vecs.div Divergence, surface integrals, and flux

Flux and surface integrals

Consider a surface S. Let $\mathbf{r}(u, v) = [x(u, v), y(u, v), z(u, v)]$ be a parametric position vector on a Euclidean vector space $\mathbb{R}^3$. Furthermore, let $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^3$ be a vector-valued function of $\mathbf{r}$ and let $\mathbf{n}$ be a unit-normal vector on a surface S. The surface integral

surface integral

$$\iint_S \mathbf{F} \cdot \mathbf{n} \, dS \tag{1}$$

which integrates the normal of $\mathbf{F}$ over the surface. We call this quantity the flux of $\mathbf{F}$ out of the surface S. This terminology comes from fluid flow, for which the flux is the mass flow rate out of S. In general, the flux is a measure of a quantity (or field) passing through a surface. For more on computing surface integrals, see Schey[6] and Kreyszig.[7]

flux

6. Schey, Div, Grad, Curl, and All that: An Informal Text on Vector Calculus, pp. 21-30.

7. Kreyszig, Advanced Engineering Mathematics, § 10.6.

Continuity

Consider the flux out of a surface S that encloses a volume ΔV, divided by that volume:

$$\frac{1}{\Delta V} \iint_S \mathbf{F} \cdot \mathbf{n} \, dS. \tag{2}$$

This gives a measure of flux per unit volume for a volume of space. Consider its physical meaning when we interpret this as fluid flow: all fluid that enters the volume is negative flux and all that leaves is positive. If physical conditions are such that we expect no fluid to enter or exit the volume via what is called a source or a sink, and if we assume the density of the fluid is uniform (this is called an incompressible fluid), then all the fluid that enters the volume must exit and we get

source
sink

incompressible

$$\frac{1}{\Delta V} \iint_S \mathbf{F} \cdot \mathbf{n} \, dS = 0. \tag{3}$$

This is called a continuity equation, although typically this name is given to equations of the form in the next section. This equation is one of the governing equations in continuum mechanics.

continuity equation

Divergence

Let's take the flux-per-volume as the volume $\Delta V \to 0$ we obtain the following.

Equation 4 divergence: integral form

$$\lim_{\Delta V \to 0} \frac{1}{\Delta V} \iint_S \mathbf{F} \cdot \mathbf{n}\, dS.$$

This is called the divergence of $\mathbf{F}$ and is defined at each point in $\mathbb{R}^3$ by taking the volume to zero about it. It is given the shorthand $\operatorname{div} \mathbf{F}$.

divergence

What interpretation can we give this quantity? It is a measure of the vector field's flux outward through a surface containing an infinitesimal volume. When we consider a fluid, a positive divergence is a local decrease in density and a negative divergence is a density increase. If the fluid is incompressible and has no sources or sinks, we can write the continuity equation

$$\operatorname{div} \mathbf{F} = 0. \tag{5}$$

In the Cartesian basis, it can be shown that the divergence is easily computed from the field

$$\mathbf{F} = F_x \hat{\boldsymbol{\imath}} + F_y \hat{\boldsymbol{\jmath}} + F_z \hat{\mathbf{k}} \tag{6}$$

as follows.

Equation 7 divergence: differential form

$$\operatorname{div} \mathbf{F} = \partial_x F_x + \partial_y F_y + \partial_z F_z$$

Exploring divergence

Divergence is perhaps best explored by considering it for a vector field in $\mathbb{R}^2$. Such a field $\mathbf{F} = F_x \hat{\imath} + F_y \hat{\jmath}$ can be represented as a "quiver" plot. If we overlay the quiver plot over a "color density" plot representing $\operatorname{div} \mathbf{F}$, we can increase our intuition about the divergence. The following was generated from a Jupyter notebook with the following filename and kernel.

```
notebook filename: div_surface_integrals_flux.ipynb
notebook kernel: python3
```

First, load some Python packages.

```
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import LogLocator
from matplotlib.colors import *
from sympy.utilities.lambdify import lambdify
from IPython.display import display, Markdown, Latex
```

Now we define some symbolic variables and functions.

```
var('x,y')
F_x = Function('F_x')(x,y)
F_y = Function('F_y')(x,y)
```

