UNCONSTRAINED NONLINEAR OPTIMIZATION

Reference:

For an arbitrary smooth function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) we want to solve the unconstrained nonlinear programming problem

\[
\min_x f(x)
\]

There are fundamentally two classes of iterative methods:

- **Line search methods** choose a descent direction \( p_k \), search a suitable scalar \( \alpha_k > 0 \) such that \( f(x_k + \alpha_k p_k) < f(x_k) \), and set \( x_{k+1} = x_k + \alpha_k p_k \)

- **Trust region methods** compute a quadratic approximation \( q(x) \) of \( f \) around \( x_k \), solve

\[
p_k = \arg \min_{p: \|p\|_2 \leq \Delta} q(x_k + p)
\]

where the size \( \Delta \) of the “trust-region” of the model is shrunk until \( f(x_k + p_k) < f(x_k) \), and set \( x_{k+1} = x_k + p_k \)

The above methods converge to a local minimum (a global one if \( f \) convex)
Steepest descent is the most obvious method, as it picks up $p_k$ orthogonal to the level sets of $f$

$$p_k = -\nabla f(x_k)$$

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

From Taylor’s theorem

$$f(x_k + \alpha p_k) = f(x_k) - \alpha \|\nabla f(x_k)\|^2 + \alpha^2 p_k' \nabla^2 f(x_k + t p_k) p_k, \ t \in (0, \alpha)$$

Note that the Hessian of $f$ is not required to compute $p_k$

The method can be very slow to converge
**LINE SEARCH METHODS: NEWTON'S METHOD**

- **Newton’s method** chooses \( p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \) (Newton’s direction)

\[
x_{k+1} = x_k - \alpha_k \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)
\]

- Newton’s direction provides the minimum of the quadratic Taylor’s approximation \( q \) of \( f \) at \( x_k \):

\[
q(x_k + p) = f(x_k) + \nabla f(x_k)'p + \frac{1}{2}p'\nabla^2 f(x_k)p
\]

- If \( \nabla^2 f(x_k) > 0 \) then for some \( \sigma_k > 0 \)

\[
\nabla f(x_k)'p_k = -\nabla f(x_k)'\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \leq -\sigma_k \|p_k\|^2_2
\]

so from Taylor’s theorem we have \( f(x_k + \alpha p_k) < f(x_k) \) for \( \alpha \) small enough

©2021 A. Bemporad - Numerical Optimization
The method converges very fast, especially close to \( x^* \), where the function \( f \) and its quadratic approximation tend to coincide.

For \( \alpha_k \equiv 1 \) we have pure Newton’s method. However line search over \( \alpha \) is required to ensure convergence.

In case \( \nabla^2 f(x_k) \) is not positive definite, a possibility is to use instead \( \nabla^2 f(x_k) + \text{diag}(\delta_k) \).

For example \( \delta_k \) can be computed during a Cholesky factorization to make intermediate diagonal entries \( \geq \epsilon \) for some \( \epsilon > 0 \).
- Newton’s method requires computing $\nabla^2 f(x_k)$, which could be expensive.

- **Quasi-Newton methods** replace $\nabla^2 f(x_k)$ with a matrix $B_k$ which is easier to compute, satisfying the secant equation

  $$B_{k+1} s_k = y_k, \text{ where } s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

  and set $p_k = -B_k^{-1} \nabla f(x_k)$.

- The **BFGS formula** (Broyden, Fletcher, Goldfarb, and Shanno) updates

  $$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

  where $B_k \succ 0$ if $B_0 \succ 0$ and $s_k^T y_k > 0$ for all $k$. 

©2021 A. Bemporad - Numerical Optimization
### Line Search Methods: Quasi-Newton Methods

- Since $B_{k+1}$ differs from $B_k$ by two one-rank updates, we can update a factorization of $B_k$ recursively.

- In alternative, one can avoid $B_k$ and directly update $H_k = B_k^{-1}$

\[
H_{k+1} = H_k + \frac{s'_k y_k + y'_k B_k^{-1} y_k}{(s'_k y_k)^2} s_k s'_k - \frac{B_k^{-1} y_k s'_k + s_k (B_k^{-1} y_k)}{s'_k y_k}
\]

- For large-scale problems, **limited-memory BFGS** only stores a finite number $m$ of past values of $(s_k, y_k)$ (usually $m < 10$) and directly computes the descent direction $p_k = -H_k \nabla f(x_k)$ without storing $H_k$
Nonlinear conjugate gradient methods (CG) update the search direction $p_k$ as follows:

$$\beta_{k+1} = \frac{\nabla f(x_{k+1})' \nabla f(x_{k+1})}{\nabla f(x_k)' \nabla f(x_k)}$$

$$p_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1} p_k$$

where the scalar $\beta_k$ is chosen so that $p_k$ and $p_{k-1}$ are conjugate.

