$$f(x) = \sinh x = \frac{e^x - e^{-x}}{2} \qquad \left\{ \cosh x = \frac{e^x + e^{-x}}{2} \right.$$

Let's expand this in a Taylor series about the point $x_0 = 0$.

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)h^2}{2!} + \frac{f'''(x_0)h^3}{3!} + \cdots$$

Letting $x = x_0 + h$, we have with $x_0 = 0$

$$x = h$$

$$\therefore f(x) = f(x_0) + f'(x_0)x + \frac{f''(x_0)x^2}{2!} + \frac{f'''(x_0)x^3}{3!} + \cdots$$

Now, to determine the coeffs, we need to compute the derivatives

$$f(x) = \sinh x \longrightarrow f(x_0) = 0$$

$$f'(x) = \cosh x \longrightarrow f'(x_0) = 1$$

$$f''(x) = \sinh x \longrightarrow f''(x_0) = 0$$

$$f'''(x) = \cosh x \longrightarrow f'''(x_0) = 1$$

$$\vdots$$

$$\therefore f(x) = \sinh x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots$$

$$f(x) = \sum_{n=1}^{\infty} \frac{x^{2n-1}}{(2n-1)!}$$

Now how do we efficiently compute infinite series of this form:

write as a recurrence relation

$$f(x) = \sum_{n=1}^{\infty} T_n(x)$$

with $T_{n+1} = r_n T_n$ where $r_n = \frac{T_{n+1}}{T_n}$

(I can do this for power series)

Algorithm

1. set maxt = max # of terms
   tol = criterion for stopping scale
2. set $n = 1$ and $T = T_1$ initialize counter and first term in series
   set $f = T$ initialize partial sum and one term
   $\epsilon = 1$
3. while $\epsilon > tol$ and $n < maxt$
   compute $r = \frac{T_{n+1}}{T_n}$ (specific to function of interest)
   $T = r * T$ compute next term
   $f = f + T$ update partial sum
   $\epsilon = max(abs(T/f))$ max relative change due to current term
   $n = n + 1$ increment counter
   end

here $T_1 = x$

$$r_n = \frac{T_{n+1}}{T_n} = \frac{x^{2(n+1)-1}}{(2(n+1)-1)!} \times \frac{(2n-1)!}{x^{2n-1}}$$

x^{2n+2-1} — $(2n+1)!$

$$= \frac{x^2 x^{2n-1}}{(2n+1)(2n)(2n-1)!} \quad \frac{(2n-1)!}{x^{2n-1}} = \frac{x^2}{(2n+1)(2n)}$$

see code sinh-series.m

note

for $x = 2$ and $n = 10$

$$\frac{x^{2n-1}}{(2n-1)!} = \frac{2^{19}}{19!} = \frac{524288}{1.21645 \times 10^{17}}$$

$$= 4.30998 \times 10^{-12}$$

want to avoid calcs like this because of round-off error

for $x = 2$ and $n = 9$

$$\frac{x^{2n-1}}{(2n-1)!} = \frac{2^{17}}{17!} = \frac{131072}{3.55687 \times 10^{14}}$$