Rather than repeat code, let's write a single function `quiver_plotter_2D` to make several of these plots.

```
def quiver_plotter_2D(
  field={F_x:x*y,F_y:x*y},
  grid_width=3,
  grid_decimate_x=8,
  grid_decimate_y=8,
  norm=Normalize(),
  density_operation='div',
  print_density=True,
):
  # define symbolics
  var('x,y')
  F_x = Function('F_x')(x,y)
```

```
F_y = Function('F_y')(x,y)
field_sub = field

# compute density
if density_operation is 'div':
  den = F_x.diff(x) + F_y.diff(y)
elif density_operation is 'curl':
  # in the k direction
  den = F_y.diff(x) - F_x.diff(y)
else:
  error('div and curl are the only density operators')
den_simp = den.subs(
  field_sub
).doit().simplify()
if den_simp.is_constant():
  print(
    'Warning: density operator is constant (no density plot)'
  )
if print_density:
  print(f'The {density_operation} is:')
  display(den_simp)

# lambdify for numerics
F_x_sub = F_x.subs(field_sub)
F_y_sub = F_y.subs(field_sub)
F_x_fun = lambdify((x,y),F_x.subs(field_sub),'numpy')
F_y_fun = lambdify((x,y),F_y.subs(field_sub),'numpy')
if F_x_sub.is_constant:
  F_x_fun1 = F_x_fun # dummy
  F_x_fun = lambda x,y: F_x_fun1(x,y)*np.ones(x.shape)
if F_y_sub.is_constant:
  F_y_fun1 = F_y_fun # dummy
  F_y_fun = lambda x,y: F_y_fun1(x,y)*np.ones(x.shape)
if not den_simp.is_constant():
  den_fun = lambdify((x,y), den_simp,'numpy')

# create grid
w = grid_width
Y, X = np.mgrid[-w:w:100j, -w:w:100j]

# evaluate numerically
F_x_num = F_x_fun(X,Y)
F_y_num = F_y_fun(X,Y)
if not den_simp.is_constant():
  den_num = den_fun(X,Y)

# plot
p = plt.figure()
#   colormesh
if not den_simp.is_constant():
  cmap = plt.get_cmap('PiYG')
  im = plt.pcolormesh(X,Y,den_num,cmap=cmap,norm=norm)
  plt.colorbar()
# Abs  quiver
dx = grid_decimate_y
```

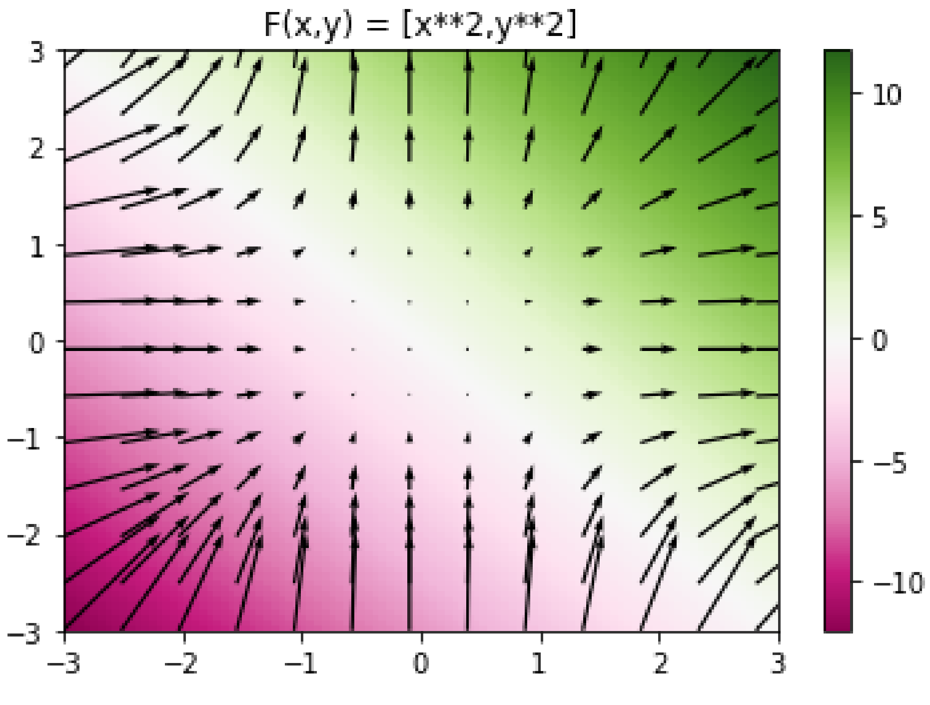

Figure div.1: png

```
  dy = grid_decimate_x
  plt.quiver(
    X[::dx,::dy],Y[::dx,::dy],
    F_x_num[::dx,::dy],F_y_num[::dx,::dy],
    units='xy', scale=10
  );
  plt.title(f'F(x,y) = [{F_x.subs(field_sub)},{F_y.subs(field_sub)}]')
  return p
```

Note that while we're at it, we included a hook for density plots of the curl of **F**, and we'll return to this in a later lecture.
Let's inspect several cases.

```
p = quiver_plotter_2D(
  field={F_x:x**2,F_y:y**2}
)
```

```
The div is:
```

$2x + 2y$

```
p = quiver_plotter_2D(
  field={F_x:x*y,F_y:x*y}
)
```

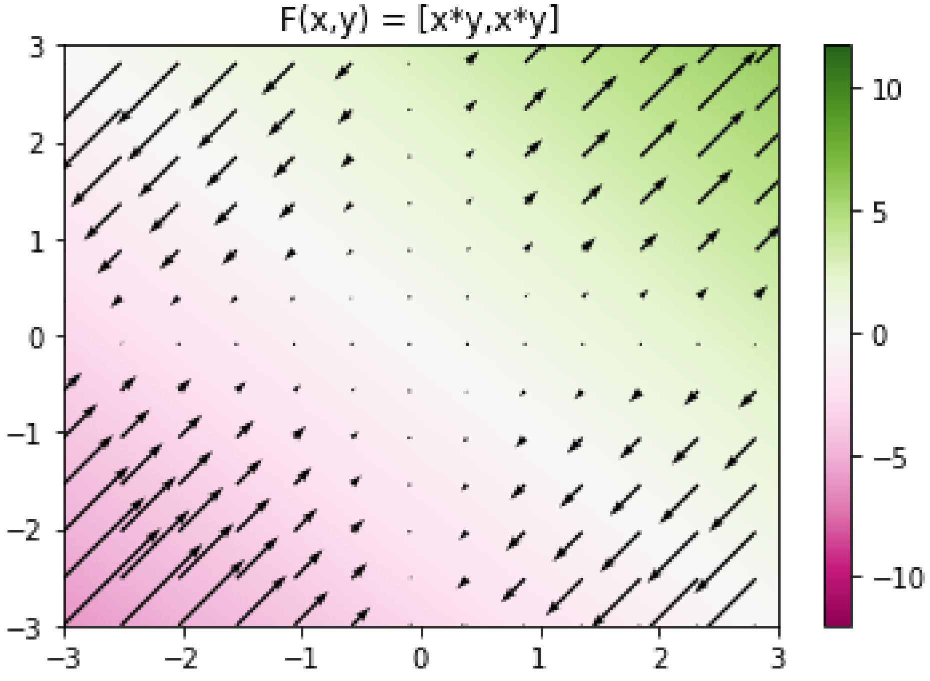

Figure div.2: png

```
The div is:
```

$x + y$

```
p = quiver_plotter_2D(
  field={F_x:x**2+y**2,F_y:x**2+y**2}
)
```

```
The div is:
```

$2x + 2y$

```
p = quiver_plotter_2D(
  field={F_x:x**2/sqrt(x**2+y**2),F_y:y**2/sqrt(x**2+y**2)},
  norm=SymLogNorm(linthresh=.3, linscale=.3)
)
```

```
The div is:
```

$$\frac{-x^3 - y^3 + 2\left(x + y\right)\left(x^2 + y^2\right)}{\left(x^2 + y^2\right)^{\frac{3}{2}}}$$

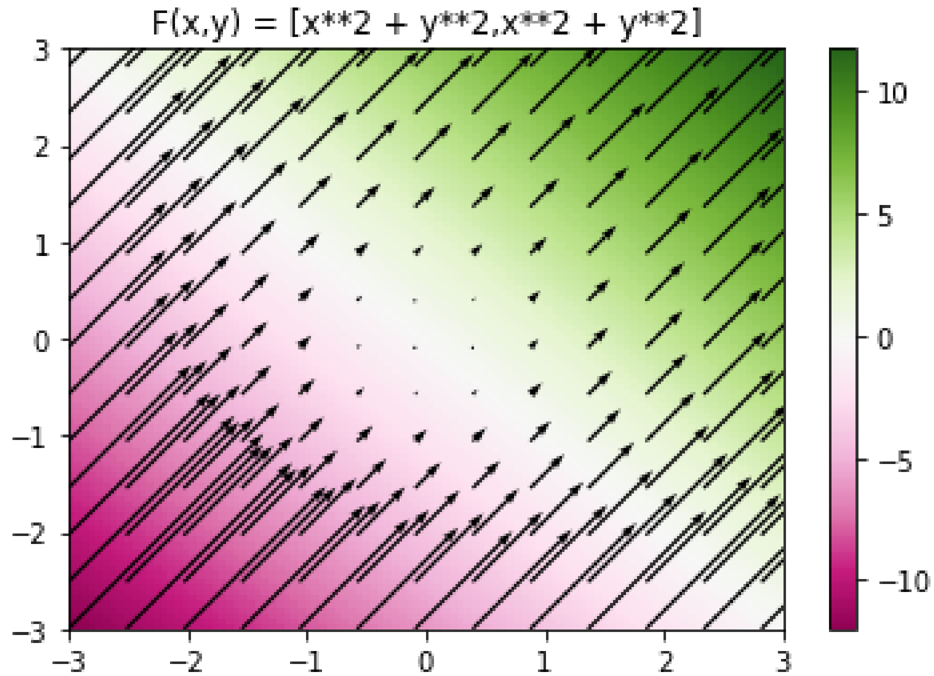

Figure div.3: png

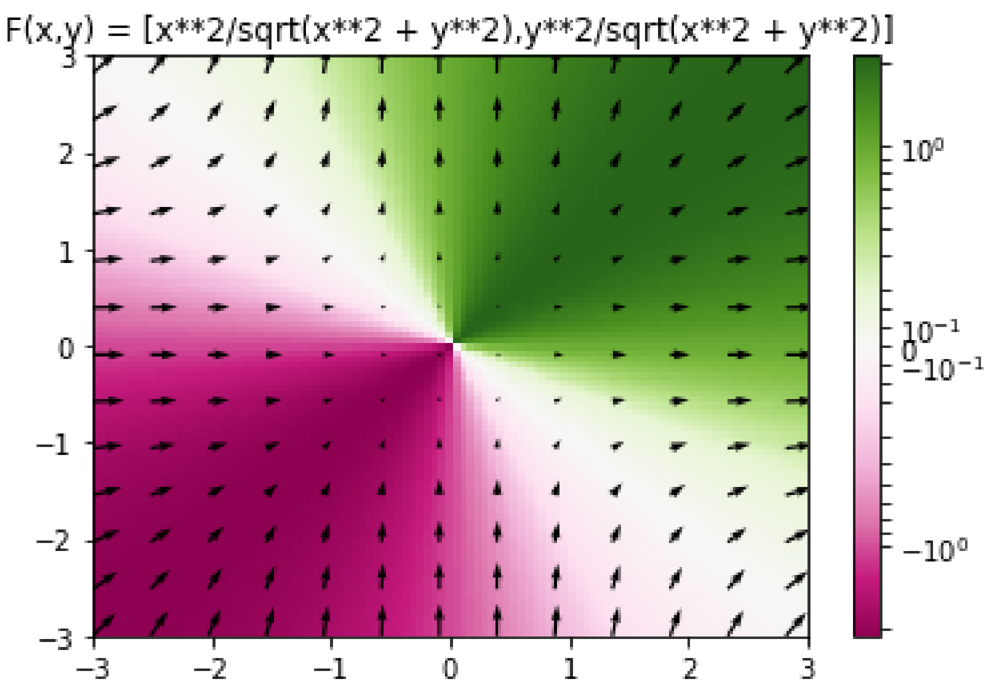

Figure div.4: png