- The method does not require storage of matrices.
- The method is almost as simple as steepest descent but usually more efficient (although it does not converge as fast as (quasi-)Newton methods).
- As for steepest descent, nonlinear CG may be sensitive to problem scaling.
LINE SEARCH

- Given a descent direction $p_k$, ideally one should choose $x_{k+1} = x_k + \alpha_k p_k$ with

$$\alpha_k = \arg\min_{\alpha > 0} f(x_k + \alpha p_k)$$

- Such a scalar nonlinear optimization may be difficult to solve and require a lot of evaluations of $f$, so we look for simpler methods

- Simply imposing $f(x_k + \alpha_k p_k) < f(x_k)$ may not work, as the improvement may become smaller and smaller as $k$ grows

- Sufficient decrease is provided by satisfying **Armijo condition**

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)' p_k \quad c_1 \in (0, 1)$$

where usually $c_1$ is small (e.g., $c_1 = 10^{-4}$)
- **Wolfe conditions** include Armijo condition + the **curvature condition**

\[
\frac{df(x_k + \alpha p_k)}{d\alpha} = \begin{vmatrix} \nabla f(x_k + \alpha p_k)' p_k \geq c_2 \nabla f(x_k)' p_k \end{vmatrix} \quad c_2 \in (c_1, 1)
\]

(the condition is **strong** if \(|\nabla f(x_k + \alpha p_k)' p_k| \leq c_2 |\nabla f(x_k)' p_k|\) is imposed)

- The curvature condition avoid values of \(\alpha\) that are too small, when \(f\) is still decaying fast (=very negative derivative)

- Usually \(c_1 = 0.9\) in (quasi-)Newton methods and \(c_1 = 0.1\) in the nonlinear CG method

- It is possible to prove that if \(f\) is continuously differentiable and \(f\) bounded below along the descent direction \(x_k + \alpha p_k, \alpha \geq 0\), the (strong) Wolfe conditions can be satisfied
• The following is a simple practical algorithm for selecting a step size $\alpha_k$ satisfying Armijo formula:

Choose $\bar{\alpha} > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$. Set $\alpha = \bar{\alpha}$.

Repeat until $f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f(x_k)' p_k$

$\alpha \leftarrow \rho \alpha$

end repeat

Set $\alpha_k = \alpha$.

• Possible choices are $\bar{\alpha} = 1$ (e.g., in Newton’s method) and $\rho = \frac{1}{2}$ (bisection)
THEOREM (Zoutendijk)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be bounded below and differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} = \{x : f(x) \leq f(x_0)\}$, and let $\nabla f$ Lipschitz continuous on $\mathcal{N}$, that is
\[
\|\nabla f(x_1) - \nabla f(x_2)\| \leq L\|x_1 - x_2\|, \forall x_1, x_2 \in \mathcal{N}
\]
for some $L > 0$. Any line search method with $p_k$ a descent direction and $\alpha_k$ satisfying the Wolfe conditions is such that
\[
\sum_{k=0}^{\infty} \cos^2(\theta_k) \|\nabla f(x_k)\|^2 < \infty, \quad \cos(\theta_k) = \frac{-\nabla f(x_k)'p_k}{\|\nabla f(x_k)\|\|p_k\|}
\]

- If we choose $p_k$ such that $\cos \theta_k \geq \delta > 0$, $\forall k \geq 0$, then $\lim_{k \to \infty} \|\nabla f(x_k)\| = 0$. 

©2021 A. Bemporad - Numerical Optimization
The condition $\cos \theta_k \geq \delta > 0, \forall k \geq 0$, holds for the steepest descent method.

It also holds for (quasi-)Newton methods when $B_k \succ 0$ with uniformly bounded condition number.

The convergence result show that the algorithm converges to a stationary point $\nabla f(x) = 0$. 

©2021 A. Bemporad - Numerical Optimization
• In analyzing the speed of convergence of iterative algorithms, we refer to convergence rates. Let \( x_k : \mathbb{N} \rightarrow \mathbb{R}^n \) be a converging sequence, \( \lim_{k \rightarrow \infty} x_k = x^* \). We define

\[
\lim_{k \rightarrow \infty} \frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|} = r, \quad r \in (0, 1)
\]

linear convergence

\[
\frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|} = r_k, \quad \lim_{k \rightarrow \infty} r_k = 0
\]

superlinear convergence

\[
\lim_{k \rightarrow \infty} \frac{\| x_{k+1} - x^* \|}{\| x_k - x^* \|^2} > 0
\]

quadratic convergence

• Convergence only relates to the asymptotic behavior of the algorithm. The transient is often more relevant, especially stopping tolerances are not small
• When $f$ is twice differentiable and $\nabla^2 f(x^*) \succ 0$ we can show that steepest descent has the linear convergence rate

$$f(x_{k+1}) - f(x^*) \leq r^2(f(x_k) - f(x^*)), \quad \frac{\lambda_{\text{max}} - \lambda_{\text{min}}}{\lambda_{\text{max}} + \lambda_{\text{min}}} < r < 1$$

where $\lambda_{\text{max}}, \lambda_{\text{min}}$ are the max/min eigenvalues of $\nabla^2 f(x^*)$

• When $f$ is twice differentiable and $\nabla^2 f(x^*) \succ 0$ and $x_0$ is sufficiently close to $x^*$ Newton’s method has the quadratic convergence rate

$$\|x_{k+1} - x^*\| \leq \tilde{L}\|x_k - x^*\|^2, \quad \|\nabla f(x_{k+1})\| \leq 2L\|\nabla^2 f(x^*)^{-1}\|^2\|\nabla f(x_k)\|^2$$

while quasi-Newton methods have a superlinear convergence rate
**LINE SEARCH METHODS: COORDINATE DESCENT**

- **Coordinate descent** consists of successively optimizing only one coordinate $x_{i_k}$ at each step $k$, $i_k \in \{1, \ldots, n\}$

