CX 4640 Assignment 8

Wesley Ford

October 26th, 2020

1. In the previous assignment, you constructed a polynomial of degree 6 that interpolates 7 points such that the polynomial approximates the square root function in the interval [0,6]. Now construct a degree 6 least squares polynomial approximation of the square root function in the same interval using Legendre basis functions. Plot this least squares polynomial approximation against the square root function. Also plot the error in the approximation. Compare your results to the results you obtained last week using polynomial interpolation.

   When finding the least square polynomial, we are trying to solve the matrix equation

   $$Bc = b$$

   where the elements of B defined using the basis functions ϕ by

   $$B_{jk} = \int_a^b \phi_j(x)\phi_k(x)dx,$$

   Legendre polynomials form an orthogonal basis, meaning that

   $$\int_{-1}^{1} \phi_j(x)\phi_k(x)dx = \begin{cases} 0 & i \neq k \\ \dfrac{2}{2j+1} & i = j \end{cases}$$

   Changing the basis using $t = \dfrac{(b-a)x + (a+b)}{2}$, this is generalized to

   $$\int_a^b \phi_j(t)\phi_k(t)dt = \begin{cases} 0 & i \neq k \\ \dfrac{b-a}{2j+1} & i = j \end{cases}$$

   So using Legendre polynomials, B becomes a diagonal matrix, with

   $$B_{jj} = \frac{b-a}{2j+1}$$

   where the interval of interpolation is $[a, b]$. On the interval $[0, 6]$ using a degree 6 polynomial

   $$B = \begin{bmatrix} 6 & & & & & & \\ & 2 & & & & & \\ & & 6/5 & & & & \\ & & & 6/7 & & & \\ & & & & 2/3 & & \\ & & & & & 6/11 & \\ & & & & & & 6/13 \end{bmatrix}$$

   The elements of the vector b has elements

   $$b_j = \int_a^b f(x)\phi_k(x)dx$$

   When $f(x) = \sqrt{x}$, and the interval is $[0, 6]$, this becomes

   $$b_j = \int_0^6 \sqrt{x}\phi_k(x)dx$$

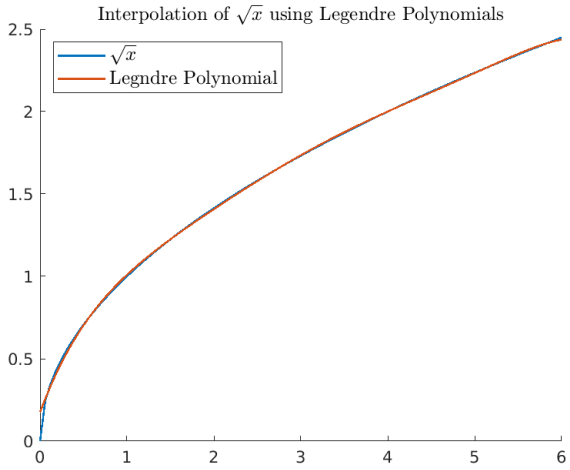
   Using Matlab, we can numerically calculate these values, and solve the Matrix equation $Bc = b$ to get the coefficients for our polynomial. The polynomial that approximates $\sqrt{x}$ over $[0, 6]$ is

   $$\sum_{j=0}^{6} c_j\phi_j(t)$$

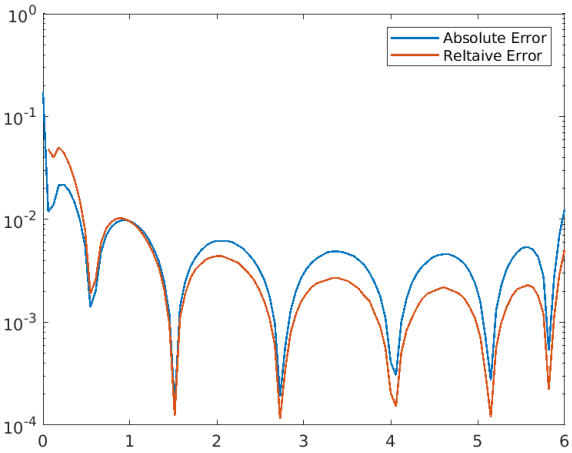
With $t = 3x + 3$, and using Matlab to solve for c, the best approximation 6 degree polynomial of $sqrt(x)$ on the interval $[0, 6]$ using Legendre polynomials is

$$-5.8786 * 10^{-4}t^6 + 0.0119t^5 - 0.0959t^4 + 0.3905t^3 - 0.8787t^2 + 1.4063t + 0.1760$$

