16.323 Principles of Optimal Control
Spring 2008

16.323 Lecture 1

Nonlinear Optimization

- Unconstrained nonlinear optimization
- Line search methods

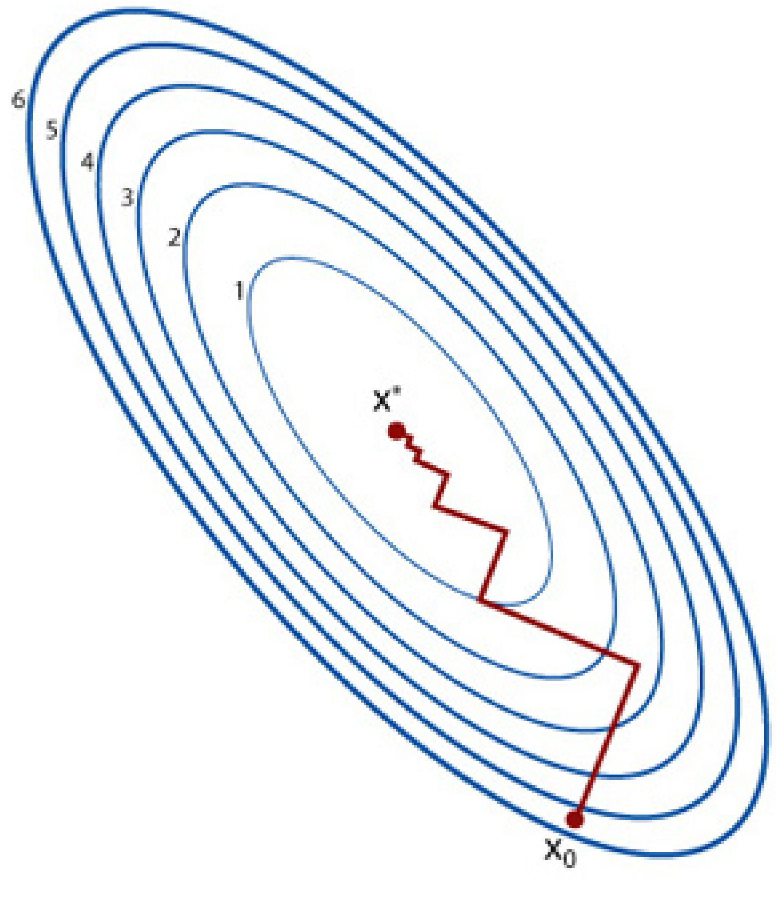


Figure by MIT OpenCourseWare.

Basics – Unconstrained

- Typical objective is to minimize a nonlinear function $F(\mathbf{x})$ of the parameters $\mathbf{x}$.
  - Assume that $F(\mathbf{x})$ is scalar $\Rightarrow \mathbf{x}^\star = \arg\min_{\mathbf{x}} F(\mathbf{x})$
- Define two types of minima:
  - **Strong**: objective function increases locally in all directions

  A point $\mathbf{x}^\star$ is a strong minimum of a function $F(\mathbf{x})$ if a scalar $\delta > 0$ exists such that $F(\mathbf{x}^\star) < F(\mathbf{x}^\star + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $0 < \|\Delta\mathbf{x}\| \leq \delta$

  - **Weak**: objective function remains same in some directions, and increases locally in other directions

  Point $\mathbf{x}^\star$ is a weak minimum of a function $F(\mathbf{x})$ if is not a strong minimum and a scalar $\delta > 0$ exists such that $F(\mathbf{x}^\star) \leq F(\mathbf{x}^\star + \Delta\mathbf{x})$ for all $\Delta\mathbf{x}$ such that $0 < \|\Delta\mathbf{x}\| \leq \delta$

- Note that a minimum is a **unique global minimum** if the definitions hold for $\delta = \infty$. Otherwise these are **local** minima.

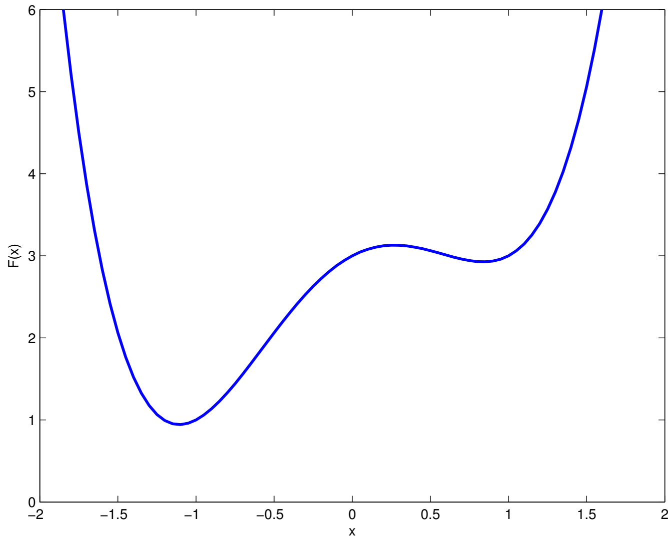


Figure 1.1: $F(x) = x^4 - 2x^2 + x + 3$ with local and global minima

First Order Conditions

- If $F(\mathbf{x})$ has continuous second derivatives, can approximate function in the neighborhood of an arbitrary point using Taylor series:

$$F(\mathbf{x}+\Delta\mathbf{x}) \approx F(\mathbf{x}) + \Delta\mathbf{x}^T \mathbf{g}(\mathbf{x}) + \frac{1}{2}\Delta\mathbf{x}^T G(\mathbf{x})\Delta\mathbf{x} + \dots$$

where $\mathbf{g} \sim$ gradient of F and $G \sim$ second derivative of F

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{g} = \left(\frac{\partial F}{\partial \mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix}, G = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}$$

- **First-order condition** from first two terms (assume $\|\Delta\mathbf{x}\| \ll 1$)
  - Given **ambiguity of sign** of the term $\Delta\mathbf{x}^T\mathbf{g}(\mathbf{x})$, can only avoid cost decrease $F(\mathbf{x}+\Delta\mathbf{x}) < F(\mathbf{x})$ if $\mathbf{g}(\mathbf{x}^\star) = 0$
    $\Rightarrow$ Obtain further information from higher derivatives
  - $\mathbf{g}(\mathbf{x}^\star) = 0$ is a necessary and sufficient condition for a point to be a **stationary point** – a necessary, but not sufficient condition to be a minima.
  - Stationary point could also be a maximum or a saddle point.

- Additional conditions can be derived from the Taylor expansion if we set $\mathbf{g}(\mathbf{x}^\star) = 0$, in which case:

$$F(\mathbf{x}^\star + \Delta\mathbf{x}) \approx F(\mathbf{x}^\star) + \frac{1}{2}\Delta\mathbf{x}^T G(\mathbf{x}^\star)\Delta\mathbf{x} + \ldots$$

  – For a strong minimum, need $\Delta\mathbf{x}^T G(\mathbf{x}^\star)\Delta\mathbf{x} > 0$ for all $\Delta\mathbf{x}$, which is sufficient to ensure that $F(\mathbf{x}^\star + \Delta\mathbf{x}) > F(\mathbf{x}^\star)$.

  – To be true for arbitrary $\Delta\mathbf{x} \neq 0$, **sufficient** condition is that $G(\mathbf{x}^\star) > 0$ (PD). [1]

- Second order **necessary** condition for a strong minimum is that $G(\mathbf{x}^\star) \geq 0$ (PSD), because in this case the higher order terms in the expansion can play an important role, i.e.

$$\Delta\mathbf{x}^T G(\mathbf{x}^\star)\Delta\mathbf{x} = 0$$

but the third term in the Taylor series expansion is positive.

- **Summary**: require $\mathbf{g}(\mathbf{x}^\star) = 0$ and $G(\mathbf{x}^\star) > 0$ (sufficient) or $G(\mathbf{x}^\star) \geq 0$ (necessary)

[1] Positive Definite Matrix

Solution Methods

- Typically solve minimization problem using an iterative algorithm.
  - Given: An initial estimate of the optimizing value of $\mathbf{x} \Rightarrow \hat{\mathbf{x}}_k$ and a search direction $\mathbf{p}_k$
  - Find: $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k$, for some scalar $\alpha_k \neq 0$

- Sounds good, but there are some questions:
  - How find $\mathbf{p}_k$?
  - How find α_k ? $\Rightarrow$ "**line search**"
  - How find initial condition $\mathbf{x}_0$, and how sensitive is the answer to the choice?

- **Search direction:**
  - Taylor series expansion of $F(\mathbf{x})$ about current estimate $\hat{\mathbf{x}}_k$

$$
\begin{aligned}
F_{k+1} \equiv F(\hat{\mathbf{x}}_k + \alpha \mathbf{p}_k) &\approx F(\hat{\mathbf{x}}_k) + \frac{\partial F}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k) \\
&= F_k + \mathbf{g}_k^T(\alpha_k \mathbf{p}_k)
\end{aligned}
$$

    ◇ Assume that $\alpha_k > 0$, and to ensure function decreases (i.e. $F_{k+1} < F_k$), set

$$\mathbf{g}_k^T \mathbf{p}_k < 0$$

    ◇ $\mathbf{p}_k$'s that satisfy this property provide a **descent direction**
  - **Steepest descent** given by $\mathbf{p}_k = -\mathbf{g}_k$

- **Summary:** gradient search methods (first-order methods) using estimate updates of the form:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k - \alpha_k \mathbf{g}_k$$

Line Search

- Line Search - given a search direction, must decide how far to "step"
  - Expression $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ gives a new solution for all possible values of α - what is the right value to pick?
  - Note that $\mathbf{p}_k$ defines a slice through solution space – is a very specific combination of how the elements of $\mathbf{x}$ will change together.
- Would like to pick α_k to minimize $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$
  - Can do this line search in gory detail, but that would be very time consuming
    - ◇ Often want this process to be fast, accurate, and easy
    - ◇ Especially if you are not that confident in the choice of $\mathbf{p}_k$

- Consider simple problem: $F(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2$ with

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_0 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \Rightarrow \mathbf{x}_1 = \mathbf{x}_0 + \alpha \mathbf{p}_0 = \begin{bmatrix} 1 \\ 1 + 2\alpha \end{bmatrix}$$

which gives that $F = 1 + (1 + 2\alpha) + (1 + 2\alpha)^2$ so that

$$\frac{\partial F}{\partial \alpha} = 2 + 2(1 + 2\alpha)(2) = 0$$

with solution $\alpha^\star = -3/4$ and $\mathbf{x}_1 = [1 \ \ -1/2]^T$

  - This is hard to generalize this to N-space – need a better approach

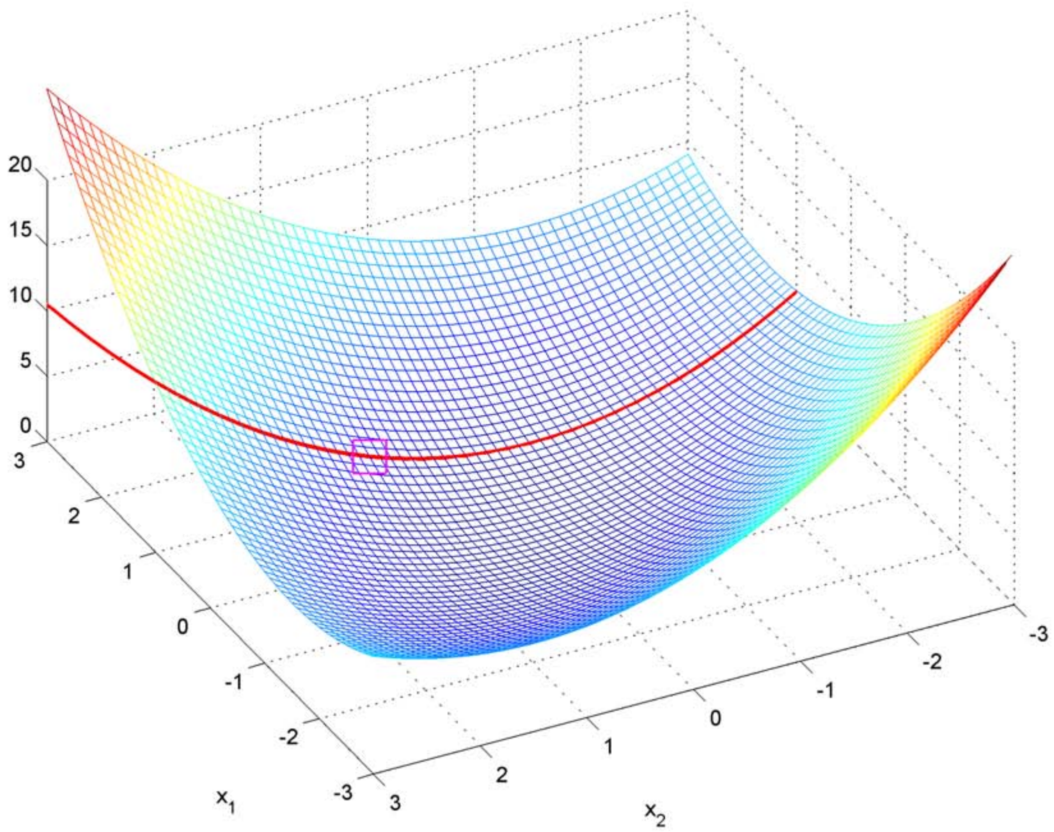


Figure 1.2: $F(x) = x_1^2 + x_1x_2 + x_2^2$ doing a line search in arbitrary direction

Line Search – II

- First step: search along the line until you think you have bracketed a "local minimum"

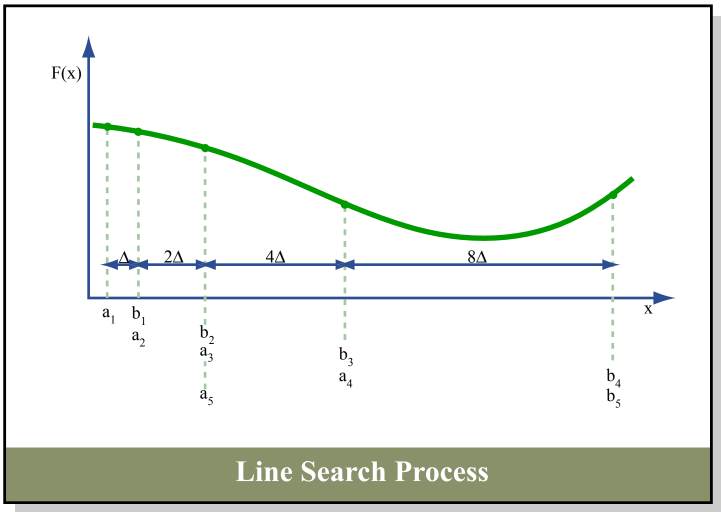


Figure by MIT OpenCourseWare.

Figure 1.3: Line search process

- Once you think you have a bracket of the local min – what is the smallest number of function evaluations that can be made to reduce the size of the bracket?
  - Many ways to do this:
    - ◇ Golden Section Search
    - ◇ Bisection
    - ◇ Polynomial approximations
  - First 2 have linear convergence, last one has "superlinear"

- Polynomial approximation approach
  - Approximate function as quadratic/cubic in the interval and use the minimum of that polynomial as the estimate of the local min.
  - Use with care since it can go very wrong – but it is a good termination approach.

- Cubic fits are a favorite:

$$\begin{aligned}\hat{F}(x) &= px^3 + qx^2 + rx + s \\ \hat{g}(x) &= 3px^2 + 2qx + r \ (\ = 0 \text{ at min})\end{aligned}$$

  Then $x^\star$ is the point (pick one) $x^\star = (-q \pm (q^2 - 3pr)^{1/2})/(3p)$ for which $\hat{G}(x^\star) = 6px^\star + 2q > 0$

- Great, but how do we find $x^\star$ in terms of what we know ($F(x)$ and $g(x)$ at the end of the bracket $[a, b]$)?

$$x^\star = a + (b-a)\left[1 - \frac{g_b + v - w}{g_b - g_a + 2v}\right]$$

  where

$$v = \sqrt{w^2 - g_a g_b} \quad \text{and} \quad w = \frac{3}{b-a}(F_a - F_b) + g_a + g_b$$

Figure 1.4: Cubic line search [Scales, pg. 40]

- **Observations**:
  - Tends to work well "near" a function local minimum (good convergence behavior)
  - But can be very poor "far away" $\Rightarrow$ use a hybrid approach of bisection followed by cubic.
- **Rule of thumb**: do not bother making the linear search too accurate, especially at the beginning
  - A waste of time and effort
  - Check the min tolerance – and reduce it as it you think you are approaching the overall solution.

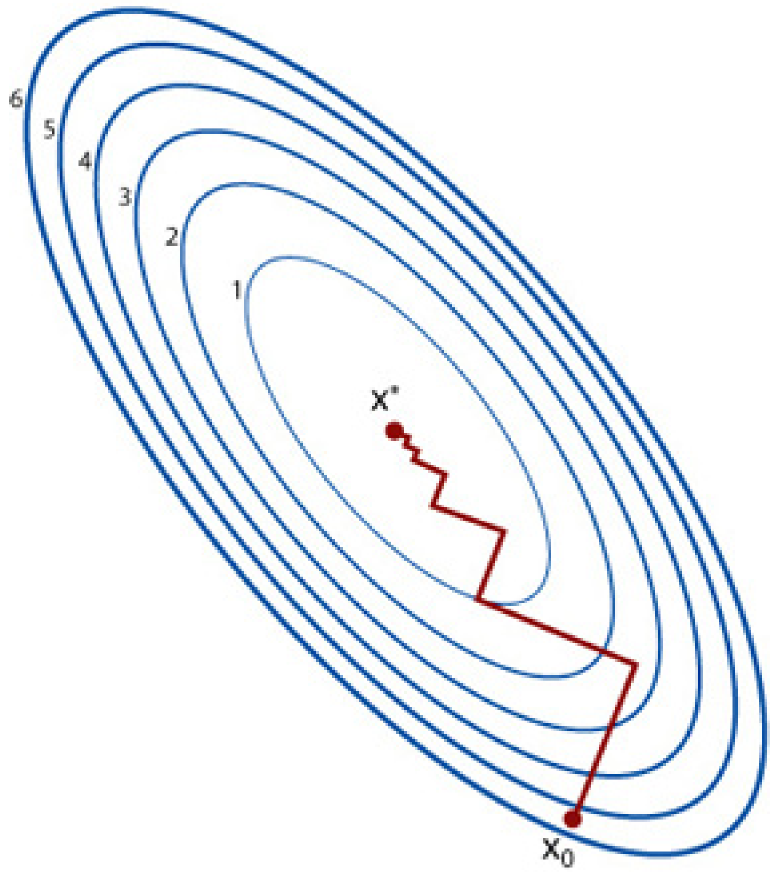


Figure by MIT OpenCourseWare.

Figure 1.5: zig-zag typical of steepest decent line searches

Second Order Methods

- Second order methods typically provide faster termination
  - Assume F is quadratic, and expand gradient $\mathbf{g}_{k+1}$ at $\hat{\mathbf{x}}_{k+1}$

$$\begin{aligned}\mathbf{g}_{k+1} \equiv \mathbf{g}(\hat{\mathbf{x}}_k + \mathbf{p}_k) &= \mathbf{g}_k + G_k(\hat{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_k) \\ &= \mathbf{g}_k + G_k \mathbf{p}_k\end{aligned}$$

  where there are no other terms because of the assumption that F is quadratic and

$$\mathbf{x}_k = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{g}_k = \left(\frac{\partial F}{\partial \mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix}_{\hat{\mathbf{x}}_k}$$

$$G_k = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}_{\hat{\mathbf{x}}_k}$$

  - So for $\hat{\mathbf{x}}_{k+1}$ to be at the minimum, need $\mathbf{g}_{k+1} = 0$, so that

$$\mathbf{p}_k = -G_k^{-1} \mathbf{g}_k$$

- Problem is that $F(\mathbf{x})$ typically not quadratic, so the solution $\hat{\mathbf{x}}_{k+1}$ is not at the minimum $\Rightarrow$ need to iterate

- Note that for a complicated $F(\mathbf{x})$, we may not have explicit gradients (should always compute them if you can)
  - But can always approximate them using finite difference techniques – but pretty expensive to find G that way
  - Use Quasi-Newton approximation methods instead, such as **BFGS** (Broyden-Fletcher-Goldfarb-Shanno)

FMINUNC Example

- Function minimization without constraints
  - Does quasi-Newton and gradient search
  - No gradients need to be formed
  - Mixture of cubic and quadratic line searches
- Performance shown on a complex function by Rosenbrock

$$F(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

  - Start at $x = [-1.9\ 2]$. Known global min it is at $x = [1\ 1]$

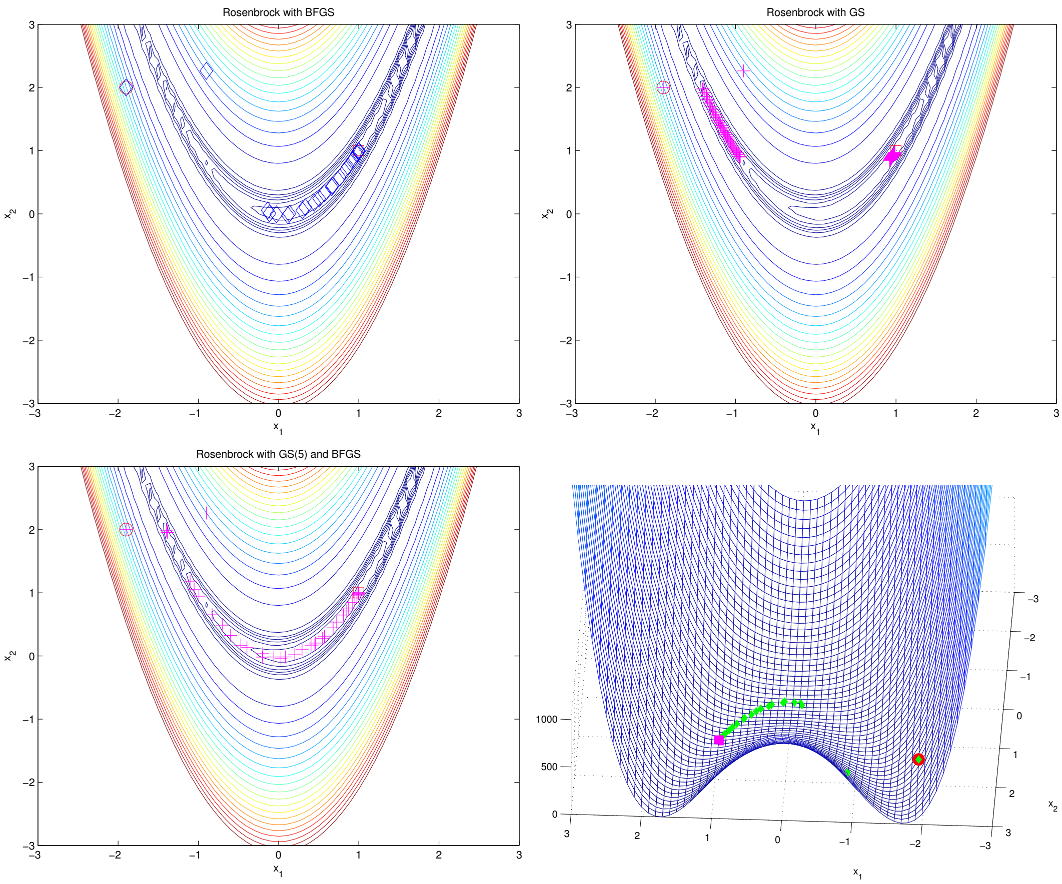
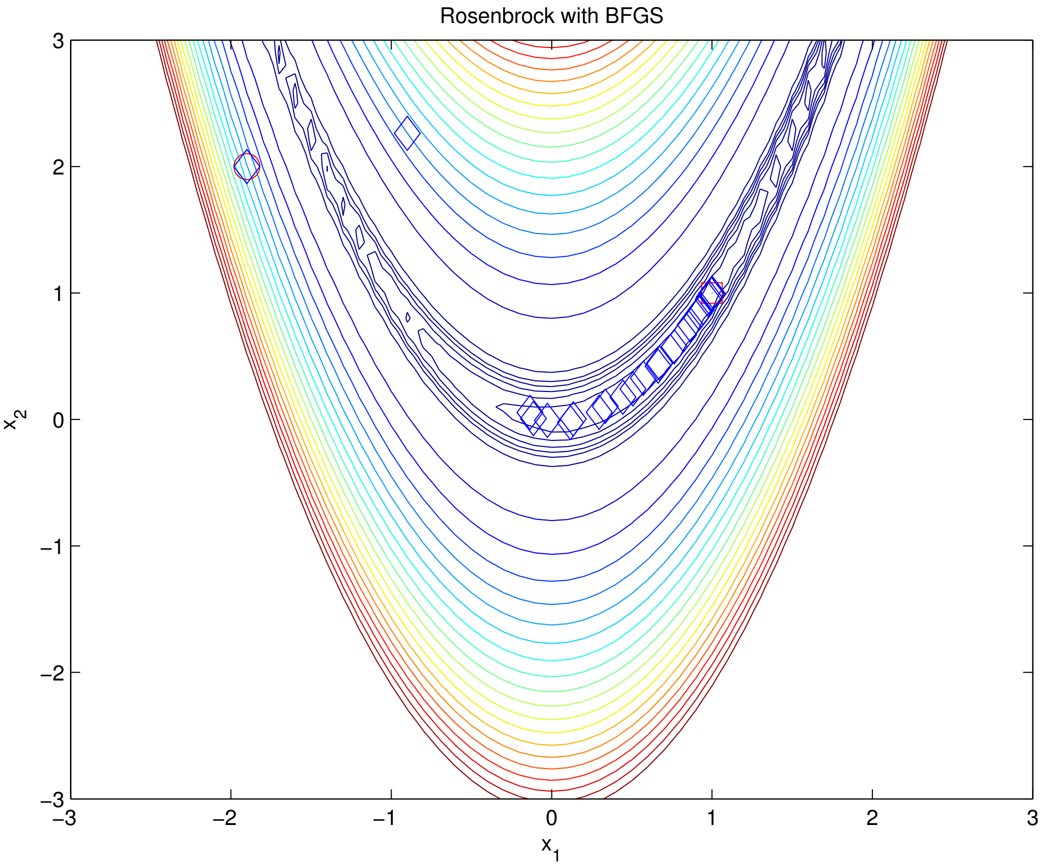
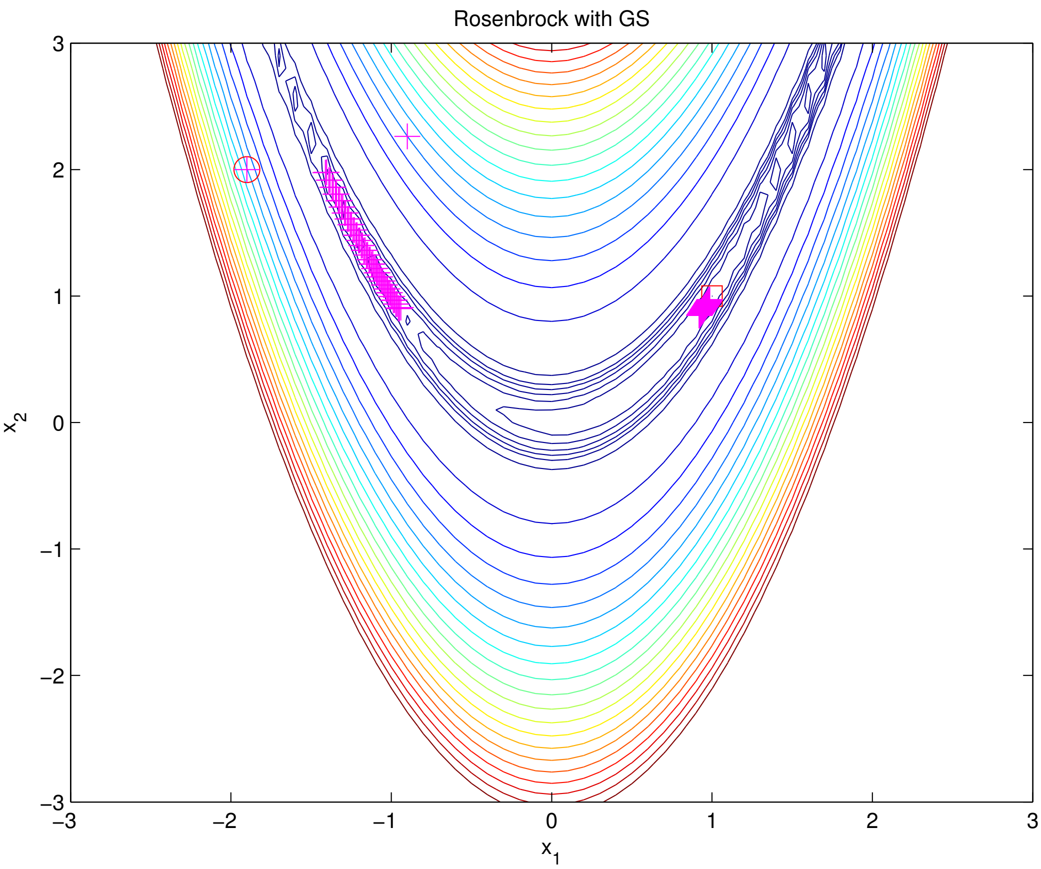
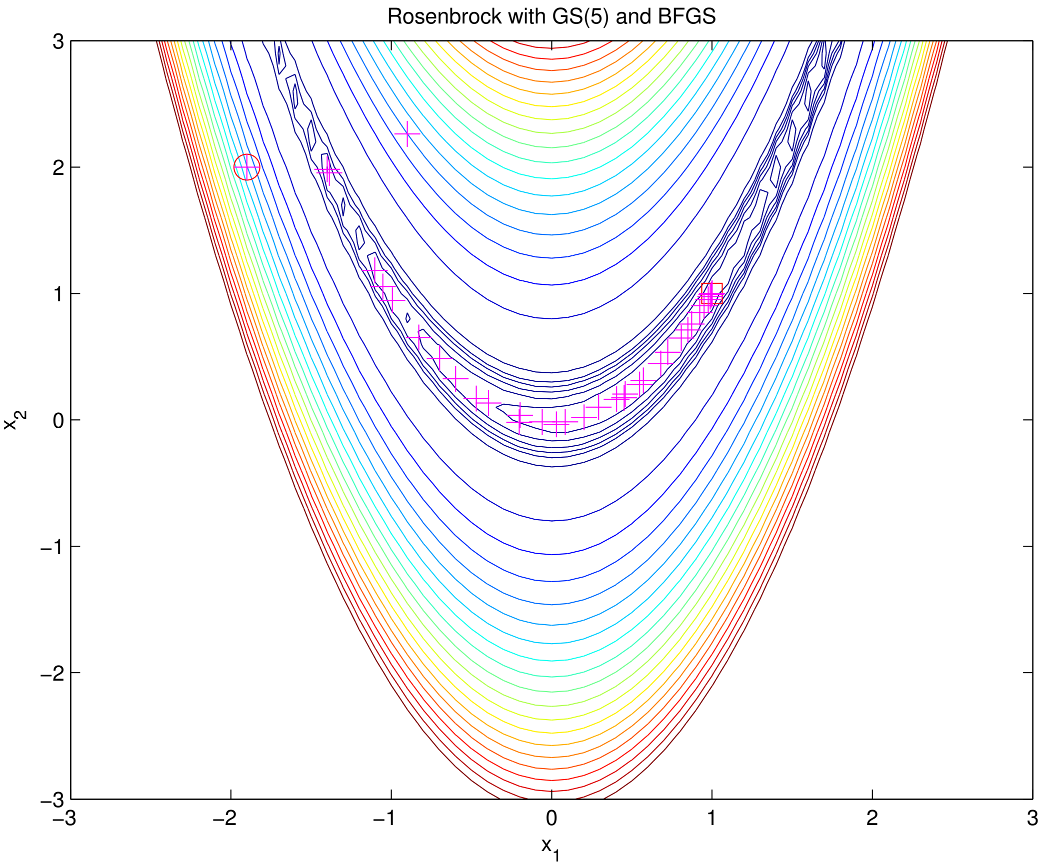


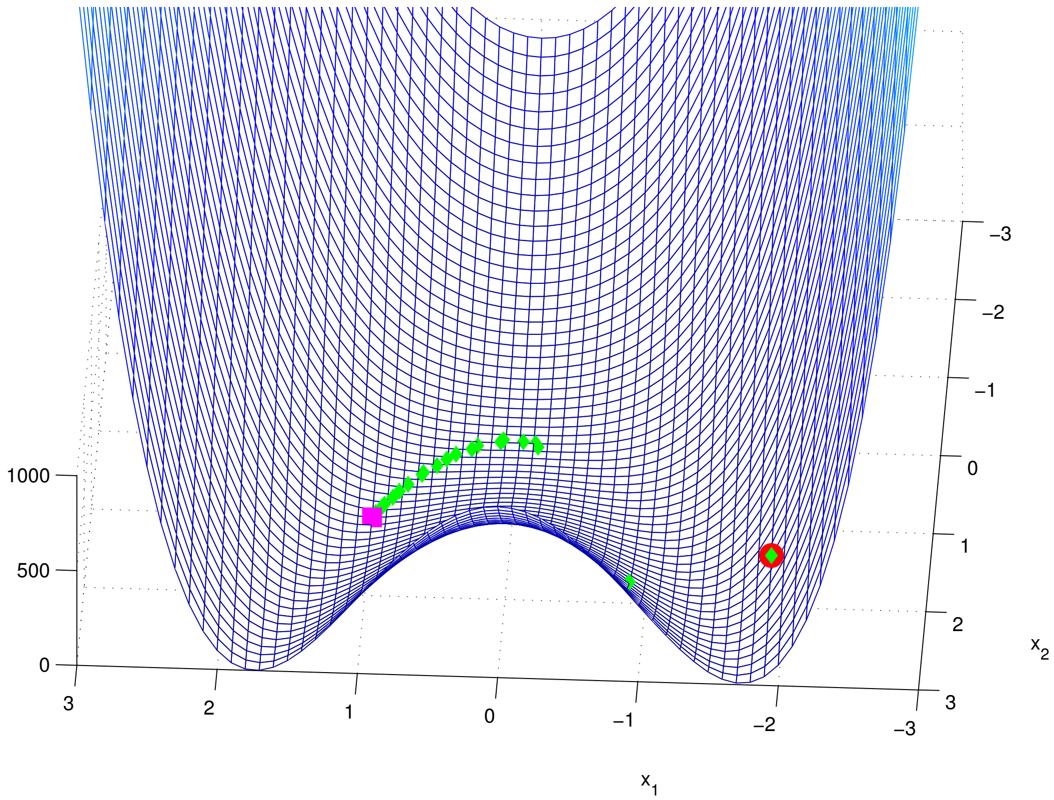
Figure 1.6: How well do the algorithms work?

- Quasi-Newton (BFGS) does well - gets to optimal solution in 26 iterations (35 ftn calls), but gradient search (steepest descent) fails (very close though), even after 2000 function calls (550 iterations).


Rosenbrock with BFGS
x1
x2


Rosenbrock with GS
x1
x2


Rosenbrock with GS(5) and BFGS
x1
x2


x1
x2
1000
500
0

- **Observations:**

1. Typically not a good idea to start the optimization with QN, and I often find that it is better to do GS for 100 iterations, and then switch over to QN for the termination phase.

2. $\hat{\mathbf{x}}_0$ tends to be very important – standard process is to try many different cases to see if you can find consistency in the answers.

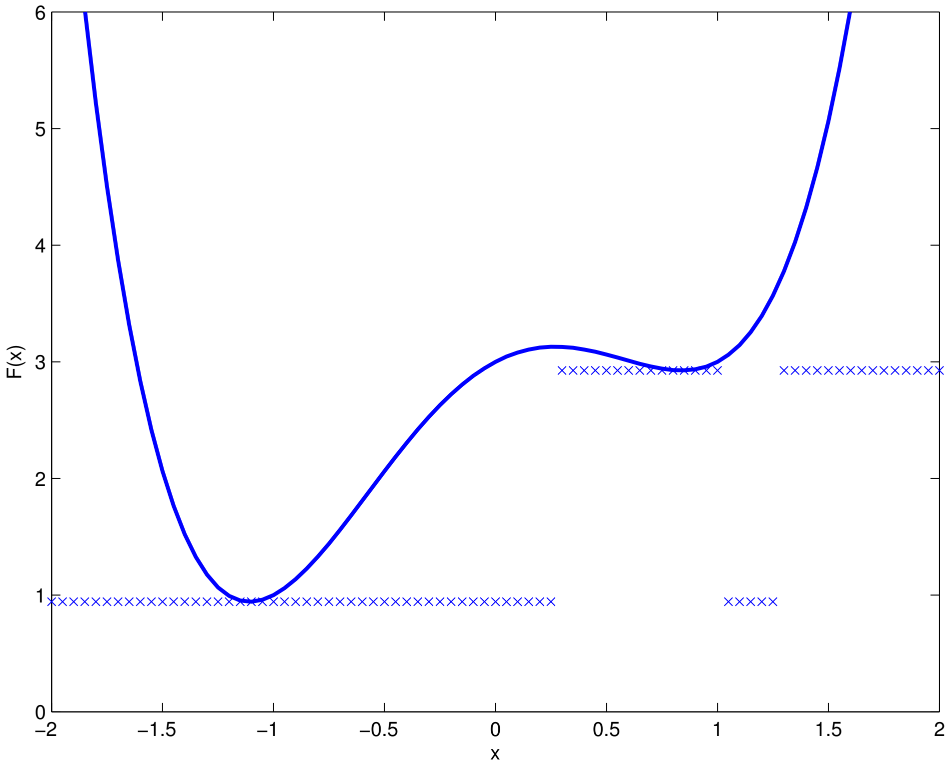


Figure 1.7: Shows how the point of convergence changes as a function of the initial condition.

3. Typically the convergence is to a local minimum and can be slow

4. Are there any guarantees on getting a good final answer in a reasonable amount of time? Typically yes, but not always.

Unconstrained Optimization Code

```
function [F,G]=rosen(x)
%global xpath

%F=100*(x(1)^2-x(2))^2+(1-x(1))^2;

if size(x,1)==2, x=x'; end

F=100*(x(:,2)-x(:,1).^2).^2+(1-x(:,1)).^2;
G=[100*(4*x(1)^3-4*x(1)*x(2))+2*x(1)-2; 100*(2*x(2)-2*x(1)^2)];

return

%
% Main calling part below - uses function above
%

global xpath

clear FF
x1=[-3:.1:3]'; x2=x1; N=length(x1);
for ii=1:N,
    for jj=1:N,
        FF(ii,jj)=rosen([x1(ii) x2(jj)]');
    end,
end

% quasi-newton
%
xpath=[];t0=clock;
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','bfgs','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity',...
    'MaxFunEvals',150,'OutputFcn', @outftn);

x0=[-1.9 2]';

xout1=fminunc('rosen',x0,opt) % quasi-newton
xbfgs=xpath;

% gradient search
%
xpath=[];
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','steepdesc','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity','MaxFunEvals',2000,'MaxIter',1000,'OutputFcn', @outftn);
xout=fminunc('rosen',x0,opt)
xgs=xpath;


% hybrid GS and BFGS
%
xpath=[];
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','steepdesc','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity','MaxFunEvals',5,'OutputFcn', @outftn);
xout=fminunc('rosen',x0,opt)
opt=optimset('fminunc');
opt=optimset(opt,'Hessupdate','bfgs','gradobj','on','Display','Iter',...
    'LargeScale','off','InitialHessType','identity','MaxFunEvals',150,'OutputFcn', @outftn);
xout=fminunc('rosen',xout,opt)

xhyb=xpath;

figure(1);clf
contour(x1,x2,FF',[0:2:10 15:50:1000])
hold on
plot(x0(1),x0(2),'ro','Markersize',12)
```

```
plot(1,1,'rs','Markersize',12)
plot(xbfgs(:,1),xbfgs(:,2),'bd','Markersize',12)
title('Rosenbrock with BFGS')
hold off
xlabel('x_1')
ylabel('x_2')
print -depsc rosen1a.eps;jpdf('rosen1a')

figure(2);clf
contour(x1,x2,FF',[0:2:10 15:50:1000])
hold on
xlabel('x_1')
ylabel('x_2')
plot(x0(1),x0(2),'ro','Markersize',12)
plot(1,1,'rs','Markersize',12)
plot(xgs(:,1),xgs(:,2),'m+','Markersize',12)
title('Rosenbrock with GS')
hold off
print -depsc rosen1b.eps;jpdf('rosen1b')

figure(3);clf
contour(x1,x2,FF',[0:2:10 15:50:1000])
hold on
xlabel('x_1')
ylabel('x_2')
plot(x0(1),x0(2),'ro','Markersize',12)
plot(1,1,'rs','Markersize',12)
plot(xhyb(:,1),xhyb(:,2),'m+','Markersize',12)
title('Rosenbrock with GS(5) and BFGS')
hold off
print -depsc rosen1c.eps;jpdf('rosen1c')

figure(4);clf
mesh(x1,x2,FF')
hold on
plot3(x0(1),x0(2),rosen(x0')+5,'ro','Markersize',12,'MarkerFaceColor','r')
plot3(1,1,rosen([1 1]),'ms','Markersize',12,'MarkerFaceColor','m')
plot3(xbfgs(:,1),xbfgs(:,2),rosen(xbfgs)+5,'gd','MarkerFaceColor','g')
%plot3(xgs(:,1),xgs(:,2),rosen(xgs)+5,'m+')
hold off
axis([-3 3 -3 3 0 1000])
hh=get(gcf,'children');
xlabel('x_1')
ylabel('x_2')
set(hh,'View',[-177 89.861],'CameraPosition',[-0.585976 11.1811 5116.63]);%
print -depsc rosen2.eps;jpdf('rosen2')

```

```
function stop = outftn(x, optimValues, state)

global xpath
stop=0;
xpath=[xpath;x'];

return
```
