Non-Linear Programming (NLP): Multivariable, Unconstrained

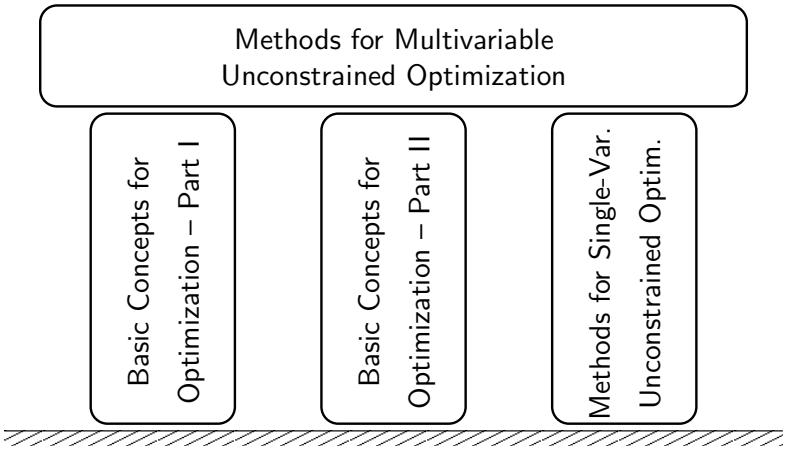
Benoît Chachuat <benoit@mcmaster.ca>

McMaster University
Department of Chemical Engineering

ChE 4G03: Optimization in Chemical Engineering

Solving Multivariable, Unconstrained NLPs

Prerequisites:



Solving Multivariable, Unconstrained NLPs

- "Aren't multivariable problems always constrained?" — Yes, but the optimum is sometimes unconstrained
- "Won't the methods for multivariable constrained problems work in the unconstrained case?" — Yes

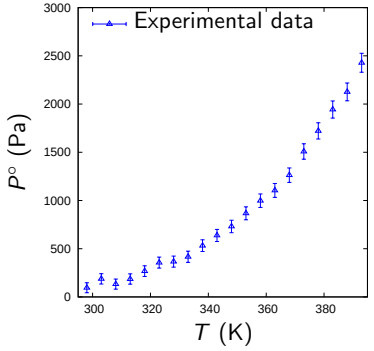
But,

1. A few important problems are unconstrained
   - E.g., nonlinear regression
2. This will give us insight into constrained optimization solution techniques
   - E.g., use of linesearch techniques
3. Unconstrained optimization is a subproblem for many nonlinear, constrained, multivariable optimization methods and software!
   - E.g., penalty and barrier methods

Example of Multivariable, Unconstrained NLP

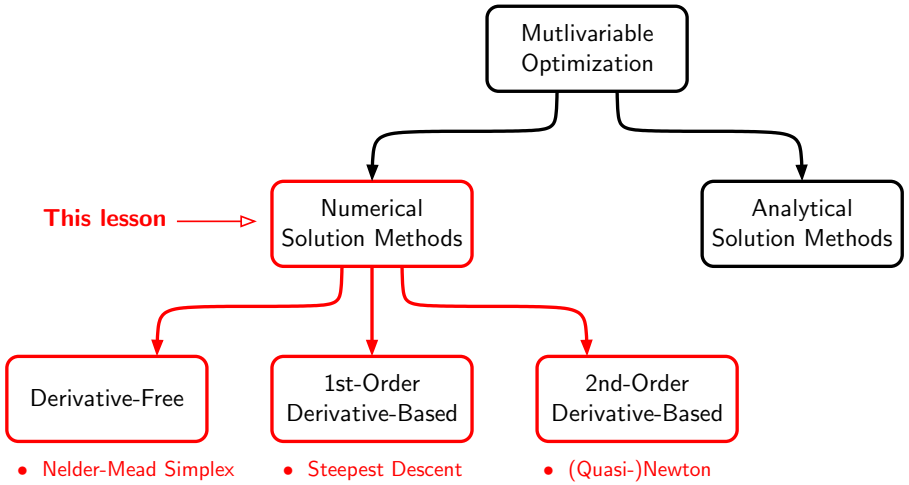
Problem: Calibrate an empirical model relating temperature (T) and vapor pressure (P°) for a pure component (using data)

Clausius-Clapeyron Approximate Equation: $\ln P^\circ = A - \dfrac{B}{C+T}$



T_k (K)	P_k° (Pa)	T_k (K)	P_k° (Pa)
298	95.2	348	730.8
303	187.4	353	867.0
308	133.1	358	997.7
313	185.9	363	1104.5
318	267.7	368	1262.6
323	356.0	373	1507.8
328	366.2	378	1721.8
333	416.8	383	1943.3
338	531.6	388	2126.2
343	637.7	393	2428.2

Outline



For additional details, see Rardin (1998), Chapter 13.5-13.8

Continuous Improving Search Algorithm

- **Step 0: Initialization**
  - Choose any starting feasible point $\mathbf{x}^0$ and let index $k \leftarrow 0$.
- **Step 1: Move Direction**
  - If no improving feasible direction $\Delta\mathbf{x}$ exists at current point $\mathbf{x}^k$, stop.
  - Otherwise, construct an improving feasible direction at $\mathbf{x}^k$ as $\Delta\mathbf{x}^{k+1}$.
- **Step 2: Step Size**
  - If there is no limit on step sizes for which direction $\Delta\mathbf{x}^{k+1}$ continues to both improve the objective function and retain feasibility, stop — The model is unbounded.
  - Otherwise, choose the largest step size α^{k+1}.
- **Step 3: Update**
  - $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha^{k+1}\Delta\mathbf{x}^{k+1}$
  - Increment index $k \leftarrow k+1$ and return to step 1.

Remarks:

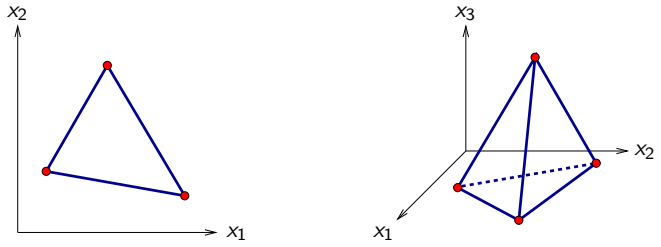
- This basic algorithm may terminate at a suboptimal point
- Moreover, it does not distinguish between local and global optima

Nelder-Mead Derivative-Free Search

Iteratively consider the cost value at $n+1$ carefully chosen points:

- A simplex is a geometrical figure formed by $n+1$ points, $\{\mathbf{x}^1, \ldots, \mathbf{x}^{n+1}\}$, in an n-dimensional system

2-Dimensional Case ($n=2$) 3-Dimensional Case ($n=3$)



- The objective function is evaluated at each point $\mathbf{x}^i$, $i=1,\ldots,n$, then the points are rearranged in nonimproving sequence:
  - **Maximize Problem:** $f(\mathbf{x}^1) > f(\mathbf{x}^2) > \cdots > f(\mathbf{x}^{n+1})$
  - **Minimize Problem:** $f(\mathbf{x}^1) < f(\mathbf{x}^2) < \cdots < f(\mathbf{x}^{n+1})$

Nelder-Mead Derivative-Free Search (cont'd)

How do we construct a search directions $\Delta\mathbf{x}$ without the aid of (partial) derivatives?

- Nelder-Mead procedure adopts an away-from-worst approach

Away-from-Worst Search Direction

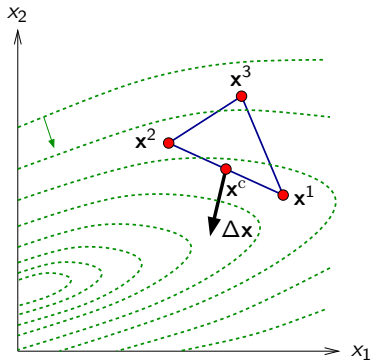
For an ordered set of points $\{\mathbf{x}^1, \ldots, \mathbf{x}^{n+1}\}$:

- Best-n centroid defined as

$$\mathbf{x}^c \triangleq \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^i$$

- Search direction $\Delta\mathbf{x}$ defined so as to *move away from worst point*,

$$\Delta\mathbf{x} \triangleq \mathbf{x}^c - \mathbf{x}^{n+1}$$



Nelder-Mead Derivative-Free Search (cont'd)

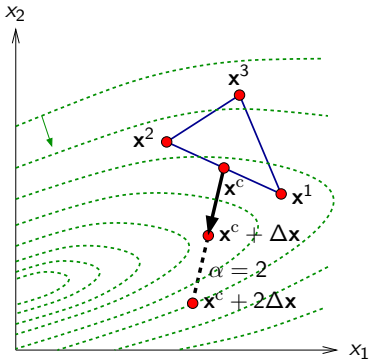
How do we decide the step size α in the direction $\Delta\mathbf{x}$?