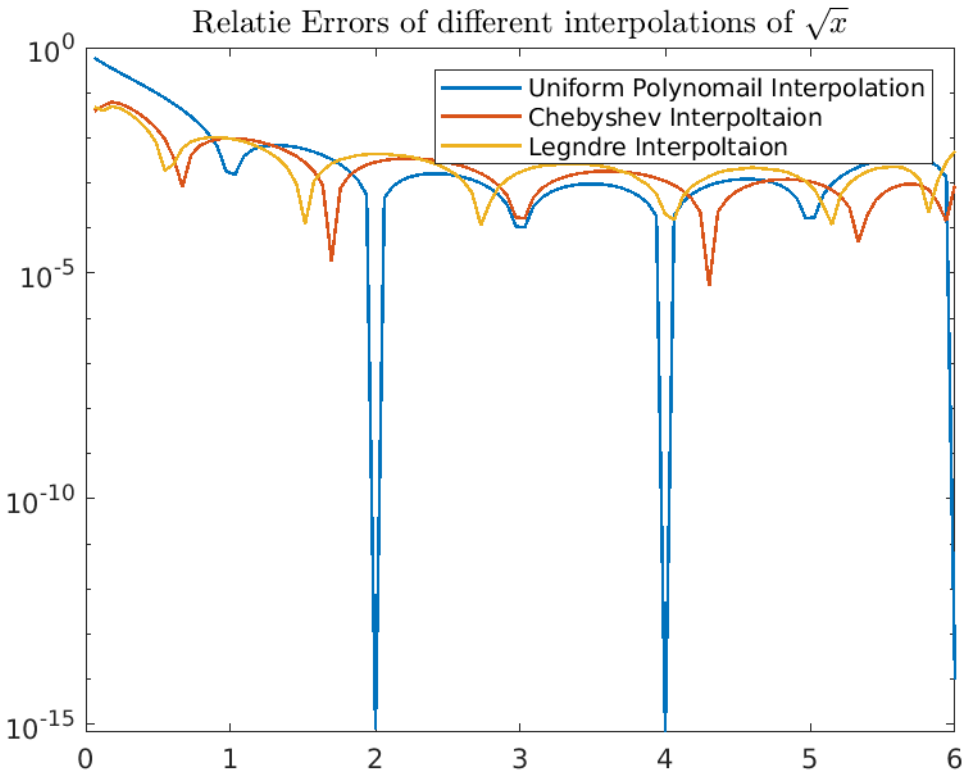
Plotting this polynomial against $\sqrt{x}$, it appears that the polynomial approximates $\sqrt{x}$ well over the interval.



Plotting both the absolute and relative error shows that the polynomial does indeed approximate $\sqrt{x}$ well.