$$= 3.68503 \times 10^{-10}$$

---

for $x = 8$ and $n = 10$

$$\frac{8^{19}}{19!} = \frac{1.44115 \times 10^{17}}{1.21645 \times 10^{17}} = 1.18472$$

```
%
%   SINH_SERIES.M   Evaluate sinh(x) using Taylor Series
%
%   Inputs:
%     x - vector of independent variable values
%     maxT - maximum number of terms in series (optional, default value = 10)
%     tol - error tolerance for truncating series (optional, default value = 0.0001)
%
%   Outputs:
%     f - value of sinh(x) evaluated at all x values
%
%   File prepared by J. R. White, UMass-Lowell (last update: Oct. 2017)
%

     function f = sinh_series(x,maxT,tol)
%
%  set defaults if no inputs for maxT or tol, and check on minimum value for tol
     if nargin == 1, maxT = 10; tol = 0.0001;  end
     if nargin == 2,  tol = 0.0001;  end
     if isempty(maxT) || maxT < 1, maxT = 10;  end
     if tol < 100*eps, tol = 100*eps;  end
%
%  initialize variables and perform computational loop
     T = x;  f = T;   rerr = 1.0;  n = 1;
     while rerr > tol && n < maxT
       r = x.*x/((2*n+1)*2*n);
       T = r.*T;        % next termin series
       f = f + T;       % update running sum
       i = find(f);     % finds indices of nonzero values of f
       rerr = max(abs(T(i)./f(i))); % calc max relative error
       n = n+1;
     end
%
%  display warning if hit max # of terms
     if n == maxT
       disp(' ')
       disp('  *** WARNING ***  Hit max # of terms in sinh_series.m')
       disp(' ')
     end
%
%   end of function
```

ts_deriv_2

10.317 Applied Problem Solving with Matlab

A Short Quiz on
Working with Taylor Series -- Derivative Approximations

A Taylor series expansion for the functions f(x+h) and f(x–h) can be written in terms of the function f(x) and all its derivatives evaluated at the point x, where $h = |\Delta x|$. Using a discrete notation (i.e. f_i, f_{i+1}, etc.), for convenience, these are given as follows:

$$f_{i+1} = f_i + f_i' h + \frac{f_i'' h^2}{2!} + \frac{f_i''' h^3}{3!} + \frac{f_i'''' h^4}{4!} + \cdots \qquad \text{(Forward Taylor Series)} \qquad (1)$$

and

$$f_{i-1} = f_i - f_i' h + \frac{f_i'' h^2}{2!} - \frac{f_i''' h^3}{3!} + \frac{f_i'''' h^4}{4!} - \cdots \qquad \text{(Backward Taylor Series)} \qquad (2)$$

a. Now, using the expressions from above, **derive a central** difference approximation to df/dx (i.e. the **first derivative**) at the point x_i, and obtain an estimate of the order of error. This should be a formal development!

Subtract eqn (2) from (1)

← Terms "on the order of" h^3

$$f_{i+1} - f_{i-1} = 2 f_i' h + \mathcal{O}(h^3)$$

→ solving for f_i' gives

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + \mathcal{O}(h^2)$$

↑ This is the derivative approximation used in practice. It is referred to as the "central" approximation, since information on both sides of point i is used to compute f_i'.

↑ This is the order of error that is $\epsilon = \alpha h^2$

error is proportional to the square of the step size

∴ The approximation is referred to as 2nd order

b. An object falling through a viscous medium, where the friction force is proportional to the velocity, can be modeled with the following ODE:

$$m\frac{dv}{dt} = mg - kv \qquad \text{or} \qquad \frac{dv}{dt} = g - cv \qquad \text{where } c = \frac{k}{m}$$

In this equation, c is the drag coefficient per unit mass (with units of s^{-1}, for example) and $g = 9.8\ m/s^2$ is the gravitational acceleration.

In a particular situation the position of an object versus time was measured over a period of 4 seconds at 1 second intervals as noted in the table below:

Table I Experimental Data – Position vs Time

i	t_i (sec)	y_i (m)
1	0	0.00
2	1	4.18
3	2	14.4
4	3	28.4
5	4	44.5

Noting that velocity is the rate of change of position, v = dy/dt, use these measured data to estimate the drag coefficient, c, in the above mathematical model. Explain your solution logic and support your results with the appropriate computations. **Hint:** Evaluate the ODE at time point i = 3 and solve for c.

