Optimization Methods
Decision making

Examples:

• determining which ingredients and in what quantities to add to a mixture being made so that it will meet specifications on its composition
• allocating available funds among various competing agencies
• deciding which route to take to go to a new location in the city

• Decision making always involves making a choice between various possible alternatives
Categories of Decision making problems

Category 1:
• The set of possible alternatives for the decision is a finite discrete set typically consisting of a small number of elements.
  – Example: “A teenage girl knows four boys all of whom she likes, and has to decide who among them to go steady with.”
• Solution: scoring methods

Category 2:
• The number of possible alternatives is either infinite, or finite but very large, and the decision may be required to satisfy some restrictions and constraints
• Solution: unconstrained and constrained optimization methods
The scoring method – an example

Rita has been dating 4 boys off and on over the last 3 years, and has come to know each of them very well. Who among the four boys would be her best choice?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Rating for</th>
<th>Type</th>
<th>Range</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Ability to support</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Friendliness</td>
<td>70</td>
<td>40</td>
<td>60</td>
<td>80</td>
</tr>
<tr>
<td>Honesty</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Respect for women</td>
<td>50</td>
<td>60</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>Handsomeness</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Interest in appearance</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Degree of reciprocity</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

*{*A = Bill, B = Raj, C = Tom, D = Dick*}
Category 2 Decision problems

1. Get a precise definition of the problem, all relevant data and information on it.
   - Uncontrollable factors (random variables)
   - Controllable inputs (decision variables)

2. Construct a mathematical (optimization) model of the problem.
   - Build objective functions and constraints

3. Solve the model
   - Apply the most appropriate algorithms for the given problem

4. Implement the solution
Optimization models

• Single x Multiobjective models

• Static x Dynamic models

• Deterministic x Stochastic models
Problem specification

Suppose we have a cost function (or objective function)

\[ f(x) : \mathbb{R}^N \rightarrow \mathbb{R}. \]

Our aim is to find values of the parameters (decision variables) \( x \) that minimize this function

\[ x^* = \arg \min_x f(x). \]

Subject to the following constraints:

- equality: \( c_i(x) = 0 \)
- nonequality: \( c_j(x) \geq 0 \)

If we seek a maximum of \( f(x) \) (profit function) it is equivalent to seeking a minimum of \(-f(x)\)
Books to read

- **Practical Optimization**

- **Practical Optimization: Algorithms and Engineering Applications**
  - Andreas Antoniou and Wu-Sheng Lu, 2007

- Both cover unconstrained and constrained optimization. Very clear and comprehensive.
Further reading and web resources

- **Numerical Recipes in C (or C++): The Art of Scientific Computing**
  - William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling
  - Good chapter on optimization
  - Available online at

- **NEOS Guide**

- **This powerpoint presentation**
  - [www.utia.cas.cz](http://www.utia.cas.cz)
Types of minima

- which of the minima is found depends on the starting point
- such minima often occur in real applications
Unconstrained **univariate** optimization

Assume we can start close to the global minimum

How to determine the minimum?

- Search methods (Dichotomous, Fibonacci, Golden-Section)
- Approximation methods
  1. Polynomial interpolation
  2. Newton method
- Combination of both (alg. of Davies, Swann, and Campey)
Search methods

- Start with the interval ("bracket") \([x_L, x_U]\) such that the minimum \(x^*\) lies inside.
- Evaluate \(f(x)\) at two points inside the bracket.
- Reduce the bracket.
- Repeat the process.

- Can be applied to any function and differentiability is not essential.
Search methods

Dichotomous

Fibonacci: 1 1 2 3 5 8 …
1D function

As an example consider the function

\[ f(x) = 0.1 + 0.1x + \frac{x^2}{(0.1 + x^2)} \]

(assume we do not know the actual function expression from now on)
Gradient descent

Given a starting location, $x_0$, examine $df/dx$ and move in the \textit{downhill} direction to generate a new estimate, $x_1 = x_0 + \delta x$

How to determine the step size $\delta x$?
Polynomial interpolation

• Bracket the minimum.
• Fit a quadratic or cubic polynomial which interpolates $f(x)$ at some points in the interval.
• Jump to the (easily obtained) minimum of the polynomial.
• Throw away the worst point and repeat the process.
Polynomial interpolation

- Quadratic interpolation using 3 points, 2 iterations
- Other methods to interpolate?
  - 2 points and one gradient
  - Cubic interpolation
Newton method

Fit a quadratic approximation to $f(x)$ using both gradient and curvature information at $x$.