- The index $i_k$ can be selected **cyclically** $i_{k+1} = [i_k \mod n] + 1$ or **randomly**

- In case $f$ is differentiable, the update is
  
  $x_{k+1} = x_k - \alpha_k \frac{\partial f(x_k)}{\partial x_{i_k}} e_{i_k}, \quad \alpha_k > 0, \quad e_i = i\text{th column of } I$

- In case of perfect line search $x_{k+1} = \arg\min_\alpha f(x_k + \alpha e_{i_k})$

- The method can be applied even if $f$ is nonsmooth and some $x_i$ discrete

- Although $f(x_{k+1}) \leq f(x^k)$ the method may not converge to a local minimum

- The method stops if there is no improvement in $f(x^k)$ after one full cycle
We want to solve the **nonlinear least-squares** problem

$$\min_x \frac{1}{2} \sum_{j=1}^{m} r_j^2(x) = \frac{1}{2} \|r(x)\|_2^2$$

where each residual $r_j : \mathbb{R}^n \to \mathbb{R}$ is smooth, $\forall j = 1, \ldots, m$ (assume $m \geq n$)

Let $J(x)$ be the **Jacobian** associated with $r(x)$

$$J(x) = \begin{bmatrix} \nabla r_1(x)' \\ \vdots \\ \nabla r_m(x)' \end{bmatrix}$$

The gradient $\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)'r(x)$, the Hessian is

$$\nabla^2 f(x) = \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)' + r_j(x) \nabla^2 r_j(x) = J(x)'J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x)$$

Gauss-Newton method approximates $\nabla^2 f(x_k) \approx J(x_k)'J(x_k)$
Gauss-Newton does not require computing the Hessian matrices $\nabla^2 r_j(x_k)$.

$$x_{k+1} = x_k + \alpha_k p_k, \quad p_k = - (J(x_k)' J(x_k))^{-1} J(x_k)' r(x_k)$$

In many problems $J(x_k)' J(x_k)$ dominates over the neglected term $\sum_{j=1}^m \nabla r_j(x_k) \nabla^2 r_j(x_k)$ close to $x^*$, so convergence speed can get very close to Newton method.

When $J(x_k)$ is full rank, $p_k$ is a descent direction:

$$p_k' \nabla f(x_k) = p_k' J(x_k)' r(x_k) = -p_k' (J(x_k)' J(x_k)) p_k = -\| J(x_k) p_k \|^2_2$$

Note that $p_k$ can be obtained by solving the least-squares problem

$$p_k = \arg \min_p \frac{1}{2} \| J(x_k) p + r(x_k) \|^2_2$$

which is the linearized version of the problem at $x_k$, $r(x) \approx r(x_k) + J(x_k) p$. 
Any technique can be used to solve each least-squares problem

The Gauss-Newton (GN) method converges under mild assumptions

\[ \lim_{k \to \infty} \nabla f(x_k) = \lim_{k \to \infty} J(x_k)'r(x_k) = 0 \]

The Levenberg-Marquardt (LM) method is a damped version of GN, based on selecting \( p_k \) by solving the regularized system

\[ p_k = -(\rho_k I + J(x_k)'J(x_k))^{-1} J(x_k)'r(x_k) \]

The parameter \( \rho_k \) can be selected at each iteration by simple rules.

Note that LM \( \approx \) GN for \( \rho_k \ll 1 \), LM \( \approx \) gradient descent for \( \rho_k \gg 1 \).

The LM method can be reinterpreted also as a trust-region method.
• We have a data set of $N = 10000$ samples $(u_k, y_k)$

$$y_k = x_1^2 u_1k + x_1 x_2 u_2k - x_2 u_3^2k + n_k$$

where $x_1 = 0.5$, $x_2 = -1$ are unknown and noise $n_k \sim N(0, \sigma^2)$, $\sigma = 0.01$

• We want to estimate the parameter vector $x$ by minimizing

$$\min_x \frac{1}{2} \sum_{i=1}^{N} \|y_k - x_1^2 u_1k + x_1 x_2 u_2k - x_2 u_3^2k\|_2^2$$

• Gauss-Newton method converges in 6 ms$^1$ after 8 iterations, with stopping tolerance

$$\|\nabla f(x_k)\| \leq 10^{-4}$$

$^1$Macbook 3 GHz Intel Core i7, MATLAB R2016b
The spread of Coronavirus COVID-19 can be modeled by the **logistic model**\(^2\)

\[
n(t) = \frac{K}{1 + Ae^{-rt}}
\]

where \(n(t)\) = number of confirmed infected at time \(t\) and \(K\) = final epidemic size

We want to fit \((K, r, A)\) to data available for different countries\(^3,4\)

\[
\min_{K, r, A} \frac{1}{2} \sum_{j=1}^{m} \left\| n(t_j) - \frac{K}{1 + Ae^{-rt_j}} \right\|^2
\]

nonlinear least squares

Here \(r_j = n(t_j) - \frac{K}{1 + Ae^{-rt_j}}\), \(\nabla r_j = \frac{1}{(1 + Ae^{-rt_j})^2} \begin{bmatrix} 1 + Ae^{-rt_j} \\ -Ke^{-rt_j} \\ KAte^{-rt_j} \end{bmatrix}\)

---

\(^2\)See also (Batista, 2020) https://www.researchgate.net/publication/339240777


\(^4\)Data for Italy available at https://github.com/pcm-dpc/COVID-19
Results for China:

\[ K = 80917.6142 \]
\[ r = 0.2221 \]
\[ A = 51.6394 \]

Results for Italy:

\[ K = 59939.3989 \]
\[ r = 0.2344 \]
\[ A = 157.5456 \]

Problem solved using derivative-free Particle Swarm Optimization (Eberhart, Kennedy, 1995) via the `pyswarm` interface [https://pythonhosted.org/pyswarm/](https://pythonhosted.org/pyswarm/)
Reference:

We consider the equality-constrained NLP problem

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad h(x) = 0
\end{align*}
\]

with \( f : \mathbb{R}^n \to \mathbb{R} \) and \( h : \mathbb{R}^n \to \mathbb{R}^m \) smooth functions.

The Lagrangian function and its derivatives are

\[
\begin{align*}
\mathcal{L}(x, \nu) &= f(x) + \nu' h(x) \\
\nabla_x \mathcal{L}(x, \nu) &= \nabla f(x) + A'(x) \nu, \quad A'(x) = [ \nabla h_1(x) \ldots \nabla h_m(x) ] \\
\nabla_{xx} \mathcal{L}(x, \nu) &= \nabla^2 f(x) + \sum_{i=1}^{m} \nu_i \nabla^2 h_i(x)
\end{align*}
\]

Assume \( A(x) \) full row rank, \( d' \nabla^2_{xx} \mathcal{L}(x, \nu) d > 0, \forall d \neq 0 \) such that \( A(x)d = 0 \)
• For all \( \nu \in \mathbb{R}^m \), the original problem is equivalent to solving

\[
\begin{align*}
    \min & \quad f(x) + \nu' h(x) = \mathcal{L}(x, \nu) \\
    \text{s.t.} & \quad h(x) = 0
\end{align*}
\]

• Consider a pair \((x_k, \nu_k)\) and the quadratic approximation of the problem around \(x_k\)

\[
\begin{align*}
    \min & \quad f(x_k) + \nu_k' h(x_k) + (\nabla f(x_k) + A'(x_k)\nu_k)' p + \frac{1}{2} p' \nabla^2_{xx} \mathcal{L}(x_k, \nu_k) p \\
    \text{s.t.} & \quad h(x_k) + A(x_k) p = 0
\end{align*}
\]

• By exploiting \(\nu_k'(h(x_k) + A(x_k)p) = 0\), the QP is equivalent to

\[
\begin{align*}
    \min & \quad \frac{1}{2} p' \nabla^2_{xx} \mathcal{L}(x_k, \nu_k) p + \nabla f(x_k)' p \\
    \text{s.t.} & \quad h(x_k) + A(x_k) p = 0
\end{align*}
\]
SQP for Equality-Constrained NLP

- The optimality conditions for the QP are
  \[
  \begin{bmatrix}
  \nabla^2_{xx} \mathcal{L}(x_k, \nu_k) & A'(x_k) \\
  A(x_k) & 0
  \end{bmatrix}
  \begin{bmatrix}
  p_k \\
  \nu_{k+1}
  \end{bmatrix}
  =
  \begin{bmatrix}
  -\nabla f(x_k) \\
  -h(x_k)
  \end{bmatrix}
  \]

- From the solution \( p_k \) we set \( x_{k+1} = x_k + \alpha_k p_k \) (Newton’s step)

- Also, we decided to update \( \nu_{k+1} \) as the vector of Lagrange multipliers of the approximated QP

- Sequential quadratic programming (SQP) for equality-constrained NLP’s iterates the above steps from an initial pair \((x_0, \nu_0)\) until convergence

- Note that in case \( h(x) = Ax - b \) we have \( \nabla^2_{xx} \mathcal{L}(x, \nu) = \nabla^2 f(x) \)
• A similar reasoning applies to general NLP problems

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad g_i(x) \leq 0, \ i \in I \\
& \quad g_j(x) = 0, \ j \in E
\end{align*}
\]

with \( f : \mathbb{R}^n \rightarrow \mathbb{R}, g_i : \mathbb{R}^n \rightarrow \mathbb{R} \) smooth functions, \( \forall i = 1, \ldots, m \).

• The quadratic approximation is

\[
\begin{align*}
\min & \quad \frac{1}{2} p' \nabla^2 \mathcal{L}(x_k, \nu_k) p + \nabla f(x_k)' p \\
\text{s.t.} & \quad g_i(x_k) + \nabla g_i(x_k)' p \leq 0, \ i \in I \\
& \quad g_j(x_k) + \nabla g_j(x_k)' p = 0, \ j \in E
\end{align*}
\]

• We solve the QP, get the primal-dual solution \((p_k, \nu_{k+1})\), and update

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

• Several variants of the SQP method exist (including quasi-Newton methods)
References:


Consider the convex programming problem

\[
\begin{align*}
\text{min} & \quad f(x) \\
\text{s.t.} & \quad g_i(x) \leq 0, \ i = 1, \ldots, m \\
& \quad Ax = b
\end{align*}
\]

Assumptions:
- \( f, g_i \) convex and twice continuously differentiable
- \( A \in \mathbb{R}^{p \times n} \) has rank \( A = p \)
- an optimizer \( x^* \) exists and \( f^* = f(x^*) \in \mathbb{R} \)
- the problem is strictly feasible

\[
\exists x \ \text{dom} \ f : g_i(x) < 0, \ \forall i = 1, \ldots, m, \ Ax = b
\]

which ensures strong duality, i.e., \( f(x^*) = q(\lambda^*, \nu^*) \), \( q = \) dual function
Denote by $I : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ the \textbf{indicator function} of the negative reals ($I(\alpha) = 0$ if $\alpha \leq 0$, $I(\alpha) = +\infty$ if $\alpha > 0$). The problem can be rewritten as

$$\min_{x} f(x) + \sum_{i=1}^{m} I(g_i(x))$$

s.t. $Ax = b$

If we approximate $I(\alpha)$ with the smooth \textbf{logarithmic barrier} function $-\frac{1}{t} \log(-\alpha)$, $t > 0$, we get

$$\min_{x} f(x) - \frac{1}{t} \sum_{i=1}^{m} \log(-g_i(x))$$

s.t. $Ax = b$

The larger $t$ the better the approximation
• The logarithmic barrier function

\[ \phi(x) = - \sum_{i=1}^{m} \log(-g_i(x)) \]

has the following properties:

- \( \text{dom } \phi = \{ x : g_i(x) < 0, \ i = 1, \ldots, m \} \)
- \( \phi \) is convex, since \( -\log \) is monotonic and \( g_i \) is convex
- \( \phi \) is twice continuously differentiable and

\[
\nabla \phi(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)} \nabla g_i(x)
\]

\[
\nabla^2 \phi(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)^2} \nabla g_i(x)\nabla g_i(x)' - \sum_{i=1}^{m} \frac{1}{g_i(x)} \nabla^2 g_i(x)
\]
For $t \geq 0$ let $x^*(t)$ be the optimizer of the approximated problem

$$\min \quad tf(x) + \phi(x)$$
$$\text{s.t.} \quad Ax = b$$

(assume for now that $x^*(t)$ is unique for all $t$)

We call **central path** the curve $\{x^*(t)\}_{t>0}$

Example: central path for the linear program

$$\min \quad c'x$$
$$\text{s.t.} \quad Gx \leq g, \quad g \in \mathbb{R}^5$$

and level sets of $\phi(x)$
• The original problem satisfies the optimality conditions

\[ \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + A' \nu^* = 0 \]
\[ \lambda^* \geq 0, \quad Ax^* = b, \quad g_i(x^*) \leq 0, \quad \lambda_i^* g_i(x^*) = 0 \]

• The approximated optimizer \( x^*(t) \) satisfies the optimality conditions

\[ \nabla f(x^*(t)) - \sum_{i=1}^{m} \frac{1}{tg_i(x^*(t))} \nabla g_i(x^*(t)) + \frac{1}{t} A' w^*(t) = 0 \]
\[ Ax^*(t) = b \]

where \( w^*(t) \) is the corresponding vector of Lagrange multipliers.
• If we set $\lambda_i^*(t) \triangleq -\frac{1}{tg_i(x^*(t))}$, $\nu^*(t) = \frac{1}{t}w^*(t)$, for all $t > 0$ we have

$$\nabla f(x^*(t)) + \sum_{i=1}^{m} \lambda_i^*(t)\nabla g_i(x^*(t)) + A'\nu^*(t) = 0, \quad Ax^*(t) = b$$

$$\lambda_i^*(t) \geq 0, \quad \lambda_i^*g_i(x^*(t)) = -\frac{1}{t}$$

• These are the same KKT conditions of the original problem, except for the relaxation of the complementary slackness condition to $\lambda_i^*g_i(x^*(t)) = -\frac{1}{t}$
• The dual function $q$ of the original problem evaluated at $\lambda^*(t), \nu^*(t)$ is

\[
q(\lambda^*(t), \nu^*(t)) = \min_x \mathcal{L}(x, \lambda^*(t), \nu^*(t)) = \min_x f(x) + \sum_{i=1}^{m} \lambda_i^* (t) g_i(x) + (Ax - b)' \nu^*(t) \\
= f(x^*(t)) - \frac{m}{t}
\]

as $x^*(t)$ also satisfies the optimality condition $\nabla_x \mathcal{L}(x, \lambda^*(t), \nu^*(t)) = 0$

• Since $q(\lambda^*(t), \nu^*(t)) \leq f(x^*)$, and since $f(x^*) \leq f(x^*(t))$ as $x^*(t)$ is feasible, we get

\[
f(x^*(t)) - \frac{m}{t} \leq f(x^*) \leq f(x^*(t))
\]

which confirms the intuition $f(x^*(t)) \to f(x^*)$ as $t \to +\infty$
• Consider the **feasibility problem**

\[
\text{find } x \text{ such that } g_i(x) \leq 0, \ i = 1, \ldots, m, \ Ax = b
\]

• The **basic phase I method** consists of solving the following convex problem with \( n + 1 \) variables

\[
(x^*_0, s^*_0) = \arg \min_{x,s} s \\
\text{s.t. } g_i(x) - s \leq 0, \ i = 1, \ldots, m \\
Ax = b
\]

from any initial guess \( x_0 \) such that \( Ax_0 = b, s_0 > \max g_i(x_0) \)

• If \( s^*_0 < 0 \) then \( x^*_0 \) is strictly feasible for the original problem
Barrier method: Given an initial strictly feasible $x$, execute:

0. Let $t_0 > 0$, $t = t_0$, $\beta > 1$, tolerance $\epsilon > 0$

1. Compute $x \leftarrow \arg \min_t tf(x) + \phi(x)$ (centering step)
   
   s.t. $Ax = b$

2. If $\frac{m}{t} \leq \epsilon$ stop

3. Otherwise increase $t \leftarrow \beta t$ and go to 1

Newton’s method solves the centering step, with the last $x$ as initial guess

Tradeoff: a large $\beta$ makes fewer centering steps but more Newton iterations at each step. Typically $\beta = 10 \div 20$

The algorithm terminates with $f(x) - f^*(x) \leq \frac{m}{t} \leq \epsilon$ in exactly $\left\lceil \frac{\log \frac{m}{\epsilon t_0}}{\log \beta} \right\rceil$ centering steps + computation of initial $x^*(t_0)$
• Consider the general NLP problem

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad g(x) \leq 0 \\
& \quad h(x) = 0
\end{align*}
\]

with \( f : \mathbb{R}^n \to \mathbb{R} \), \( g : \mathbb{R}^n \to \mathbb{R}^m \), \( h : \mathbb{R}^n \to \mathbb{R}^p \) smooth functions.

• The optimality condition for the NLP with slacks can be written as

\[
\nabla f(x) + \nabla g(x)'z + \nabla h(x)'y = 0 \\
g(x) + s = 0 \quad S = \text{diag}(s) \\
h(x) = 0 \quad Z = \text{diag}(z) \\
SZe = 0 \quad e = [1 \ldots 1]' \\
s, z \geq 0
\]
Let us now relax the optimality conditions as

\[
\nabla f(x) + \nabla g(x)'z + \nabla h(x)'y = 0 \\
g(x) + s = 0 \\
h(x) = 0 \\
SZe = \mu e \\
s, z \geq 0
\]

Let \( x^*(\mu), s^*(\mu), y^*(\mu), z^*(\mu) \) be the solution of the relaxed KKT equations

For \( \mu > 0 \) the curve \((x^*(\mu), s^*(\mu), y^*(\mu), z^*(\mu))\) is the \textbf{primal-dual central path}

Note that \( \mu = s^*_i(\mu)z^*_i(\mu) = -g_i(x^*(\mu))z^*_i(\mu) = \frac{1}{t} \)

For \( \mu \to 0 \), under suitable assumptions, the central path converges to the primal/dual optimizer \((x^*, s^*, y^*, z^*)\)
• Primal-dual interior point methods apply a Newton step to solve the system of relaxed KKT equations with decreasing values of $\mu$

• They are more efficient than barrier method when high accuracy is needed

• Often exhibit superlinear asymptotic convergence

• They can start at infeasible points
Let us consider the LP

\[
\begin{align*}
\min_x & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad Ex = f
\end{align*}
\]

By introducing the slack vector \( s = b - Ax \), the KKT conditions

\[
\begin{align*}
c + A'z + E'y &= 0 \\
Ax + s &= b \\
Ex &= f \\
z_is_i &= 0, \ i = 1, \ldots, m \\
z, s &\geq 0
\end{align*}
\]

can be rewritten as

\[
F(x, z, y, s) = \begin{bmatrix}
A'z + E'y + c \\
Ax + s - b \\
Ex - f \\
ZSe
\end{bmatrix} = 0, \quad z, s \geq 0
\]

where \( Z = \text{diag}(z_1, \ldots, z_m) \), \( S = \text{diag}(s_1, \ldots, s_m) \), \( e = [1 \ldots 1]' \)
• We want to solve the nonlinear system \( F(x, z, y, s) = 0 \) by Newton’s method

• Starting from a candidate solution \( z > 0, s > 0, x, y, \) Newton’s step \( \Delta x, \Delta z, \Delta y, \Delta s \) is given by solving the linear system

\[
0 = F(x, z, y, s) + \nabla F(x, z, y, s) \begin{bmatrix} \Delta x \\ \Delta z \\ \Delta y \\ \Delta s \end{bmatrix}
\]

• Let

\[
\begin{bmatrix} r^c \\ r^b \\ r^f \end{bmatrix} = \begin{bmatrix} A'z + E'y + c \\ Ax + s - b \\ Ex - f \end{bmatrix}
\]

. The linear system to solve is

\[
\begin{bmatrix}
0 & A' & E' & 0 \\
A & 0 & 0 & I \\
E & 0 & 0 & 0 \\
0 & S & 0 & Z
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta z \\
\Delta y \\
\Delta s
\end{bmatrix} =
\begin{bmatrix}
-r^c \\
-r^b \\
-r^f \\
-ZSe
\end{bmatrix}
\]

• To preserve \( z, s \geq 0 \) we set

\[
\begin{bmatrix} x \\ z \\ y \\ s \end{bmatrix} \leftarrow \begin{bmatrix} x \\ z \\ y \\ s \end{bmatrix} + \alpha \begin{bmatrix} \Delta x \\ \Delta z \\ \Delta y \\ \Delta s \end{bmatrix}
\]

with \( \alpha \) sufficiently small
To prevent excessively small $\alpha$, given the current $x_k, s_k, \lambda_k, x_k, s_k > 0$, primal-dual interior-point method solve instead the relaxed system

$$
\begin{bmatrix}
0 & A' & E' & 0 \\
A & 0 & 0 & I \\
E & 0 & 0 & 0 \\
0 & S_k & 0 & Z_k
\end{bmatrix}
\begin{bmatrix}
\Delta x_k \\
\Delta z_k \\
\Delta y_k \\
\Delta s_k
\end{bmatrix}
= 
\begin{bmatrix}
-r_c^k \\
-r_b^k \\
-r_f^k \\
-Z_kS_ke + \sigma_k\mu_ke
\end{bmatrix}
$$

where $\mu_k = \frac{1}{m} z'_k s_k$ is the current duality measure and $\sigma_k \in [0, 1]$ is the centering parameter, that is the factor we want to reduce the current $\mu_k$.

- The performance of the method depends on how $\alpha_k$ and $\sigma_k$ are chosen.

- Mehrotra’s predictor-corrector algorithm is one of the most used IP methods for LP (Mehrotra, 1992).

- Homogeneous and self-dual formulations are useful to easily recognize infeasibility and unboundedness (Yu, Todd, Mizuno, 1994) (Xu, Hung, Ye, 1996).
• Consider the convex QP

$$\min_x \frac{1}{2} x' Q x + c' x$$

s.t. $Ax \leq b$

$$Ex = f$$

$$Q = Q' \succeq 0$$

• By introducing the slack vector $s = b - Ax$, the KKT conditions

$$Qx + c + E'y + A'z = 0$$

$$Ex = f$$

$$Ax + s = b$$

$$z_i s_i = 0, \quad i = 1, \ldots, m$$

$$z, s \geq 0$$

can be rewritten as

$$0 = F(x, z, y, s) = \begin{bmatrix} Qx + E'y + A'z + c \\ Ex - f \\ Ax + s - b \\ ZSe \end{bmatrix} \triangleq \begin{bmatrix} r_Q \\ r_E \\ r_A \\ r_S \end{bmatrix}, \quad z, s \geq 0$$
Start from a candidate solution \( z > 0, s > 0, x, y \)

As for LP, we want to solve \( F(x, z, y, s) = 0 \) by Newton’s method

We use a variant of Mehrotra’s predictor-corrector algorithm (Mehrotra, 1992)

First, we solve the linear system (predictor step)

\[
\begin{pmatrix}
Q & E' & A' & 0 \\
E & 0 & 0 & 0 \\
A & 0 & 0 & I \\
0 & 0 & S & Z \\
\end{pmatrix}
\begin{pmatrix}
\Delta x_{aff} \\
\Delta y_{aff} \\
\Delta z_{aff} \\
\Delta s_{aff} \\
\end{pmatrix}
= 
\begin{pmatrix}
-rQ \\
-rE \\
-rA \\
-rS \\
\end{pmatrix}
\]
• Next, we solve the linear system (centering-corrector step)

\[
\begin{bmatrix}
Q & E' & A' & 0 \\
E & 0 & 0 & 0 \\
A & 0 & 0 & I \\
0 & 0 & S & Z
\end{bmatrix}
\begin{bmatrix}
\Delta x_{cc} \\
\Delta y_{cc} \\
\Delta z_{cc} \\
\Delta s_{cc}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
-\Delta S_{aff} \Delta Z_{aff} e + \sigma \mu e
\end{bmatrix}
\]

where the centering parameter $\sigma \in [0, 1)$ is chosen as

\[
\alpha_{aff} = \arg \max_\alpha \{ \alpha \in [0, 1] : \begin{bmatrix}
z + \alpha \Delta z_{aff} \\
\text{for S} + \alpha \Delta s_{aff}
\end{bmatrix} \geq 0\}
\]

\[
\mu_{aff} = (z + \alpha_{aff} \Delta z_{aff})' (s + \alpha_{aff} \Delta s_{aff}) / m
\]

\[
\mu = z's/m \quad \leftarrow \text{duality gap}
\]

\[
\sigma = (\mu_{aff} / \mu)^3
\]

• **Note:** the same left-hand-side matrix is used to solve both linear systems. So such a matrix can be factorized just once at each IP iteration.
Now set
\[
\Delta x = \Delta x_{\text{aff}} + \Delta x_{\text{cc}}, \quad \Delta y = \Delta y_{\text{aff}} + \Delta y_{\text{cc}} \\
\Delta z = \Delta z_{\text{aff}} + \Delta z_{\text{cc}}, \quad \Delta s = \Delta s_{\text{aff}} + \Delta s_{\text{cc}}
\]
and choose \( \alpha_{\text{max}} = \arg \max \{ \alpha \in [0, 1] : \left[ \frac{z + \alpha \Delta z}{s + \alpha \Delta s} \right] \geq 0 \} \), so that \( z, s \) remain nonnegative

The actual step-length is chosen as \( \alpha = \gamma \alpha_{\text{max}} \), with the step-factor \( \gamma \in (0, 1) \) close to 1, see (Mehrotra, 1992)

For even better choices of the step-length \( \alpha \) see (Curtis, Nocedal, 2007)

For reducing the number of factorizations, execute multiple corrections steps (Gondzio, 1996)

