Newton method based on numerical differentiation

Jacobian calculation

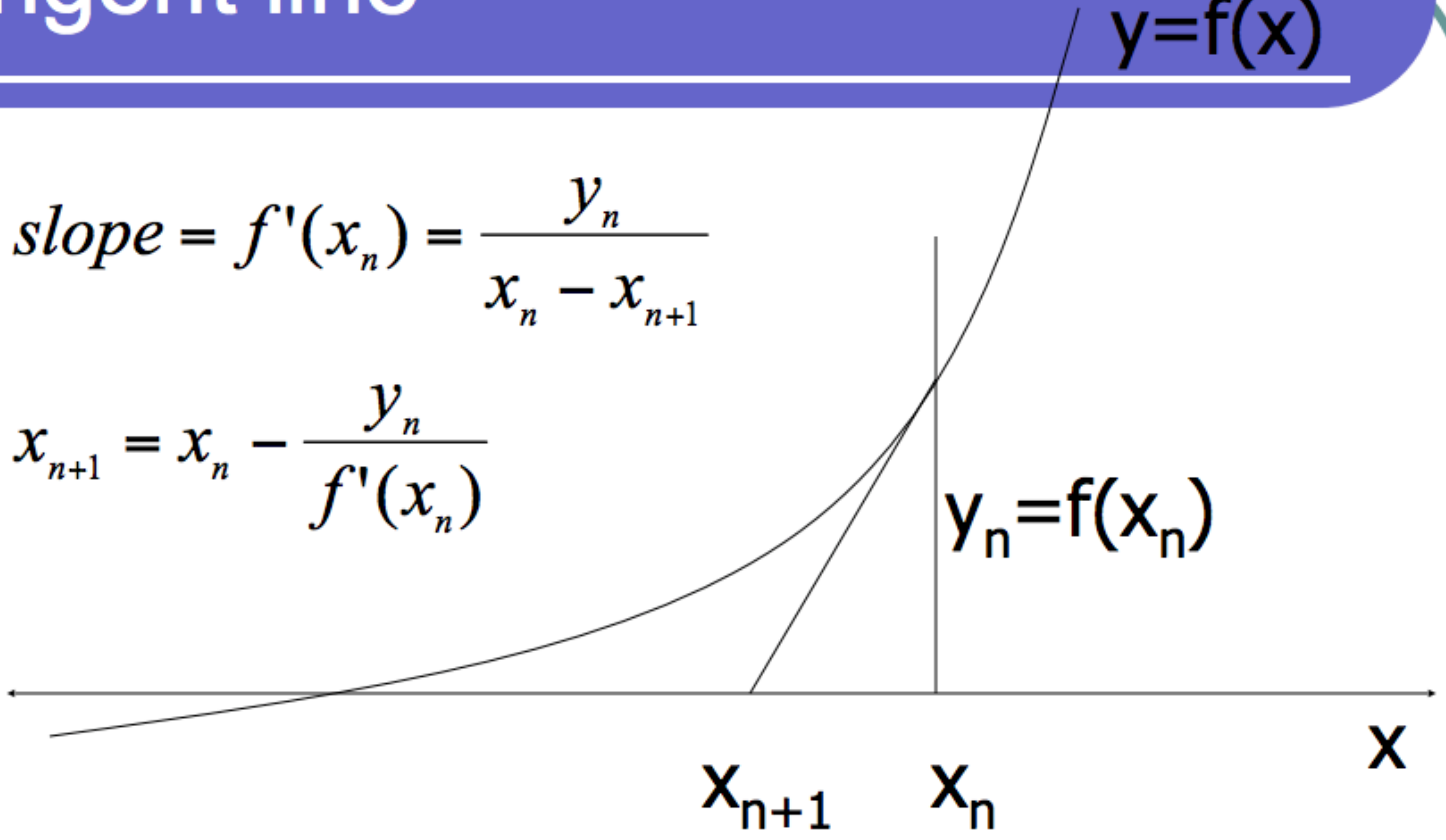
Newton method for root finding

- An iterative approach
    - Updating rule
    - While-looping
    - Halting condition
- Symbolic differentiation
-

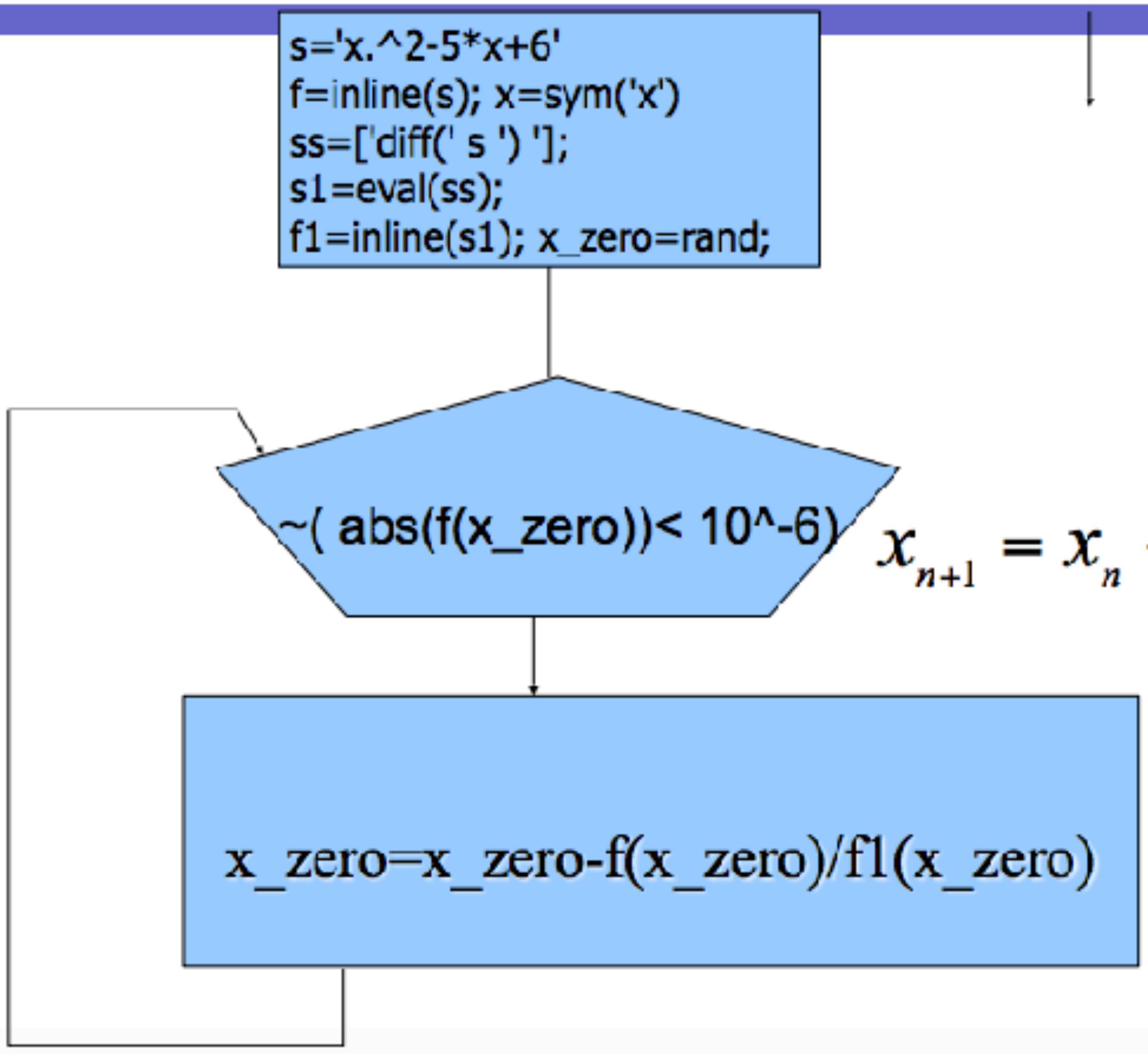
Tangent line

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

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



Iterative approach



```
s='x.^2-5*x+6'
f=inline(s); x=sym('x')
ss=['diff(' s ') '];
s1=eval(ss);
f1=inline(s1); x_zero=rand;
```

```
~( abs(f(x_zero))< 10^-6)
```

```
x_zero=x_zero-f(x_zero)/f1(x_zero)
```

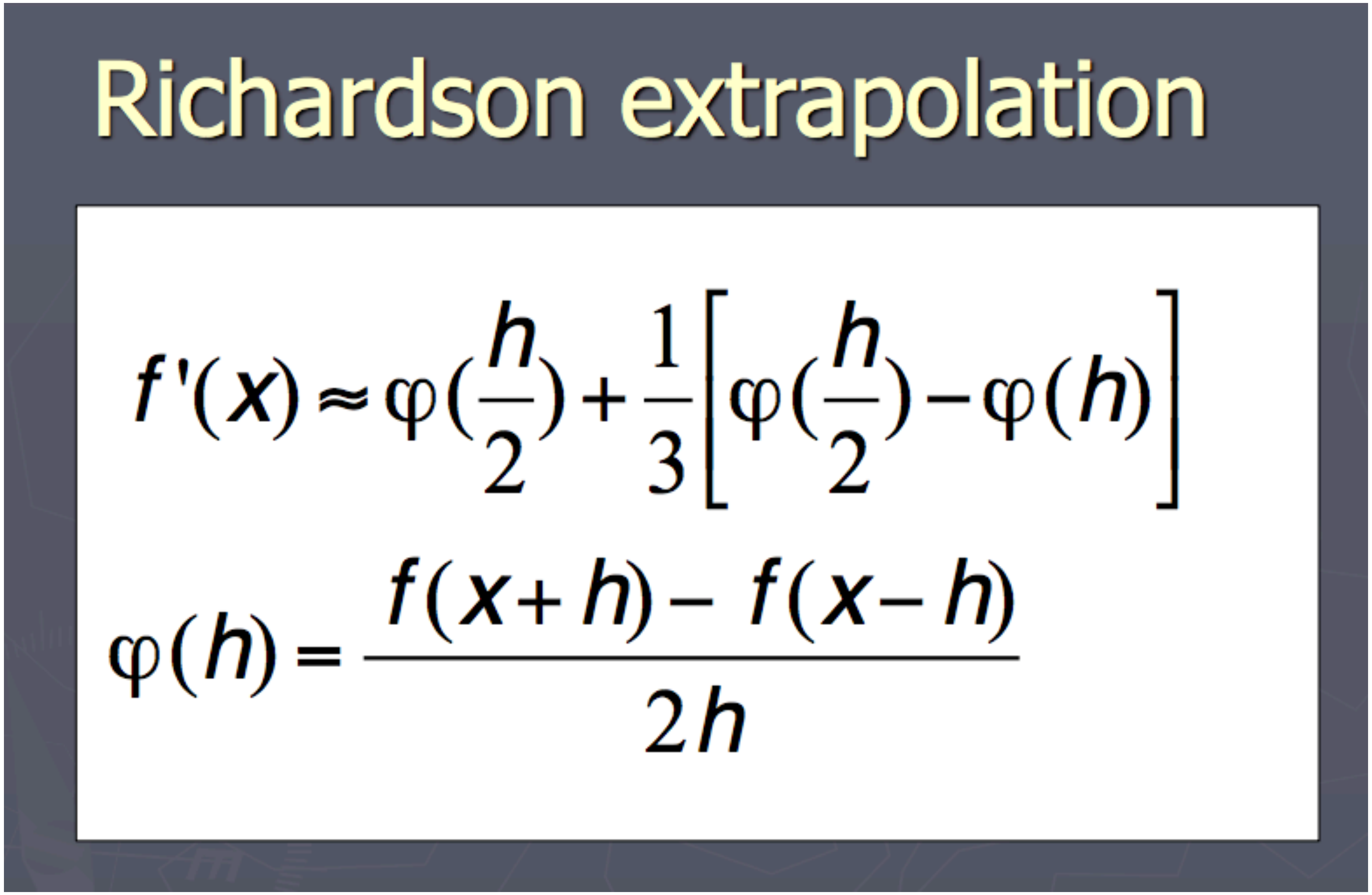
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

```
s='x.^2-5*x+6'
f=inline(s); x=sym('x')
ss=['diff(' s ') '];
s1=eval(ss);
f1=inline(s1);
x_zero=rand;
while   ~( abs(f(x_zero))< 10^-6)
    x_zero=x_zero-f(x_zero)/f1(x_zero)
end
```

symbolic differentiation

halting cond.

updating rule



Richardson extrapolation

$$f'(x) \approx \varphi\left(\frac{h}{2}\right) + \frac{1}{3}\left[\varphi\left(\frac{h}{2}\right) - \varphi(h)\right]$$

$$\varphi(h) = \frac{f(x+h) - f(x-h)}{2h}$$

```
s='x.^2-5*x+6'
f=inline(s);
% x=sym('x')
% ss=['diff(' s ') '];
% s1=eval(ss);
% f1=inline(s1);
x_zero=rand;
while   ~( abs(f(x_zero))< 10^-6)
    fn=…
    x_zero=x_zero-f(x_zero)/fn
end
```

symbolic differentiation

halting cond.

Apply numerical differentiation of Richardson Extrapolation to determine f'(x_zero)

```
s1='3*x1-cos(x2*x3)-1/2';
s2='x1^2-81*(x2+0.1)^2+sin(x3)+1.06';
s3='exp(-x1*x2)+20*x3+1/3*(10*pi-3)';
x1=sym('x1');x2=sym('x2');x3=sym('x3');
f=inline([str2sym(s1);str2sym(s2);str2sym(s3)]);
A=jacobian([str2sym(s1);str2sym(s2);str2sym(s3)],[x1 x2 x3]);
j=inline(A);
x=rand(3,1)-0.5;
y=f(x(1),x(2),x(3))
j(x(1),x(2),x(3))
```

```
y =

  -2.5738
   1.5538
   18.7591


ans =

   3.0000   -0.0135    0.0025
  -0.7162   -3.5253    0.9148
   0.0761    0.3482   20.0000
```

Calculate Jacobian by symbolic differentiation

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[J(\mathbf{x}_n)\right]^{-1} F(\mathbf{x}_n)$$

$$J(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \dfrac{\partial f_2(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \text{M} \\ \dfrac{\partial f_n(\mathbf{x})}{\partial x_1} & \dfrac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \dfrac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

```
s1='3*x1-cos(x2*x3)-1/2';
s2='x1^2-81*(x2+0.1)^2+sin(x3)+1.06';
s3='exp(-x1*x2)+20*x3+1/3*(10*pi-3)';
x1=sym('x1');x2=sym('x2');x3=sym('x3');
f=inline([str2sym(s1);str2sym(s2);str2sym(s3)]);
% A=jacobian([str2sym(s1);str2sym(s2);str2sym(s3)],[x1 x2 x3]);
% j=inline(A);
x=rand(3,1)-0.5;
y=f(x(1),x(2),x(3))
% j(x(1),x(2),x(3))
…
…
…
```

Try to calculate Jacobian by numerical differentiation of Richardson Extrapolation and compare it with previous results

```
y =

  -2.5738
   1.5538
  18.7591


ans =

   3.0000   -0.0135    0.0025
  -0.7162   -3.5253    0.9148
   0.0761    0.3482   20.0000
```

http://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf