Lecture 4

Interpolatin Techniques for Line search

Interpolation

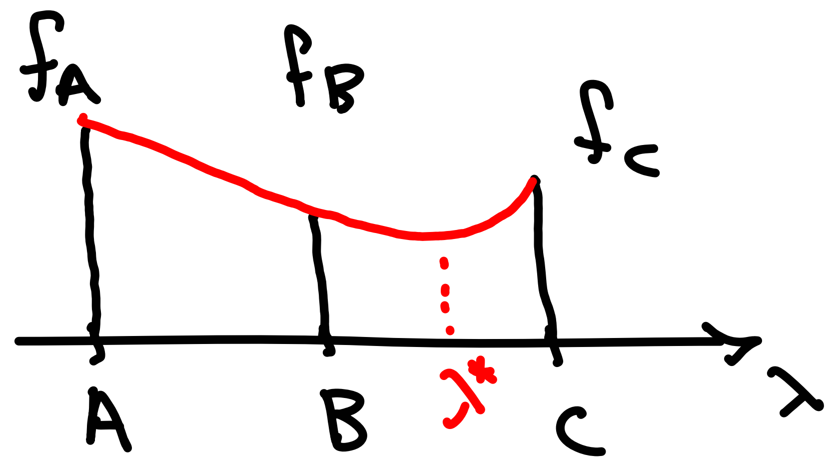
* A model is constructed using available data samples.
* The minimum of the model is used to approximate the function minimum.
* The accuracy of the model is evaluated and the model is "refitted" using extra data points.

Quadratic Model

* $h(\lambda) = a + b\lambda + c\lambda^2$

  $\frac{dh}{d\lambda} = 0 \Rightarrow b + 2\lambda c = 0$

  $\lambda^* = -\frac{b}{2c}$



Local minimum if $\frac{d^2h}{d\lambda^2} = +ve \Rightarrow c > 0$

* How do we determine the coefficients a, b, c given the data $(A, f_A), (B, f_B), (C, f_C)$

Quadratic Model (Cont'd)

* $h(\lambda) = a + b\lambda + c\lambda^2$

$\Rightarrow f_A = a + bA + cA^2,\ f_B = a + bB + cB^2,\ f_C = a + bC + cC^2$

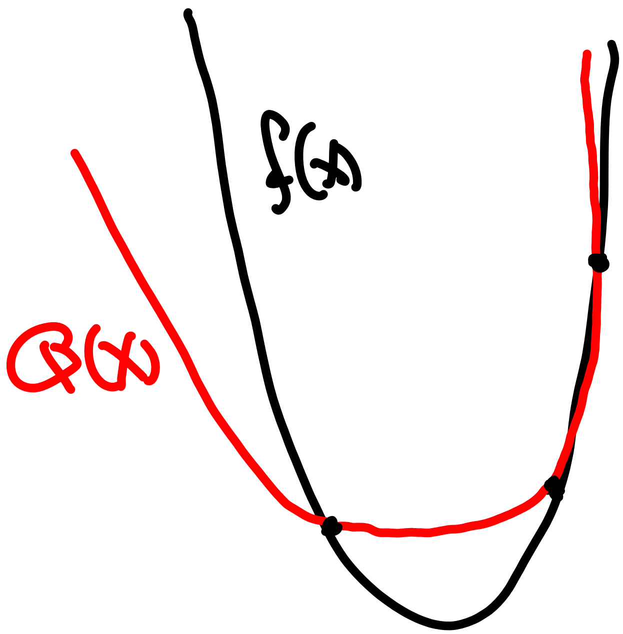
3 eqns in 3 unknowns (a, b, c).

* closed form solution exists!

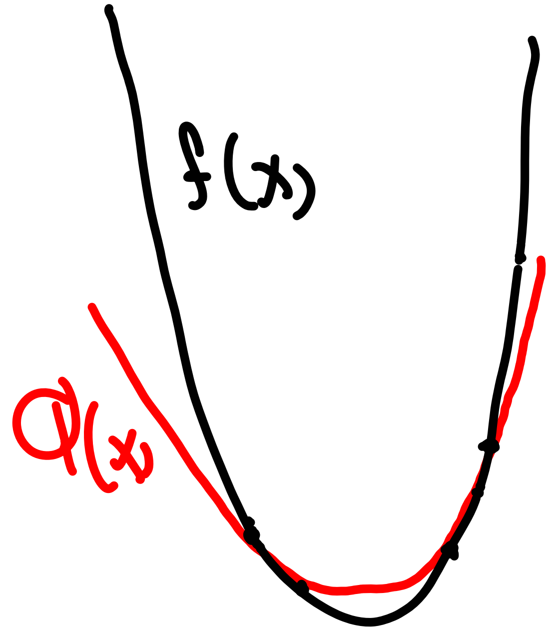
$$\lambda^* = -\frac{b}{2c} = \frac{f_A(B^2 - C^2) + f_B(C^2 - A^2) + f_C(A^2 - B^2)}{2[f_A[B - C] + f_B(C - A) + f_C(A - B)]}$$

