$f: \mathbb{R}^m \to \mathbb{R}^n$ $\quad x \in \mathbb{R}^m$

$g(x) = x + A f(x)$

$m \quad m \times n \quad n$

$g: \mathbb{R}^m \to \mathbb{R}^m$

How to choose A so that the iteration $x_{k+1} = g(x_k)$ converges?

How to choose optimally.

$Dg(x) = Dx + DA\, f(x)$

$= I + A\, Df(x)$

what's this...

Review of multiple variable calculus.

$Df(x)$ is related to the best linear approximation of f at x in the same way that in one dimension the tangent line approximates the function

Thus $f(x+v) - f(x) \approx Df(x)\, v$

$Df(x)$: matrix

$$\lim_{\|v\| \to 0} \frac{\| f(x+v) - f(x) - Df(x)v \|}{\|v\|} = 0.$$

It turns out, if there is such a matrix $Df(x)$ that satisfies the above limit then

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

row, column.

Let's suppose $m = n$ and the matrix $Df(r)$ is invertible. Then minimize Dg with

$$I + A\,Df(r) \approx 0$$

solve for A

$$A\,Df(r) \approx -I$$

matrix

$$A \approx -(Df(r))^{-1} \approx -(Df(x_k))^{-1}$$

Newton's method for systems

$$x_{k+1} \approx x_k - (Df(x_k))^{-1} f(x)$$

A

In code use left matrix division

$$x_{k+1} \approx x_k - Df(x_k) \backslash f(x)$$

$$x_{k+1} = g(x_k) \quad \text{where } g(x) = x - Df(x) \backslash f(x)$$

Example:

$$f_1(x_1, x_2, x_3) = -x_1 \cos(x_2) - 1$$
$$f_2(x_1, x_2, x_3) = x_1 x_2 + x_3$$
$$f_3(x_1, x_2, x_3) = e^{-x_3} \sin(x_1 + x_2) + x_1^2 - x_2^2.$$

```
julia> F(x)=[f1(x),f2(x),f3(x)]
F (generic function with 1 method)

julia> f1(x)=-x[1]*cos(x[2])-1
       f2(x)=x[1]*x[2]+x[3]
       f3(x)=exp(-x[3])*sin(x[1]+x[2])+x[1]^2-x[2]^2
f3 (generic function with 1 method)

julia> F([1,2,3])
3-element Vector{Float64}:
 -0.5838531634528576
  5.0
 -2.99297404851065
```

```
julia> using Symbolics

julia> @variables x1,x2,x3
3-element Vector{Num}:
 x1
 x2
 x3
```

```
julia> Jsym=Symbolics.jacobian(F([x1,x2,x3]),[x1,x2,x3])
3×3 Matrix{Num}:
                   -cos(x2)                           x1*sin(x2)                          0
                         x2                                   x1                          1
 2x1 + exp(-x3)*cos(x1 + x2)   -2x2 + exp(-x3)*cos(x1 + x2)   -exp(-x3)*sin(x1 + x2)
```

$$\mathbf{J}(x) = \begin{bmatrix} -\cos(x_2) & x_1\sin(x_2) & 0 \\ x_2 & x_1 & 1 \\ e^{-x_3}\cos(x_1+x_2)+2x_1 & e^{-x_3}\cos(x_1+x_2)-2x_2 & -e^{-x_3}\sin(x_1+x_2) \end{bmatrix}.$$

same as in the book...

```
julia> as="J(x1,x2,x3)="*string(Jsym)
"J(x1,x2,x3)=Num[-cos(x2) x1*sin(x2) 0; x2
cos(x1 + x2) -2x2 + exp(-x3)*cos(x1 + x2)
]"

julia> eval(Meta.parse(as))
J (generic function with 1 method)

julia> J(1,2,3)
3×3 Matrix{Num}:
 0.416147   0.909297            0
        2          1            1
  1.95071  -4.04929  -0.00702595
```

define a matrix
with parameter a vector...

```
julia> DF(x)=J(x[1],x[2],x[3])
DF (generic function with 1 method)
```

```
julia> DF([1,2,3])
3×3 Matrix{Num}:
 0.416147  0.909297           0
        2         1           1
  1.95071  -4.04929  -0.00702595
```

$$G(x) = x - (DF(x))^{-1} F(x)$$

$$G(x) = x - DF(x) \backslash F(x)$$

left matrix division

```
julia> G(x)=x-DF(x)\F(x)
G (generic function with 1 method)
```

need an initial guess

```
julia> using LinearAlgebra
```

```
julia> xk=[1.0,2.0,3.0]
       for k=1:15
           xk=G(xk)
           println("xk=$xk |F(xk)|=",norm(F(xk)))
       end
xk=Num[2.455763274274508, 1.9758515561023375, -4.887378104651353] |F(xk)|=125.2859968369057
xk=Num[1.888431728835991, 2.089197401300674, -4.009611333943298] |F(xk)|=41.70129426842139
xk=Num[1.3609985357657672, 2.2877178796579645, -3.2182869742494686] |F(xk)|=15.51652737249202
xk=Num[0.3978644296893281, 3.0076029127640225, -1.889964045457231] |F(xk)|=10.653655644298308
xk=Num[1.0776576921287906, 1.7282437492647924, -2.7321549098556677] |F(xk)|=3.4527226865387184
xk=Num[-5.128092188240146, 3.423278903924739, 7.0359255971316355] |F(xk)|=18.928097593342788
xk=Num[-7.581360593552638, 9.233624441395444, 55.749099344634594] |F(xk)|=32.34746493041192
xk=Num[-4.681625636515559, 5.3483318891203275, 13.772569052551447] |F(xk)|=13.221832769537052
xk=Num[-4.829151290060229, 4.852315382681558, 23.505740249337066] |F(xk)|=0.4027847292804838 7
xk=Num[-4.9183569428982015, 4.917986381574402, 24.182554240566507] |F(xk)|=0.008022032939327163
xk=Num[-4.917186115359714, 4.917186041100551, 24.178717990879555] |F(xk)|=1.3292478470012917e-6
xk=Num[-4.9171859252871375, 4.917185925287135, 24.178717423841892] |F(xk)|=3.582633105135331e-14
xk=Num[-4.917185925287132, 4.917185925287132, 24.17871742384187] |F(xk)|=1.1102230246251565e-15
xk=Num[-4.917185925287132, 4.917185925287132, 24.178717423841874] |F(xk)|=3.722145839155691e-15
xk=Num[-4.917185925287132, 4.917185925287132, 24.178717423841874] |F(xk)|=3.722145839155691e-15
```

far from my initial guess, so not surprising that it didn't converge right away..

like (1/2)^3

looks like quadratic convergence...

What to do if you can't compute the Jacobian matrix at all, even with builtin computer algebra system...

For one variable these were the inverse interpolation methods avoid use of f' in the iteration.

In multivariate settings it's more complicated... instead of inverse interpolation we'll focus on approximating $A = (Df(x))^{-1}$ in an efficient way that doesn't require knowing Df.

Talk about loss of precision again... and do something like the secant method.