EXERCISE 1

1. (20 points) Let f be a smooth function of one variable. Suppose we are given the values of f and its first derivative f' at two distinct points, say, $x = a$ and $x = b$. Find a cubic polynomial that interpolates that data. That is, find a cubic p such that $p(a) = f(a)$, $p(b) = f(b)$, $p'(a) = f'(a)$, and $p'(b) = f'(b)$. Next, repeat the exercise with data extended by one more derivative, i.e., find a polynomial of degree five which matches the values of f, f', and f'' at $x = a$ and $x = b$. Bonus points for working out the general case which can be viewed as a generalization of Taylor's formula.

Solution: To find both the third order and fifth order polynomials, I solved a linear system of equations. For the cubic polynomial, the following linear system was formed:

$$\begin{bmatrix} 1 & a & a^2 & a^3 \\ 1 & b & b^2 & b^3 \\ 0 & 1 & 2\,a & 3\,a^2 \\ 0 & 1 & 2\,b & 3\,b^2 \end{bmatrix} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} = \begin{bmatrix} f(a) \\ f(b) \\ f'(a) \\ f'(b) \end{bmatrix}$$

where the the polynomial in general is given by

$$p(x) = \sum_{n=0}^{3} \xi_n x^n.$$

By solving for the ξ_n, substituting them back into the equation, and further simplifying, the polynomial we obtain is

$$\begin{aligned} p(x) =& f(a)\left[\frac{(x-b)^2(3a-b-2x)}{(a-b)^3}\right] + f'(a)\left[\frac{(x-b)^2(x-a)}{(a-b)^2}\right] + \\ & f(b)\left[\frac{(x-a)^2(a-3b+2x)}{(a-b)^3}\right] + f'(b)\left[\frac{(x-a)^2(x-b)}{(a-b)^2}\right]. \end{aligned}$$

We can verify that this is indeed the polynomial by substituting both $x = a$ and $x = b$ into $p(x)$ and $p'(x)$. By performing the same process, for the fifth degree polynomial, the result I obtained is

$$\begin{aligned} p(x) =& f(a)\left[\frac{(x-b)^3(5a(2a-b-3x)+b^2+3bx+6x^2)}{(a-b)^5}\right] + f'(a)\left[\frac{(x-b)^3(x-a)(4a-b-3x)}{(a-b)^4}\right] + \\ & f''(a)\left[\frac{(x-b)^3(x-a)^2}{2(a-b)^3}\right] - f(b)\left[\frac{(x-a)^3(a(a-5b-3x)+10b^2-15bx+6x^2)}{(a-b)^5}\right] - \\ & f'(a)\left[\frac{(x-a)^3(x-b)(a-4b+3x)}{(a-b)^4}\right] - f''(b)\left[\frac{(x-a)^3(x-b)^2}{2(a-b)^3}\right]. \end{aligned}$$

Again, this can be verified by substituting $x = a$ and $x = b$ into p, p', and p''.

I thought about trying to derive a general formula and I was honestly overwhelmed, so I scoured the web to see if I could find anything, and what I found was a paper written by José L. López and Nico M. Temme. The general formula for the Two-point Taylor formula is

$$
\begin{aligned}
P_n(x_1,x_2;x) &= \sum_{k=0}^{n-1}[a_k(x_1,x_2)(x-x_1)+a_k(x_2,x_1)(x-x_2)](x-x_1)^k(x-x_2)^k,\\
a_0(x_1,x_2) &= \frac{f(x_2)}{(x_2-x_1)}\\
a_n(x_1,x_2) &= \sum_{k=0}^{n}\frac{(n+k-1)!}{k!(n-k)!}\frac{(-1)^{n+1}nf^{(n-k)}(x_2)+(-1)^k k f^{(n-k)}(x_1)}{n!(x_1-x_2)^{n+k-1}}.
\end{aligned}
$$

EXERCISE 2

2. (20 points) Let P_N denote the vector space of polynomials of degree N restricted to the interval $[-1,1]$ and equipped with the standard L^2 inner product. For each N, there exists a function of two variables $K_N(x,y)$ such that

$$p(y) = \langle K_N(x,y), p(x)\rangle$$

for all $p \in P_N$ For example, it can be easily confirmed that

$$a + by = \int_{-1}^{1} \left(\frac{1}{2} + \frac{3}{2}xy\right)(a+bx)dx.$$

Therefore,

$$K_1(x,y) = \frac{1}{2} + \frac{3}{2}xy.$$

For quadratics, one can write K_2 as

$$K_2(x,y) = \frac{9}{8} + \frac{3}{2}xy + \frac{45}{8}x^2y^2 - \frac{15}{8}x^2 - \frac{15}{8}y^2,$$

which, again, is easily confirmed by direct computation. Derive an expression for K_3. Bonus points if you find a general formula for K_N.