Given a starting point \( \bar{x}, \bar{y}, \bar{z}, \bar{s} \), a good initial guess is to solve for \( \Delta z_{\text{aff}}, \Delta s_{\text{aff}} \) and set \( x_0 = \bar{x}, y_0 = \bar{y}, z_0 = \max\{1, |\bar{z} + \Delta z_{\text{aff}}|\}, s_0 = \max\{1, |\bar{s} + \Delta s_{\text{aff}}|\} \)
• Let $\Delta \tilde{z} = Z^{-1} \Delta z$. We can eliminate $\Delta s = Z^{-1} r_S - S \Delta \tilde{z}$ and get the system

$$
\begin{bmatrix}
Q & E' & A'Z \\
E & 0 & 0 \\
A & 0 & -S
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta \tilde{z}
\end{bmatrix}
= 
\begin{bmatrix}
-r_Q \\
-r_E \\
Z^{-1} r_S - r_A
\end{bmatrix}
$$

• The above system can be made symmetric by multiplying the last rows by $Z$

• We can further easily eliminate $Z^{-1} \Delta z = S^{-1}(A \Delta x + r_A - Z^{-1} r_S)$ and get

$$
\begin{bmatrix}
Q + A' Z S^{-1} A & E' \\
E & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta y
\end{bmatrix}
= 
\begin{bmatrix}
-r_Q + A' S^{-1}(r_S - Z r_A) \\
-r_E
\end{bmatrix}
$$

• Note that $Z^{-1} S$ is positive and diagonal.
**Example: NLP Solution via IPOPT & CasADi**

- **IPOPT** (Interior Point OPTimizer\(^5\)) is a software package based on an IP method to solve the NLP (Wächter, Biegler, 2006)

  \[
  \begin{align*}
  \min_x & \quad f(x) \\
  \text{s.t.} & \quad g_\ell \leq g(x) \leq g_u \\
  & \quad x_\ell \leq x \leq x_u
  \end{align*}
  \]

- **CasADi\(^6\)** is a modeling language for NLP problems. It implements automatic differentiation for computing gradients (Andersson, Gillis, Horn, Rawlings, Diehl, 2019)

- **CasADi + IPOPT** greatly simplifies formulating and solving nonlinear optimization problems via interior-point methods in MATLAB, \(\text{\footnotesize\href{https://web.casadi.org/}{\texttt{\textcopyright}}\text{\footnotesize\href{https://web.casadi.org/}{\texttt{\textregistered}}} \texttt{\textcopyright2021 A. Bemporad - Numerical Optimization}}\), \(\text{\href{https://web.casadi.org/}{\texttt{python}}}, \text{\href{https://web.casadi.org/}{\texttt{\textcopyright}}} \text{\href{https://web.casadi.org/}{\texttt{C++}}}

---

\(^5\)https://coin-or.github.io/Ipopt/

\(^6\)https://web.casadi.org/
Let us minimize the course-logo function

\[ f(x, y) = -e^{-(x^2+y^2)} + 0.3 \sin \left( \frac{1}{10} x^3 + y^2 \right) + 1.2 \]

**Python**

```python
import casadi.*
x=SX.sym('x');
y=SX.sym('y');
f=-exp(-(x**2+y**2))+.3*sin(x**3/10+y**2)+1.2;
P=struct('f',f,'x',[x;y]);
F=nlpsol('F','ipopt',P);
r=F('x0',[-1,-1]);
xopt=full(r.x);
fopt=full(r.f);
```

**MATLAB**

```matlab
from casadi import *
x=SX.sym('x')
y=SX.sym('y')
f=-exp(-(x**2+y**2))+.3*sin(x**3/10+y**2)+1.2
P=dict(x=vertcat(x,y), f=f)
F=nlpsol('F','ipopt',P)
r=F(x0=[-1,-1])
xopt=r['x'].full()
fopt=r['f'].full()
```

**Optimizer** \( x^* = 0, y^* = 0, \) optimum \( f^* = 0.2 \)
Example: NLP Solution via IPOPT & CasADi

Number of objective function evaluations = 18
Number of objective gradient evaluations = 11
Number of equality constraint evaluations = 0
Number of inequality constraint evaluations = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations = 10

Let’s add the constraint

\[ 1 \leq (x + 2)^2 - \frac{1}{2} y^3 \leq 3 \]

\[
g = (x+2)^2 - y^3/2\;
P = \text{struct}(\text{f}, f, \text{'x'}, [x;y], \text{'g'}, g);
F = \text{nlpsol}(\text{'F'}, \text{'ipopt'}, P);
r = F(\text{'x0'}, [-1;-1], \text{'ubg'}, 3, \text{'lbg'}, 1);
xopt = \text{full}(r.x);
fopt = \text{full}(r.f);
lam_g_opt = \text{full}(r.lam_g);
\]

MATLAB

Python

\[ g = (x+2)^2 - y^3/2 \]
\[ P = \text{dict}(x=\text{vertcat}(x,y), f=f, g=g) \]
\[ F = \text{nlpsol}(\text{'F'}, \text{'ipopt'}, P) \]
\[ r = F(x0=[-1,-1], ubg=3, lbg=1) \]
\[ xopt = r[\text{'x'}].\text{full()} \]
\[ fopt = r[\text{'f'}].\text{full()} \]
\[ lam_g_opt = r[\text{'lam_g'}].\text{full()} \]

New optimizer \[ x^* = -0.2679, y^* = 0, \text{optimum } f^* = 0.2687 \]
• **Caveat**: the NLP is non convex.

• If we start from $x_0 = -2, y_0 = -2$ we get the different local minimum

$$x^* = -1.5078, y^* = -1.7668, \text{ optimum } f^* = 1.3019!$$

• If function/constraints are not convex, one may need to test different initial conditions, or switch to global optimization methods.