Practice final exam problems with solutions

1

Estimate $y(5)$ for the initial-value problem

$$\frac{dy}{dt} = \sqrt{y} - t + 1, \;\; y(1) = 3$$

using Euler's method (RK1) with step size $h = 1$.

Solution: Using the iteration $y_{i+1} = y_i + h(\sqrt{y_i} - t_i + 1)$, we get $4.7321, 5.9074, 6.3379, \mathbf{5.8554}$

2

(a) Use Euler's method with step size $h = 1/4$ to estimate $y(1)$ given $\frac{dy}{dt} = -2y, y(0) = 3$.
(b) Find the absolute error of this estimate compared to the analytic solution $y(t) = 3e^{-2t}$.
(c) Write Python code to solve for and display $y(1)$ using `solve_ivp`.

Solution: (a) With $\frac{dy}{dt} = -2y$, the Euler method iteration is $y_{i+1} = y_i + hy'(t_i, y_i) = (1 - 2h)y_i$.
With $h = 1/4$, this gives the estimated values

t	y
0	3
1/4	1.5
1/2	0.75
3/4	0.375
1	**0.1875**.

(b) $E = |3e^{-2} - 0.1875| = \mathbf{0.21851}$.
(c)

```
from scipy.integrate import solve_ivp
f = lambda t, y: -2*y
sol = solve_ivp(f, [0, 1], [3])
sol.y[0, -1]
```

3

(a) Use Euler's method with a step size h of 1/3 to estimate $y(t = 1)$ if $y(t = 0) = 0, dy/dt = y + 2\sqrt{t}$.
(b) Estimate $y(t = 1)$ for the same problem with the RK4 method and $h = 1$.
(c) Write Python code that uses `scipy.integrate.solve_ivp` to estimate $y(t = 1)$ and displays the estimated value.

Solution: (a) The estimates of y after each step are 0, 0.38490, **1.05753**.
(b) $K_1 = 0, K_2 = 1.4142, K_3 = 2.1213, K_4 = 4.1213, y_1 = \mathbf{1.86540}$
(c)

```
from numpy import sqrt
from scipy.integrate import solve_ivp
f = lambda t, y:  y + 2 * sqrt(t)
ts = [0, 1]
y0 = [0]
sol = solve_ivp(f, ts, y0)
sol.y[0,-1]
```

(The result is 2.060)

4

Let $y(t)$ be the solution to $y' = 4te^{-y}$ satisfying $y(0) = 3$. Use Euler's Method with time step $h = 0.1$ to approximate $y(0.1), y(0.2), \ldots, y(0.5)$.

Solution: The estimated values of y at the first 5 timesteps are $3, 3.002, 3.006, 3.012, 3.020$

5

(a) Use Euler's method with a step size h of 1/3 to estimate $y(t = 1)$ if $y(t = 0) = 0, dy/dt = y + 4t^2$.
(b) Estimate $y(t = 1)$ for the same problem with the RK4 method and $h = 1$.

Solution: (a) The estimates of y after each step are 0, 4/27, **64/81**.
(b) $K_1 = 0, K_2 = 1, K_3 = 1.5, K_4 = 5.5, y_1 = \mathbf{1.75}$

6

Estimate $y(3)$ for the initial-value problem
$y' = \frac{\sqrt{y}}{2}, y(1) = 1$
using Heun's method (RK2) with step size $h = 1$.

Solution: The result is 1.5562 after 1 step ($K_1 = 0.5, K_2 = 0.61237$) and **2.2372** after 2 steps ($K_1 = 0.62374, K_2 = 0.73823$).

7

Estimate $y(3)$ for the initial-value problem $y'' = \frac{\sqrt{y}}{2}, y(1) = 1, y'(1) = 0$ using Heun's method (RK2) with step size $h = 1$.

Solution: The result for $[y\, y']$ is [1.25 0.5] after 1 step ($K_1 = [0\, 0.5], K_2 = [0.5\, 0.5]$) and [**2.02951** 1.11023] after 2 steps ($K_1 = [0.5\, 0.55902], K_2 = [1.05902\, 0.66144]$).

8

(a) Use 1 step of the RK4 method to solve for $y(x = 6)$ if $y(x = 0) = 2$ and $y' = x\sqrt{y}$.

(b) Write Python code for solving this problem and displaying the result.