Solution: Since I wasn't able to get the extra credit on the last exercise, I was able to get it for this exercise. But before I provide the formula for K_n, I will first provide the formula for K_3. In order to compute K_3, I first looked at K_1 and K_2 to see if I noticed any trends. What I noticed is that only even order terms were present in both K_1 and K_2 (xy, x^2, y^2, x^2y^2,etc), but I wasn't entirely sure. So, I used a general third order bivariate multinomial in x and y. Let us call it $q(x,y)$. Then, I took the inner product of q with each term of p. This led to the following:

$$\int_{-1}^{1} q(x,y)\ 1\ dx = \frac{2}{3}y^2a_7 + \frac{2}{3}a_3 + 2a_4y^2 + 2a_1 = 1$$
$$\int_{-1}^{1} q(x,y)\ x\ dx = \frac{2}{5}y^3a_8 + \frac{2}{5}ya_6 + \frac{2}{3}y^3a_5 + \frac{2}{3}ya_2 = y$$
$$\int_{-1}^{1} q(x,y)\ x^2\ dx = \frac{2}{5}y^2a_7 + \frac{2}{5}a_3 + \frac{2}{3}a_4y^2 + \frac{2}{3}a_1 = y^2$$
$$\int_{-1}^{1} q(x,y)\ x^3\ dx = \frac{2}{7}y^3a_8 + \frac{2}{7}ya_6 + \frac{2}{5}y^3a_5 + \frac{2}{5}ya_2 = y^3.$$

What we obtain from above are systems of equations. After solving these equations for their respective coefficients, we find that K_3 is

$$\boxed{K_3(x,y) = \frac{175}{8}x^3y^3 - \frac{105}{8}y^3x + \frac{45}{8}x^2y^2 - \frac{105}{8}yx^3 - \frac{15}{8}y^2 + \frac{75}{8}xy - \frac{15}{8}x^2 + \frac{9}{8}}$$

In order to determine $K_n(x,y)$, I first tried to use what I found concerning that only even orders existed. This led me to write K_n using the binomial theorem. However, this was foolish. Then, I tried to be clever. I knew that odd ordered terms vanished and that I was working in $L^2[-1,1]$. So, I decided to determine if I could use orthogonal polynomials. I tried all of the classical orthogonal polynomials defined on $[-1,1]$, and I found that I could express K_n as some constant times the integral of the product of Jacobi polynomials with $a = 1$ and $b = 0$:

$$K_n(x,y) = c\int_{-1}^{1} P_n^{(1,0)}(s\ x)\ P_n^{(1,0)}(s\ y)\ ds.$$

At this point, I knew that I was close. What I noticed from $n = 1$, $n = 2$, and $n = 3$ is that I needed to scale the integral by $(1)^2$, $(3/2)^2$, and $(4/2)^2$, respectively. So, I found that $c = \left(\frac{n+1}{2}\right)^2$. Thus, I found the formula to be

$$\boxed{K_n(x,y) = \left(\frac{n+1}{2}\right)^2 \int_{-1}^{1} P_n^{(1,0)}(s\ x)\ P_n^{(1,0)}(s\ y)\ ds.}$$

In testing the formula in Maple for $n = 1, \ldots, 7$, the inner product of K_n and the n-th degree polynomial in x returned the exact n-th degree polynomial in y, as expected.

EXERCISE 3

3. (20 points) Quadratures are usually constructed from the values of the integrand. However, suppose that in addition to the values of the function f we have access to the values of its first derivative f'. Find an optimal quadrature of the form

$$Q(f) = w_1 f(x_1) + w_2 f(x_2) + w_3 f'(x_1) + w_4 f'(x_2)$$

for approximating $\int_{-1}^{1} f(x)\,dx$. Organize your solution as follows. First, choose an optimality criterion. Then use that criterion to find the nodes and weights. Finally, derive an error bound of the form:

$$|E(f)| \le C \max_{-1\le\xi\le1} |f^{(n)}(\xi)|.$$

Note: If the problem is too simple for two nodes, you can challenge yourself by the generalization to N nodes (for extra credit of course).

