4.6 Skip part of this

Scalar equations: $f: \mathbb{R} \to \mathbb{R}$ Systems: $F: \mathbb{R}^n \to \mathbb{R}^n$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

even this might be difficult to compute

$$x_{k+1} = x_k - \left(DF(x_k)\right)^{-1} F(x_k)$$

difficult to compute

assuming $x_n \to r$

$$f'(r) \approx f'(x_n) \approx \frac{f(x_n+h) - f(x_n)}{h}$$

got better as $x_n \to r$

this approximation depends on h and doesn't get better if h is fixed.

Idea:

$$x_{n+1} = x_n - \frac{f(x_n)\, h}{f(x_n+h) - f(x_n)}$$

since $x_n \to r$ then $x_{n-1} - x_n \to 0$

h

secant method

$$x_{n+1} = x_n - \frac{f(x_n)(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}$$

same as inverse interpolation on 2 point

Point of view is

$$f'(x_n) \approx \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}$$

don't need to know f' to compute.

approximate Newton's method by approximating the derivative.

$$x_{k+1} = x_k - \left(DF(x_k)\right)^{-1} F(x_k)$$

Try to approximate $DF(x_k)$ ... to get a Quasi-Newton method for systems...

only update the derivative every 3 steps...

$$x_{k+1} = x_k - \left(DF(x_k)\right)^{-1} F(x_k) \qquad \text{for } k = 3\ell$$

$$x_{k+2} = x_{k+1} - \left(DF(x_k)\right)^{-1} F(x_{k+1})$$

↑ approximation of $\left(DF(x_{k+1})\right)^{-1}$

maybe 3x faster if most of the time was spent computing DF each iteration...

$$x_{k+3} = x_{k+3} - \left(DF(x_k)\right)^{-1} F(x_{k+3})$$

↑ approximation of $\left(DF(x_{k+2})\right)^{-1}$

Note that as $x_k \to r$ then $DF(x_k) \approx DF(r)$

But still compute the derivative once in a while!

$$DF(x)e_j = \underbrace{\mathbf{J}(\mathbf{x})}_{DF(x)}\mathbf{e}_j \approx \frac{\mathbf{f}(\mathbf{x}+\delta\mathbf{e}_j) - \mathbf{f}(\mathbf{x})}{\delta}, \qquad j = 1, \ldots, n.$$

↑ means the j^{th} column.

$$Df(x) = \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \cdots & \partial f_1/\partial x_m \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & & \vdots \\ \vdots & & \ddots & \\ \partial f_n/\partial x_1 & \partial f_n/\partial x_2 & \cdots & \partial f_n/\partial x_m \end{bmatrix}$$

$j=2$ column

$$\frac{\partial f_1}{\partial x_2} = \lim_{\delta \to 0} \frac{f_1(x+\delta e_2) - f_1(x)}{\delta}$$

what is $\delta = ?$

Maybe take $\delta = \|x_{k-1} - x_k\|$

Now as $x_k \to r$

we have $\delta \to 0$.

Preserve super-linear convergence by making the approximation of Df get better as x_n approaches the root r.

Using difference quotients to approximate derivatives leads to loss of precision when δ is too small (or h before).

Numerical Example.

```
julia> using Plots

julia> f(x)=sin(x)
f (generic function with 1 method)

julia> df(x)=cos(x)
df (generic function with 1 method)

julia> dfh(x,h)=(f(x+h)-f(x))/h
dfh (generic function with 1 method)
```

what happens if h is too big?