Solution: (a) The stages are $K_1 = 0, K_2 = 25.456, K_3 = 69.079, K_4 = 303.51$, giving $y_1 = \mathbf{84.10}$

(b)
```
from numpy import sqrt
from scipy.integrate import solve_ivp
f = lambda x, y:  x * sqrt(y)
xs = [0, 6]
y0 = [2]
sol = solve_ivp(f, xs, y0)
sol.y[0,-1]
```
(The result is 108.49)

9

Estimate $y(t = 2)$ if $y(t = 0) = 0, dy/dt = -2y + 3\sqrt{t}$
with the RK4 method and $h = 1$.

Solution: First step $K_1, K_2, K_3, K_4, y_{i+1}$: 0, 2.1213, 0, 3, 1.2071.
Second step: 0.58579, 0.67423, 0.58579, 0.65685, **1.8342**.

10

(a) Use the implicit Euler method with step size $h = 1/6$ to estimate $y(1)$ given $\frac{dy}{dt} = -2y, y(0) = 3$.
(b) Find the absolute error of the estimated $y(1)$ compared to the analytic solution $y(t) = 3e^{-2t}$.
(c) Write Python code to solve for and display $y(1)$.

Solution: (a) To use the implicit Euler method, we need to solve (explic-

itly) for y_{i+1} given y_i. This is straightforward in this case:

$$y_{i+1} = y_i + h \cdot f(t_{i+1}, y_{i+1}) = y_i - 2hy_{i+1} \rightarrow y_{i+1} = \frac{y_i}{1+2h}.$$

So at each step the estimated y is multiplied by a constant factor of $\frac{1}{1+2h} = \frac{3}{4}$, giving after 6 steps $\mathbf{\frac{2187}{4096}}$ or 0.53394
(b) Since the true value is $3e^{-2} = 0.40601$, $E_a = |0.53394 - 0.40601| = \mathbf{0.128}$
(c)

```
from numpy import sqrt
from scipy.integrate import solve_ivp
f = lambda t, y:  -2*y
ts = [0, 1]; y0 = [3]
sol = solve_ivp(f, ts, y0)
sol.y[0,-1]
```

(The result is 0.406)

11

Estimate $y(3)$ for the initial-value problem
$y'' = \frac{\sqrt{y}}{2}, y(1) = 5, y'(1) = 0$
using Heun's method (RK2) with step size $h = 1$.

Solution: Writing as a first-order system $\mathbf{z} = \begin{pmatrix} y \\ y' \end{pmatrix}, \mathbf{z}' = \begin{pmatrix} z_2 \\ \frac{\sqrt{z_1}}{2} \end{pmatrix}, \mathbf{z}(1) = \begin{pmatrix} 5 \\ 0 \end{pmatrix}$,

the result is $\begin{pmatrix} 5.5590 \\ 1.1180 \end{pmatrix}$ after 1 step ($K_1 = \begin{pmatrix} 0 \\ 1.11803 \end{pmatrix}, K_2 = \begin{pmatrix} 1.11803 \\ 1.11803 \end{pmatrix}$) and $\begin{pmatrix} 7.2665 \\ 2.3535 \end{pmatrix}$ after 2 steps ($K_1 = \begin{pmatrix} 1.11803 \\ 1.1789 \end{pmatrix}, K_2 = \begin{pmatrix} 2.2969 \\ 1.2920 \end{pmatrix}$), so $y(3) \approx \mathbf{7.2665}$.

12

For the mass-spring system with driving force as given below, use 2 equal-length steps of Euler's method to estimate $x(t)$ at $t = 1$. The initial conditions at $t = 0$ are $x = 1, x' = 0$.

$$x'' + x = 3\cos(t)$$

Solution: After writing the differential equation in standard form ($x'' = 3\cos(t) - x$) and converting to a first-order system ($y_1 \equiv x, y_2 \equiv x'$) $y_1' = y_2, y_2' = 3\cos(t) - y_1$ with $y_1(0) = 1, y_2(0) = 0$, the Euler method gives $y_1 = 1, y_2 = 1$ after one step of length $h = 0.5$, and $y_1 = \mathbf{1.5}, y_2 = 1.8164$ after two steps.