Solution: To begin, the optimality condition that I wanted to utilize was that the quadrature rule be exact for polynomials of degree $2n-1$ because the interval of definition is $[-1,1]$. Let us define the error functional E as

$$E(f) = \int_{-1}^{1} f(x)\,dx - w_1 f(x_1) - w_2 f(x_2) - w_3 f'(x_1) - w_4 f'(x_2).$$

To satisfy the optimality condition, we must have

$$\begin{aligned}
E(1) &= 2 - w_1 - w_2 = 0\\
E(x) &= -w_1 x_1 - w_2 x_2 - w_3 - w_4 = 0\\
E(x^2) &= \frac{2}{3} - w_1 {x_1}^2 - w_2 {x_2}^2 - 2w_3 x_1 - 2w_4 x_2 = 0\\
E(x^3) &= -w_1 {x_1}^3 - w_2 {x_2}^3 - 3w_3 {x_1}^2 - 3w_4 {x_2}^2 = 0.
\end{aligned}$$

To make finding the nodes and weights easier, we can assume that $w_1 = w_2$, $w_3 = w_4$, and $x_1 = -x_2$. By making these assumptions and solving the system, we find that

$$x_1 = \frac{1}{\sqrt{3}}, \quad x_2 = -\frac{1}{\sqrt{3}}, \quad w_1 = w_2 = 1, \quad w_3 = w_4 = 0.$$

So, the quadrature rule becomes standard Gaussian Quadrature:

$$\int_{-1}^{1} f(x)\,dx = f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right).$$

This, however is by no means the only quadrature rule for which polynomials of degree $2n-1$ are terminated. I could have even chosen that the optimality condition require that the error function annihilate polynomials with degree up to $3n-1$. Generally, without assumptions, the weights for this quadrature rule are given by

$$\begin{aligned} w_1 &= \frac{2(3x_1{x_2}^2 - {x_2}^3 + x_1 + x_2)}{(x_1 - x_2)^3} \\ w_2 &= \frac{2({x_1}^3 - 3{x_1}^2 x_2 - x_1 - x_2)}{(x_1 - x_2)^3} \\ w_3 &= -\frac{2(3x_1{x_2}^2 + x_1 + 2x_2)}{3\,(x_1 - x_2)^2} \\ w_4 &= -\frac{2(3{x_1}^2 x_2 + 2x_1 + x_2)}{3\,(x_1 - x_2)^2} \end{aligned}$$

For this general case, we can choose any $-1 < x_1 < 1$ and any $-1 < x_2 < 1$ to obtain a quadrature rule. For the error term for Gaussian Quadrature, we utilize Peano's Kernel Theorem. The error functional is:

$$E(f) = \int_{-1}^{1} f(x)dx - f\big(-1/\sqrt{3}\big) - f\big(1/\sqrt{3}\big). \tag{1}$$

By applying our error functional to monomials of increasing degree, we find that the error functional annihilates up to cubic monomials, for the optimality condition. For quartic monomials, the error we obtain is

$$E(x^4) = \int_{-1}^{1} x^4 dx - \left(-\frac{1}{\sqrt{3}}\right)^4 - \left(\frac{1}{\sqrt{3}}\right)^4 = \frac{8}{45}.$$

Thus, by application of Peano's Kernel Theorem, because our error functional is linear and annihilates monomials (and thus polynomials) of degree 3, we have

$$E(f) = \int_{-1}^{1} f^{(4)} K(t)dt,$$

where

$$K(t) = \frac{1}{3!} E_x((x-t)_+^3).$$

Now, to determine an explicit form for the kernel $K(t)$, we substitute $(x-t)_+^3$ into equation (1). By doing so, we obtain

$$E_x((x-t)_+^3) = \int_{-1}^{1} (x-t)_+^3 dx - \left(-\frac{1}{\sqrt{3}} - t\right)_+^3 - \left(\frac{1}{\sqrt{3}} - t\right)_+^3$$