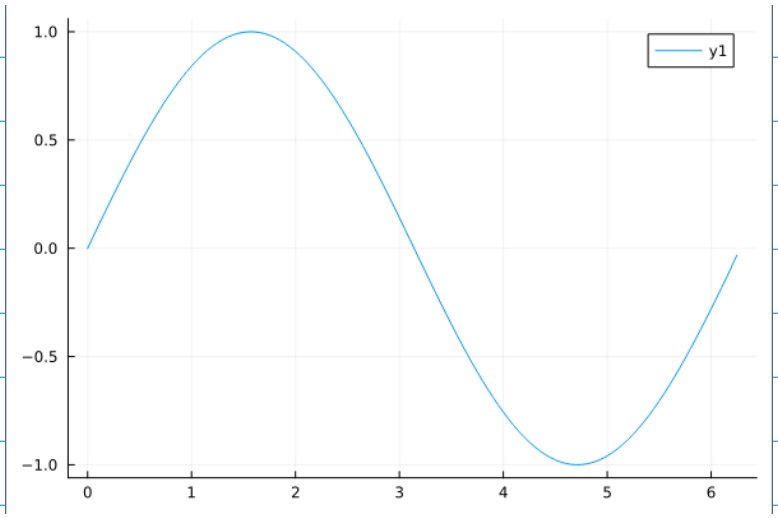
what happens if h is too small?

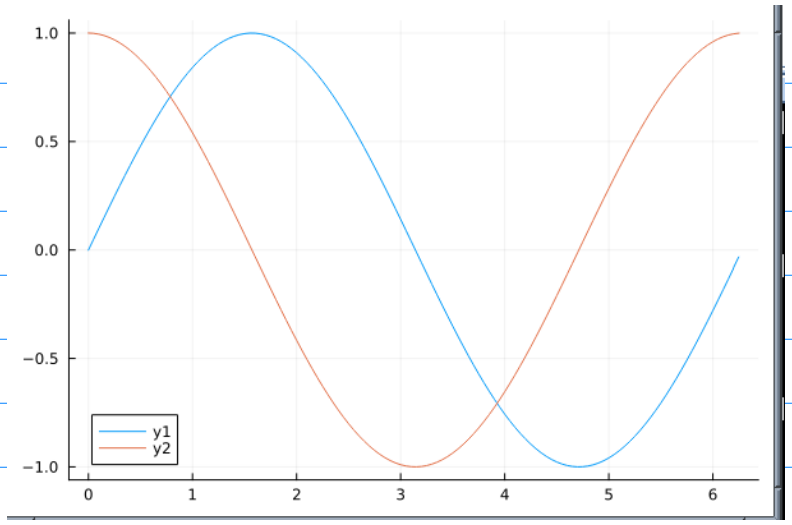
```
julia> xs=0:0.05:2*pi
0.0:0.05:6.25

julia> plot(f,xs)
```