13

(a) Write a Python function, similar to the RK2 one we did in lab, to implement Euler's method for solving a system of first-order differential equations with given initial values. The first line should be
`def rk1(f,tspan,y0,n):`
(b) Write a Python script that uses this `rk1` and a step size h =0.01 to display $y(2)$ if $y'' = \sin(1/y)$ and $y(1) = 10, y'(1) = 3$.

Solution: (a)

```
def rk1(f,tspan,y0,n):
            from numpy import linspace, empty
            a = tspan[0]
            b = tspan[1]
            h = (b-a)/n
            T = linspace(a, b, n+1)
            Y = empty([n+1, len(y0)])
            Y[0, :]  = y0
            for i in range(n):
                        K1 = h * f(T[i], Y[i, :])
                        Y[i+1, :]  = Y[i, :]  + K1
            return T, Y
```

(b)

```
from numpy import sin, array
dzdt = lambda t, z:  array([z[1], sin(1 / z[0])])
tspan = [1, 2]
y0 = [10, 3]
h = 0.01
n = round((tspan[1] - tspan[0]) / h)
T, Z = rk1(dzdt, tspan, y0, n)
Z[-1, 0]
```

(The result is 13.045)

14

Write a Python script that solves numerically the initial-value problem
$y'' + y' + y^2 = 1, y(0) = 2, y'(0) = -2$
and plot $y(t)$ for $0 \leq t \leq 10$.

Solution: Need to convert to a first-order ODE system ($z_1 = y, z_2 = y'$), after which an IVP solver like `solve_ivp` can be called:

```
from scipy.integrate import solve_ivp
from matplotlib.pyplot import plot
f = lambda t, z:  [z[1], -z[1] - z[0]**2 + 1]
```

```
zi = [2, -2]
ts = [0, 10]
sol = solve_ivp (f, ts, zi)
plot (sol.t, sol.y[0, :])
```
($y(x = 10) \approx 1.008$)

15

Write a Python script that (a) numerically estimates $\Theta(2)$ if $\Theta'' = -9\Theta$ and $\Theta(0) = 1, \Theta'(0) = 0$ and (b) finds the absolute error of this estimate relative to the analytic solution $\Theta(t) = \cos(3t)$.

Solution: With $z_1 = \Theta, z_2 = \Theta'$,

```
from scipy.integrate import solve_ivp
from math import cos
f = lambda t, z:  [z[1], -9*z[0]]
zi = [1, 0]
ts = [0, 2]
sol = solve_ivp (f, ts, zi)
err = abs (sol.y[0, -1] - cos(3*ts[-1]))
```

16

(a) Set up a system of linear equations (in matrix form) using finite difference with $n = 4$ to solve the equation $y'' + y' = x + 2$ subject to the boundary conditions $y(x = 0) = 0$ and $y(x = 3) = 1$.
(b) Solve this system using Gaussian elimination or LU decomposition, and sketch $y(x)$.

Solution: A finite-difference approximation to the ODE would be

$$(h^{-2} - (2h)^{-1})y_{i-1} - 2h^{-2}y_i + (h^{-2} + (2h)^{-1})y_{i+1} = x_i + 2.$$

Applying this here with $h = \frac{3}{4}$ for $i = 1, 2, 3$, $x_i = 0.75, 1.5, 2.25$, we get

$$\left(\begin{array}{ccc|c} -32/9 & 22/9 & 0 & 2.75 \\ 10/9 & -32/9 & 22/9 & 3.5 \\ 0 & 10/9 & -32/9 & 4.25 - (22/9)(1), \end{array}\right.$$

which can be solved to get the estimates $y(0.75) = -2.67, y(1.5) = -2.76, y(2.25) = -1.37$. Draw a smooth curve connecting these with the boundary conditions to sketch $y(x)$.

17

Set up a linear system in matrix form to solve the boundary-value problem $y'' + 5y' = 10$, $y(0) = 1$, $y(4) = 3$ for values of the function y(x), using second-order-accurate centered finite-difference approximations for the derivatives, with n=5. You do not need to solve the linear system.