To compute the integral of the truncated polynomial, we know that as long as $x < t$, the integral will return zero by definition of the truncated polynomial. So, our lower bound is t instead of -1. By performing the integration, we obtain

$$K(t) = \frac{(1-t)^4}{4} - \left(-\frac{1}{\sqrt{3}} - t\right)_+^3 - \left(\frac{1}{\sqrt{3}} - t\right)_+^3,$$

which is our kernel. So, the error becomes

$$E(f) = \frac{1}{6}\int_{-1}^{1} f^{(4)}(t)\left[\frac{(1-t)^4}{4} - \left(-\frac{1}{\sqrt{3}} - t\right)_+^3 - \left(\frac{1}{\sqrt{3}} - t\right)_+^3\right]dt.$$

Now, we apply the GMVT. However, we must confirm that the kernel doesn't change sign. This is confirmed with the following MATLAB code and plot.

```matlab
%% Exercise 4

t = linspace(−1,1);
K = @(t) (((1−t).^4)/4 − ((−1/sqrt(3) − t).^3).*(−1 ≤ t ≤ −1/sqrt(3))...
    − ((1/sqrt(3) − t).^3).*(−1 ≤ t ≤ 1/sqrt(3)))/6;

plot(t,K(t));
```

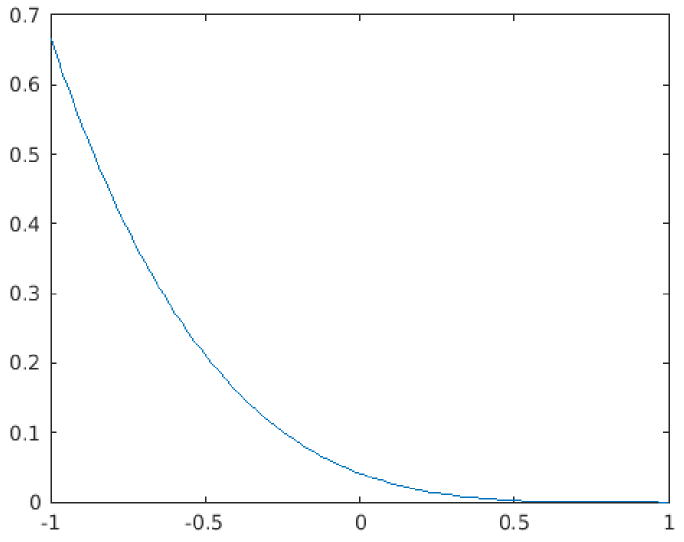

Figure 1: A plot of the Kernel. It indeed does not change sign over the interval $[-1, 1]$.

Because the Kernel doesn't change sign, we can apply the GMVT and evaluate the integral:

$$\begin{aligned}
E(f) &= \frac{1}{6}\int_{-1}^{1} f^{(4)}(t)\left[\frac{(1-t)^4}{4} - \left(-\frac{1}{\sqrt{3}} - t\right)_+^3 - \left(\frac{1}{\sqrt{3}} - t\right)_+^3\right]dt \\
&= \frac{f^{(4)}(\xi)}{6}\int_{-1}^{1}\left[\frac{(1-t)^4}{4} - \left(-\frac{1}{\sqrt{3}} - t\right)_+^3 - \left(\frac{1}{\sqrt{3}} - t\right)_+^3\right]dt \quad \text{by the GMVT} \\
&= \frac{f^{(4)}(\xi)}{6}\left[\int_{-1}^{1}\frac{(1-t)^4}{4}dt - \int_{-1}^{-\frac{1}{\sqrt{3}}}\left(-\frac{1}{\sqrt{3}} - t\right)^3 dt - \int_{-1}^{\frac{1}{\sqrt{3}}}\left(\frac{1}{\sqrt{3}} - t\right)^3 dt\right] \\
&= \boxed{\frac{f^{(4)}(\xi)}{135}, \quad \text{where} \quad -1 \le \xi \le 1}.
\end{aligned}$$

I tried to generalize the problem to N nodes, but it was rough. It was difficult because the general error estimate for Gaussian quadrature relies on knowledge of Hermite interpolation, and because I couldn't find a proof for the error of Hermite interpolation, or muster up the courage to do it myself. However, the general error term for Gaussian quadrature is given by

$$E(f) = \frac{f^{(2n)}(\xi)}{(2n)!}\int_{-1}^{1}\pi_n(x)^2\,dx, \quad \text{or} \quad E(f) = \frac{2^{2n+1}\,(n!)^4}{(2n+1)\,[(2n)!]^3}f^{(2n)}(\xi),$$

where $\pi_n(x) = \prod_{i=1}^{n}(x - x_i)$, or the nodal polynomial whose roots are roots of the n-th Legendre polynomial.

