(Custom CSS files are not reliable for controlling Jupyter font style. To establish the same appearance as the original notebook, depend on the browser to control the font, by setting the desired font faces in the browser settings. For example, Chrome 135 or Firefox 134 can do this. In this notebook series, Bookerly font is for markdown and Monaco is for code.)

Chapter 19 Further Numerical Methods for Solving 1st Order Differential Equations. For this section further numerical methods simply refers to a certain procedure for entering successive problems into Wolfram Alpha.

19.1 Use the modified Euler's method to solve $y' = y - x; \quad y(0) = 2$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y - x; y(0) = 2} from 0 to 1 using Dormand-Prince method |!

Of ten numerical methods available to W|F, the D-P method is the most accurate, with a global error on the present problem listed as -6.34×10^{-9}. For this problem Wolfram Alpha happens to have the exact solution available. The idea is that by entering the D-P method as the preferred option, the most accurate result will be obtained in case the exact solution is not available for some reason.

Exact solution: $y(x) = x + e^x + 1$

19.2 Use the modified Euler's method to solve $y = y^2 + 1; \quad y(0) = 0$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y^2 + 1; y(0) = 0} from 0 to 1 using Dormand-Prince method |!

The Dormand-Prince method would have produced a global error of 1.62×10^{-8} if the exact solution had not been available.

Exact solution: $y(x) = \tan(x)$

19.3 Find $y(1.6)$ for $y, = 2x; \quad y(1) = 1$ using the modified Euler's method with $h = 0.2$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = 2 * x; y(1) = 1} using Dormand-Prince method |!

The Dormand-Prince method, along with a few others, would have produced a global error of zero if the exact solution had not been available.

Exact solution: $y(x) = x^2$

$(1.6)^2 = 2.56$

19.4 Use the Runge-Kutta method to solve $y' = y - x; \quad y(0) = 2$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y - x; y(0) = 2} from 0 to 1 using Dormand-Prince method |!

The Dormand-Prince method would have produced a global error of -6.34×10^{-9} if the exact solution had not been available.

Exact solution: $y(x) = x + e^x + 1$

19.5 Use the Runge-Kutta method to solve $y' = y; \quad y(0) = 1$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y; y(0) = 1} from 0 to 1 using Dormand-Prince method |!

The Dormand-Prince method would have produced a global error of -5.48×10^{-9} if the exact solution had not been available.

Exact solution: $y(x) = e^x$

19.6 Use the Runge-Kutta method to solve $y' = y^2 + 1; \quad y(0) = 0$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y^2 + 1; y(0) = 0} from 0 to 1 using Dormand-Prince method |!

The Dormand-Prince method would have produced a global error of 1.62×10^{-8} if the exact solution had not been available.

Exact solution: $y(x) = \tan(x)$

19.7 Use the Adams-Bashforth-Moulton method to solve $y' = y - x\,; \quad y(0) = 2$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y - x; y(0) = 2} from 0 to 1 using Dormand-Prince method |!

The Dormand-Prince method would have produced a global error of -6.34×10^{-9} if the exact solution had not been available.

Exact solution: $y(x) = x + e^{x} + 1$

19.8 Use the Adams-Bashforth-Moulton method to solve $y' = y^2 + 1\,; \quad y(0) = 0$ on the interval [0, 1] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

!| solve {y' = y^2 + 1; y(0) = 0} from 0 to 1 using Dormand-Prince method |!

The Dormand-Prince method would have produced a global error of 1.62×10^{-8} if the exact solution had not been available.

Exact solution: $y(x) = \tan\,(x)$

19.9 Use the Adams-Bashforth-Moulton method to solve $y' = 2xy/(x^2 - y^2)\,; \quad y(1) = 3$ on the interval [1, 2] with $h = 0.1$.