Solution: At each interior grid point, the differential equation can be approximated with the finite differences as $\frac{y_{i-1}-2y_i+y_{i+1}}{h^2} + 5\frac{-y_{i-1}+y_{i+1}}{2h} = 10$, with $h = 0.8$ here. Including also the boundary conditions for y_0 and y_5 as the first and last equations, the linear system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{25}{16} & -\frac{25}{8} & \frac{75}{16} & 0 & 0 & 0 \\ 0 & -\frac{25}{16} & -\frac{25}{8} & \frac{75}{16} & 0 & 0 \\ 0 & 0 & -\frac{25}{16} & -\frac{25}{8} & \frac{75}{16} & 0 \\ 0 & 0 & 0 & -\frac{25}{16} & -\frac{25}{8} & \frac{75}{16} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 10 \\ 10 \\ 10 \\ 10 \\ 3 \end{pmatrix}$$

18

Set up a system of algebraic equations (in matrix form) using finite difference with $n = 5$ for $y'' - y' + y = 2$ subject to the boundary conditions $y(0) = 0$ and $y'(3) = 2$. Use second-order-accurate (centered where possible) finite difference approximations. You only need to write down the system, not to solve it.

Solution: Replacing the derivatives with second-order-accurate centered finite difference approximations, we have for the interior grid points

$$\left(\frac{1}{h^2} + \frac{1}{2h}\right) y_{i-1} + \left(1 - \frac{2}{h^2}\right) y_i + \left(\frac{1}{h^2} - \frac{1}{2h}\right) y_{i+1} = 2$$

and for the first-derivative boundary condition

$$\frac{1}{2h} y_{n-2} - \frac{2}{h} y_{n-1} + \frac{3}{2h} y_n = 2.$$

Here $h = 3/5$, so the linear system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 65/18 & -41/9 & 35/18 & 0 & 0 & 0 \\ 0 & 65/18 & -41/9 & 35/18 & 0 & 0 \\ 0 & 0 & 65/18 & -41/9 & 35/18 & 0 \\ 0 & 0 & 0 & 65/18 & -41/9 & 35/18 \\ 0 & 0 & 0 & 5/6 & -10/3 & 5/2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

The unknowns y_i are the values of $y(x)$ at the grid points, which are equally spaced x_i from $x = 0$ to $x = 3$.

19

Set up a system of algebraic equations (in matrix form) using finite difference with $n = 5$ to solve $y'' + y' - y = 4$ subject to the boundary conditions $y(x = 0) = 0$ and $y'(x = 3) = 2$. Use second-order-accurate (centered where possible) finite difference approximations. For full credit, solve this system and sketch $y(x)$.

Solution: Replacing the derivatives with second-order-accurate finite difference approximations, we have for the interior grid points

$$\left(\frac{1}{h^2} - \frac{1}{2h}\right) y_{i-1} + \left(-1 - \frac{2}{h^2}\right) y_i + \left(\frac{1}{h^2} + \frac{1}{2h}\right) y_{i+1} = 4$$

and for the first-derivative boundary condition

$$\frac{1}{2h} y_{n-2} - \frac{2}{h} y_{n-1} + \frac{3}{2h} y_n = 2.$$

Here $h = 3/5$, so the linear system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 35/18 & -59/9 & 65/18 & 0 & 0 & 0 \\ 0 & 35/18 & -59/9 & 65/18 & 0 & 0 \\ 0 & 0 & 35/18 & -59/9 & 65/18 & 0 \\ 0 & 0 & 0 & 35/18 & -59/9 & 65/18 \\ 0 & 0 & 0 & 5/6 & -10/3 & 5/2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 4 \\ 4 \\ 4 \\ 2 \end{pmatrix}$$

The unknowns y_i are the values of $y(x)$ at the grid points, which are equally spaced x_i from $x = 0$ to $x = 3$. The solution to the linear system is

$$\mathbf{y} = \begin{pmatrix} 0 \\ -1.921 \\ -2.380 \\ -2.178 \\ -1.565 \\ -0.561 \end{pmatrix},$$

which points can be connected to sketch $y(x)$.

20

Set up a system of algebraic equations (in matrix form) using finite difference with $n = 5$ to solve $y'' + y' + y = 2$ subject to the boundary conditions $y(x = 0) = 0$ and $y'(x = 1) = 2$. For full credit, solve this system and sketch $y(x)$.

