerical integration, instead of the two-dimensional numerical integration in the last lecture.

Doing the same integral as in the previous lecture numerically, but this time over the boundary of the triangle instead of the triangle area.

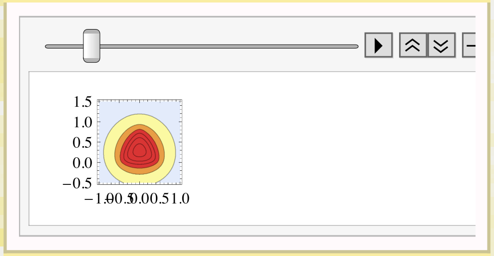
```
Pot[X_, Y_, Z_] := NIntegrate[Evaluate[
   integrand /. {x → X, y → Y, z → Z}], {t, 0, 1}]
```
1

We will create contourplots (level sets of constant potential) at as a function of different heights. We check the timing of the computation to compare to method in the last lecture.

```
ncplot[h_] :=
 ncplot[h] = ContourPlot[Pot[a, b, h],
   {a, -1, 1}, {b, -.5, 1.5}, Contours →
    Table[v, {v, .25, 2, .25}], ColorFunction →
    ColorData["TemperatureMap"],
   ColorFunctionScaling -> False,
   PlotPoints → 11 , ImageSize → {96, 72}]
Timing[ncplot[.05]]
```
2

```
Row[{TextCell[
   "Computing ContourPlots a different
     h: Progress:   ", "Text"],
  ProgressIndicator[Dynamic[h], {0, .5}]}]
ncplots = Table[ncplot[h],
   {h, .025, .5, .025}];
```
3

```
ListAnimate[ncplots]
```
4



1: There is no free lunch—the closed form of the integral is either unknown or takes too long to compute. 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.

2: A `ContourPlot` showing the level sets of the scalar potential field at a particular height h is obtained by a single call to the function *ncplot* . `Timing` shows that a speed-up factor of two is obtained for a single plot.

3: Here, we calculate a sequence of contour plots and store them for subsequent animation. Because this calculation takes a while to finish, we add a `ProgressIndicator`.

4: This is an animation for the potential in a plane as we increase the height of the plane above the triangular patch.

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.

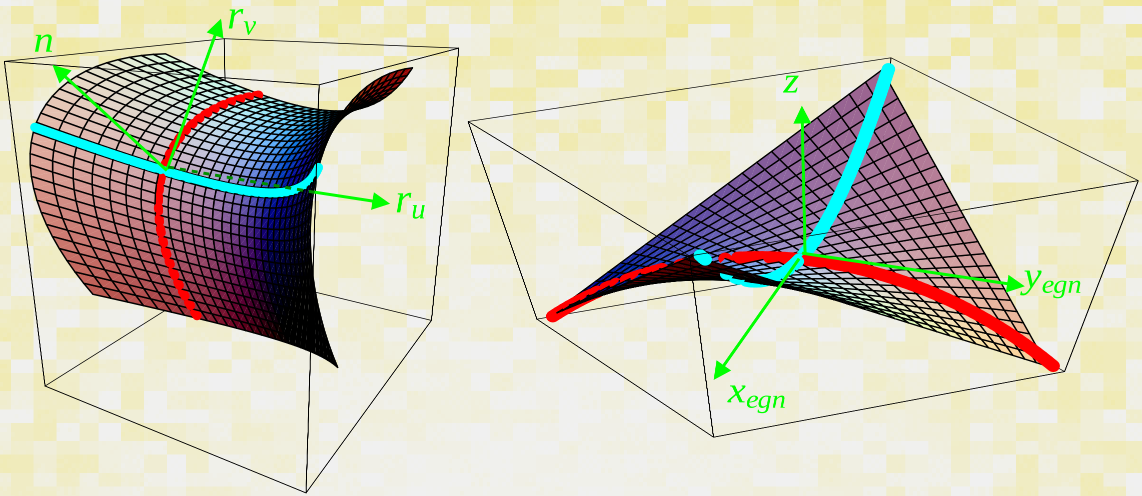


Figure 15-14: **Parabolic approximation to a surface and local eigenframe.** The surface on the left is a second-order approximation of a surface at the point where the coordinate axes are drawn. The surface has a local normal at that point which is related to the cross product of the two tangents of 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*.