EXERCISE 4

4. (20 points) Consider the task of evaluating a double integral

$$Q = \int_{-1}^{1}\int_{-1}^{1} f(x,y)\ dx\ dy$$

for a highly oscillatory integrand. Computing Q is equivalent to computing two one-dimensional integrals both of which can be approximated using Gaussian quadrature. Let $F(y) = \int_{-1}^{1} f(x,y)dx$. Then,

$$Q = \int_{-1}^{1} F(y)\ dy \approx \sum_{n=1}^{N} w_n F(\xi_n),$$

where ξ_n are the Gauss-Legendre nodes and w_n's are the corresponding weights. Now,

$$F(\xi_n) = \int_{-1}^{1} f(x,\xi_n)\ dx \approx \sum_{m=1}^{N} w_m f(\xi_m,\xi_n).$$

Hence

$$Q \approx \sum_{n=1}^{N}\sum_{m=1}^{N} w_n w_m f(\xi_m,\xi_n)$$

Use this scheme with $N = 50$ to approximate Q for $f = \cos(10(x^2+y^2))$. What is the accuracy of this approximation?

Solution: To approximate the integral, I modified my Gaussian Quadrature code in 1-D to accommodate the 2-D computation. The code and its output is below.

```
%% Exercise 4

f = @(x,y) cos(10*(x.^2 + y.^2));
g = @(s) cos(10*s);

I = integral2(f,-1,1,-1,1);
Q = quad_gauss_exam(g,50,eps('double'));
E = I - Q;

I =

  -0.112630611326338

Q =

  -0.112630611326196

```

```
E =

  -1.420807915764044e-13
```

```
function Q = quad_gauss_exam(f,n,tol)

x0 = cos(pi*(0.75 + (0:n-1))/(n + 0.5));
y0 = cos(pi*(0.75 + (0:n-1))/(n + 0.5));

for k = 1:1000
    [p,dp] = legendrep(n,x0);
    x0 = x0 - (p./dp).*(x0.^2 - 1);
    if norm(p,inf) < tol
        break
    end
end

for k = 1:1000
    [p,dp] = legendrep(n,y0);
    y0 = y0 - (p./dp).*(y0.^2 - 1);
    if norm(p,inf) < tol
        break
    end
end

w1 = 2./(dp.^2).*(1 - x0.^2);
w2 = 2./(dp.^2).*(1 - y0.^2);

o = ones(size(x0));
p = (x0'*o).^2 + (o'*y0).^2;

Q = w1*f(p)*w2';
end

function [p,dp] = legendrep(n,x)

if n == 0
    p = ones(size(x));
    dp = zeros(size(x));
elseif n == 1
    p = x;
    dp = ones(size(x));
else
    p1 = x;
    p2 = ones(size(x));
    for m = 2:n
        p = ((2*m - 1)*(x.*p1) - (m-1)*p2)/m;
        p2 = p1;
```

```
            p1 = p;
        end
        dp = n.*(x.*p - p2);
    end
end
```

From my Gaussian Quadrature code above, I computed the nodes for both the x and y variables. Then, because the quadrature method is a double sum, in order to compute the function values, I constructed a matrix A whose elements consisted of $x_i^2 + y_j^2$ for all $i, j = 1, \dots, N$. Then, I evaluated the function over the matrix. Thus, in MATLAB, $Q = w_1 A\, w_2^T$, where $A = p$ and $f = \cos(10s)$ in the code. The relative accuracy of the Quadrature method compared with MATLAB's global adaptive quadrature `integral2` is $\mathcal{O}(10^{-13})$, which is decent (almost double precision), even though the exact value of the integral is

$$\int_{-1}^{1}\int_{-1}^{1} \cos(10(x^2+y^2))\ dx\ dy = \frac{\pi}{5}\text{FresnelC}\left(\frac{2\sqrt{5}}{\sqrt{\pi}}\right)^2 - \frac{\pi}{5}\text{FresnelS}\left(\frac{2\sqrt{5}}{\sqrt{\pi}}\right)^2.$$

The order of accuracy of a quadrature method is the highest order polynomial for which the quadrature method is exact. For one-dimensional Gaussian Quadrature, the accuracy is $2n-1$, which means that given n-nodes, the Gaussian quadrature is exact for polynomials of degree less than or equal to $2n-1$. So, because we are treating the quadrature in two-dimensions as the product of two one-dimensional quadrature routines, we can show that the order of accuracy is $4n-2$, or $2n-1$ for each dimension. For example, for $n = 2$ our quadrature routine is

$$Q(f) = w_1\ v_1\ f(x_1, y_1) + w_1\ v_2\ f(x_1, y_2) + w_2\ v_1\ f(x_2, y_1) + w_2\ v_2\ f(x_2, y_2).$$

The rule $Q(f) = \int_{-1}^{1}\int_{-1}^{1} f(x, y)\ dx\ dy$ is exact if the degree of f in $x \le 2n-1$ and if the degree of f in $y \le 2n-1$[1]. Suppose $f(x, y)$ has degree ≤ 3 in x and y separately. Then, let $q(y) = f(x_1, y)$ is a polynomial in y of degree ≤ 3. So,

$$v_1\ q(y_1) + v_2\ q(y_2) = \int_{-1}^{1} q(y)\ dy.$$

Now, let $r(y) = f(x_2, y)$ be a polynomial in y of degree ≤ 3. So,

$$v_1\ r(y_1) + v_2\ r(y_2) = \int_{-1}^{1} r(y)\ dy.$$

Next let $p(x) = \int_{-1}^{1} f(x, y)\ dy$ be a polynomial in x of degree $\le 2n-1$, so that

$$\int_{-1}^{1} p(x)\ dx = w_1\ p(x_1) + w_2\ p(x_2)$$

[1]Credit of the proof for $n = 2$ to Lawrence A. Fialkow from SUNY:`http://cs.newpaltz.edu/~fialkowl/files/proj04/`

.
Now,

$$\begin{aligned}
Q(f) &= w_1\ v_1\ f(x_1,y_1) + w_1\ v_2\ f(x_1,y_2) + w_2\ v_1\ f(x_2,y_1) + w_2\ v_2\ f(x_2,y_2)\\
&= w_1\ (v_1\ f(x_1,y_1) + v_2\ f(x_1,y_2)) + w_2\ (v_1\ f(x_2,y_1) + w_2\ v_2\ f(x_2,y_2))\\
&= w_1\ (v_1\ q(y_1) + v_2\ q(y_2)) + w_2\ (v_1\ r(y_1) + v_2\ r(y_2))\\
&= w_1\ p(x_1) + w_2\ p(x_2)\\
&= \int_{-1}^{1} p(x)\ dx\\
&= \int_{-1}^{1}\int_{-1}^{1} f(x,y)\ dy\ dx
\end{aligned}$$

So, the general degree of accuracy of this scheme is $2n-1$ for f in x and $2n-1$ for f in y. So, this particular scheme, because $N = 50$, the method terminates polynomials in x and y of degree less than or equal to 99 for each, or a total degree of less than or equal to 198.

EXERCISE 5

5. (20 points) Consider the following IVP on the interval $[0, 20]$*:*

$$\frac{dy}{dt} = y\ (2-y)\ (3-y) + cos(t), \quad y(0) = 0.$$

Solve the IVP using implicit Euler's method with constant step size $h = \frac{1}{2^n}$*, where* $n = 0, \ldots, 6$*. Present well-documented code and the plots.*