Both errors are quite large near $x = 0$, but drop dramatically to errors on the order of 10^{-2}. Comparing the Legendre interpolation with the interpolations from last week, we can see that it reduces error dramatically compared to using uniform points in the , and performs just as well as using Chebyshev points, if not slightly better in the interval $[0, 1]$.




```
%Code for Q1. Interpolating sqrt(x) using Legendre Polynomials
clf
n=6;
b = zeros(n+1,1);
B = zeros(n);
interval = [0,6];

syms x t;
x = (2*t - interval(1) - interval(2)) / (interval(2) - interval(1));
for i=0:n
f = sqrt(t) * legendreP(i,x);
b(i+1) = int(f, interval);
B(i+1, i+1) = (interval(2)-interval(1)) / (2*i + 1);
end
c = B\b;
v = 0;
for i = 0:n
v = v + c(i+1) * legendreP(i,x);
end
t = linspace(0,6);
vy = subs(v);

hold on
plot(t, sqrt(t), 'LineWidth',1.2);
plot(t, vy, 'LineWidth',1.2);
l = legend('$\sqrt{x}$', 'Legndre Polynomial');
legend('Location', 'northwest');
ti = title('Interpolation of $\sqrt{x}$ using Legendre Polynomials');
```

```
set(l, 'Interpreter', 'latex', 'fontsize',12);
set(ti, 'Interpreter', 'latex', 'fontsize',12);
hold off
diff = abs(sqrt(t) - vy);
plot(t, diff, 'LineWidth', 1.2);
hold on
plot(t, diff ./ sqrt(t), "LineWidth", 1.2);
legend('Absolute Error', "Reltaive Error")
hold off

%Use last weeks hw code to get values for polyvals
x = 0:6;
y = sqrt(x);
P1 = polyfit(x,y,6);
%Compute chebyshev points for [-1,.1], then add 1 and scale by 3 to get nodes for
%the interval [0,6]

chebyX = cos((2*x + 1) / (2*6+2) * pi);
chebyX = 3 * (chebyX+1);
chebyY = sqrt(chebyX);
P2 = polyfit(chebyX, chebyY, 6);

diffP1 = abs(sqrt(t) - polyval(P1, t));
diffP2 = abs(sqrt(t) - polyval(P2, t));

plot(t, diffP1 ./ sqrt(t), 'LineWidth', 1.2);
hold on
plot(t, diffP2 ./ sqrt(t), 'LineWidth', 1.2);
plot(t, diff ./ sqrt(t), 'LineWidth', 1.2);
legend("Uniform Polynomail Interpolation", "Chebyshev Interpoltaion", "Legndre Interpolt
ti = title("Relatie Errors of different interpolations of $\sqrt{x}$");
set(ti, 'Interpreter', 'latex', 'fontsize',12);
```

2. In this problem you will investigate the imaginary forward difference:

$$f'(x) \approx Im\left(\frac{f(x+ih)-f(x)}{h}\right)$$

Here, $Im(z)$ is the imaginary part of a complex number z.

(a) What is the order of the approximation?

Using the Taylor expansion at x to rewrite $f(x+ih)$.

$$f(x+ih) = f(x) + ihf'(x) + \frac{(ih)^2}{2}f''(x) + \frac{(ih)^3}{6}f'''(\xi)$$

$$f'(x) \approx Im\left(\frac{f(x) + ihf'(x) + \frac{(ih)^2}{2}f''(x) + \frac{(ih)^3}{6}f'''(\xi) - f(x)}{h}\right)$$

$$f'(x) \approx Im\left(\frac{ihf'(x) - \frac{h^2}{2}f''(x) - \frac{ih^3}{6}f'''(\xi)}{h}\right)$$

$$f'(x) \approx f'(x) - \frac{h^2}{6}f'''(\xi)$$

We can see that the approximation has a truncation error of $\frac{h^2}{6}f'''(\xi)$, so the order of the approximation is $O(h^2)$

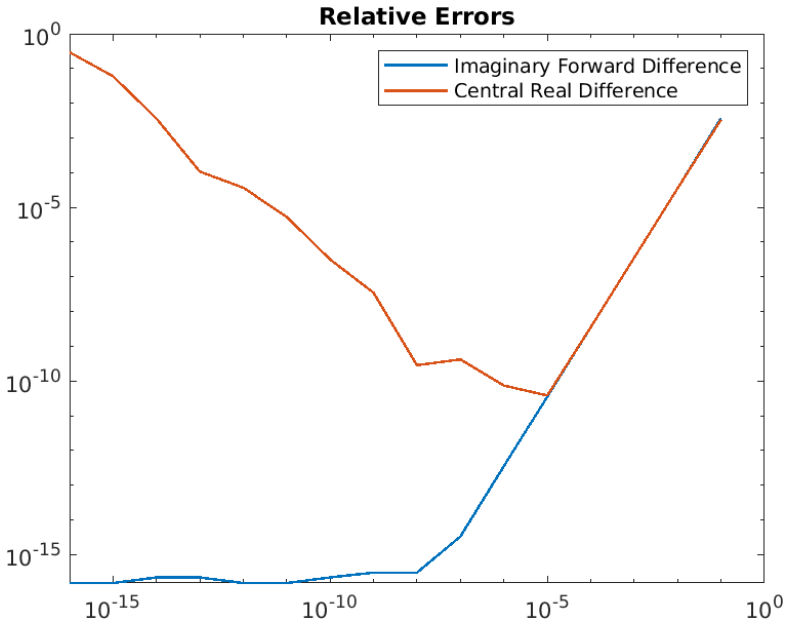
(b) Consider $f(x) = -e^{2x}\sin(\pi x)$. What is the exact value of $f'(1)$?

$$f'(x) = \frac{d}{dx} - e^{2x}\sin(\pi x) = -2e^{2x}\sin(\pi x) - \pi e^{2x}\cos(\pi x)$$

$$f'(1) = \pi e^2$$

(c) Implement the imaginary forward difference and the real central difference for $f'(x)$ for the above function $f(x)$ at $x = 1$.

(d) Plot the relative errors of the two methods for `h = logspace(-16, -1, 16)`. If an error is equal to zero, set the error to `eps`. Describe the features of the graph in detail and explain what causes them.



We can see that for $h < 10^{-5}$, the relative error the Real Central Difference (RCD) introduces into the calculation increases as the values of h get smaller. At the smallest value of h, 10^{-16}, the RCD produces relative errors near 1! (i.e. nowhere near the real answer). This is because of the subtractive cancellation introduced by the RCD formula $\left(f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}\right)$. As each step size decreases, $f(x+h) \approx f(x-h)$ causing subtractive cancellation. The Imaginary Forward Difference (IFD) does not suffer from this issue is as long as $f(x)$ is real when evaluating the derivative at x. Because IFD only takes the Imaginary components of $\frac{f(x+ih) - f(x)}{h}$, the $f(x)$ term is not used when calculating the approximation and thus cannot cause subtractive cancellation. Because of this, IFD continues to get more accurate as h decreases until the accuracy is limited by finite precision. For $h > 10^{-5}$, $f(x+h) \not\approx f(x-h)$, and so subtractive cancellation errors are insignificant compared to the truncation errors.

(e) What are the pros and cons of using the imaginary forward, the RCD difference?
A pro of using imaginary forward is that for it does not suffer from subtractive cancellation like RCD does, and thus can get extremely good approximates for $f'(x)$. A con is that to implement imaginary forwarding, the software used must be able to handle imaginary computations. RCD on the other hand can be implemented without software to handle imaginary computations, and still performs well. In the example above, it was still able to approximate the derivative to a relative error of 10^{-10}. If the precision of calculations does not need to be better than that, RCD can perform just as well as imaginary forwarding without a computer that can handle imaginary computations.

```
%Code for Question 2 HW8
clf;
clear;
f = @(x) -exp(2.*x) .* sin(pi .* x);

%Part C
%Implementing imaginary forward difference
ifd = @(x,h) imag((f(x+ 1i.*h) - f(x)) / h);

%real central difference
```

```
rcd = @(x,h) (f(x+h) - f(x-h)) / (2.*h);


%Part D
real = pi * exp(2);

x=1;
h = logspace(-16, -1, 16);
ifd_err = zeros(1,16);
rcd_err = zeros(1,16);
for j=1:length(h)
        ifd_val = ifd(x,h(j));
        rcd_val = rcd(x,h(j));
        ifd_err(j) = abs(real - ifd_val) / real;
        rcd_err(j) = abs(real - rcd_val) / real;
end
ifd_err(ifd_err == 0) = eps;
rcd_err(rcd_err == 0) = eps;
loglog(h, ifd_err, h, rcd_err, 'Linewidth', 1.2);
legend('Imaginary Forward Difference', 'Central Real Difference');
title("Relative Errors")
```