* $c > 0 \Rightarrow [f_A(B - C) + f_B(C - A) + f_C(A - B)] < 0$

Illustration



widely separated points



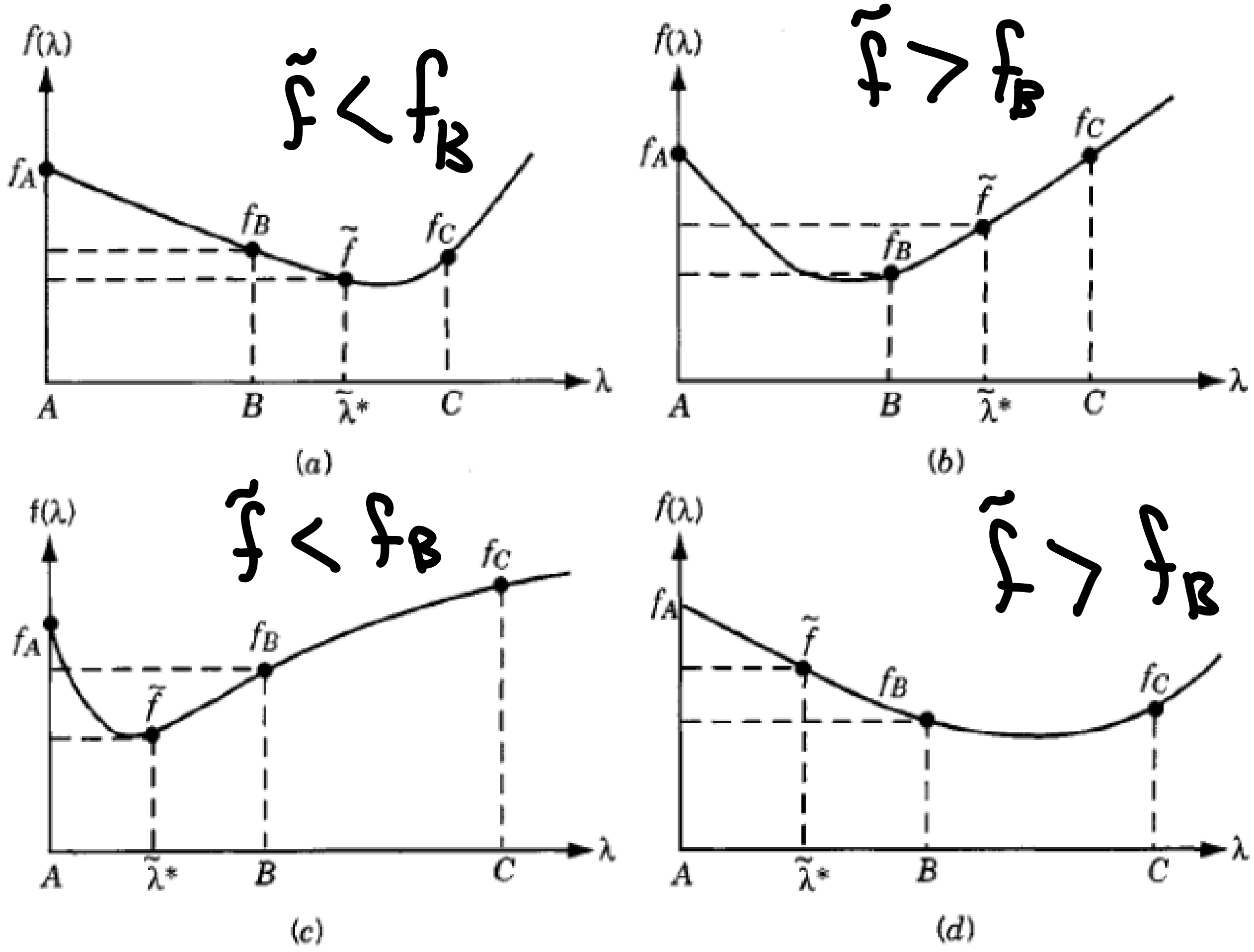
closely separated points

Line Search with a Quadratic Model

* $f(\lambda) = f(\underline{x}^0 + \lambda \underline{S})$ $\Rightarrow$ $\underline{S}$ should be normalized so that λ values are reasonable.
* Solution is first bracketed using an elimination algorithm.
* Accuracy of the quadratic model $h(\lambda)$ is checked $\Rightarrow$ $\left| \dfrac{h(\lambda^*) - f(\lambda^*)}{f(\lambda^*)} \right| \le \varepsilon_1$

  or $\left| \dfrac{f(\lambda^* + \Delta\lambda) - f(\lambda^* - \Delta\lambda)}{2\Delta\lambda} \right| \le \varepsilon_2$

Model Refitting



Example

The function $f(x) = -5x^5 + 4x^4 - 12x^3 + 11x^2 - 2x + 1$ is unimodal in the interval $[-0.5, 0.5]$. Use Quadratic approximation to find its minimum with a range of uncertainty less than 10^{-5}.