- Expand $f(x)$ locally using a Taylor series.

\[
  f(x + \delta x) = f(x) + f'(x)\delta x + \frac{1}{2}f''(x)\delta x^2 + o(\delta x^2)
\]

- Find the $\delta x$ which minimizes this local quadratic approximation.

\[
  \delta x = -\frac{f'(x)}{f''(x)}
\]

- Update $x$.

\[
  x_{n+1} = x_n - \delta x = x_n - \frac{f'(x)}{f''(x)}
\]
Newton method

- avoids the need to bracket the root
- quadratic convergence (decimal accuracy doubles at every iteration)
Newton method

• Global convergence of Newton’s method is poor.
• Often fails if the starting point is too far from the minimum.

• In practice, must be used with a globalization strategy which reduces the step length until function decrease is assured
Extension to N (multivariate) dimensions

• How big N can be?
  – problem sizes can vary from a handful of parameters to many thousands

• We will consider examples for N=2, so that cost function surfaces can be visualized.
An Optimization Algorithm

- Start at \( x_0, k = 0 \).

1. Compute a search direction \( p_k \)

2. Compute a step length \( \alpha_k \), such that \( f(x_k + \alpha_k p_k) < f(x_k) \)

3. Update \( x_k = x_k + \alpha_k p_k \)

4. Check for convergence (stopping criteria)
   e.g. \( df/dx = 0 \)

Reduces optimization in \( N \) dimensions to a series of (1D) line minimizations
Taylor expansion

A function may be approximated locally by its Taylor series expansion about a point $x^*$

$$f(x^* + x) \approx f(x^*) + \nabla f^T x + \frac{1}{2} x^T H x$$

where the gradient $\nabla f(x^*)$ is the vector

$$\nabla f(x^*) = \begin{bmatrix} \frac{\partial f}{x_1} & \cdots & \frac{\partial f}{x_N} \end{bmatrix}^T$$

and the Hessian $H(x^*)$ is the symmetric matrix

$$H(x^*) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$
Quadratic functions

\[ f(x) = a + g^T x + \frac{1}{2} x^T H x \]

• The vector \( g \) and the Hessian \( H \) are constant.
• Second order approximation of any function by the Taylor expansion is a quadratic function.

We will assume only quadratic functions for a while.
Necessary conditions for a minimum

\[ f(x) = a + g^T x + \frac{1}{2}x^T H x \]

Expand \( f(x) \) about a stationary point \( x^* \) in direction \( p \)

\[ f(x^* + \alpha p) = f(x^*) + g(x^*)^T \alpha p + \frac{1}{2} \alpha^2 p^T H p \]

\[ = f(x^*) + \frac{1}{2} \alpha^2 p^T H p \]

since at a stationary point \( g(x^*) = 0 \)

At a stationary point the behavior is determined by \( H \)
- $H$ is a symmetric matrix, and so has orthogonal eigenvectors

$$Hu_i = \lambda_i u_i \quad ||u_i|| = 1$$

$$f(x^* + \alpha u_i) = f(x^*) + \frac{1}{2}\alpha^2 u_i^T Hu_i$$

$$= f(x^*) + \frac{1}{2}\alpha^2 \lambda_i$$

- As $|\alpha|$ increases, $f(x^* + \alpha u_i)$ increases, decreases or is unchanging according to whether $\lambda_i$ is positive, negative or zero
Examples of quadratic functions

Case 1: both eigenvalues positive

\[ f(x) = a + g^T x + \frac{1}{2} x^T H x \]

with \( a = 0 \), \( g = \begin{bmatrix} -50 \\ -50 \end{bmatrix} \), \( H = \begin{bmatrix} 6 & 4 \\ 4 & 6 \end{bmatrix} \) positive definite

minimum
Examples of quadratic functions

Case 2: eigenvalues have different sign

\[ f(x) = a + g^T x + \frac{1}{2} x^T H x \]

with

\[ a = 0, \quad g = \begin{bmatrix} -30 \\ 20 \end{bmatrix}, \quad H = \begin{bmatrix} 6 & 0 \\ 0 & -4 \end{bmatrix} \]

saddle point
Examples of quadratic functions

Case 3: one eigenvalues is zero

\[ f(x) = a + g^T x + \frac{1}{2} x^T H x \]

with

\[ a = 0, \quad g = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix} \]

positive semidefinite

parabolic cylinder
Optimization for quadratic functions

Assume that $\mathbf{H}$ is positive definite

$$f(x) = a + \mathbf{g}^T x + \frac{1}{2} x^T \mathbf{H} x$$

$$\nabla f(x) = \mathbf{g} + \mathbf{H} x$$

There is a unique minimum at

$$x^* = -\mathbf{H}^{-1} \mathbf{g}$$

If $N$ is large, it is not feasible to perform this inversion directly.
Steepest descent

- Basic principle is to minimize the N-dimensional function by a series of 1D line-minimizations:

\[ x_{k+1} = x_k + \alpha_k p_k \]

- The steepest descent method chooses \( p_k \) to be parallel to the gradient

\[ p_k = -\nabla f(x_k) \]

- Step-size \( \alpha_k \) is chosen to minimize \( f(x_k + \alpha_k p_k) \).

For quadratic forms there is a closed form solution:

\[ \alpha_k = \frac{p_k^T p_k}{p_k^T H p_k} \]

Prove it!
Steepest descent

- The gradient is everywhere perpendicular to the contour lines.
- After each line minimization the new gradient is always orthogonal to the previous step direction (true of any line minimization).
- Consequently, the iterates tend to zig-zag down the valley in a very inefficient manner
Conjugate gradient

• Each $p_k$ is chosen to be conjugate to all previous search directions with respect to the Hessian $H$:

  $$p_i^T Hp_j = 0, \quad i \neq j$$

• The resulting search directions are mutually linearly independent.

• *Remarkably*, $p_k$ can be chosen using only knowledge of $p_{k-1}$, $\nabla f(x_{k-1})$, and $\nabla f(x_k)$

  $$p_k = \nabla f_k + \left( \frac{\nabla f_k^T \nabla f_k}{\nabla f_{k-1}^T \nabla f_{k-1}} \right) p_{k-1}$$

Prove it!
Conjugate gradient

- An N-dimensional quadratic form can be minimized in at most N conjugate descent steps.

- 3 different starting points.
- Minimum is reached in exactly 2 steps.
Optimization for General functions

\[ f(x, y) = \exp(x)(4x^2 + 2y^2 + 4xy + 2x + 1) \]

Apply methods developed using quadratic Taylor series expansion
Rosenbrock’s function

\[ f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \]

Minimum at [1, 1]
Steepest descent

• The 1D line minimization must be performed using one of the earlier methods (usually cubic polynomial interpolation)

• The zig-zag behaviour is clear in the zoomed view
• The algorithm crawls down the valley
Conjugate gradient

- Again, an explicit line minimization must be used at every step

- The algorithm converges in 98 iterations
- Far superior to steepest descent
Newton method

Expand $f(x)$ by its Taylor series about the point $x_k$

$$f(x_k + \delta x) \approx f(x_k) + g_k^T \delta x + \frac{1}{2} \delta x^T H_k \delta x$$

where the gradient is the vector

$$g_k = \nabla f(x_k) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_N} \end{bmatrix}^T$$

and the Hessian is the symmetric matrix

$$H_k = H(x_k) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$
Newton method

For a minimum we require that \( \nabla f(x) = 0 \), and so

\[
\nabla f(x) = g_k + H_k \delta x = 0
\]

with solution \( \delta x = -H_k^{-1}g_k \). This gives the iterative update

\[
x_{k+1} = x_k - H_k^{-1}g_k
\]

- If \( f(x) \) is quadratic, then the solution is found in one step.
- The method has quadratic convergence (as in the 1D case).
- The solution \( \delta x = -H_k^{-1}g_k \) is guaranteed to be a downhill direction.
- Rather than jump straight to the minimum, it is better to perform a line minimization which ensures global convergence

\[
x_{k+1} = x_k - \alpha_k H_k^{-1}g_k
\]

- If \( H=I \) then this reduces to steepest descent.
• The algorithm converges in only 18 iterations compared to the 98 for conjugate gradients.