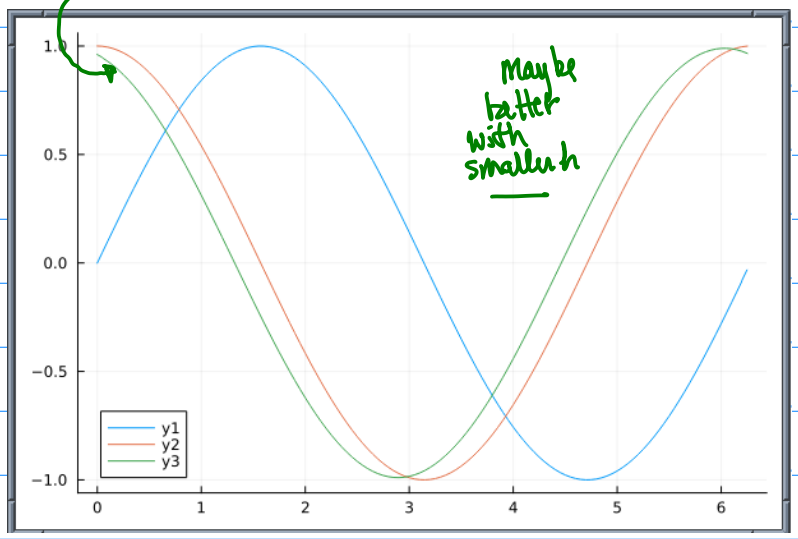
```
julia> plot!(df,xs)
```

overlay on top of the previous plot



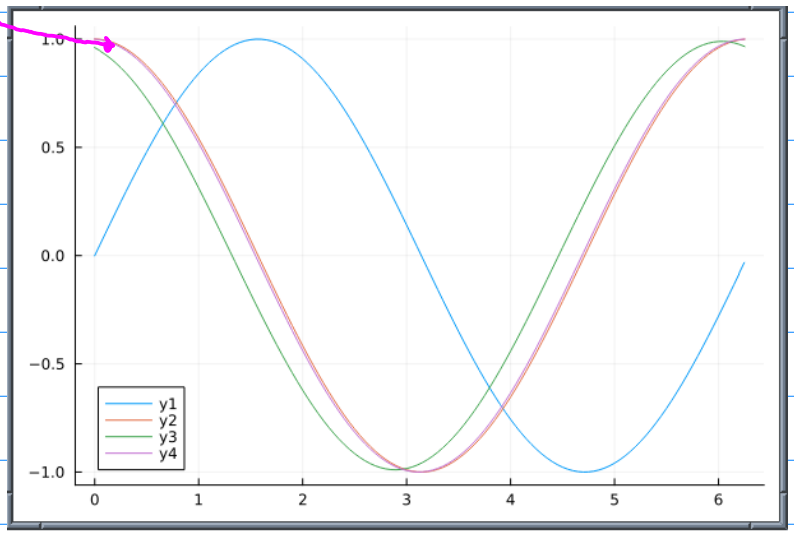



```
julia> plot!(x->dfh(x,0.5),xs)
```




```
julia> plot!(x->dfh(x,0.05),xs)
```

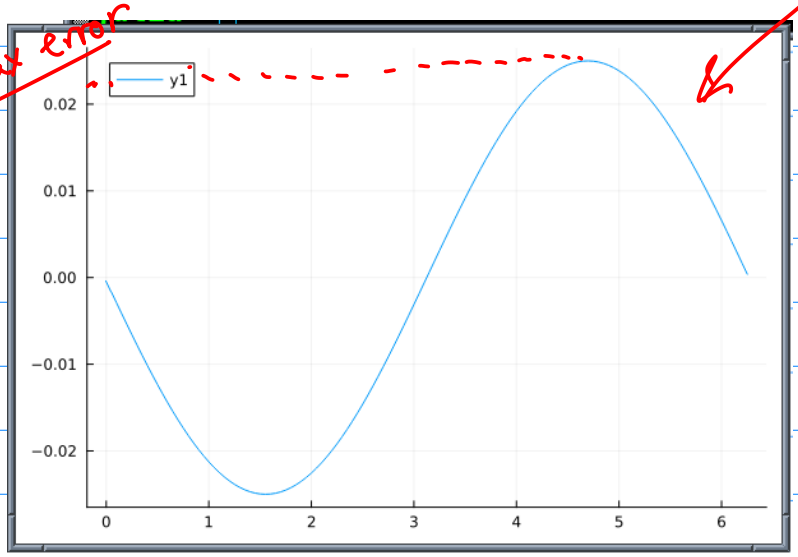
almost visually the same...



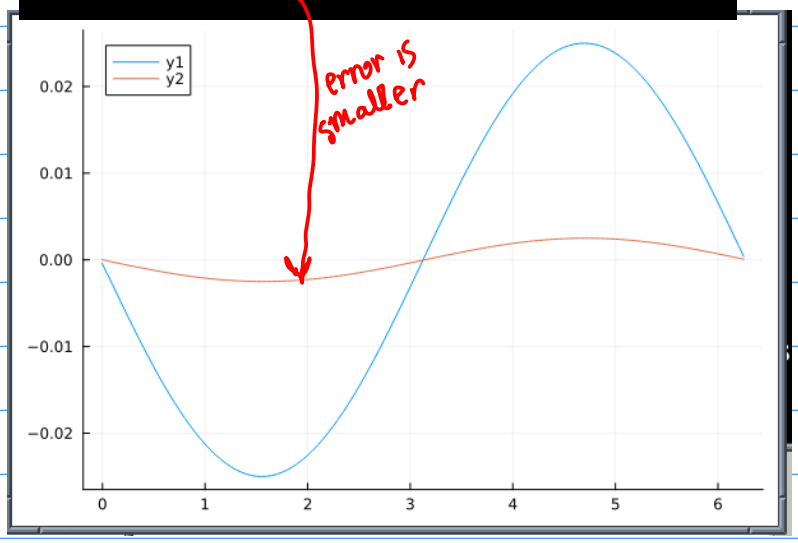
Idea... plot the error...

```
julia> plot(x->(dfh(x,0.05)-df(x)),xs)
```

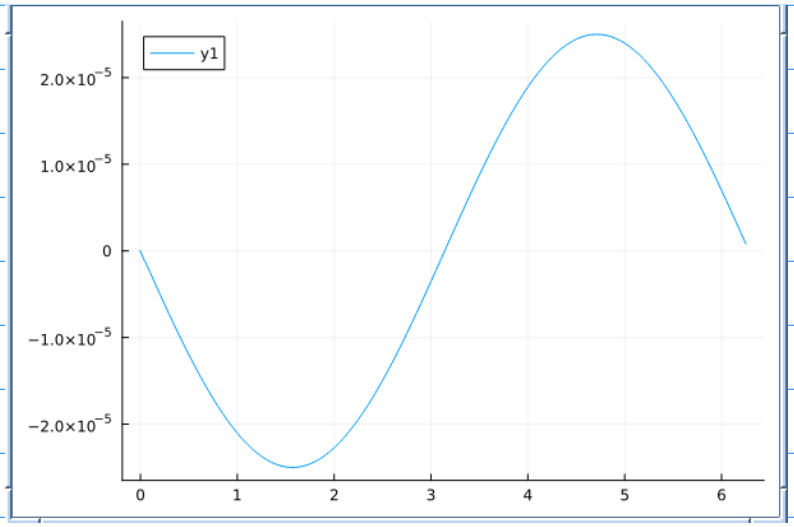
looks like sin function again. but not. look at the scale



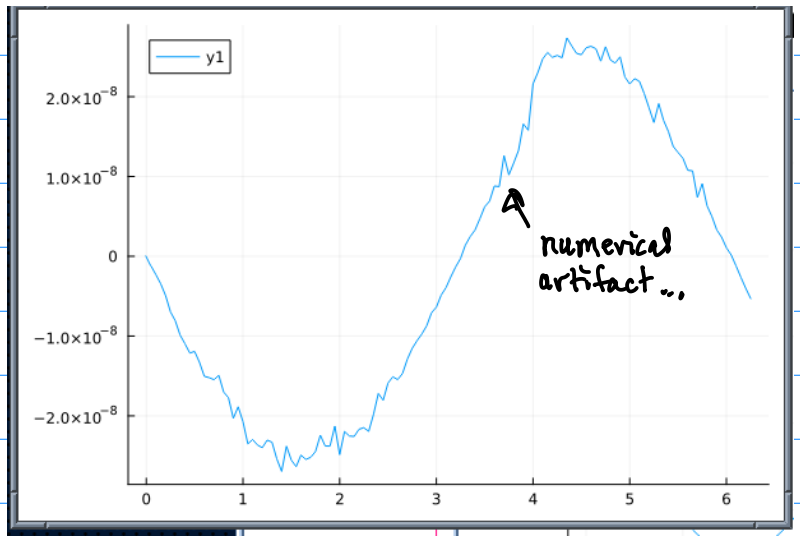

```
julia> plot!(x->(dfh(x,0.005)-df(x)),xs)
```



```
julia> plot(x->(dfh(x,0.00005)-df(x)),xs)
```

