Numerical Methods Homework 5

Adam Kit

19 May 2020

1 Mueller's Method

We want to find the root of the equation $x^3 + 2x^2 + 10x - 20 = 0$ correct upto three decimal places using Mueller's method. The initial approximations $x_0 = 0$, $x_1 = 1$, $x_2 = 2$.

Mueller's method is based on the secant method, just instead of 2 points, we use 3 points and construct a parabola. Kinda like the Lagrange polynomial method, just without the Lagrange polynomials! As usual the code for this homework can be found here

The parabola is constucted with the three points $(x_{k-1}, f(x_{k-1}))$, $(x_{k-2}, f(x_{k-2}))$, $(x_{k-3}, f(x_{k-3}))$, and resembles

$$y_k(x) = f(x_{k-1}) + (x - x_{k-1})f[x_{k-1}, x_{k-2}] + (x - x_{k-1})(x - x_{k-2})f[x_{k-1}, x_{k-2}, x_{k-3}]$$

which leads us to the recurrance

$$x_k = x_{k-1} - \frac{2f(x_{k-1})}{\omega \pm \sqrt{\omega^2 - 4f[x_{k-1}, x_{k-2}, x_{k-3}]}}$$

where $\omega = f[x_{k-1}, x_{k-2}] + f[x_{k-1}, x_{k-3}] - f[x_{k-2}, x_{k-3}]$ and divided difference defined by

$$f[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

and

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

The program is attached in Figure 2, and the output is 1.369, and a graphic can be found in Figure 1

2 Gaussian Elimination

For Gaussian Elimination we are allowed 3 types of operations to get to the row reduced form:

- Swapping two rows
- Multiplying a row by a nonzero number
- Adding a multiple of one row to another row

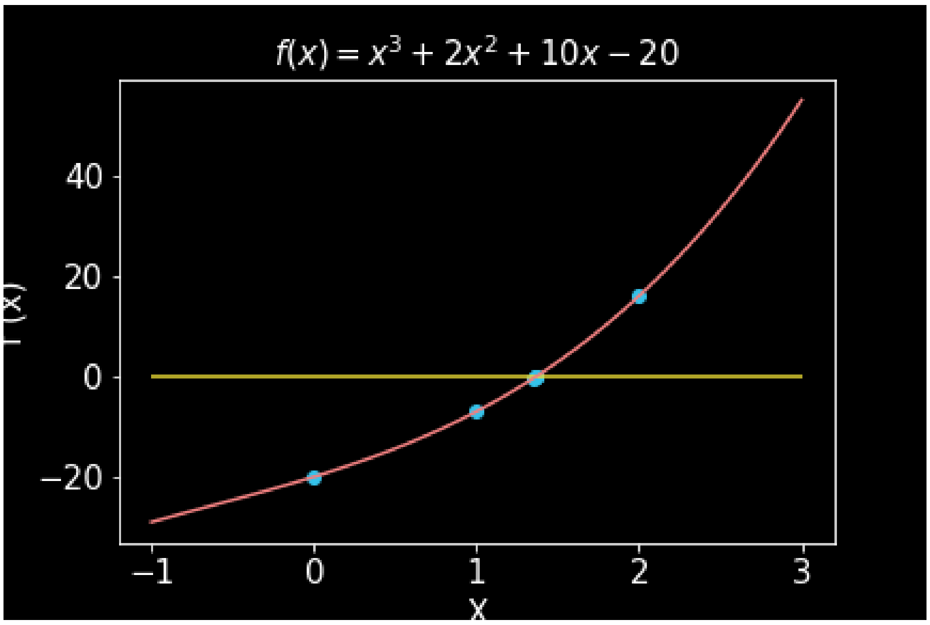


Figure 1: Mueller Algortihm, the blue dots represent the x values chosen for the parabola construction. See *mullersmethod.py* in the repository

So using these three rules, we can solve $AX = B$, where A is defined as

$$A = \begin{bmatrix} 10 & -7 & 3 & 5 \\ -6 & 8 & -1 & -4 \\ 3 & 1 & 4 & 11 \\ 5 & -9 & -2 & 4 \end{bmatrix}; B = \begin{bmatrix} 6 \\ 5 \\ 2 \\ 7 \end{bmatrix}$$

We start with: $\left|\begin{array}{cccc|c} 10 & -7 & 3 & 5 & 6 \\ -6 & 8 & -1 & -4 & 5 \\ 3 & 1 & 4 & 11 & 2 \\ 5 & -9 & -2 & 4 & 7 \end{array}\right.$ and try to end with something like $\left|\begin{array}{cccc|c} 1 & 0 & 0 & 0 & h \\ 0 & 1 & 0 & 0 & i \\ 0 & 0 & 1 & 0 & j \\ 0 & 0 & 0 & 1 & k \end{array}\right.$

Step 1: $R_1/10 \to R_1$ $\left|\begin{array}{cccc|c} 1 & -0.7 & 0.3 & 0.5 & 0.6 \\ -6 & 8 & -1 & -4 & 5 \\ 3 & 1 & 4 & 11 & 2 \\ 5 & -9 & -2 & 4 & 7 \end{array}\right.$

Step 2: $R_2 + 6R_1 \to R_2$; $R_3 - 3R_1 \to R_3$; $R_4 - 5R_1 \to R_4$ $\left|\begin{array}{cccc|c} 1 & -0.7 & 0.3 & 0.5 & 0.6 \\ 0 & 3.8 & 0.8 & -1 & 8.6 \\ 0 & 3.1 & 3.1 & 9.5 & 0.2 \\ 0 & -5.5 & -3.5 & 1.5 & 4 \end{array}\right.$

Step 3: $R_2/3.8 \to R_2$ $\left|\begin{array}{cccc|c} 1 & -0.7 & 0.3 & 0.5 & 0.6 \\ 0 & 1 & \frac{4}{19} & -\frac{5}{19} & \frac{43}{19} \\ 0 & 3.1 & 3.1 & 9.5 & 0.2 \\ 0 & -5.5 & -3.5 & 1.5 & 4 \end{array}\right.$

Step 4: $R_1 + 0.7R_2 \to R_1$; $R_3 - 3.1R_2 \to R_3$; $R_4 + 5.5R_2 \to R_4$ $\left|\begin{array}{cccc|c} 1 & 0 & \frac{17}{38} & \frac{6}{19} & \frac{83}{38} \\ 0 & 1 & \frac{4}{19} & -\frac{5}{19} & \frac{43}{19} \\ 0 & 0 & \frac{93}{38} & \frac{196}{19} & -\frac{259}{38} \\ 0 & 0 & -\frac{89}{38} & \frac{1}{19} & \frac{625}{38} \end{array}\right.$

Step 5: $R_3/\frac{93}{38} \to R_3$ $\left|\begin{array}{cccc|c} 1 & 0 & \frac{17}{38} & \frac{6}{19} & \frac{83}{38} \\ 0 & 1 & \frac{4}{19} & -\frac{5}{19} & \frac{43}{19} \\ 0 & 0 & 1 & \frac{392}{93} & -\frac{259}{93} \\ 0 & 0 & -\frac{89}{38} & \frac{1}{19} & \frac{625}{38} \end{array}\right.$

Step 6: $R_1 - \frac{17}{38}R_3 \to R_1$; $R_2 - \frac{4}{19}R_3 \to R_2$; $R_4 + \frac{89}{38}R_3 \to R_4$ $\left|\begin{array}{cccc|c} 1 & 0 & 0 & -\frac{146}{93} & \frac{319}{93} \\ 0 & 1 & 0 & -\frac{107}{93} & \frac{256}{93} \\ 0 & 0 & 1 & \frac{392}{93} & -\frac{259}{93} \\ 0 & 0 & 0 & \frac{923}{93} & \frac{923}{93} \end{array}\right.$