Solution: Well, my code is below and the plots are below.

```
%% Exercise 5
clc;
close all;
clear;

rhs = @(t,y) y.^3 - 5*y.^2 + 6*y + cos(t); % rhs of ODE
t0 = 0; % initial time
tf = 20; % final time

% step-sizes
n = linspace(0,6,7);
h = 1./(2.^n);

E = zeros(1,7);

% initial condition
y0 = 0;

% plots
figure
hold on
subplot(4,2,[1 2])
[y,t] = ode_solver(rhs,t0,tf,h(1),y0); % Backward Euler solution
[tt,yy] = ode45(rhs,[t0 tf],y0); % ode45 solution
plot(t,y,'b-'); % plot of my solution
plot(tt,yy,'r.'); % plot of ode45 solution
xlabel('$t$','interpreter','latex');
ylabel('$y$','interpreter','latex');

E(1) = abs(norm(y - yy,'inf'));
% plots with stepsizes h = 0.5 to 0.015625
for k = 0:5
    [y,t] = ode_solver(rhs,t0,tf,h(k+2),y0);
    [tt,yy] = ode45(rhs,[t0 tf],y0);
    subplot(4,2,k+3)
    E(k+2) = abs(norm(y - yy,'inf'));
    hold on
    plot(t,y,'b-');
    plot(tt,yy,'r.');
    xlabel('$t$','interpreter','latex');
    ylabel('$y$','interpreter','latex');
end

% plotting for the order of the method
n = linspace(0,15,16);
h = 1./(2.^n);
E = zeros(1,16);

% computing global error
for k = 1:16
    [y,t] = ode_solver(rhs,t0,tf,h(k),y0);
    [tt,yy] = ode45(rhs,[t0 tf],y0);
    E(k) = abs(norm(y - yy,'inf'));
end

% fit of the log of the step-size and global error
x = abs(log(h(3:end)));
y = abs(log(E(3:end)));
p = polyfit(x,y,1);
xx = linspace(0,11,100);
pp = polyval(p,xx);

% plot of data and the fit
figure
hold on
grid on
plot(xx,pp,'b-');
plot(x,y,'bo');
legend('Fit','Log-Log Data');
title(sprintf("Order = %1.5f",p(1)));

% ODE
function [y,t] = ode_solver(rhs,t0,tf,h,y0)
    t = t0:h:tf; % time vector
    y = zeros(size(t)); % solution vector
    y(1) = y0; % initial condition

    for k = 1:length(t)-1
        ynew = y(k) + rhs(t(k),y(k))*h; % forward euler

        % Using forward euler's and newton's methods to approximate y(n+1)
        % with 100 iterations
        for j = 1:100
            g = ynew - y(k) + h*(ynew^3 -5*ynew^2 + 6*ynew + cos(t(k+1)));
            dg = 1 + h*(3*ynew^2 - 10*ynew + 6);
            ynew = ynew - g/dg;
        end

        % computing the sequence
        y(k+1) = y(k) + h*rhs(t(k+1),ynew);
    end
end
```

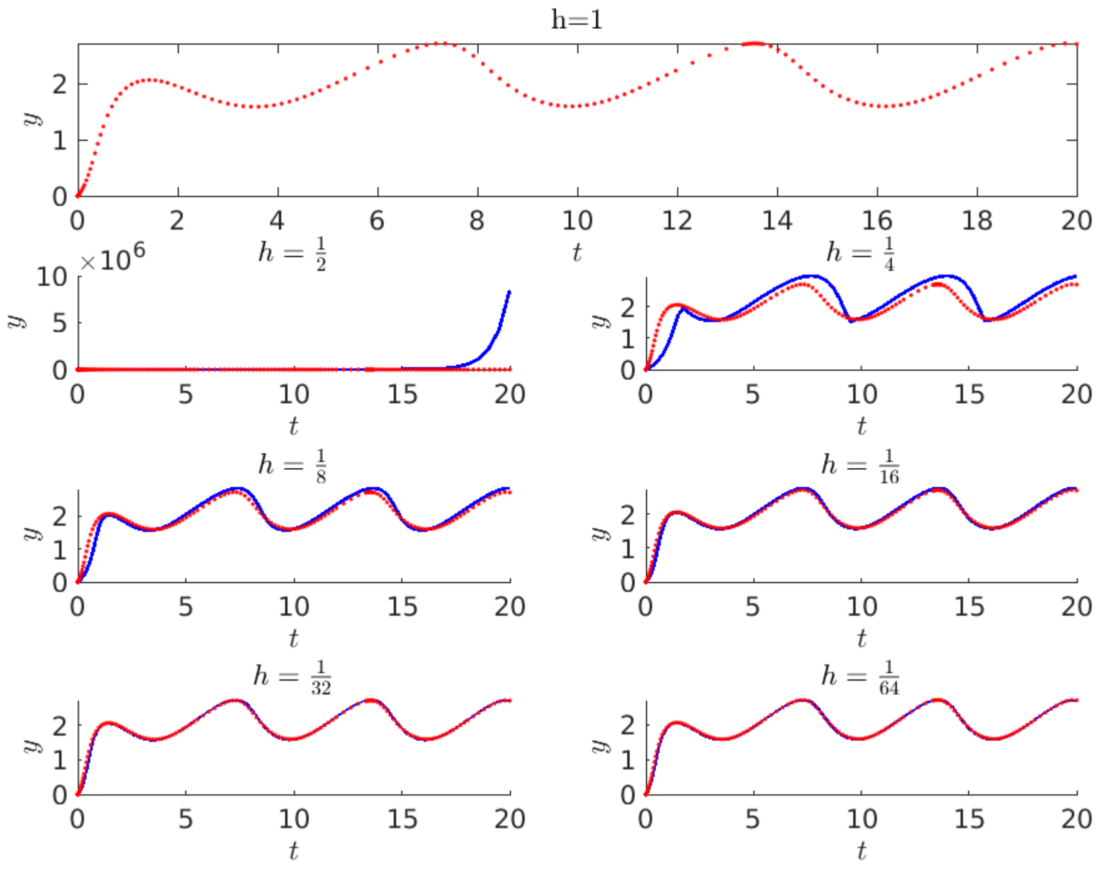

Figure 2: Plots of all of the values of h used for implicit Euler's method.