• However, the method requires computing the Hessian matrix at each iteration – this is not always feasible
Summary of the 1\textsuperscript{st} lecture

- Minimization of 1-D functions
  - Search methods
  - Approximation methods

- N-D functions \( \rightarrow \) finding the descent direction
- Taylor series \( \rightarrow \) Quadratic functions
- Steepest descent
- Conjugate Gradient
- Newton method
Quasi-Newton methods

- If the problem size is large and the Hessian matrix is dense then it may be infeasible/inconvenient to compute it directly.
- Quasi-Newton methods avoid this problem by keeping a “rolling estimate” of $H(x)$, updated at each iteration using new gradient information.
- Common schemes are due to Broyden, Goldfarb, Fletcher and Shanno (BFGS), and also Davidson, Fletcher and Powell (DFP).
- The idea is based on the fact that for quadratic functions holds
  \[ g_{k+1} - g_k = H(x_{k+1} - x_k) \]
  and by accumulating $g_k$’s and $x_k$’s we can calculate $H$. 
Quasi-Newton BFGS method

• Set $H_0 = I$.
• Update according to

$$
H_{k+1} = H_k + \frac{\gamma_k \gamma_k^T}{\gamma_k^T \delta_k} - \frac{H_k \gamma_k \gamma_k^T H_k}{\delta_k^T H_k \delta_k}
$$

where

$$
\gamma_k = g_{k+1} - g_k \\
\delta_k = x_{k+1} - x_k
$$

• The matrix inverse can also be computed in this way.
• Directions $\delta_k$‘s form a conjugate set.
• $H_{k+1}$ is positive definite if $H_k$ is positive definite.
• The estimate $H_k$ is used to form a local quadratic approximation as before.
BFGS example

- The method converges in 34 iterations, compared to 18 for the full-Newton method.
Non-linear least squares

• It is very common in applications for a cost function $f(x)$ to be the sum of a large number of squared residuals

$$f(x) = \sum_{i=1}^{M} r_{i}(x)$$

• If each residual depends non-linearly on the parameters $x$ then the minimization of $f(x)$ is a non-linear least squares problem.
Non-linear least squares

\[ f(x) = \sum_{i=1}^{M} r_i^2(x) \]

- The M × N Jacobian of the vector of residuals \( r \) is defined as

\[ J(x) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \ldots & \frac{\partial r_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial x_1} & \ldots & \frac{\partial r_M}{\partial x_N} \end{bmatrix} \]

- Consider

\[ \frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_i r_i^2 = \sum_i 2r_i \frac{\partial r_i}{\partial x_k} \]

- Hence

\[ \nabla f(x) = 2J^T r \]
Non-linear least squares

- For the Hessian holds

\[
\frac{\partial^2 f}{\partial x_k \partial x_l} = 2 \sum_i \frac{\partial r_i}{\partial x_l} \frac{\partial r_i}{\partial x_k} + 2 \sum_i r_i \frac{\partial^2 r_i}{\partial x_k \partial x_l}
\]

- Note that the second-order term in the Hessian is multiplied by the residuals \( r_i \).
- In most problems, the residuals will typically be small.
- Also, at the minimum, the residuals will typically be distributed with mean = 0.
- For these reasons, the second-order term is often ignored.
- Hence, explicit computation of the full Hessian can again be avoided.

Gauss-Newton approximation
Gauss-Newton example

- The minimization of the Rosenbrock function

\[ f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \]

- can be written as a least-squares problem with residual vector

\[
\mathbf{r} = \begin{bmatrix}
10(y - x^2) \\
(1 - x)
\end{bmatrix}
\]

\[
\mathbf{J} = \begin{bmatrix}
\frac{\partial r_1}{\partial x} & \frac{\partial r_1}{\partial y} \\
\frac{\partial r_2}{\partial x} & \frac{\partial r_2}{\partial y}
\end{bmatrix} = \begin{bmatrix}
-20x & 10 \\
-1 & 0
\end{bmatrix}
\]
Gauss-Newton example

\[ x_{k+1} = x_k - \alpha_k H_k^{-1} g_k \]

\[ H_k = 2J_k^T J \]

- minimization with the Gauss-Newton approximation with line search takes only 11 iterations
Comparison

CG

Quasi-Newton

Newton

Gauss-Newton
Simplex
Constrained Optimization

\[ f(x) : \mathbb{R}^N \rightarrow \mathbb{R} \]

\[ x^* = \arg \min_x f(x) \]

Subject to:

- Equality constraints:
  \[ a_i(x) = 0 \quad i = 1, 2, \ldots, p \]

- Nonequality constraints:
  \[ c_j(x) \geq 0 \quad j = 1, 2, \ldots, q \]

- Constraints define a feasible region, which is nonempty.

- The idea is to convert it to an unconstrained optimization.
Equality constraints

- Minimize $f(x)$ subject to: $a_i(x) = 0$ for $i = 1, 2, \ldots, p$

- The gradient of $f(x)$ at a local minimizer is equal to the linear combination of the gradients of $a_i(x)$ with Lagrange multipliers as the coefficients.

$$\nabla f(x^*) = \sum_{i=1}^{p} \lambda_i^* \nabla a_i(x^*)$$
\( f_3 > f_2 > f_1 \)

\( \tilde{x} \) is not a minimizer

\( x^* \) is a minimizer, \( \lambda^* > 0 \)

\( f_3 > f_2 > f_1 \)

\( x^* \) is a minimizer, \( \lambda^* < 0 \)

\( f_3 > f_2 > f_1 \)

\( x^* \) is not a minimizer
3D Example

\[ a_1(x) = -x_1 + x_3 - 1 = 0 \]

\[ a_2(x) = x_1^2 + x_2^2 - 2x_1 = 0 \]
3D Example

\[ f(x) = x_1^2 + x_2^2 + \frac{1}{4}x_3^2 \]

Gradients of constraints and objective function are linearly independent.
3D Example

$f(x) = x_1^2 + x_2^2 + \frac{1}{4}x_3^2$

Gradients of constraints and objective function are linearly dependent.
Inequality constraints

- Minimize $f(x)$ subject to: $c_j(x) \geq 0 \quad \text{for} \quad j = 1, 2, \ldots, q$

- The gradient of $f(x)$ at a local minimizer is equal to the linear combination of the gradients of $c_j(x)$, which are active ($c_j(x) = 0$)

- and Lagrange multipliers must be positive, $\mu_j \geq 0, j \in A$

\[ \nabla f(x^*) = \sum_{j \in A} \mu_j^* \nabla c_j(x^*) \]
No active constraints at $x^*$, $\nabla f(x) = 0$

$x^*$ is not a minimizer, $\mu < 0$

$x^*$ is a minimizer, $\mu > 0$
Lagrangian

- We can introduce the function \( L(x, \lambda, \mu) \)

\[
L(x, \lambda, \mu) = f(x) - \sum_{i=1}^{p} \lambda_i a_i(x) - \sum_{j=1}^{q} \mu_j c_j(x)
\]

- The necessary condition for the local minimizer is

\[
\nabla_x L(x, \lambda, \mu) = 0
\]

and it must be a feasible point (i.e. constraints are satisfied).

- These are Karush-Kuhn-Tucker conditions
Quadratic Programming (QP)

- Like in the unconstrained case, it is important to study quadratic functions. **Why?**
- Because general nonlinear problems are solved as a sequence of minimizations of their quadratic approximations.
- QP with constraints
  
  Minimize  
  \[ f(x) = \frac{1}{2} x^T H x + x^T p \]
  
  subject to linear constraints.

- \( H \) is symmetric and positive semidefinite.
QP with Equality Constraints

• Minimize

\[ f(x) = \frac{1}{2} x^T H x + x^T p \]

Subject to:

\[ A x = b \]

• Ass.: \( A \) is \( p \times N \) and has full row rank \((p<N)\)

• Convert to unconstrained problem by variable elimination:

\[ x = Z \phi + A^+ b \]

\( Z \) is the null space of \( A \)

\( A^+ \) is the pseudo-inverse.

Minimize

\[ \hat{f}(\phi) = \frac{1}{2} \phi^T \hat{H} \phi + \phi^T \hat{p} \]

\[ \hat{H} = Z^T H Z \]

\[ \hat{p} = Z^T (H A^+ b + p) \]

This quadratic unconstrained problem can be solved, e.g., by Newton method.
QP with inequality constraints

- Minimize
  \[ f(x) = \frac{1}{2} x^T H x + x^T p \]
  Subject to:
  \[ A x \geq b \]

- First we check if the unconstrained minimizer \( x^* \) is feasible.
  If yes we are done.
  If not we know that the minimizer must be on the boundary and we proceed with an active-set method.

- \( x_k \) is the current feasible point
- \( A_k \) is the index set of active constraints at \( x_k \)
- Next iterate is given by
  \[ x_{k+1} = x_k + \alpha_k d_k \]
Active-set method

- How to find $d_k$?
  - To remain active
  - The objective function at $x_k + d$ becomes

$$f_k(d) = \frac{1}{2}d^T H d + d^T g_k + f(x_k)$$

where $g_k = \nabla f(x_k)$

- The major step is a QP sub-problem

$$d_k = \arg \min_d \frac{1}{2}d^T H d + d^T g_k$$

subject to: $a_j^T d = 0 \quad j \in A_k$

- Two situations may occur: $d_k = 0$ or $d_k \neq 0$
Active-set method

- \( d_k = 0 \)
  We check if KKT conditions are satisfied

\[
\nabla_x L(x, \mu) = Hx_k + p - \sum_{j \in \mathcal{A}_k} \mu_j a_j = 0 \quad \text{and} \quad \mu_j \geq 0
\]

If YES we are done.
If NO we remove the constraint from the active set \( \mathcal{A}_k \) with the most negative \( \mu_j \) and solve the QP sub-problem again but this time with less active constraints.

- \( d_k \neq 0 \)
  We can move to \( x_{k+1} = x_k + d_k \) but some inactive constraints may be violated on the way.
In this case, we move by \( \alpha_k d_k \) till the first inactive constraint becomes active, update \( \mathcal{A}_k \), and solve the QP sub-problem again but this time with more active constraints.
General Nonlinear Optimization

• Minimize $f(x)$
  subject to: $a_i(x) = 0$
  $c_j(x) \geq 0$

  where the objective function and constraints are nonlinear.

1. For a given \{x_k, \lambda_k, \mu_k\} approximate Lagrangian by Taylor series → QP problem
2. Solve QP → descent direction \{\delta_x, \delta_\lambda, \delta_\mu\}
3. Perform line search in the direction $\delta_x \rightarrow x_{k+1}$
4. Update Lagrange multipliers $\rightarrow \{\lambda_{k+1}, \mu_{k+1}\}$
5. Repeat from Step 1.
General Nonlinear Optimization

Lagrangien

\[ L(x, \lambda, \mu) = f(x) - \sum_{i=1}^{p} \lambda_i a_i(x) - \sum_{j=1}^{q} \mu_j c_j(x) \]

At the \( k \)th iterate: \( \{x_k, \lambda_k, \mu_k\} \)

and we want to compute a set of increments: \( \{\delta_x, \delta_\lambda, \delta_\mu\} \)

First order approximation of \( \nabla_x L \) and constraints:

\[ \nabla_x L(x_{k+1}, \lambda_{k+1}, \mu_{k+1}) \approx \nabla_x L(x_k, \lambda_k, \mu_k) + \nabla^2_x L(x_k, \lambda_k, \mu_k) \delta_x + \nabla^2_{x\lambda} L(x_k, \lambda_k, \mu_k) \delta_\lambda + \nabla^2_{x\mu} L(x_k, \lambda_k, \mu_k) \delta_\mu = 0 \]

\[ c_i(x_k, \delta_x) \approx c_i(x_k) + \delta_x^T \nabla_x c_i(x_k) \geq 0 \]

\[ a_i(x_k, \delta_x) \approx a_i(x_k) + \delta_x^T \nabla_x a_i(x_k) = 0 \]

These approximate KKT conditions corresponds to a QP program
SQP example

Minimize
subject to:

\[ f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \]

\[ 1.5 - x_1^2 - x_2^2 \geq 0 \]
Linear Programming (LP)

- LP is common in economy and is meaningful only if it is with constraints.
- Two forms:
  1. Minimize \( f(x) = c^T x \) subject to: \( Ax = b \) \( x \geq 0 \)
  2. Minimize \( f(x) = c^T x \) subject to: \( Ax \geq b \)

- QP can solve LP.
- If the LP minimizer exists it must be one of the vertices of the feasible region.
- A fast method that considers vertices is the Simplex method.

\( A \) is \( p \times N \) and has full row rank \((p<N)\)

Prove it!