MATLAB CODE

```
A=-0.5; %start of interval
C=0.5;  %end of interval
B=0.5*(A+C); %interval middle point
L=C-A; %current interval length
epsilon=1.0e-5; %termination condition
fA=getFunction(A);
fB=getFunction(B);
fC=getFunction(C);
while (L>1.0e-5)
    Lambda=0.5*(fA*(B*B-C*C)+fB*(C*C-A*A)+fC*(A*A-B*B))/
                           (fA*(B-C)+fB*(C-A)+fC*(A-B));
    fLambda=getFunction(Lambda); %get function value at new
point
    %solution in first interval
    if((A<Lambda)&(Lambda<B))
        if(fLambda<fB) %move right point to current B.
           C=B;
           fC=fB;
           B=Lambda;
           fB=fLambda;
        else  %move left point to lambda
            A=Lambda;
            fA=fLambda;
        end
     end
    %solution in second interval
    if((B<Lambda)&(Lambda<C))
        if(fLambda<fB) %move left edge
            A=B;
            fA=fB;
            B=Lambda;
            fB=fLambda;
        else  %move right edge
            C=Lambda;
            fC=fLambda;
        end
    end
    L=C-A;
    A
    B
    C
end
```

Code Output

A=-0.5 B=0 C=0.5

A = 0 B = 0.2214 C =0.5000

A = 0 B = 0.1316 C = 0.2214

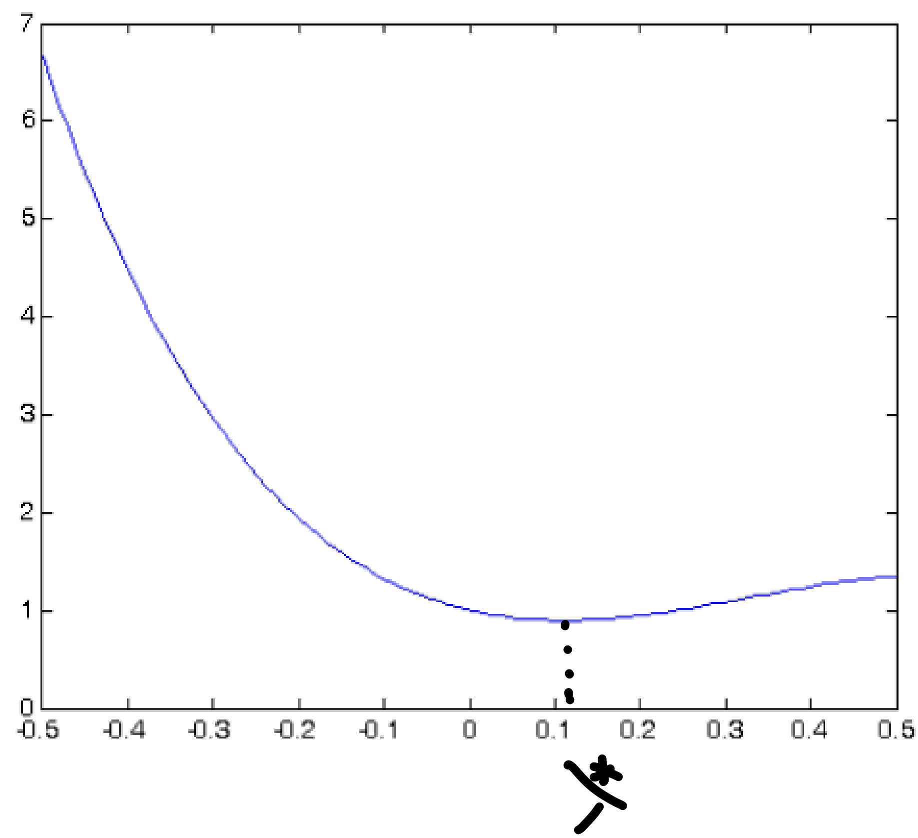
A = 0 B = 0.1193 C = 0.1316

.

.