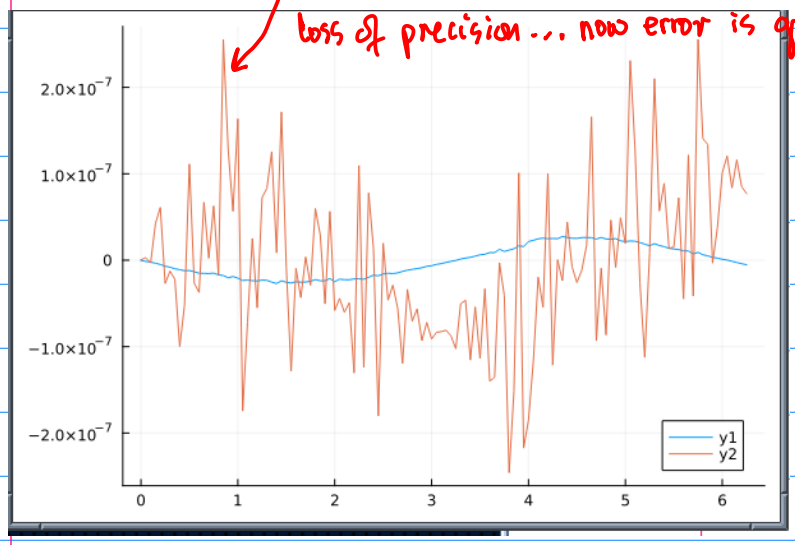



```
julia> plot(x->(dfh(x,0.00000005)-df(x)),xs)
```

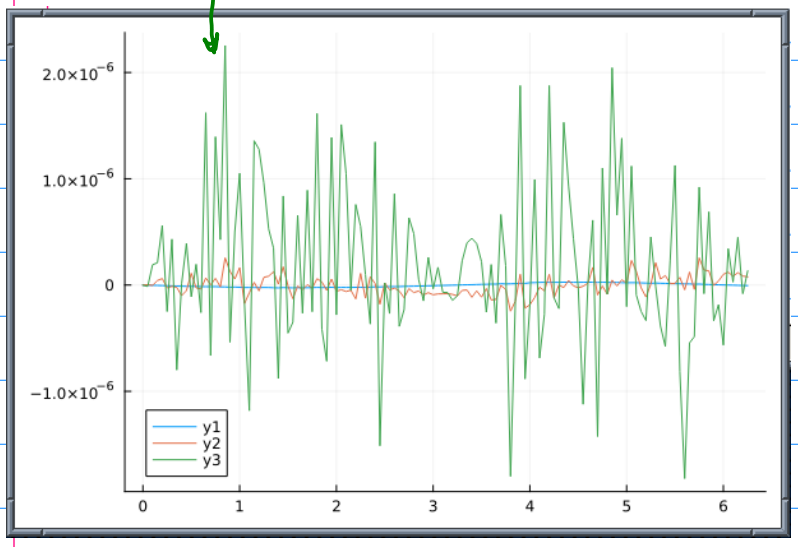


```
plot!(x->(dfh(x,0.0000000005)-df(x)),xs)
```

loss of precision... now error is greater




```
julia> plot!(x->(dfh(x,0.00000000005)-df(x)),xs)
```



How best to choose h... Think about it analytically...

$f(x+h) - f(x)$ ← exact calculation of numerator.

$fl(fl(f(x+h)) - fl(f(x)))$ ← have to round the values of $f(x)$ and $f(x+h)$ and then round the result of subtracting them.

Every rounding operation makes a relative error on the order of ε_{mach}.

$fl(x) = x + \varepsilon x$ where $|\varepsilon| \leq \frac{1}{2}\varepsilon_{mach}$

relative error in rounding

$$fl(f(x)) \simeq f(x)(1+\varepsilon_1)$$

$$fl(f(x+h)) \simeq f(x+h)(1+\varepsilon_2)$$

rounding from the subtraction

$$fl\left(fl(f(x+h)) - fl(f(x))\right) = \left(f(x)(1+\varepsilon_1) - f(x+h)(1+\varepsilon_2)\right)(1+\varepsilon_3)$$

Multiply this out to figure out how much rounding error there is in total and use this to figure of what it should be.