HW1

Gavin DeBrun

1a)

$(x_i, y_i) \in \{(0,0), (\frac{1}{2}, 1), (1,0)\}$

$L_0 = \frac{(x-\frac{1}{2})(x-1)}{(0-\frac{1}{2})(0-1)} = 2(x-\frac{1}{2})(x-1)$

$L_1 = \frac{(x-0)(x-1)}{(\frac{1}{2}-0)(\frac{1}{2}-1)} = -4x(x-1)$

$L_2 = \frac{(x-0)(x-\frac{1}{2})}{(1-0)(1-\frac{1}{2})} = 2x(x-\frac{1}{2})$

$P_2(x) = 0 \cdot L_0 + 1 \cdot L_1 + 0 \cdot L_2 = L_1$

$P_2(\frac{1}{4}) = -4(\frac{1}{4})(\frac{1}{4}-1) = \boxed{\frac{3}{4}}$

1b)

$P_2'(x) = -8x + 4$

$\boxed{P_2'(\frac{1}{4}) = 2}$

2)

```
import numpy as np

def lagrange(x, f, xx):

    yy = 0
    for i in range(len(x)):
        Lj = 1
        for j in range(len(x)):
            if i != j:
                Lj *= (xx - x[j]) / (x[i] - x[j])
        yy += Lj * f[i]
    return yy
```

3a)

We have $g''(0) = 0 = g''(4)$, $h = 1$ and $b_i = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$ for $i \neq \{0, 4\}$

Thus, $A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, $x = \begin{bmatrix} g''(x_0) \\ g''(x_1) \\ g''(x_2) \\ g''(x_3) \\ g''(x_4) \end{bmatrix}$ and using the aforementioned formula for the applicable b_i's, we obtain:

$b = \begin{bmatrix} 0 \\ 1 \\ -12 \\ 9 \\ 0 \end{bmatrix}$

Now we can solve the system of equations for x:

```python
import numpy as np
A = np.array([[1, 0, 0, 0, 0],
              [1/6, 2/3, 1/6, 0, 0],
              [0, 1/6, 2/3, 1/6, 0],
              [0, 0, 1/6, 2/3, 1/6],
              [0, 0, 0, 0, 1]])
b = np.array([0, 1, -12, 9, 0])
gpp = np.linalg.solve(A, b)
print(gpp)
```

```
[  0.          7.71428571 -24.85714286  19.71428571   0.        ]
```

Thus, we have $g''(0) = 0$, $g''(1) = 7.714$, $g''(2) = -24.857$, $g''(3) = 19.714$ and $g''(4) = 0$.

3b)

We have $i = 2$, $i + 1 = 3$, $y_i = 7$, $y_{i+1} = -1$, $g''(x_i) = -24.857$, and $g''(x_{i+1}) = 19.714$. So using Moin eq. 1.6, we have:

$g_2(2.4) = \frac{-24.857}{6} \cdot [(.6)^3 - .6] + \frac{19.714}{6}[(.4)^3 - .4] + 3.8 = \boxed{4.287}$

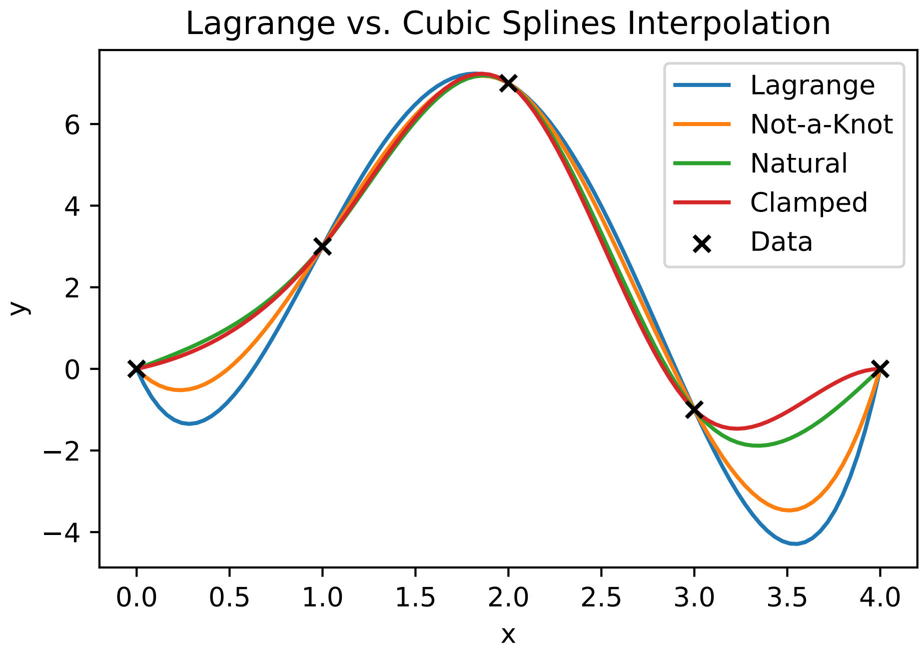
3c)

```python
import numpy as np
from scipy.interpolate import CubicSpline

x = np.array([0, 1, 2, 3, 4])
y = np.array([0, 3, 7, -1, 0])

cs = CubicSpline(x, y, bc_type="natural")
print(cs(2.4))
```

```
4.286857142857143
```

4)

```python
import numpy as np
from scipy.interpolate import CubicSpline
import matplotlib.pyplot as plt


def lagrange(x, f, xx):
    yy = 0
    for i in range(len(x)):
        Lj = 1
        for j in range(len(x)):
            if i != j:
                Lj *= (xx - x[j]) / (x[i] - x[j])
        yy += Lj * f[i]
    return yy


x = np.array([0, 1, 2, 3, 4])
y = np.array([0, 3, 7, -1, 0])
xx = np.linspace(0, 4, 100)

lg_interp = lagrange(x, y, xx)

cs1 = CubicSpline(x, y, bc_type="not-a-knot")
nak_interp = cs1(xx)

cs2 = CubicSpline(x, y, bc_type="natural")
nat_interp = cs2(xx)

cs3 = CubicSpline(x, y, bc_type=((1, 1), (1, 0)))
clamp_interp = cs3(xx)

plt.plot(xx, lg_interp, label="Lagrange")
plt.plot(xx, nak_interp, label="Not-a-Knot")
plt.plot(xx, nat_interp, label="Natural")
plt.plot(xx, clamp_interp, label="Clamped")
plt.scatter(x, y, label="Data", marker="x", zorder=10, color="black")
plt.legend()
plt.xlabel("x")
plt.ylabel("y")
plt.title("Lagrange vs. Cubic Splines Interpolation")
plt.show()
```



Lagrange vs. Cubic Splines Interpolation
Lagrange
Not-a-Knot
Natural
Clamped
Data
y
6
4
2
0
−2
−4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x