- Nelder-Mead procedure first reflects about centroid $\mathbf{x}^c$ (i.e., with step size $\alpha = 1$)
- If the new point $\mathbf{x}^c + \Delta\mathbf{x}$ is neither better than $\mathbf{x}^1$ nor worse than $\mathbf{x}^n$, it is adopted without further trials

Nelder-Mead Expansion

In case $\mathbf{x}^c + \Delta\mathbf{x}$ is a new best:

- Attempt expansion with step $\alpha = 2$
- Accept expansion $\mathbf{x}^c + 2\Delta\mathbf{x}$ only in case of an improvement!

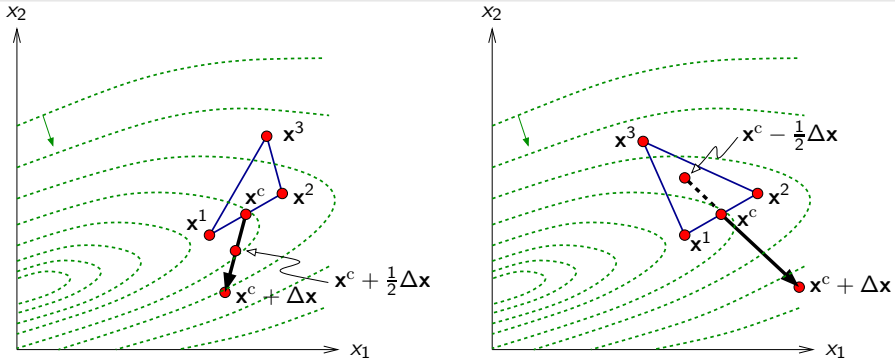


Nelder-Mead Derivative-Free Search (cont'd)

Nelder-Mead Contraction

In case $\mathbf{x}^c + \Delta\mathbf{x}$ would still remain worse than $\mathbf{x}^n$:

1. Attempt contraction with $\alpha = \frac{1}{2}$, if $f(\mathbf{x}^c + \Delta\mathbf{x})$ better than $f(\mathbf{x}^{n+1})$
2. Attempt contraction with $\alpha = -\frac{1}{2}$, otherwise
3. Accept contraction $\mathbf{x}^c \pm \frac{1}{2}\Delta\mathbf{x}$ only in case of an improvement!

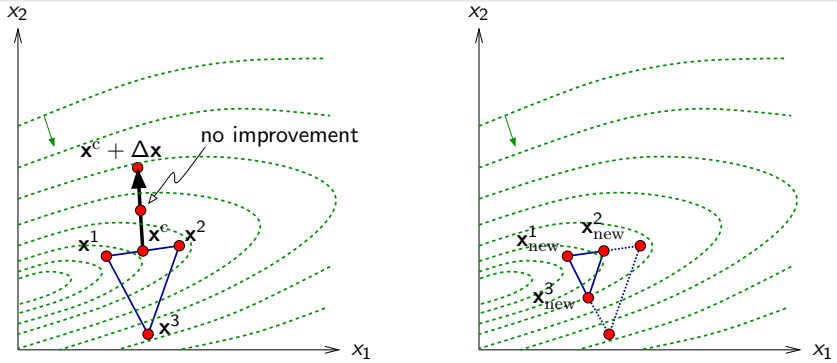


Nelder-Mead Derivative-Free Search (cont'd)

Nelder-Mead Shrinking

In case neither the reflection point nor a contraction alternative yields an improving point w.r.t. $\mathbf{x}^n$, the procedure shrinks the whole array toward current best $\mathbf{x}^1$,

$$\mathbf{x}^i_{\text{new}} \triangleq \frac{1}{2}\left[\mathbf{x}^1 + \mathbf{x}^i\right], \quad \text{for all } i = 2, \ldots, n+1$$



Nelder-Mead Derivative-Free Search Algorithm

- **Step 0: Initialization**
  - Choose $(n+1)$ distinct points $\{\mathbf{x}^1, \ldots, \mathbf{x}^{n+1}\}$, and evaluate $f(\mathbf{x}^1), \ldots, f(\mathbf{x}^{n+1})$; set stopping tolerance $\epsilon > 0$
- **Step 1: Move Direction**
  - Rearrange the $\mathbf{x}^i$ in nonimproving sequence, and compute best-n centroid:

$$\mathbf{x}^c \triangleq \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^i$$

  - If $|f(\mathbf{x}^i) - f(\mathbf{x}^c)| < \epsilon$, stop — Report the best of current $\mathbf{x}^c$, $\mathbf{x}^1$
  - Otherwise, compute away-from-worst move direction as

$$\Delta\mathbf{x} \triangleq \mathbf{x}^c - \mathbf{x}^{n+1};$$

proceed to step 2

Nelder-Mead Derivative-Free Search Algorithm (cont'd)

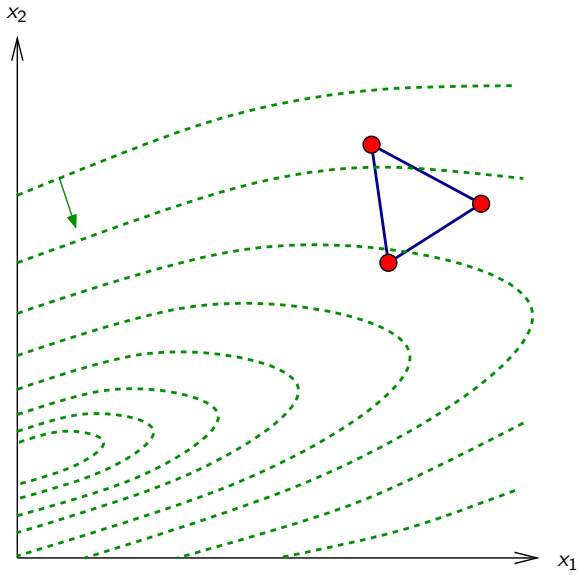
- **Step 2: Step Size**
  - If $f(\mathbf{x}^c + \Delta\mathbf{x})$ improves on current best $f(\mathbf{x}^1)$: expand by trying $f(\mathbf{x}^c + 2\Delta\mathbf{x})$; set $\alpha = 2$ if further improvement is obtained, $\alpha = 1$ otherwise; proceed to step 3
  - If $f(\mathbf{x}^c + \Delta\mathbf{x})$ does not improve on second worst $f(\mathbf{x}^n)$: contract by trying $f(\mathbf{x}^c + \frac{1}{2}\Delta\mathbf{x})$ if $f(\mathbf{x}^c + \Delta\mathbf{x})$ is better than $f(\mathbf{x}^{n+1})$, or trying $f(\mathbf{x}^c - \frac{1}{2}\Delta\mathbf{x})$ otherwise; set $\alpha = \pm\frac{1}{2}$ if improvement is obtained w.r.t. $f(\mathbf{x}^{n+1})$; proceed to step 3
- **Step 3: Update**
  - If contraction is nonimproving, shrink the current simplex toward best $\mathbf{x}^1$ by setting:

$$\mathbf{x}^i_{\text{new}} \triangleq \frac{1}{2}\left(\mathbf{x}^1 - \mathbf{x}^i\right), \quad \text{for all } i = 2, \ldots, n+1;$$

    compute new function values $f(\mathbf{x}^2), \ldots, f(\mathbf{x}^{n+1})$; return to step 1
  - Otherwise, replace $\mathbf{x}^{n+1}$ in the set of points by $\mathbf{x}^c + \alpha\Delta\mathbf{x}$; return to step 1

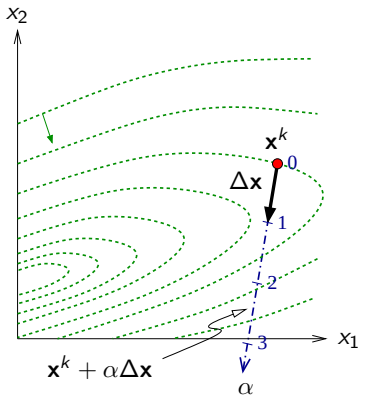
Applying Nelder-Mead Derivative-Free Search

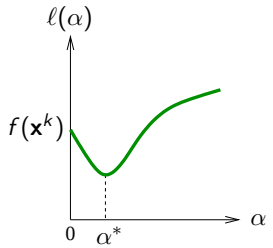
Class Exercise: Sketch a few iterations of Nelder-Mead search



Derivative-Based Methods: Principles (Minimize Case)

- Both first- and second-order methods determine the **search direction**, $\Delta\mathbf{x}$, using **derivatives** (gradient & Hessian)
  - Perform a search along the direction $\Delta\mathbf{x}$
  - How far should we move?
- **Linesearch:** a 1-d search in a multidimensional set





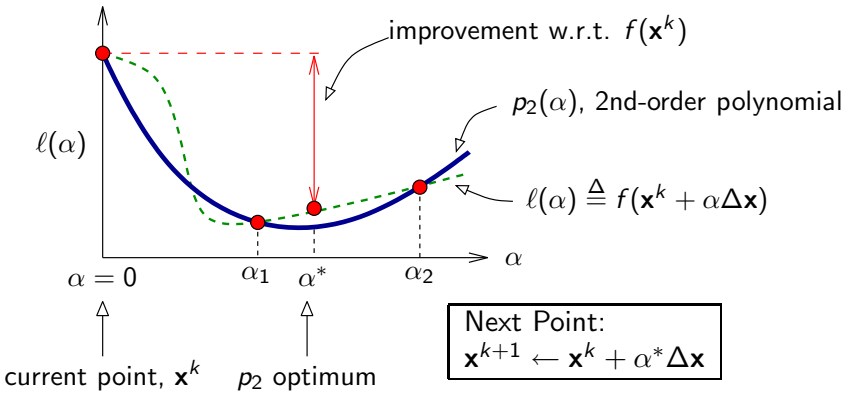
Linesearch Problem:

$$\min_{\alpha \geq 0} \ \ell(\alpha) \triangleq f(\mathbf{x}^k + \alpha\Delta\mathbf{x})$$

Derivative-Based Methods: Linesearch (Minimize Case)

For example, use interpolation methods such as **quadratic fit search**

- Typically, perform only one iteration of linesearch at each point $\mathbf{x}^k$
- Must achieve a minimum amount of improvement, not necessarily the best possible improvement



Next Point:
$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha^*\Delta\mathbf{x}$

Applying Linesearch for Derivative-Based Methods

Class Exercise: Perform a linesearch on $f(\mathbf{x})$ in the direction $\Delta\mathbf{x}$ at the point $\mathbf{x}^\circ$, with:

$$f(\mathbf{x}) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2, \qquad \Delta\mathbf{x} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \qquad \mathbf{x}^\circ = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

First-Order Methods: Steepest Descent

At any current point $\mathbf{x}^k$ with gradient $\nabla f(\mathbf{x}^k \neq \mathbf{0}$, 1st-order methods pursue move direction:

$$\Delta\mathbf{x} \triangleq \pm\nabla f(\mathbf{x}^k)$$

($+$ for a maximize, $-$ for a minimize)

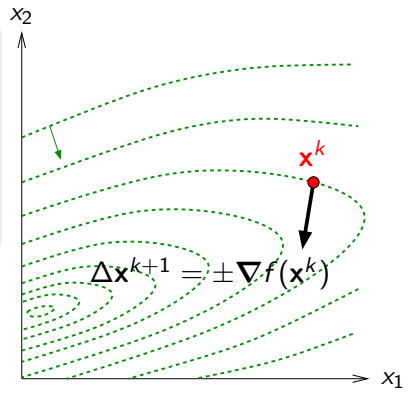
Steepest Descent/Ascent Direction

At any $\mathbf{x}^k$ with $\nabla f(\mathbf{x}^k) \neq \mathbf{0}$, the locally steepest rate of improvement is obtained with:

$$\Delta\mathbf{x}^{k+1} \triangleq \pm\nabla f(\mathbf{x}^k)$$

Can you see/explain why?

This is a **local** property!



Steepest Descent Algorithm (Minimize Case)

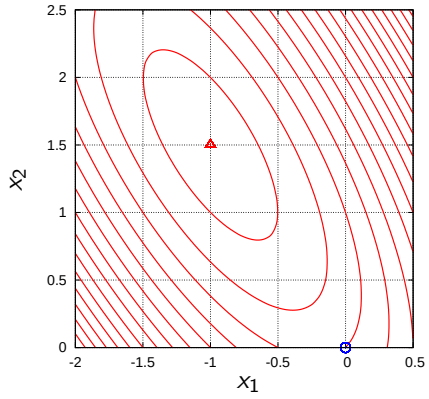
- **Step 0: Initialization**
  - Choose an initial guess $\mathbf{x}^0$, as well as stopping tolerance $\epsilon > 0$
  - Set $k \leftarrow 0$
- **Step 1: Gradient**
  - Compute objective function gradient $\nabla f(\mathbf{x}^k)$
- **Step 2: Stopping**
  - If $\|\nabla f(\mathbf{x}^k)\| < \epsilon$, stop — report $\mathbf{x}^k$ as an approximate solution (stationary point)
- **Step 3: Steepest Descent Step**
  - Direction: Set move direction, $\Delta\mathbf{x}^{k+1} \leftarrow -\nabla f(\mathbf{x}^k)$
  - Linesearch: Solve 1-d linesearch problem (at least approximately), $\min_\alpha \ell(\alpha) \triangleq f(\mathbf{x}^k + \alpha\Delta\mathbf{x}^{k+1})$, to compute the step α^{k+1}
  - Update: $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha^{k+1}\Delta\mathbf{x}^{k+1}$
  - Increment $k \leftarrow k+1$ and return to step 1

Applying Steepest-Descent Search

Class Exercise: Consider the problem to minimize

$$f(\mathbf{x}) \triangleq x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$$

- Calculate the steepest descent direction at point $\mathbf{x}^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$
- Apply linesearch along that direction to determine $\mathbf{x}^1$



Pros and Cons of Steepest-Descent Search

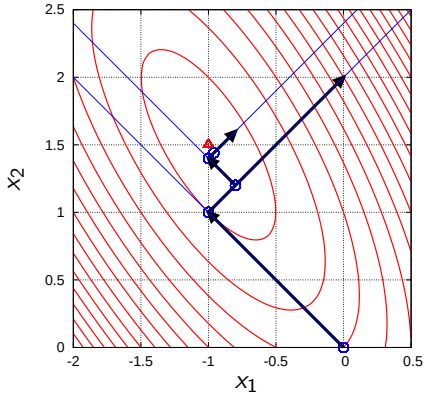
Pros:

- Approach is appealingly straightforward
- Approach is robust (following the steepest descent direction *always* yields improvement)

Cons:

- Can converge to a saddle point
- Convergence is typically slow due to zigzagging, especially near an optimal solution
- Sensitive to numerical errors

More sophistication is required!



Second-Order Methods: Newton Step

- **Idea:** Incorporate second-order information to enhance convergence
- **How?** Construct, then optimize, the quadratic approximation p_2 of f passing through a current iterate $\mathbf{x}^k$,

$$p_2(\mathbf{x}^k + \Delta\mathbf{x}) \triangleq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\mathsf{T}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\mathsf{T}\mathbf{H}(\mathbf{x}^k)\Delta\mathbf{x}$$

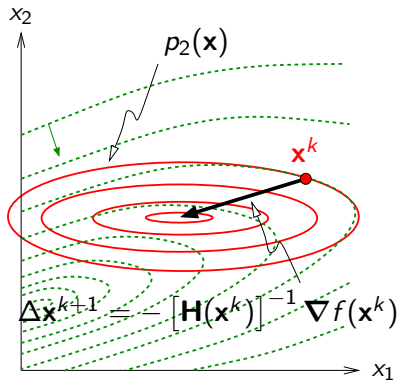
Newton Step

At any $\mathbf{x}^k$ with $\nabla f(\mathbf{x}^k) \neq \mathbf{0}$, the Newton step $\Delta\mathbf{x}^{k+1}$ is obtained such that:

$$\nabla_{\Delta\mathbf{x}}\Big[p_2(\mathbf{x}^k + \Delta\mathbf{x})\Big] = \mathbf{0}$$

That is, one has to solve the linear system:

$$\mathbf{H}(\mathbf{x}^k)\,\Delta\mathbf{x} = -\nabla f(\mathbf{x}^k)$$



Newton Search Algorithm (Minimize Case)

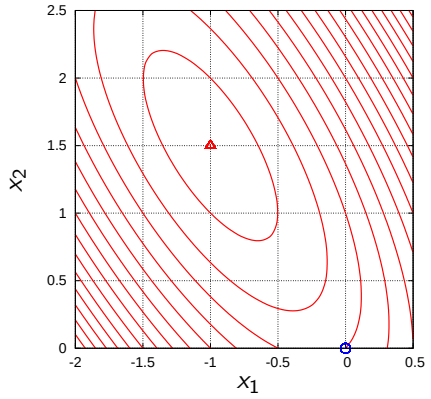
- **Step 0: Initialization**
  - Choose an initial guess $\mathbf{x}^0$, as well as stopping tolerance $\epsilon > 0$
  - Set $k \leftarrow 0$
- **Step 1: Derivatives**
  - Compute objective function gradient $\nabla f(\mathbf{x}^k)$ and Hessian matrix $\mathbf{H}(\mathbf{x}^k)$
- **Step 2: Stopping**
  - If $\left\|\nabla f(\mathbf{x}^k)\right\| < \epsilon$, stop — report $\mathbf{x}^k$ as an approximate solution (stationary point)
- **Step 3: Newton Step**
  - Direction: Calculate Newton move $\Delta\mathbf{x}^{k+1}$ by solving the linear system $\mathbf{H}(\mathbf{x}^k)\,\Delta\mathbf{x} = -\nabla f(\mathbf{x}^k)$
  - Linesearch: Solve 1-d linesearch problem (at least approximately), $\min_\alpha \ell(\alpha) \triangleq f(\mathbf{x}^k + \alpha\Delta\mathbf{x}^{k+1})$, to compute the step α^{k+1}
  - Update: $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha^{k+1}\Delta\mathbf{x}^{k+1}$
  - Increment $k \leftarrow k+1$ and return to step 1

Applying Newton Search

Class Exercise: Consider the problem to minimize

$$f(\mathbf{x}) \triangleq x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$$

- Calculate the Newton step $\Delta\mathbf{x}^1$ at point $\mathbf{x}^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ — **Why?**



Pros and Cons of Newton Search

Pros:

- Excellent performance of Newton search close to the optimum
- Less sensitive to numerical errors than steepest descent search

Cons:

- Very sensitive to starting point $\mathbf{x}^0$ — Can fail to converge when starting relatively far from a local optimum!
- Hessian matrix needed at each iteration, as well as solution of a linear system — Very burdensome task, especially for large-scale systems!

Need to mitigate these deficiencies!

Derivative-Based Methods Revisited: Deflection Matrices

Idea: Blend first- and second-order methods so as to conserve their respective advantages:

- Use only first partial derivatives and guarantee convergence, as with steepest descent search
- Speed-up convergence with some higher-order information, as with Newton search

How? Deflection Matrices

Deflection matrices $\mathbf{D}^k$ produce modified gradient search direction:

$$\Delta\mathbf{x}^{k+1} = -\mathbf{D}^k\nabla f(\mathbf{x}^k)$$

Remark: Both steepest-descent and Newton search are special instances of deflection matrices:

$$\text{Steepest Descent: } \mathbf{D}^k \triangleq \mathbf{I}, \qquad \text{Newton: } \mathbf{D}^k \triangleq \mathbf{H}(\mathbf{x}^k)^{-1}$$

Quasi-Newton Method

1. **Convergence Speed — Approximate Newton Search**
   - The Hessian $\mathbf{H}(\mathbf{x}^k)$ reflects the rate of change in gradient $\nabla f(\mathbf{x}^k)$,
   $$\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k) \approx \mathbf{H}(\mathbf{x}^k)\left[\mathbf{x}^{k+1} - \mathbf{x}^k\right]$$
   - Deflection matrices $\mathbf{D}^k$ approximate the inverse Hessian matrix $\mathbf{H}(\mathbf{x}^k)^{-1}$ by satisfying the quasi-Newton condition:
   $$\mathbf{D}^k\left[\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)\right] = \mathbf{x}^{k+1} - \mathbf{x}^k$$
   $$\mathbf{D}^k = (\mathbf{D}^k)^\mathsf{T} \quad \text{(symmetric)}$$
2. **Robustness — Guarantee Directions Improve** (minimize problem)
   - For $\Delta\mathbf{x}$ to be improving: $\nabla f(\mathbf{x}^k)^\mathsf{T}\Delta\mathbf{x} < 0$
   - With $\Delta\mathbf{x}^{k+1} = -\mathbf{D}^k\nabla f(\mathbf{x}^k)$, one has: $\nabla f(\mathbf{x}^k)^\mathsf{T}\mathbf{D}^k\nabla f(\mathbf{x}^k) > 0$
   - Sufficient condition for direction improving is:
   $$\mathbf{D}^k \succ \mathbf{0} \quad \text{(positive definite)}$$

BFGS Formula

- A variety of deflection matrices can satisfy the foregoing requirements
- The **BFGS formula**[1] has proved more effective than all the others:

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \left(1 + \frac{\mathbf{g}^\mathsf{T}\mathbf{D}^k\mathbf{g}}{\mathbf{d}^\mathsf{T}\mathbf{g}}\right)\frac{\mathbf{d}\mathbf{d}^\mathsf{T}}{\mathbf{d}^\mathsf{T}\mathbf{g}} - \frac{\mathbf{D}^k\mathbf{g}\mathbf{d}^\mathsf{T} + \mathbf{d}\mathbf{g}^\mathsf{T}\mathbf{D}^k}{\mathbf{d}^\mathsf{T}\mathbf{g}}$$

  with $\mathbf{d} \triangleq \mathbf{x}^{k+1} - \mathbf{x}^k$, and $\mathbf{g} \triangleq \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$
  - Initialize with symmetric, positive definite $\mathbf{D}^0$ — e.g., $\mathbf{D}^0 \triangleq \mathbf{I}$
  - *Every* subsequent iterate $\mathbf{D}^k$ is symmetric
  - *Every* subsequent iterate $\mathbf{D}^k$ is positive definite

- Upon convergence to a local optimum $\mathbf{x}^*$, deflection matrices $\mathbf{D}^k$ approximate the inverse Hessian matrix $\mathbf{H}(\mathbf{x}^*)^{-1}$

[1] Combined work of C. Broyden, R. Fletcher, D. Goldfarb, and D. Shanno

Quasi-Newton Search Algorithm (Minimize Case)

- **Step 0: Initialization**
  - Choose an initial guess $\mathbf{x}^0$, initial deflection matrix $\mathbf{D}^0 = \mathbf{I}$, and stopping tolerance $\epsilon > 0$
  - Set $k \leftarrow 0$
- **Step 1: Stopping**
  - If $\left\|\nabla f(\mathbf{x}^k)\right\| < \epsilon$, stop — report $\mathbf{x}^k$ (approximate stationary point)
- **Step 2: Quasi-Newton Step**
  - Direction: Calculate move direction, $\Delta\mathbf{x}^{k+1} \leftarrow -\mathbf{D}^k \nabla f(\mathbf{x}^k)$
  - Linesearch: Solve 1-d linesearch problem (at least approximately), $\min_\alpha \ell(\alpha) \stackrel{\Delta}{=} f(\mathbf{x}^k + \alpha\Delta\mathbf{x}^{k+1})$, to compute the step α^{k+1}
- **Step 3: Update**
  - Iterate: $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \alpha^{k+1}\Delta\mathbf{x}^{k+1}$
  - Deflection matrix: $\mathbf{D}^{k+1} \leftarrow \mathbf{D}^k + \left(1 + \frac{\mathbf{g}^T\mathbf{D}^k\mathbf{g}}{\mathbf{d}^T\mathbf{g}}\right)\frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d}^T\mathbf{g}} - \frac{\mathbf{D}^k\mathbf{g}\mathbf{d}^T + \mathbf{d}\mathbf{g}^T\mathbf{D}^k}{\mathbf{d}^T\mathbf{g}}$, with $\mathbf{d} = \mathbf{x}^{k+1} - \mathbf{x}^k$, $\mathbf{g} = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$
  - Increment $k \leftarrow k+1$ and return to step 1

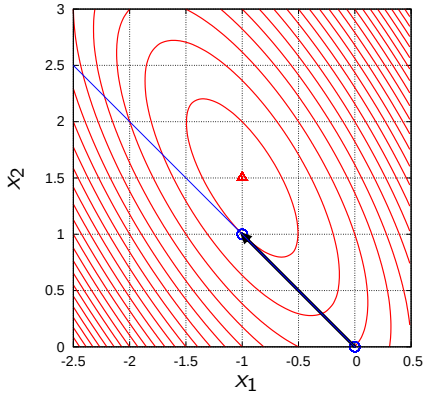
Applying BFGS Quasi-Newton Search

Class Exercise: Consider the problem to minimize

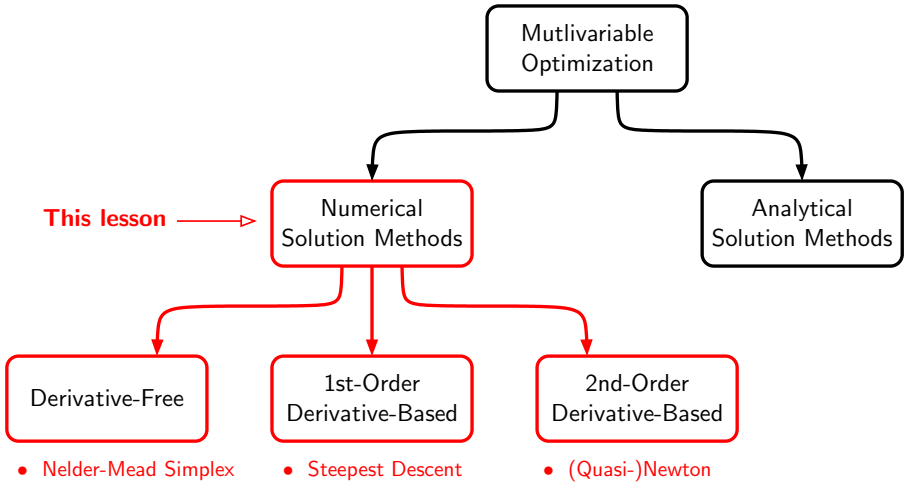
$$f(\mathbf{x}) \stackrel{\Delta}{=} x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$$

Starting with $\mathbf{x}^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $\mathbf{D}^0 = \mathbf{I}$, the next point is $\mathbf{x}^1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

- Calculate $\mathbf{D}^1$ using BFGS
- Calculate quasi-Newton move direction $\Delta\mathbf{x}^2$ at $\mathbf{x}^1$



Multivariable, Unconstrained NLP — The Final Words



Do it yourself!
Describe an example for each approach, with some pros and cons?