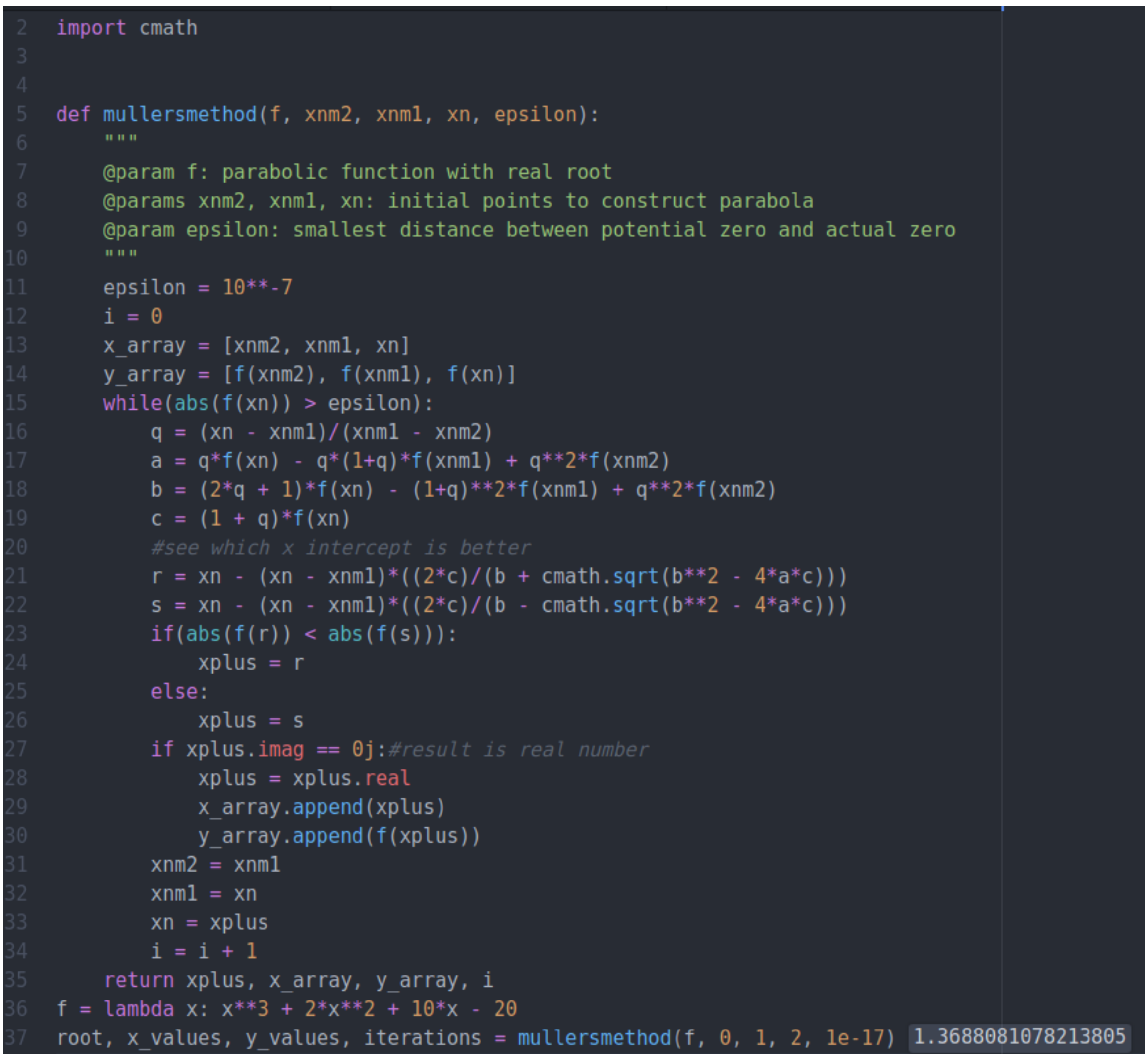
Step 7: $R_4/\frac{923}{93} \to R_4$; $\left|\begin{array}{cccc|c} 1 & 0 & 0 & -\frac{146}{93} & \frac{319}{93} \\ 0 & 1 & 0 & -\frac{107}{93} & \frac{256}{93} \\ 0 & 0 & 1 & \frac{392}{93} & -\frac{259}{93} \\ 0 & 0 & 0 & 1 & 1 \end{array}\right.$

Step 8: $R_1 + \frac{146}{93}R_4 \to R_1$; $R_2 + \frac{107}{93}R_4 \to R_2$; $R_3 = \frac{392}{93}R_4 \to R_3$ $\left|\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 & -7 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right.$

Thus $X = \begin{bmatrix} 5 \\ 4 \\ -7 \\ 1 \end{bmatrix}$ and to check we can plug in these values into the equations, which I give the first for example:

$$10(5) - 7(4) + 3(-7) + 5(1) = 6$$

(hopefully its okay I do not show the rest, but trust me they work out ;))



```python
import cmath


def mullersmethod(f, xnm2, xnm1, xn, epsilon):
    """
    @param f: parabolic function with real root
    @params xnm2, xnm1, xn: initial points to construct parabola
    @param epsilon: smallest distance between potential zero and actual zero
    """
    epsilon = 10**-7
    i = 0
    x_array = [xnm2, xnm1, xn]
    y_array = [f(xnm2), f(xnm1), f(xn)]
    while(abs(f(xn)) > epsilon):
        q = (xn - xnm1)/(xnm1 - xnm2)
        a = q*f(xn) - q*(1+q)*f(xnm1) + q**2*f(xnm2)
        b = (2*q + 1)*f(xn) - (1+q)**2*f(xnm1) + q**2*f(xnm2)
        c = (1 + q)*f(xn)
        #see which x intercept is better
        r = xn - (xn - xnm1)*((2*c)/(b + cmath.sqrt(b**2 - 4*a*c)))
        s = xn - (xn - xnm1)*((2*c)/(b - cmath.sqrt(b**2 - 4*a*c)))
        if(abs(f(r)) < abs(f(s))):
            xplus = r
        else:
            xplus = s
        if xplus.imag == 0j:#result is real number
            xplus = xplus.real
            x_array.append(xplus)
            y_array.append(f(xplus))
        xnm2 = xnm1
        xnm1 = xn
        xn = xplus
        i = i + 1
    return xplus, x_array, y_array, i
f = lambda x: x**3 + 2*x**2 + 10*x - 20
root, x_values, y_values, iterations = mullersmethod(f, 0, 1, 2, 1e-17) 1.3688081078213805
```

Figure 2: Code can also be found in Link to Github