Solution: Replacing the derivatives with second-order-accurate finite difference approximations, we have for the interior grid points

$$\left(\frac{1}{h^2} - \frac{1}{2h}\right) y_{i-1} + \left(1 - \frac{2}{h^2}\right) y_i + \left(\frac{1}{h^2} + \frac{1}{2h}\right) y_{i+1} = 2$$

Here $h = 1/5$, so the linear system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 22.5 & -49 & 27.5 & 0 & 0 & 0 \\ 0 & 22.5 & -49 & 27.5 & 0 & 0 \\ 0 & 0 & 22.5 & -49 & 27.5 & 0 \\ 0 & 0 & 0 & 22.5 & -49 & 27.5 \\ 0 & 0 & 0 & 2.5 & -10 & 7.5 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

The unknowns y_i are the values of $y(x)$ at the grid points, which are equally spaced x_i from $x = 0$ to $x = 1$. The solution to the linear system is

$$\mathbf{y} = \begin{pmatrix} 0 \\ 1.433 \\ 2.627 \\ 3.580 \\ 4.303 \\ 4.810 \end{pmatrix},$$

which points can be connected to sketch $y(x)$.

21

(a) Set up a system of linear algebraic equations (in matrix form) using finite difference with $n = 5$ to solve $y'' + y = x + 4$ subject to the boundary conditions $y(x = 0) = 0$ and $y(x = 5) = 2$. Explain what the unknowns in this system represent.
(b) Write Python code to solve the same problem numerically and to plot $y(x)$.

Solution: (a) Replacing the second derivative with a second-order-accurate finite difference approximation, we have for the interior grid points

$$\frac{1}{h^2} y_{i-1} + \left(1 - \frac{2}{h^2}\right) y_i + \frac{1}{h^2} y_{i+1} = x_i + 4$$

Here $h = 1$, so the linear system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ 6 \\ 7 \\ 8 \\ 2 \end{pmatrix}$$

The unknowns y_i are the values of $y(x)$ at the grid points x_i.

(b)

```
from scipy.integrate import solve_bvp
```

```
from matplotlib.pyplot import plot
dzdx = lambda x, z:  [z[1], x + 4 - z[0]]
bc = lambda za, zb:  [za[0], zb[0] - 2]
xi = [0, 5]
zi = [[1, 0.4], [1, 0.4]]
sol = solve_bvp(dzdx, bc, xi, zi)
plot (sol.x, sol.y[0, :])
```
(For example, $y(x = 2.5) \approx 13.366$)

22

(a) Set up a system of linear equations (in matrix form) using finite difference with $n = 4$ to solve the equation $y'' - y = x + 3$ subject to the boundary conditions $y(x = 0) = 0$ and $y(x = 2) = 0$.
(b) Solve this system (for example, using Gaussian elimination) and sketch $y(x)$.
(c) Write Python code to solve the boundary-value problem numerically and plot $y(x)$.

Solution: A finite-difference approximation to the ODE would be

$$h^{-2}y_{i-1} - (1 + 2h^{-2})y_i + h^{-2}y_{i+1} = x_i + 3.$$

Applying this here with $h = \frac{1}{2}$ for $i = 1, 2, 3$, $x_i = 0.5, 1, 1.5$, we get

$$\left(\begin{array}{ccc|c} -9 & 4 & 0 & 3.5 \\ 4 & -9 & 4 & 4 \\ 0 & 4 & -9 & 4.5, \end{array}\right)$$

which can be solved to get the estimates $y(0.5) = -1.01, y(1) = -1.39, y(1.5) = -1.12$. Draw a smooth curve connecting these with the boundary conditions to sketch $y(x)$.
(c) Converting to a first-order system with $z_1 \equiv y, z_2 \equiv y', z_1' = z_2, z_2' = z_1 + x + 3$ and boundary conditions $z_1(0) = z_1(2) = 0$,

```
from scipy.integrate import solve_bvp
from matplotlib.pyplot import plot
from numpy import zeros
dzdx = lambda x, z:  [z[1], x + 3 + z[0]]
bc = lambda za, zb:  [za[0], zb[0]]
xi = [0, 2]
zi = zeros((2, 2))
sol = solve_bvp(dzdx, bc, xi, zi)
plot (sol.x, sol.y[0, :])
```
(For example, $y(x = 1) \approx -1.408$)

23

(a) Set up a system of linear equations (in matrix form) using finite difference with $n = 6$ to solve the equation $y'' + 3y' = y + 4$ subject to the boundary conditions $y(x = 0) = 0$ and $y(x = 1) = 0$.
(b) Solve this system (for example, using Gaussian elimination) and sketch the solution $y(x)$.