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 ($\vec{x} = (x,y,z)$, $\vec{\xi} = (\xi,\eta,\zeta)$) $\nabla F \cdot (\vec{\xi} - \vec{x}) = 0$ or $\frac{\partial F}{\partial x}(\xi - x) + \frac{\partial F}{\partial y}(\eta - y) + \frac{\partial F}{\partial z}(\zeta - z) = 0$
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 $\nabla \cdot \left(\frac{\nabla F}{\lVert \nabla F \rVert} \right)$ 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

$$\frac{\left(\frac{d\vec{x}}{du} \cdot \frac{d\vec{x}}{du}\right)\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d^2\vec{x}}{dv^2}\right) - 2\left(\frac{d\vec{x}}{du} \cdot \frac{d\vec{x}}{dv}\right)\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d^2\vec{x}}{du dv}\right) + \left(\frac{d\vec{x}}{dv} \cdot \frac{d\vec{x}}{dv}\right)\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d^2\vec{x}}{du^2}\right)}{\left(\frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv} \cdot \frac{d\vec{x}}{du} \times \frac{d\vec{x}}{dv}\right)^{3/2}}$$

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

$$\frac{(1 + \frac{\partial f}{\partial x}^2)\frac{\partial^2 f}{\partial y^2} - 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\frac{\partial^2 f}{\partial x \partial y} + (1 + \frac{\partial f}{\partial y}^2)\frac{\partial^2 f}{\partial x^2}}{\sqrt{1 + \frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}}$$

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

Representations of Surfaces: Graphs $z = f(x, y)$ (part 1)

Visualization examples of surfaces represented by the graph $z = f(x, y)$; Examples of the use of `MeshFunctions` and `ColorFunction` to visualize various surface properties are given.

```
GraphFunction[x_, y_] :=
 (x - y) (x + y) / (1 + (x + y)^2)
assump = {x ∈ Reals, y ∈ Reals}
```
1

```
plotdefault = Plot3D[GraphFunction[x, y],
  {x, -3, 3}, {y, -3, 3}, PlotLabel → "Default"]
```
2

```
plotlevels =
 Plot3D[GraphFunction[x, y], {x, -3, 3},
  {y, -3, 3}, MeshFunctions → (#3 &),
  ColorFunction → "Rainbow",
  PlotLabel → "Constant Heights"]
```
3

```
angle[x_] := ((Pi / 2 + ArcTan[x]) / Pi)
angle[x_, y_] := ((Pi / 2 + ArcTan[x, y]) / Pi)
```
4

```
plotcircles = Plot3D[
  GraphFunction[x, y], {x, -3, 3}, {y, -3, 3},
  MeshFunctions → (Sqrt[#1^2 + #2^2] &),
  ColorFunction -> (Hue[angle[#1, #2] * 0.5] &),
  ColorFunctionScaling → False,
  PlotLabel → "Cylindrical Coordinates"]
```
5

```
CurvatureOfGraph[f_, x_, y_] :=
 FullSimplify[Module[
   {dfdx = D[f[x, y], x], dfdy = D[f[x, y], y],
    d2fdx2 = D[f[x, y], {x, 2}],
    d2fdy2 = D[f[x, y], {y, 2}],
    d2fdxdy = D[f[x, y], x, y]} ,
   Return[((1 + dfdx^2) d2fdx2 - 2 dfdx
        dfdy d2fdxdy + (1 + dfdy^2) d2fdy2) /
     Sqrt[1 + dfdx^2 + dfdy^2]]],
  Assumptions → assump]
```
6

```
CurvFunc = Function[{x, y}, Evaluate[
   CurvatureOfGraph[GraphFunction, x, y]]]
```
7

1: We will use *GraphFunction* as an example to show different ways to visualize a graph over an area.

2: `Plot3D` is used to plot *GraphFunction* with default settings.

3: Here is an example of using `MeshFunctions` to draw lines at constant altitude (i.e, constant values of $f(x, y)$)

4: This function, *angle* , which maps angles to the range $(0, 1)$ will be useful for visualization examples below (e.g., **5** and the following sections **2**).

5: This will help visualize a cylindrical- in addition to the Cartesian-coordinate system. The `MeshFunctions` option is used to plot concentric circles; `ColorFunction` illustrates the angular coordinate, θ, with `Hue`.

6: Our goal is to visualize curvature on top of the graph. This is a somewhat advanced example. Here we construct a function (*CurvatureOfGraph*) that computes the curvature $H(x, y)$ of an $f(x, y)$, and uses `FullSimplify` with assumptions that the coordinate are real numbers.

7: Here we use `Function` to create a symbol representing a function of two variables for the particular instance of the curvature of $f =$ *GraphFunction* . `Evaluate` is used in the definition to ensure that the curvature computation is performed only once.

3.016 Home

Full Screen

Close

Quit

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

Representations of Surfaces: Graphs $z = f(x, y)$ (part 2)

We continue the example by visualizing the curvature and the inclination of the graph.

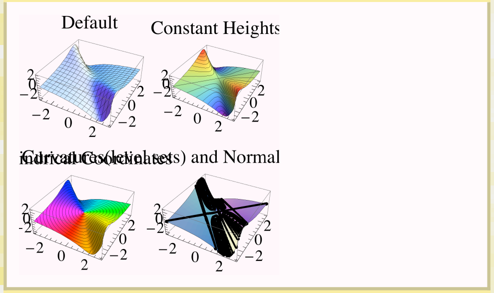
```
dfdx = Function[{x, y}, Evaluate[
   FullSimplify[D[GraphFunction[x, y], x],
    Assumptions → assump]]]
dfdy = Function[{x, y}, Evaluate[
   FullSimplify[D[GraphFunction[x, y], y],
    Assumptions → assump]]]
```
1

This is the surface with lines of constant curvature superimposed, and with colors associated with the local normal.

```
plotcurvature = Plot3D[
  GraphFunction[x, y], {x, -3, 3}, {y, 3, -3},
  MeshFunctions → (CurvFunc[#1, #2] &),
  MeshStyle → Thick, PlotLabel →
   "Curvatures(level sets) and Normals(color
     variation)", ColorFunction →
   (Glow[RGBColor[angle[dfdx[#1, #2]],
       angle[dfdy[#1, #2]], 0.75]] &),
  ColorFunctionScaling → False,
  Lighting → None]
```
2

Visualizing all the examples together.

```
GraphicsGrid[{{plotdefault, plotlevels},
  {plotcircles, plotcurvature}},
 ImageSize → 2 {72, 72}]
```
3



1: Two more symbols for functions of two arguments are created. Each represents a the slope of the tangent plane in the directions of the coordinate axes.

2: `Plot3D` is used to illustrate the local tangent-plane with `ColorFunction` which points to a red-scale for the surface slope in the x-direction and a blue-scale for the y-slope. We use `Glow` with `Lighting` set to none.

3: Finally, we use `GraphicsGrid` to illustrate the four graphic-examples together.

3.016 Home

Full Screen

Close

Quit

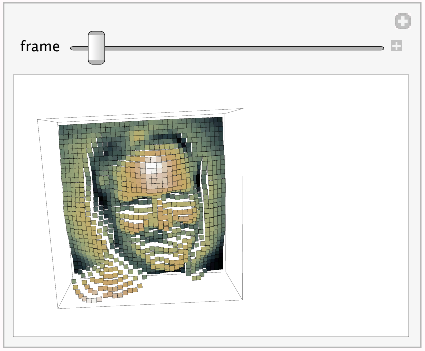
notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