Evaluating the ODE at point i gives

$$\left.\frac{dv}{dt}\right|_i = g - c v_i \qquad \text{or} \qquad c = \frac{1}{v_i}\left(g - \left.\frac{dv}{dt}\right|_i\right)$$

using the central approximation from Part a, we can compute the object's velocity at points 2, 3, and 4.

$$v_i = \frac{y_{i+1} - y_{i-1}}{2\Delta t} \quad \text{for } i = 2, 3, 4$$

$$v_2 = \frac{y_3 - y_1}{2\Delta t} = \frac{14.4 - 0}{2} = 7.2 \text{ m/s}$$

$$v_3 = \frac{y_4 - y_2}{2\Delta t} = \frac{28.4 - 4.18}{2} = 12.1 \text{ m/s}$$

$$v_4 = \frac{y_5 - y_3}{2\Delta t} = \frac{44.5 - 14.4}{2} = 15.1 \text{ m/s}$$

We can also compute the acceleration at i = 3 as

$$\left.\frac{dv}{dt}\right|_i = \frac{v_{i+1} - v_{i-1}}{2\Delta t}$$

$$\left.\frac{dv}{dt}\right|_3 = \frac{v_4 - v_2}{2\Delta t} = \frac{15.1 - 7.2}{2} = 3.95 \text{ m/s}^2$$

$$c = \frac{1}{v_3}\left(g - \left.\frac{dv}{dt}\right|_3\right)$$

$$c = \frac{9.8 - 3.95}{12.1}$$

$$c = 0.48\ s^{-1}$$

and

```
>> ts_deriv_1

   Polynomial of interest is: 2x^4 + 5x^3 - x + 1
   The exact derivative at x = 1.5 is :    59.750

   Approximate Derivatives at x = 1.5
         h      Backward Approx   Rel Error        Central Approx  Rel Error
      1.0e-04       59.745       8.284e-05           59.750       2.845e-09
      1.0e-03       59.701       8.282e-04           59.750       2.845e-07
      1.0e-02       59.257       8.256e-03           59.752       2.845e-05
      1.0e-01       54.968       8.003e-02           59.920       2.845e-03
      1.0e+00       25.250       5.774e-01           76.750       2.845e-01
```

Backward Approx:

$h = 10^{-2} \rightarrow 10^{-3}$

$n = \log(\varepsilon_1/\varepsilon_2)$

$= \log\left(8.256\times10^{-3} / 8.282\times10^{-4}\right)$

$= 0.999 \approx 1.0$

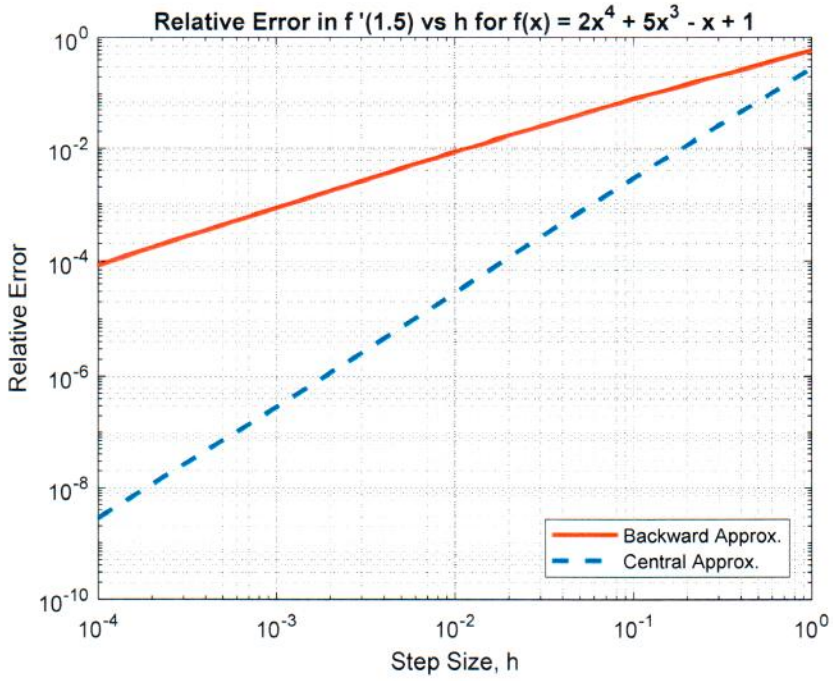
Central Approx:

$h = 10^{-2} \rightarrow 10^{-3}$

$n = \log(\varepsilon_1/\varepsilon_2)$

$= \log\left(2.845\times10^{-5} / 2.845\times10^{-7}\right)$

$= 2.0$



Notes:

Since $\varepsilon = \alpha h^n$, then $\log\varepsilon = \log\alpha + n\log h$. Thus, n is the slope of the curve on a log-log plot.

From the plot, we see that the **backward** approximation has a slope of unity (i.e. a one decade change in step size leads to about a one decade change in relative error -- thus, $n_{backward} \rightarrow 1$.

For the **central** approximation, the slope of the log-log curve is just about a factor of two larger (i.e. a one decade change in h gives a two decade change in ε -- thus, $n_{central} \rightarrow 2$.

And, of course, from the development in the notes, this is exactly what we expected!!!

Also, note that, for very large h, this approximate theory starts to break down (error is just too large when $h \approx 1$ for this problem)…

From a more quantitative view, we also have the following relationships:

$$\frac{\varepsilon_1}{\varepsilon_2} = \left(\frac{h_1}{h_2}\right)^n \qquad \text{or} \qquad n = \frac{\log(\varepsilon_1/\varepsilon_2)}{\log(h_1/h_2)}$$

and, **for a change in h of a factor of 10**, n is given by $n = \log(\varepsilon_1/\varepsilon_2)$ [**see above calculations**]

```
%
%   TS_DERIV_1.M   Approximation to the 1st Derivatives of a Simple Function
%
%   This file addresses the error associated with two different approximations
%   to the 1st derivative of a simple polynomial function:
%     f(x) = 2x^4 + 5x^3 - x + 1
%     f'(x) = 8x^3 + 15x^2 - 1   (exact representation)
%   The error versus step size is plotted.
%
%   In evaluating this function, we will use Matlab's description of a
%   polynomial as a vector of coefficients and Matlab's polyval function
%   for efficiently evaluating the polynomial.
%
%   File prepared by J. R. White, UMass-Lowell (last update: Oct. 2017)
%

     clear all,  close all,   nfig = 0;
%
%   define polynomial functions and evaluate the 1st derivative at x = 1.5
     fxc = [2 5 0 -1 1];   fpxc = [8 15 0 -1];
     x = 1.5;  fpx = polyval(fpxc,x);
     fprintf(1,'\n');
     fprintf(1,'   Polynomial of interest is: 2x^4 + 5x^3 - x + 1 \n');
     fprintf(1,'   The exact derivative at x = 1.5 is :  %8.3f \n\n',fpx);
%
%   evaluate the 1st derivative using backward & central approximations
     Nh = 5;  h = logspace(-4,0,Nh);
     fpxback = (polyval(fxc,x) - polyval(fxc,x-h))./h;
     fpxcent = (polyval(fxc,x+h) - polyval(fxc,x-h))./(2*h);
%
%   compute relative errors
     reback = abs((fpxback-fpx)/fpx);   recent = abs((fpxcent-fpx)/fpx);
%
%   tabulate results
     fprintf(1,'   Approximate Derivatives at x = 1.5  \n');
     fprintf(1,'         h      Backward Approx   Rel Error        Central Approx  Rel↙
Error \n');
     for i = 1:Nh
       fprintf(1,'  %12.1e     %8.3f    %12.3e          %8.3f    %12.3e \n', ...
             h(i),fpxback(i),reback(i),fpxcent(i),recent(i));
     end
     fprintf(1,'\n');
%
%   plot results
     nfig = nfig+1;   figure(nfig)
     loglog(h,reback,'r-',h,recent,'b--','LineWidth',2), grid
     title('TS\_Deriv\_1:  Relative Error in f ''(1.5) vs h for f(x) = 2x^4 + 5x^3 - x +↙
1');
     xlabel('Step Size, h'),ylabel('Relative Error')
     legend('Backward Approx.','Central Approx.','Location','SouthEast')
%
%   end of program
```