Solution: (a) A discretized approximation for $y'' + 3y' = y + 4$, assuming a grid with equal spacings h, is

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + 3\frac{y_{i+1} - y_{i-1}}{2h} = y_i + 4,$$

or

$(h^{-2} - 1.5h^{-1})y_{i-1} + (-2h^{-2} - 1)y_i + (h^{-2} + 1.5h^{-1})y_{i+1} = 4.$

With $h = 1/6$, the coefficients are $27, -73, 45$ respectively. Also putting in the boundary conditions, the linear system in augmented matrix form becomes

$$\left[\begin{array}{ccccc|c} -73 & 45 & 0 & 0 & 0 & 4 \\ 27 & -73 & 45 & 0 & 0 & 4 \\ 0 & 27 & -73 & 45 & 0 & 4 \\ 0 & 0 & 27 & -73 & 45 & 4 \\ 0 & 0 & 0 & 27 & -73 & 4, \end{array}\right]$$

where the unknowns $y_1, y_2, \ldots y_5$ are the approximate function values at $1/6, 1/3, \ldots, 5/6$.

(b) Since this is a tridiagonal system, it can be solved relatively quickly by Gaussian elimination (without pivoting). The steps are

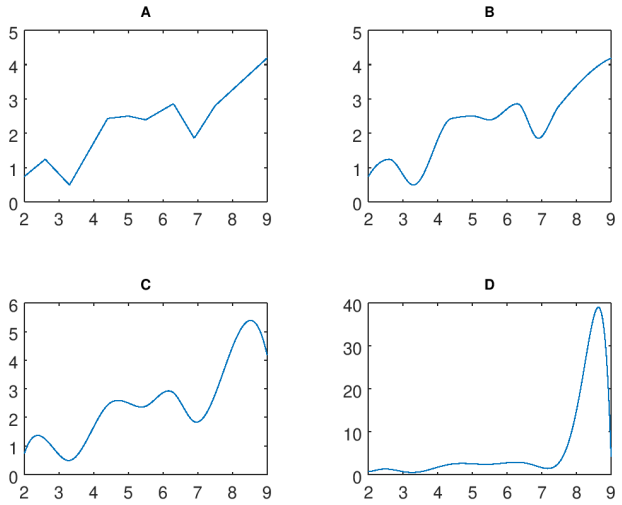
$$\left[\begin{array}{ccccc|c} -73 & 45 & 0 & 0 & 0 & 4 \\ 27 & -73 & 45 & 0 & 0 & 4 \\ 0 & 27 & -73 & 45 & 0 & 4 \\ 0 & 0 & 27 & -73 & 45 & 4 \\ 0 & 0 & 0 & 27 & -73 & 4, \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} -73 & 45 & 0 & 0 & 0 & 4 \\ 0 & -56.356 & 45 & 0 & 0 & 5.479 \\ 0 & 27 & -73 & 45 & 0 & 4 \\ 0 & 0 & 27 & -73 & 45 & 4 \\ 0 & 0 & 0 & 27 & -73 & 4, \end{array}\right] \rightarrow$$

$$\left[\begin{array}{ccccc|c} -73 & 45 & 0 & 0 & 0 & 4 \\ 0 & -56.356 & 45 & 0 & 0 & 5.479 \\ 0 & 0 & -51.441 & 45 & 0 & 6.625 \\ 0 & 0 & 27 & -73 & 45 & 4 \\ 0 & 0 & 0 & 27 & -73 & 4, \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} -73 & 45 & 0 & 0 & 0 & 4 \\ 0 & -56.356 & 45 & 0 & 0 & 5.479 \\ 0 & 0 & -51.441 & 45 & 0 & 6.625 \\ 0 & 0 & 0 & -49.381 & 45 & 7.477 \\ 0 & 0 & 0 & 0 & -48.395 & 8.088, \end{array}\right]$$

After which back substitution gives $y_5 = -0.1671, y_4 = -0.3037, y_3 = -0.3945, y_2 = -0.4122, y_1 = -0.3089$. Together with the boundary conditions $y_0 = 0, y_6 = 0$, these values can be used to sketch $y(x)$ as a downward-pointing parabola-like (but asymmetric) curve with a minimum (y less than -0.41) near $x = 0.4$.