A Frivolous Example for Graphs $z = f(x, y)$: Floating Pixels from Images in 3D

We demonstrate how to read a grey-scale image into MATHEMATICA® , and then use the pixel brightness values to displace the images according to $z = \text{brightness}(x, y)$.

```
MinMax[alist_List] :=
 Module[{flatlist = Flatten[alist]},
  Return[{Min[flatlist], Max[flatlist]}]]
mug = Import[
   "http://pruffle.mit.edu/~ccarter/ch_face
     _frames/Carter_2000_verysmall.png"];
ProgressIndicator[Dynamic[i], {1, 64}]
vp[i_] := {.1 Sin[(i - 1) Pi / 31],
   Sin[(i - 1) 2 Pi / 31], 2 Cos[2 (i - 1) Pi / 63]};
minmax = MinMax[mug[[1, 1]]];
Table[mugshot[i] =
    ListPlot3D[mug[[1, 1]], MeshStyle -> None,
     Mesh → None, InterpolationOrder → 0,
     ColorFunction → "GreenBrownTerrain",
     Axes → False, ViewPoint → vp[i],
     PlotRange → minmax,
     ImageSize → Full,
     SphericalRegion → True];, {i, 1, 64}];
Manipulate[mugshot[frame], {frame, 1, 64, 1}]
```

1



1: We first construct a function that will pick out the largest and smallest numbers in a list, and this will allow us to set `PlotRange` between the darkest and brightest pixels. (This function should probably check to ensure that the list contains only numeric entries, so that `Max` and `Min` return sensible results.) We will create a 3D rendering of pixels and "fly" through it. The function *vp* will provide the "orbit" for our flight through the pixels.

`Table` is used to create `Graphics3D` objects from different viewpoints for subsequent animation. Each graphics object is created with `ListPlot3D` with an array of pixel values for the first argument (`mug[[1,1]]`). Using `InterpolationOrder` set to zero implies that the plot's discrete values will not be continuously connected (i.e., the pixels are not "warped" to ensure continuity).

I used a modified version of this example to add an animation to my homepage

3.016 Home

Full Screen

Close

Quit

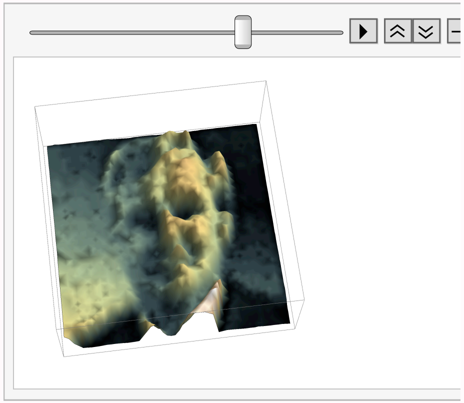
notebook (non-evaluated) pdf (evaluated, color) pdf (evaluated, b&w) html (evaluated)

A Frivolous Example for Graphs $z = f(x, y)$: Creating and Animating Surfaces from Image Sequences

We read in a sequence of images and use their pixel values to create an interpolation function for a surface $z = \text{brightness}(x, y)$. `Plot3D` calls the interpolation function produces a 3D animation from a 2D one.

```
Table[chface[read] = Import[
    "http://pruffle.mit.edu/~ccarter/ch_face
       _frames/ch_face." <>
     ToString[100 + read - 1] <> ".png"];
  facedata[read] = ListInterpolation[
    chface[read][[1, 1]], {{0, 1}, {0, 1}}];
  If[read == 1, minmax =
     MinMax[chface[read][[1, 1]]];, minmax =
    MinMax[{minmax, chface[read][[1, 1]]}]];,
  {read, 1, 28, 1}];
pface[i_] := Plot3D[facedata[i][x, y],
  {y, 0, 1}, {x, 0, 1}, PlotRange → minmax,
  ColorFunction → "GreenBrownTerrain",
  Mesh → False, Axes → False,
  ViewPoint → {-0.25, -2, 5}, ImageSize → All]
ListAnimate[Table[pface[gcomp],
  {gcomp, 1, 28, 1}], DefaultDuration → 10]
```

1



1: `Table` is used to iteratively read images that were created from a typical web-animation. (I am working on a way to do this directly from a single image file with multiple frames (with color), but haven't finished yet. `ListInterpolation` is used to create a continuous function of x and y in the domain $0 < |x|\&|y| < 1$. The height of the function corresponds to the brightness of the pixel. The function *pface* `[i]` produces a `Graphics3D` object for each frame in the animation. `ListAnimate` produces the animation from the image-functions.

3.016 Home

Full Screen

Close

Quit

Representations of Surfaces: Parametric Surfaces $\vec{x}(u, v)$

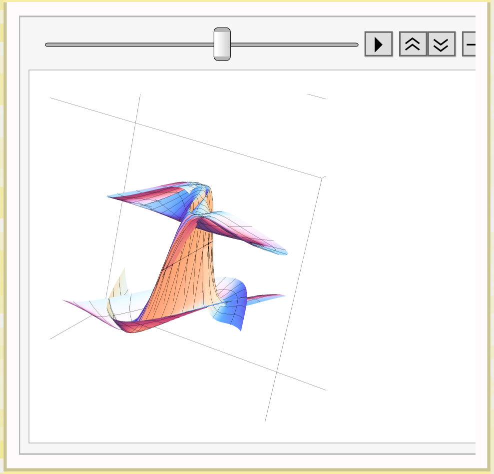
Visualization techniques for surfaces of the form $(x(u, v), y(u, v), z(u, v))$ are presented.

```
SurfaceParametric[u_, v_] := {Cos[u] v,
  u Cos[u + v], Cos[u] / (.1 + Cos[u] ^2)}
```
1

```
ParametricPlot3D[
 Evaluate[SurfaceParametric[u, v]],
 {u, -2, 2}, {v, -2, 2}]
```
2

Using Manipulate, we can vary the boundary domain, and provide a more intuitive way to understand this complicated surface.

```
evolution = Table[ParametricPlot3D[
    Evaluate[SurfaceParametric[u, v]],
    {u, -ep, ep}, {v, -ep, ep}, PlotRange →
     {{-4, 4}, {-4, 4}, {-4, 4}}, PlotPoints →
     {1 + Round[ep / .125], 1 + Round[ep / .125]},
    ImageSize → Full], {ep, .125, 4.25, .125}];
ListAnimate[evolution, ImageSize → Full]
```
3



1–2: 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.

3: The nature of parametric surfaces are typically much more complicated than for graphs. Because the surface often folds over and through itself, it is difficult to comprehend its shape. For this case, it is useful to visualize the *evolution* of the surface as the domain of (u, v) increases. Here we use `Table` to iteratively increase the size of the domain, and then use `ListAnimate` to visualize its evolution.

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

Representations of Surfaces: Level Sets constant $= f(x, y, z)$

Visualization examples of surfaces represented their level sets constant $= F(x, y, z)$ are presented. This type of surface representation is particularly convenient when surfaces are disconnected, or merge during an evolution. Level sets are used extensively in *phase field models of microstructural evolution*.

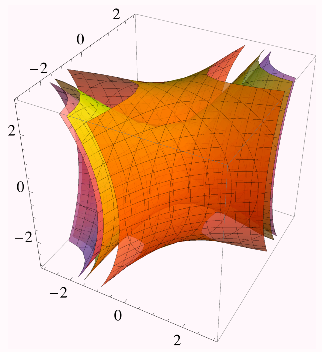
```
ConstFunction = x^2 - 4 x y + y^2 + z^2
```
1

```
ContourPlot3D[ConstFunction, {x, -1, 1},
 {y, -1, 1}, {z, -1, 1}, Contours → {2.5}]
```
2

The following statements produce contour plots of the same function, using two different methods for colorizing the surfaces...

```
cpa = ContourPlot3D[ConstFunction, {x, -3, 3},
  {y, -3, 3}, {z, -3, 3}, Contours → {0, 2, 8}]
```
3

```
cpb = ContourPlot3D[ConstFunction,
  {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
  Contours → {0, 2, 8}, ContourStyle → {
    Directive[Pink, Opacity[0.8]],
    Directive[Yellow, Opacity[0.8]],
    Directive[Orange, Opacity[0.8]]}]
```
4




```
Manipulate[
 ContourPlot3D[ConstFunction, {x, -3, 3},
  {y, -3, 3}, {z, -3, 3}, Contours -> {i},
  ImageSize → Full], {i, -2, 10, .25}]
```
5

1: *ConstFunction* will be used for the following visualization examples.

1–2: A contour in two-dimensions is a curve; we have seen examples of such curves with `ContourPlot`. A contour in three-dimensions is a surface and we will use `ContourPlot3D` to visualize the level set formulation of a surface constant $= F(x, y, z)$ given by *ConstFunction* . Here, we explicitly specify those x, y, and z for which $x^2 - 4xy + y^2 + z^2 = 2.5$.

3: Here is an example of specifying three different level sets by passing several `Contours` to `ContourPlot3D`. It is difficult to distinguish which surface belongs to a particular level set.

4: The surfaces can be distinguished from one another with by giving each a different graphics `Directive` its own color. Setting `Opacity` to a value less than one helps eliminate the 'hidden surface' problem.

5: The evolution of level sets can be visualized with `Manipulate` by varying the value that is passed to `Contours`. It is apparent why this surface representation is useful when surfaces undergo topological changes. It may be helpful to consider these changes as a higher dimensional effect: consider $t = f(x, y, z)$ as a graph 'over' 3D region, or a four-dimensional surface. As a lower dimensional example (i.e., $t = f(x, y)$), consider the curves that develop as a torus (*ummmm doughnut*) is slice sequentially from one side. Initially the perimeter is an single closed elongated loop, which eventually begins to pinch in the middle and then break into isolated curves.

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 $\vec{x}(u,v)$.

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

Example of an Integral over a Parametric Surface

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.

```
sphere[u_ , v_ ] :=
 R {Cos[v] Cos[u] , Cos[v] Sin[u] , Sin[v]}
```
1

```
Ru[u_ , v_] = D[sphere[u, v], u] // Simplify
Rv[u_ , v_] = D[sphere[u, v], v] // Simplify
```
2

```
Needs["VectorAnalysis`"]
NormalVector[u_ , v_ ] =
 CrossProduct[Ru[u, v], Rv[u, v]] // Simplify
NormalMag = FullSimplify[
  Norm[NormalVector[u, v]], Assumptions →
   {R ≥ 0, 0 ≤ u ≤ 2 π, -π / 2 < v < π / 2}]
UnitNormal[u_, v_] =
 NormalVector[u, v] / NormalMag
```
3

```
SurfaceTension[nvec_] :=
 1 + gamma111 * nvec[[1]]^2 nvec[[2]]^2 nvec[[3]]^2
```
4

```
SphericalPlot3D[
 SurfaceTension[UnitNormal[u, v]] /.
  gamma111 → 12,
 {u, 0, 2 Pi}, {v, -Pi / 2, Pi / 2}]
```
5

```
SphereEnergy = Integrate[Integrate[
   SurfaceTension[UnitNormal[u, v]] Cos[v],
   {u, 0, 2 π}], {v, -π / 2, π / 2}]
```
6

```
CubeSide = (4 π / 3) ^ (1 / 3)
```
7

```
CubeEnergy =
 6 (CubeSide^2 SurfaceTension[{1, 0, 0}])
```
8

```
EqualEnergies =
 Solve[CubeEnergy == SphereEnergy,
   gamma111] // Flatten
```
9

```
N[gamma111 /. EqualEnergies]
```
10

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 `VectorAnalysis` package to calculate a vector that is normal to the surface, $\vec{r_u} \times \vec{r_v}$, for subsequent use in the surface integral. Using `Norm` to find the magnitude of the local normal, we can produce a function to return the unit normal vector $\hat{n}$, *UnitNormal* , as a function of the surface parameters.

4: This is just an example of a $\gamma(\hat{n})$ that depends on direction that will be used for purposes of illustration.

5: Using `SphericalPlot3D`, the form of *SurfaceTension* for the particular choice of $\gamma_{111} = 12$ is visualized.

6: Using the result from $|\vec{r_u} \times \vec{r_v}|$, the total surface energy of a spherical body of radius $R = 1$ is computed by integrating $\gamma\hat{n}$ over the entire surface.

7–8: This would be the energy of a cubical body with the same volume as the sphere with unit radius. The cube is oriented so that its faces are normal to $\langle 100 \rangle$.

9–10: This calculation is not very meaningful, but it is the value of the surface anisotropy factor γ_{111} such that the cube and sphere have the same total surface energy. The total-surface-energy minimizing shape for a fixed volume is calculated using the *Wulff theorem*.

3.016 Home

Full Screen

Close

Quit

Index

3.016 Home
Full Screen
Close
Quit