A = 0.1099 B = 0.1099 C = 0.1099

Function plot



Directional Derivatives

* $\underset{\sim}{\nabla} f(\underset{\sim}{x})$ is the gradient of the function.
* $\underset{\sim}{s}^T \underset{\sim}{\nabla} f(\underset{\sim}{x})$ is called the directional derivatives in the direction $\underset{\sim}{s}$.
* It can be shown that

$$f'(\lambda) = \frac{df}{d\lambda} = \frac{d}{d\lambda} f(\underset{\sim}{x} + \lambda \underset{\sim}{s}) = \underset{\sim}{s}^T \nabla f(\underset{\sim}{x} + \lambda \underset{\sim}{s})$$

(Calculate $\underset{\sim}{\nabla} f(\underset{\sim}{x} + \lambda \underset{\sim}{s})$ and then calculate $f'(\lambda) = \underset{\sim}{s}^T \underset{\sim}{\nabla} f(\underset{\sim}{x} + \lambda \underset{\sim}{s})$)

Example: Consider the function

$f(\underset{\sim}{x}) = x_1^2 + 3x_1x_2 + x_2^2$. Given $\underset{\sim}{x}^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

and $\underset{\sim}{s} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, find the derivative

$\frac{df}{d\lambda} = \frac{d}{d\lambda}\left(f(\underset{\sim}{x}^{(0)} + \lambda \underset{\sim}{s})\right)$ at $\lambda = 2$.

Solution: $\underset{\sim}{x} = \underset{\sim}{x}^{(0)} + \lambda \underset{\sim}{s} = \begin{bmatrix} 1+\lambda \\ 1-\lambda \end{bmatrix}$

$f(\lambda) = (1+\lambda)^2 + 3(1+\lambda)(1-\lambda) + (1-\lambda)^2$

Example (cont'd)

$$\frac{df}{d\lambda} = 2(1+\lambda) - 6\lambda - 2(1-\lambda) = -2\lambda$$

$$\left.\frac{df}{d\lambda}\right)_{\lambda=2} = -4$$

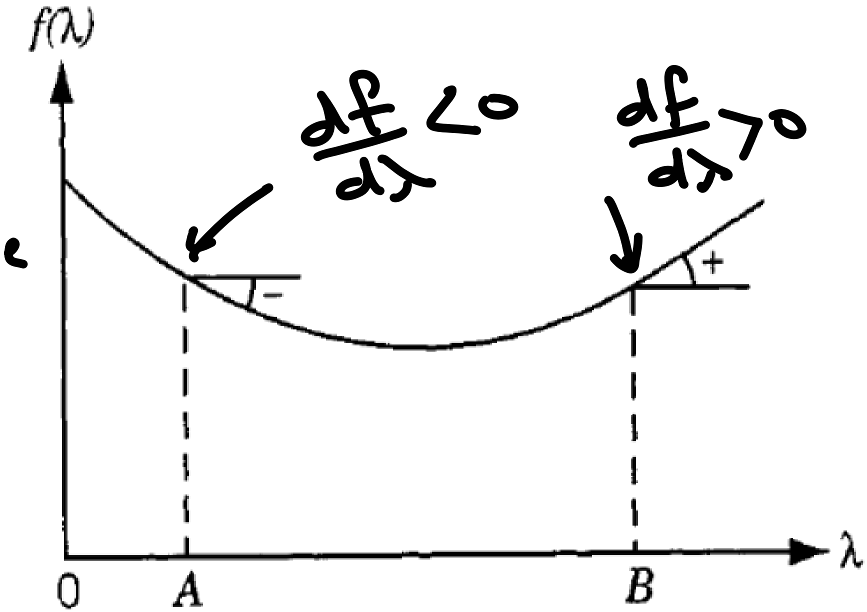
alternative solution $\quad \underset{\sim}{x}^{(1)} = \begin{bmatrix} 1+\lambda \\ 1-\lambda \end{bmatrix} \underset{\lambda=2}{=} \begin{bmatrix} 3 \\ -1 \end{bmatrix}$

$$\nabla f(\underset{\sim}{x}) = \begin{bmatrix} 2x_1 + 3x_2 \\ 3x_1 + 2x_2 \end{bmatrix} \Rightarrow \nabla f(\underset{\sim}{x}^{(1)}) = \begin{bmatrix} 3 \\ 7 \end{bmatrix}$$

$$\nabla f(\underset{\sim}{x}^{(1)})^T \underset{\sim}{s} = \begin{bmatrix} 3 & 7 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -4$$

Cubic Interpolation

* Solution is bracketed between two points with opposite derivative sign.



* A cubic model

$h(\lambda) = a + b\lambda + c\lambda^2 + d\lambda^3$

is fitted using f_A, $f'(A)$, f_B, $f'(B)$

Cubic Interpolation (Cont'd)

* The minimum is given by

$$\lambda^* = \frac{-c \pm \sqrt{c^2 - 3bd}}{3d}$$

$$\frac{d^2h}{d\lambda^2} > 0 \Longrightarrow 2c + 6d\lambda^* > 0$$

* The accuracy of the cubic model is tested using

$$\left| \frac{h(\lambda^*) - f(\lambda^*)}{f(\lambda^*)} \right| \le \varepsilon_1 \quad \text{or} \quad \left| \frac{df}{d\lambda} \right|_{\lambda^*} = \left| \left(\underset{\sim}{S}^T \nabla \underset{\sim}{f} \right)_{\lambda^*} \right| \le \varepsilon_2$$

(model refitting)

Matlab Code

```
A=0; %left point
B=2*pi; %right point
delta=1.0e-3; %delta for sensitivity analysis
fA=getFunction(A); %get function value at A
fB=getFunction(B); %get function value at B
fpA=getDerivative(A, delta); %get derivative through
finite difference at A
fpB=getDerivative(B, delta); %get derivative through
finite difference at B
L=B-A; %solution interval
Epsilon=1.0e-1; %interval termination condition
fpLambda=10; %initial gradient at solution
while(abs(fpLambda)>Epsilon) %repeat until condition
  %first we construct system of equations
  Matrix=[1    A    (A*A)   (A*A*A)
          1    B    (B*B)   (B*B*B)
          0    1    (2*A)   (3*A*A)
          0    1    (2*B)   (3*B*B)];
  RHS=[fA  fB   fpA   fpB]'; %this is the vector of RHS
  Coeff=inv(Matrix)*RHS; %Get coefficients
  b=Coeff(2,1); %2nd coefficient
  c=Coeff(3,1);  %3rd coefficient
  d=Coeff(4,1); %4th coefficient
  %two solutions exist.  We pick the one in interval
  Lambda1=(-c+sqrt(c*c-3*b*d))/(3*d);
  Lambda2=(-c-sqrt(c*c-3*b*d))/(3*d);
  if((Lambda1>A)&(Lambda1<B))
      Lambda=Lambda1;
  else
      Lambda=Lambda2;
  end
  fLambda=getFunction(Lambda); %get value at new minimi
  fpLambda=getDerivative(Lambda,delta); %get derivative
  %now we narrow down the interval
  if(fpLambda*fpA>0) %on same side of minimum
      A=Lambda;  %move left value to lambda
      fA=fLambda;
      fpA=fpLambda;
  else
      B=Lambda; %move right value to lambda
      fB=fLambda;
      fpB=fpLambda;
  end

end
```

Example

Utilize the Matlab code to find the minimum of the function

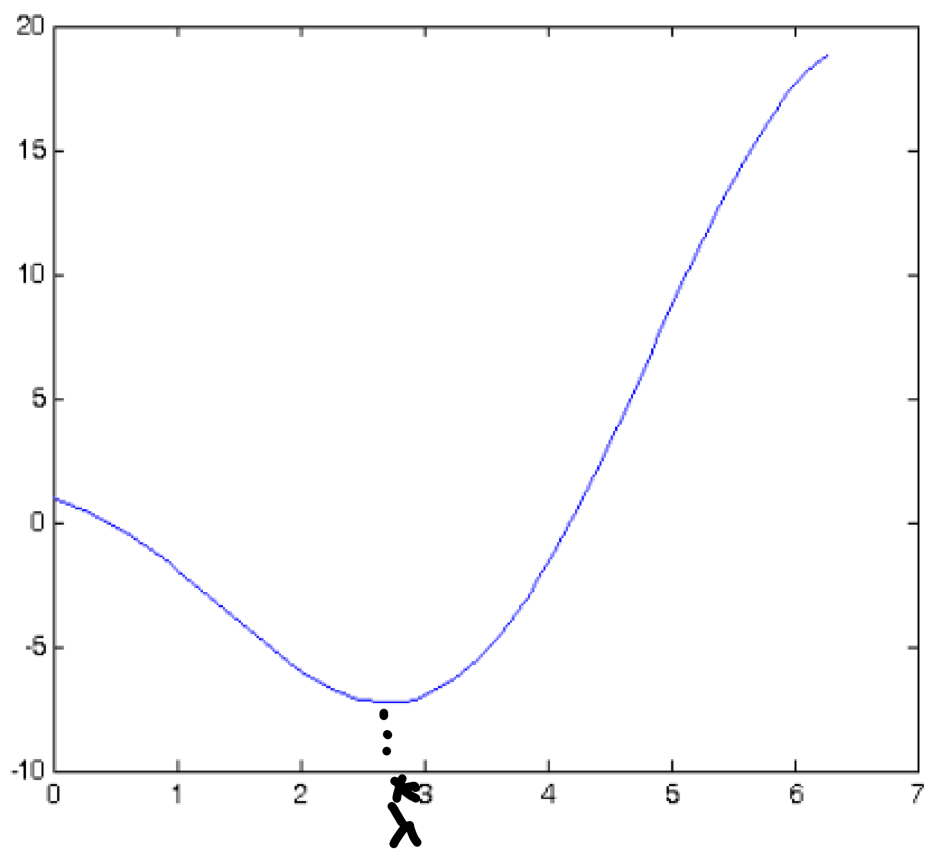
$f(x) = -3x\sin 0.75x + e^{-2x}$ in the interval

$x \in [0, 2\pi]$

Code Output

```
A = 0       B = 6.2832   Lambda = 0.7208
A = 0.7208  B = 6.2832   Lambda = 1.4511
A = 1.4511  B = 6.2832   Lambda = 2.0359
A = 2.0359  B = 6.2832   Lambda = 2.3862
A = 2.3862  B = 6.2832   Lambda = 2.5629
A = 2.5629  B = 6.2832   Lambda = 2.6442
A = 2.6442  B = 6.2832   Lambda = 2.6798
A = 2.6798  B = 6.2832   Lambda = 2.6952
```

Example



Newton Method

* Newton method requires 2nd order derivatives.
* It converges very fast if starting from a sufficiently close point to λ^*.
* It may diverge as well!
* It exploits a Taylor expansion of the gradient.

Newton Method

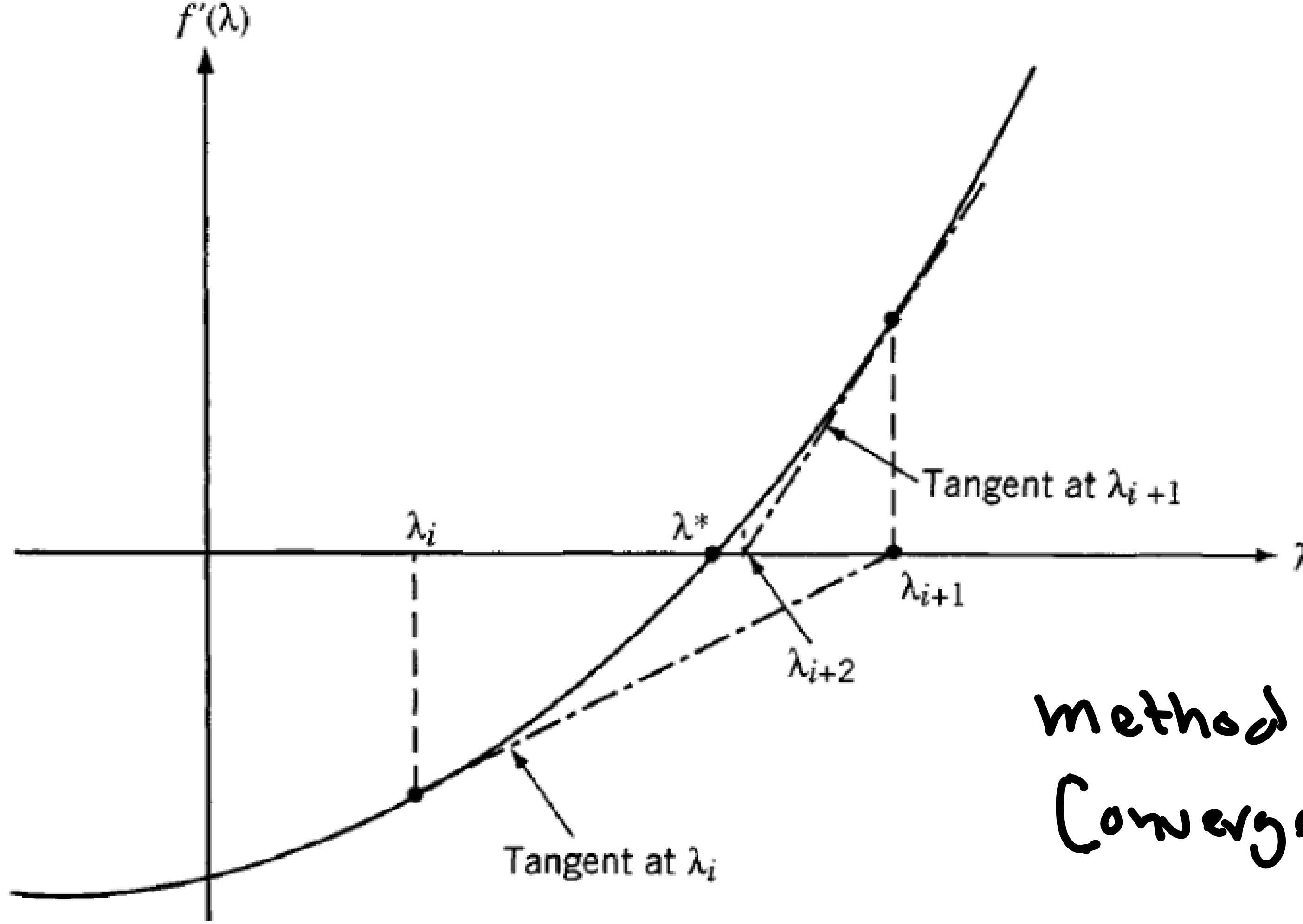
* Our target is to find a point λ^* satisfying $f'(\lambda^*) = 0$
* Using Taylor expansion we have

$$f'(\lambda) = f'(\lambda_i) + f''(\lambda_i)(\lambda - \lambda_i) = 0$$

$$\Rightarrow \lambda_{i+1} = \lambda_i - \frac{f'(\lambda_i)}{f''(\lambda_i)}$$

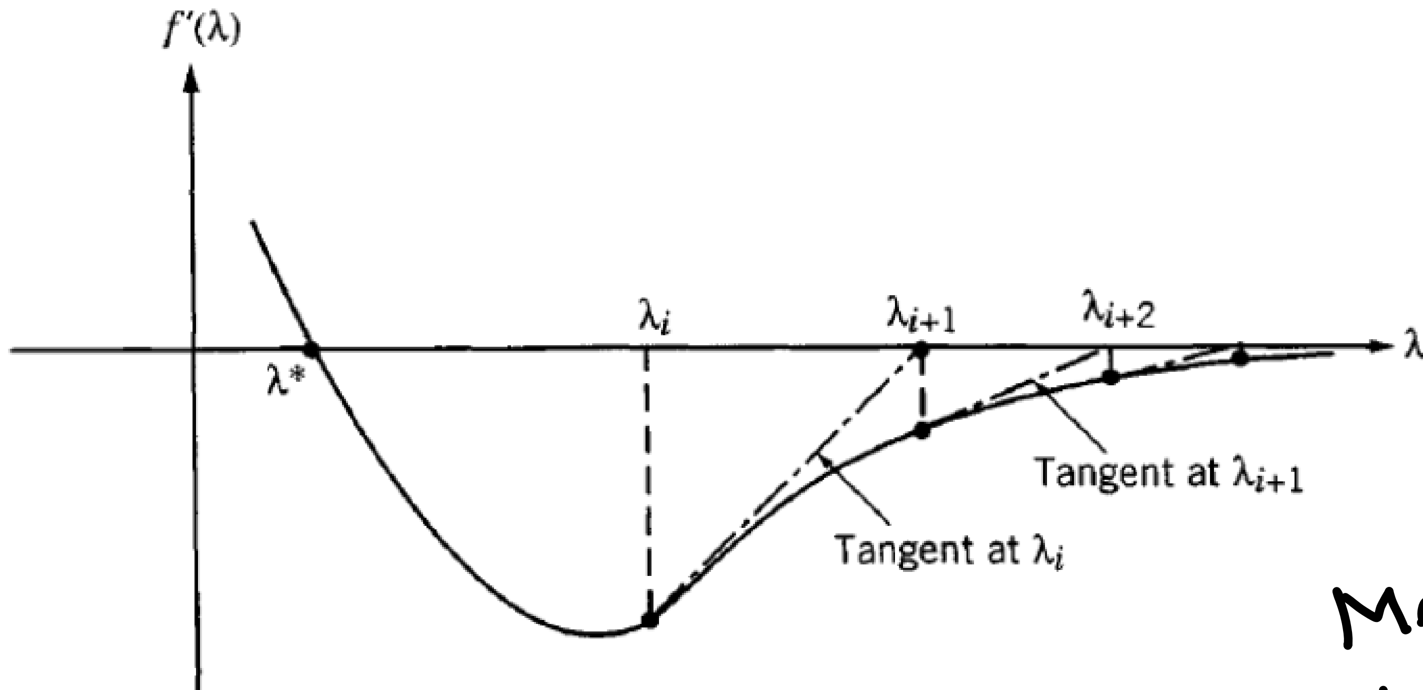
* Termination Condition $|f'(\lambda_{i+1})| \leq \varepsilon$

Convergence vs. Divergence



Method Converges!

Convergence vs. Divergence (Cont'd)



Method diverges

<u>Example</u> Using Newton's Method, find the minimizer of $f(x) = \frac{1}{2}x^2 - \sin x$. The initial value is $x^{(0)} = 0.5$. The required accuracy is $\| x^{n+1} - x^n \| < 10^{-5}$.

Solution: $f'(x) = x - \cos x$, $f''(x) = 1 + \sin x$

$$x^{(1)} = 0.5 - \left[\frac{0.5 - \cos 0.5}{1 + \sin 0.5} \right] = 0.7552$$

Example (Cont'd)

$$x^{(2)} = x^{(1)} - \frac{f'(x^{(1)})}{f''(x^{(1)})} = 0.7552 - \frac{0.02710}{1.685} = 0.7391$$

$$x^{(3)} = x^{(2)} - \frac{f'(x^{(2)})}{f''(x^{(2)})} = 0.7391 - \frac{0.009461}{1.673}$$

$$x^{(3)} \simeq 0.7390$$

$x^{(4)} = 0.7390$ (Notice that $f'' > 0$)

A Quasi-Newton Method

* This method is identical to Newton method with the difference that both the first and second derivatives are approximated through finite differences.

$$f'(\lambda_i) = \frac{f(\lambda_i + \Delta\lambda) - f(\lambda_i - \Delta\lambda)}{2\Delta\lambda}$$

$$f''(\lambda_i) = \frac{f(\lambda_i + \Delta\lambda) - 2f(\lambda_i) + f(\lambda_i - \Delta\lambda)}{(\Delta\lambda)^2}$$

MATLAB CODE

```
StartingPoint=11; %starting guess
delta=0.001; %perturbation used in derivative analysis
MaxIterationCount=100; %we do not allow more than 100
iterations
Epsilon=1.0e-4; %terminating condition for gradient
gradient=1.0e4; %initial value of gradient
Counter=0; %intialize iteration counter
CurrentPoint=StartingPoint; %initialize current point
while((Counter<MaxIterationCount)&(abs(gradient)>Epsilon))
   f=getFunction(CurrentPoint); %function at point
   fp=getFunction(CurrentPoint+delta); %function with +ve
perturbation
   fn=getFunction(CurrentPoint-delta); %function with -ve
perturbation
   gradient=(fp-fn)/(2*delta); %get first order derivative
   Hessian=(fp-2*f+fn)/(delta*delta);  %get second order
derivative
   NewPoint=CurrentPoint-(gradient/Hessian); %get the new
point
   CurrentPoint=NewPoint; %update new point
   Counter=Counter+1 %increment iteration counter
   CurrentPoint
end
```

Example

* The matlab code is applied to minimizing the function $f(x) = x^3 - 12.2x^2 + 7.45x + 42$.

* The output is

Counter=0 CurrentPoint=11.0

Counter = 1 CurrentPoint =8.5469

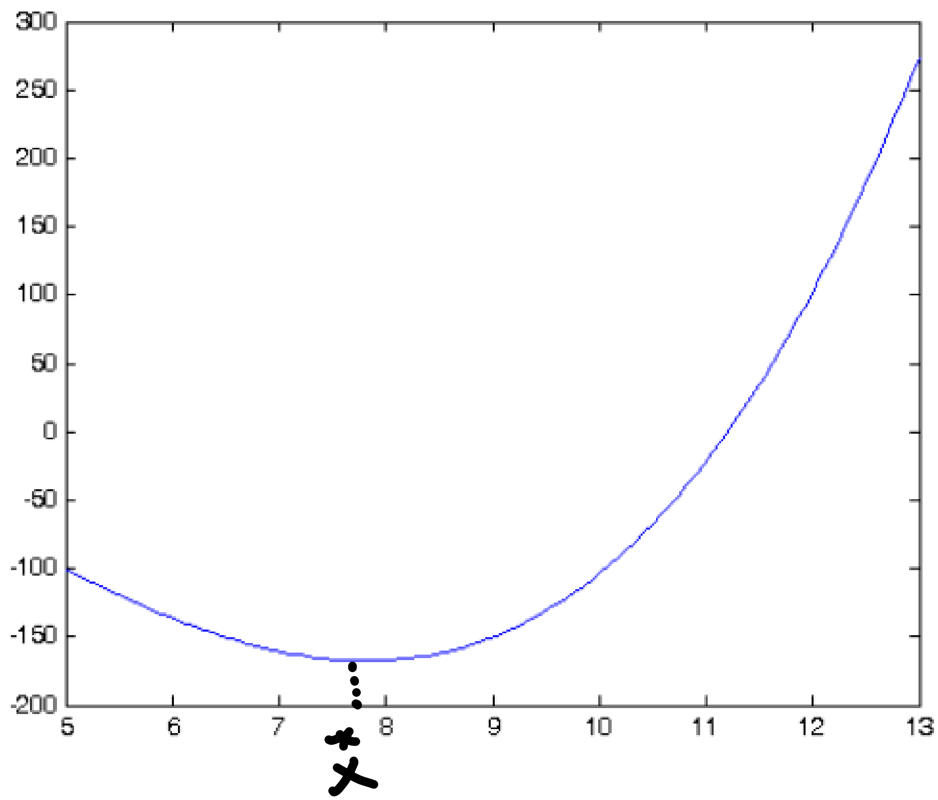
Counter =2 CurrentPoint =7.8753

Counter =3 CurrentPoint =7.8161

Counter =4 CurrentPoint =7.8156

Counter =5 CurrentPoint =7.8156

Function plot



Secant Method

* This method has the same basis as Newton method
* If the solution is bracketed between two values of $\lambda \in [A, B]$, the 2nd order derivative is approximated by

$$f''(A) = \frac{(f'(B) - f'(A))}{(B - A)}$$

$$\lambda_{i+1} = A - \frac{f'(A)\,(B - A)}{(f'(B) - f'(A))}$$

Illustration