This problem can be fed to Wolfram Alpha:

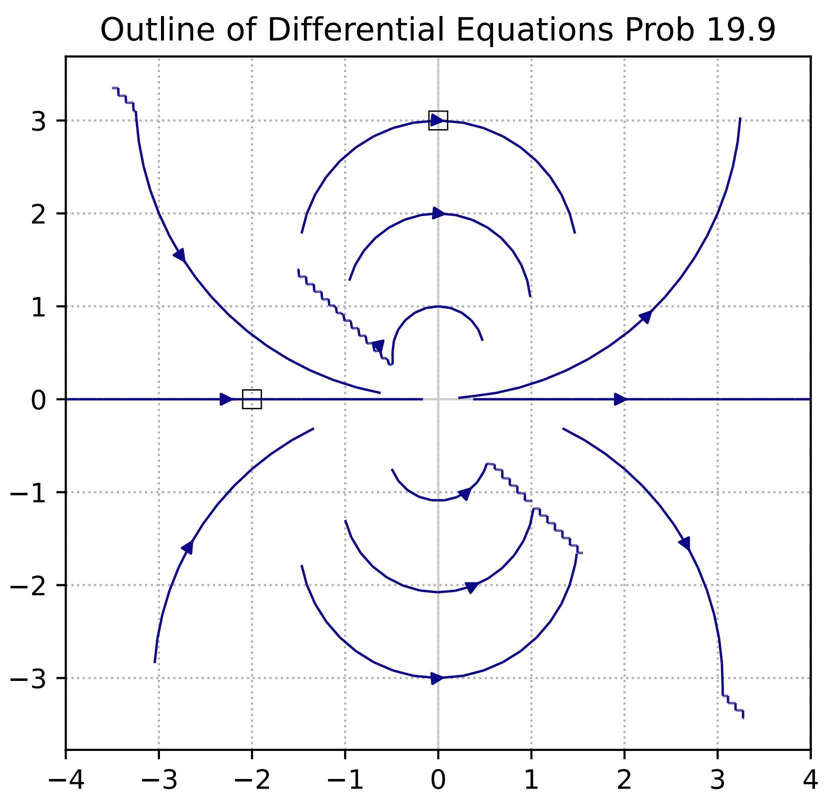
!| solve {y' = (2 * x * y)/(x^2 - y^2); y(1) = 3} from 1 to 2 using Dormand-Prince method |!

The potential error ranges for various numerical methods are not available for this problem.

Exact solution: $y(x) = \frac{1}{3}\left(\sqrt{25 - 9x^2} + 5\right)$

The text shows a direction field plot for this function, so to match that, a plot is shown below. Two points about the plot: (a) runtime warnings have been suppressed, (b) strange squiggly lines herald something (?).

In [77]:
```
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_formats = ['svg']

import warnings
with warnings.catch_warnings():
    warnings.simplefilter('ignore')

# Creating dataset
w = 4
Y, X = np.mgrid[-w:w:100j, -w:w:100j]
U = np.ones_like(X) #dxdt = 1
V = 2*X*Y/(X**2 - Y**2)
speed = np.sqrt(U**2 + V**2)

seek_points = np.array( [[-2,2.5,-3,3, 2,  -2,  -1,  -.5, 0, 0, 0,0],
                         [0,  0,   2,2,-.75,-.75,-1.3,-.75,-3, 1, 2,3]])


fig, ax = plt.subplots()
ax.grid(True, which='both', linestyle='dotted')
ax.axhline(y=0, color='0.8', linewidth=0.8)
ax.axvline(x=0, color='0.8', linewidth=0.8)

ratio = 1.0
#ratio is adjusted by eye to get squareness of x and y spacing
xleft, xright = ax.get_xlim()
ybottom, ytop = ax.get_ylim()
ax.set_aspect(abs((xright-xleft)/(ybottom-ytop))*ratio)


strm = ax.streamplot(X, Y, U, V, color = U,
                     linewidth = 0.9,
                     cmap ='plasma',
                     start_points = seek_points.T)

plt.title("Outline of Differential Equations Prob 19.9")
plt.rcParams['figure.figsize'] = [5, 5]

xpts = np.array([-2, 0])
ypts = np.array([0, 3])
plt.plot(xpts, ypts, markersize=7, color='k', marker='s', \
         mfc='none', linestyle = 'none', markeredgewidth=0.5)
plt.show()
```



In [ ]:

In [ ]: