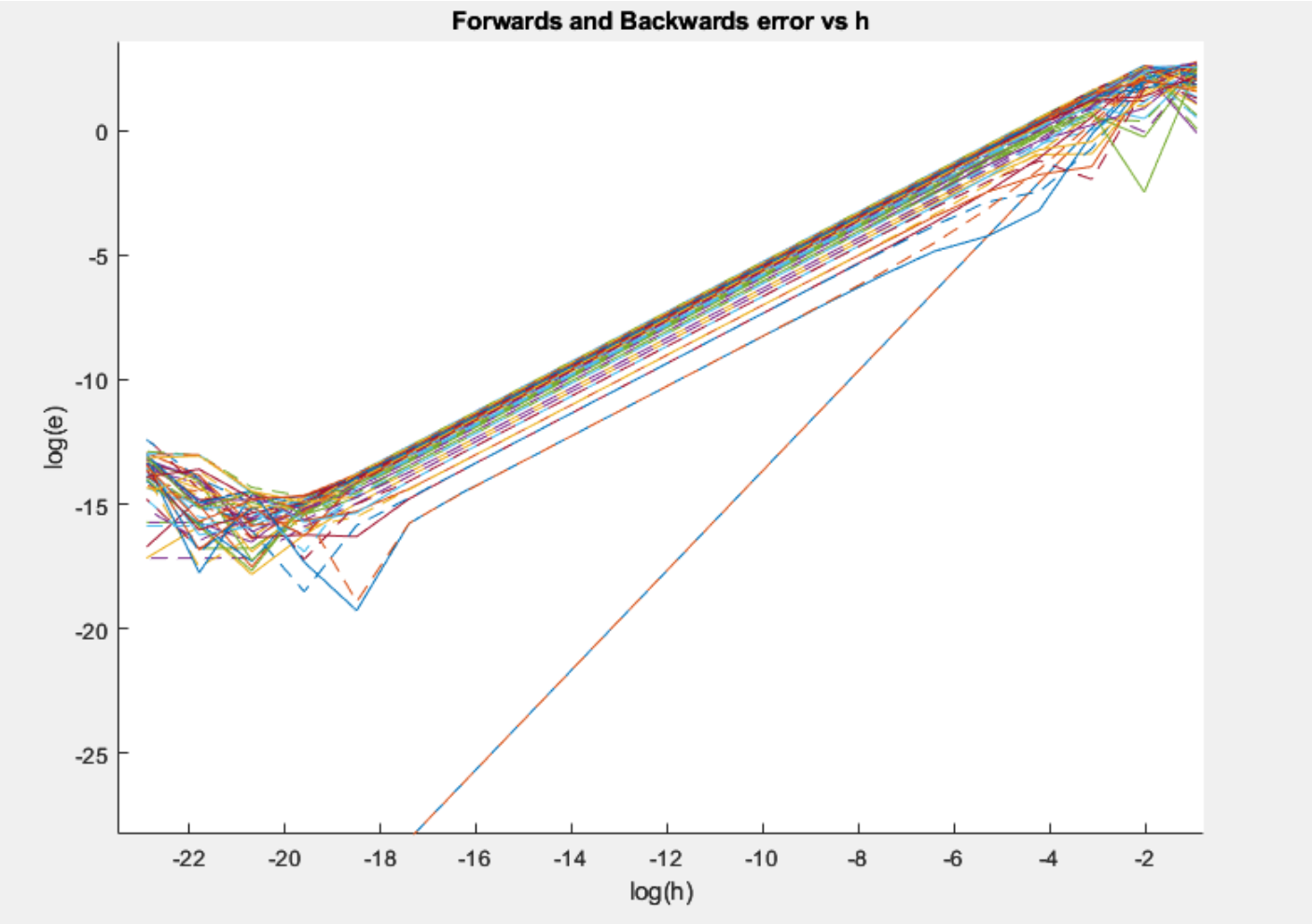
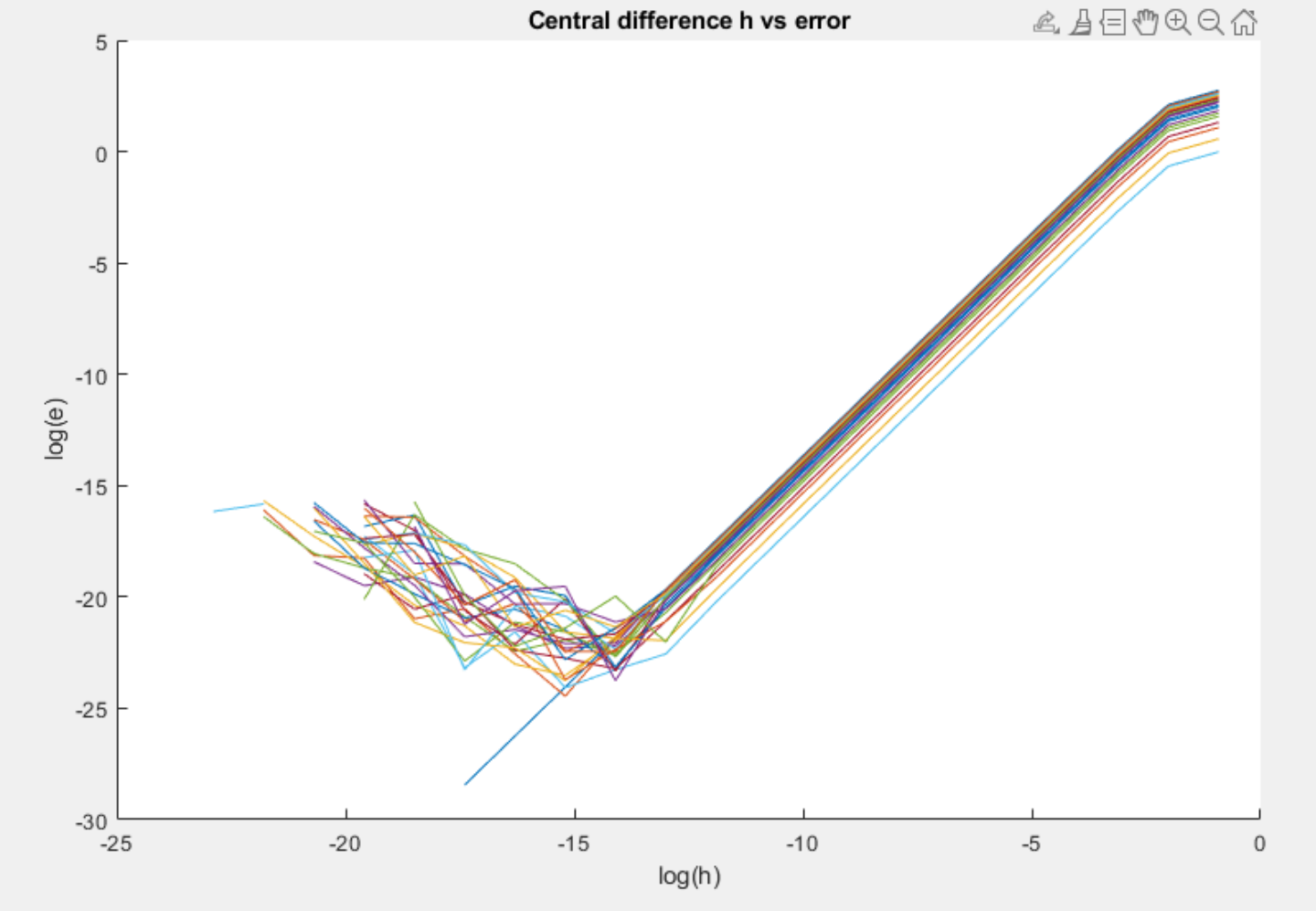
NumComp - Fall 2022
Project #10

Due –
Isaiah Thomas

Plots!




Central difference h vs error
log(e)
log(h)
5
0
-5
-10
-15
-20
-25
-30
-25
-20
-15
-10
-5
0

1. Identify the asymptotic regime to estimate the convergence rate of each method. Do the rates depend on x?

Asymptotic regimes are pretty visible from the plots. About $(10^{-18}, 10^{-}4)$ for forwards and backwards, about $(10^{-13}, 10^{-}3)$ for central. Respective lines in those ranges have all the same slope p and therefore the same rate of convergence (besides one x for forwards and backwards).

First, both forwards and backwards plots appear to be quite similar. I could not tell a difference between the methods by studying the values of x, h against their error

Second, asymptotic regime seems to be minimally impacted by most values of x. However, some values of x appear to have a slightly smaller asymptotic regime, giving it a smaller initial error value, moving it slightly down the plot.

There is one strange value of x that behaves strangely against all others. This value appears to increase convergence rate and asymptotic regime for forwards and backwards methods, and only asymptotic regime for the central method. After some reworking of plotting code, I found this value of x to be zero.

I suppose this makes sense, as where $x = 0$ in our original function, the slope is maximized and the most unchanging. I'd have to assume the value of the second derivative, and perhaps maybe all derivatives, impact the convergence rate of the error. Perhaps maybe as the value of second derivative approaches decreases, the rate of convergence and regime does as well. It could also make sense that the central method's rate of convergence is less affected by this idea, as it is a more accurate approximation of the derivative.

These ideas generally stem from the fact that the accuracy of a Taylor series is based on how many derivations values it composes.

Last thought: JU$T is by far RTJ's best song. Pharrell and Zach de la Rocha?? Stacked. Kill your slave masters.

Also One is easily Paco's food song. Mathematically significant and the song is probably about a turtle / tortoise or something? Maybe its about a maimed and tortured soldier. Maybe its both.

Code!

```
function m = approxdrv(xi, h)
    %used for later substitution of created quadratic
    syms x;

    %kept this variable to extend range of plotted polynomial if needed
    plotmoreh = h*10;

    %compute y values at each needed point relative to h and x
    %fm is the matlab function of the given sin function
    %   function y = fm(x)
    %       y = sin(4.8 * pi * x );
    %   end
    fxph = fm(xi + h);
    fxmh = fm(xi - h);
    fxc = fm(xi);

    %put in answer matrix
    b = [fxmh;fxc;fxph];

    %fwd and bwd derivative approx and true derivate calc
    fwddf = (fxph - fxc)/h;
    bwddf = (fxc - fxmh)/h;
    %df is the matlab function of the handcalculated derivative function
    %      function y = df(x)
    %           coe = 4.8 * pi;
    %           y = coe * cos(coe * x);
    %      end
    trueder = df(xi);

    %fitting a quadratic to the 3 points below
    vquad = fliplr(vander([xi - h, xi, xi + h]));
    %coe of the quadratic
    quadcoe = vquad \ b;

    %derivative of quad
    quadder = polyder(fliplr(transpose(quadcoe)));

    %adding symbolic x and exponents
    quadderpoly = newpoly(flip(transpose(quadder)));
```

```
    %substitute x into the quadratic give out central derivate approx
    cdf = subs(quadderpoly,x,xi);

    %answer set
    m = [fwddf; bwddf; cdf; trueder];
end
```

```
function xhpair = main(hintervalexp, xrange, xpoints,
                       hprecision, hstart,maxit)

    %range of xs to use and interval
    xlow = xrange(1);
    xhigh = xrange(2);
    xstep = (xhigh - xlow)/xpoints;

    k = 1;
    for i = xlow:xstep:xhigh
        disp(i);
        hit = 1;
        hcur = hstart;

        %run through a whole bunch of hs until either precision is met or
        %   max iterations
        while true
            if (hcur < hprecision || hit > maxit )
                disp(hit);
                break;
            end
            %gather all approximates and truths for current x and h
            curapprox = approxdrv(i, hcur);

            truthd(hit) = curapprox(4);

            %store hs
            hs(hit) = hcur;

            %assuming error formula is abs(true - approx)
            cerror(hit) = abs(truthd(hit) - curapprox(3));
            berror(hit) = abs(truthd(hit) - curapprox(2));
            ferror(hit) = abs(truthd(hit) - curapprox(1));

            %reduce h by some amount
            hcur = hcur/hintervalexp;
            hit = hit + 1;
        end
        errorset = {hs, {cerror, berror, ferror}};
        xhpair{k} = {i,errorset};
        %data structure below
```

```
        %oh boy, why am i like this
        %tuples galore
%                    xhpairs
%                      /\
%              xi            hs+error
%                             /\
%                   htested     herrorset
%                                /   |   \
%                               cent  bwd   fwd
%
%
%       tree{xpoints}{ [1=x] [2=next tuple] }{ [1 = hs][2 = error sets] }
%       i mean...
        k = k + 1;
    end
end
```

```
function out1 = plotfun(tuple)
    %plots backwards and forwards
    %index x values
    %change step value for amount of lines
    for i = 1:1:length(tuple)
        disp(i);
        hold on;
        %identifying anomolous x value
        if( log(min(tuple{i}{2}{2}{2})) < -25)
            plot(log(tuple{i}{2}{1}), log(tuple{i}{2}{2}{2}), Color='Red');
            plot(log(tuple{i}{2}{1}), log(tuple{i}{2}{2}{3}), Color='Red');
            out1 = tuple{i}{1};
        else
        plot(log(tuple{i}{2}{1}), log(tuple{i}{2}{2}{2}));
        plot(log(tuple{i}{2}{1}), log(tuple{i}{2}{2}{3}), LineStyle="--");
        end
    end


    hold off;
end

function out1 = plotfun2(tuple)
    %plots central difference
    %index x values
    %change step value for amount of lines
    for i = 1:1:length(tuple)
        disp(i);
        hold on;

        if( log(min(tuple{i}{2}{2}{1})) < -25)
            plot(log(tuple{i}{2}{1}), log(tuple{i}{2}{2}{1}), Color='Red');
            out1 = tuple{i}{1};
        else
        plot(log(tuple{i}{2}{1}), log(tuple{i}{2}{2}{1}), LineStyle="-.");
        end
    end


    hold off;
end
```

Great topic, I disgust myself with my code. Slowly learning matlab plotting lol