24

The subplots below show different interpolating functions fit to the same set of 10 points, which have x coordinates ranging from 2 to 9. Identify each function plotted and briefly explain how you know. The options are polynomial interpolation, linear spline, cubic spline, PCHIP, or none of those (only select none if you are sure that the other options are all impossible).



Solution: A: linear spline (points are connected by straight-line segments)
B: PCHIP (points are connected by curves that don't over or undershoot the points)
C: spline (points are connected by a smoother curve than PCHIP; more oscillation than PCHIP evident near $x = 5$ and $x = 8$)
D: polynomial (the most oscillatory, as indicated by the much larger y axis range compared to the original points and the other interpolation methods)

25

Write a Python script to draw a smooth-looking graph of a function that goes through the following points:

x	0	1	3	4	6
y	7	8	5	3	2.

Solution: You need to use a smooth interpolation method, such as the cubic spline, to generate enough points that their graph looks smooth. For example:

```
from numpy import linspace
from scipy.interpolate import CubicSpline
from matplotlib.pyplot import plot
x = [0, 1, 3, 4, 6]
y = [7, 8, 5, 3, 2]
xx = linspace (0, 6, 200)
sp = CubicSpline(x, y, bc_type = 'natural')
yy = sp(xx)
plot (x, y, 'o', xx, yy)
```

26

Given the points

$$\begin{array}{c|ccccc} x_i & -2 & -1 & 0 & 1 & 2 \\ y_i & 16 & 6 & 4 & 4 & 0, \end{array}$$

estimate $y(x = 1.3)$ using (a) polynomial interpolation through all the given points, (b) polynomial interpolation through the three closest points, (c) piecewise linear interpolation.

Solution: (a) The interpolating polynomial is $y = -x^3 + x^2 + 4$. Its value at $x = 1.3$ is **3.493**
(b) The interpolating polynomial is $y = -2x^2 + 2x + 4$. Its value at $x = 1.3$ is **3.22**
(c) The interpolating polynomial piece is $y = -4x + 8$. Its value at $x = 1.3$ is **2.8**

27

Given the points

$$\begin{array}{c|ccccc} x_i & -2 & -1 & 0 & 1 & 2 \\ y_i & 1 & -2 & 1 & -2 & 1, \end{array}$$

estimate $y(x = 1.2)$ using (a) polynomial interpolation through all the given points, (b) polynomial interpolation through the three closest points, (c) piecewise linear interpolation.

Solution: (a) The interpolating polynomial through the points is $x^4 - 4x^2 + 1$, which at $x = 1.2$ has the value -2.6864
(b) The interpolating polynomial through the 3 closest points is $3(x - 1)^2 - 2$, which at $x = 1.2$ has the value -1.88
(c) Linear interpolation using the $x = 1$ and $x = 2$ values gives -1.4

28

(a) Given the (x, y) values $(1, 2), (2, 4), (3, 10), (5, 7)$ write in matrix form the linear system to solve for the b_i coefficients in the interpolating cubic spline with natural boundary conditions.

(b) If an interpolating cubic spline is as given below, find (1) what points it interpolates and (2) its value at $x = 0$.

$$-1.27(x+1)^3 + 6.54(x+1) - 2 \text{ for } -1 \le x \le 1$$
$$9.15(x-1)^3 - 7.57(x-1)^2 - 8.59(x-1) + 2 \text{ for } 1 \le x \le 2$$
$$-6.63(x-2)^3 + 19.89(x-2)^2 + 3.74(x-2) - 6 \text{ for } 2 \le x \le 3$$

Solution:

(a) Similar to a homework problem, the linear system can be written in the form

$$\begin{bmatrix} 1 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 \\ 0 & h_2 & 2(h_2+h_3) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 3(\Delta_2 - \Delta_1) \\ 3(\Delta_3 - \Delta_2) \end{bmatrix}$$

which is

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 6 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \\ -22.5 \end{bmatrix}$$

(b) (1) The x coordinates of the interpolated points are given by the segment endpoints and the y coordinates by the spline function values at those points. So the points are $(-1, -2), (1, 2), (2, -6), (3, 11)$.
(2) $x = 0$ is in the first segment, and the value of the first cubic polynomial there is **3.27**