In the figure above, we can see that for $n = 0$ and $n = 1$ the approximation of the solution to the ODE is pretty bad. However, we can begin to see the solution for $n = 2$. Then, for the last few values, the method converges to the ode45 solution. Further, below is a plot of the order of the accuracy as a log-log plot of the global error against the step size. I removed the first two values of the step size, and thus the global error, because again they were pretty bad. We proved in class that Forward Euler's method converges with an order of 1, and implicit Euler's method also converges with an order of 1. Below is a plot using fourteen points to plot the log of the global error against the log of the step size.

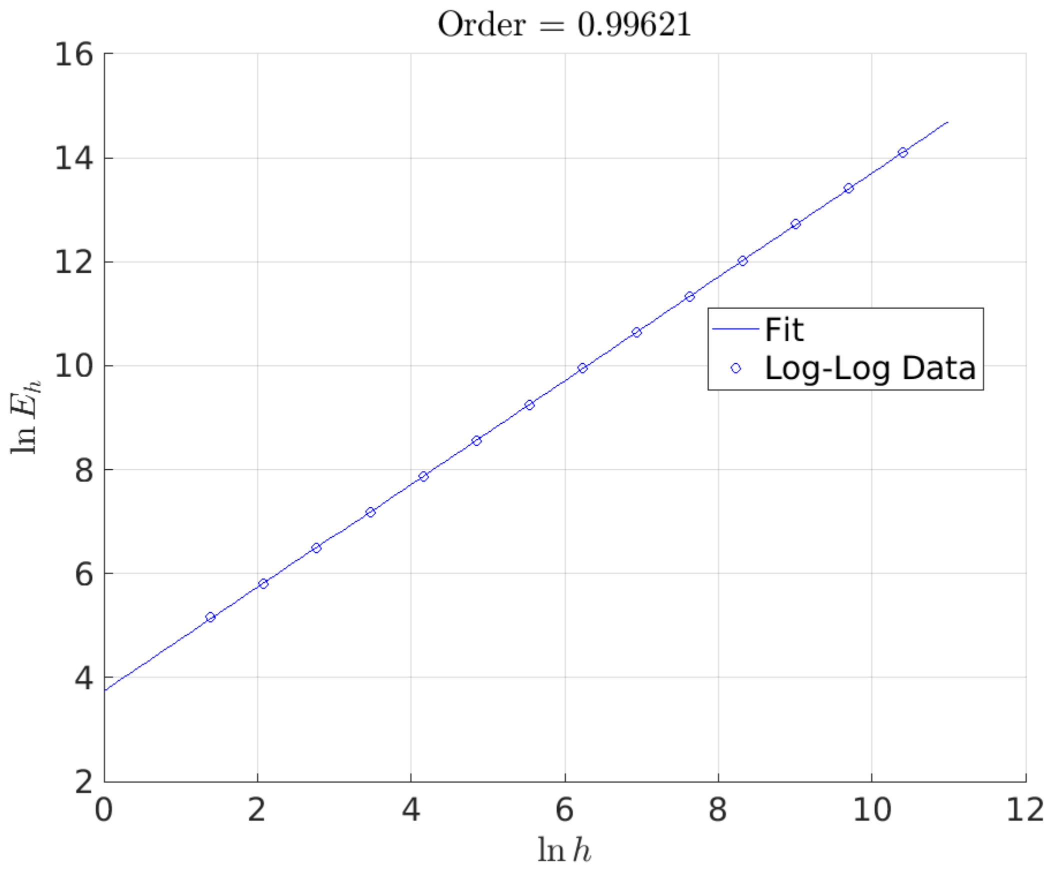

Figure 3: Log-log plot of the global error against the step size. The order is clearly 1.

We can clearly see that the order of implicit Euler's method is indeed 1.