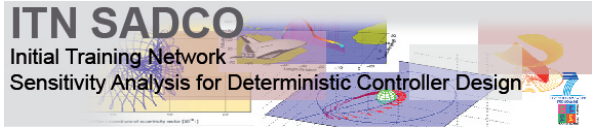
Numerical algorithms for nonlinear optimization problems

Sonja Rauški

22. Januar 2013



Outline

- Nonlinear optimization
- Numerical methods for nonlinear optimization
  - Newton-type optimization
  - Active set method
  - Sequential Quadratic Programming
  - Primal-dual interior point method
- Examples

Nonlinear optimization

$$\begin{array}{lll} \min\limits_{x} & f(x) & \\ \text{s.t.} & g(x) = 0, \\ & h(x) \leq 0. \end{array}$$

where: $x \in \mathbb{R}^n, f \in \mathbb{R}, g \in \mathbb{R}^{m_e}, h \in \mathbb{R}^{m_i}$

- Function f is the cost/ objective function
- Functions g and h equality and inequality constraints, resp.
- The set of indices i for which $h_i(x) = 0$ is named the active set
- f, g, h may be nonlinear
- Nonlinear optimization $\Leftrightarrow$ Nonlinear programming (NLP)

KKT conditions

- A most of solvers try to find an approximate KKT point.
- Find the "primal-dual" variables x^*, ν^*, λ^* such that:

$g(x^*) = 0, h(x^*) \leq 0$	*Primal feasibility*
$\nu^* \geq 0$	*Dual feasibility*
$\nu_i^* h_i(x^*) = 0, i = 1, ..., m_i$	*Complementary slackness*
$\nabla \mathcal{L}(x^*, \nu^*, \lambda^*) = 0$	*Stationarity*

where:
Lagrangian function:
$\mathcal{L}(x, \nu, \lambda) = f(x) + \sum_{i=1}^{m_e} \lambda_i g_i(x) + \sum_{i=1}^{m_i} \nu_i h_i(x)$
Lagrangian multipliers: λ_i, ν_i

KKT conditions

- A most of solvers try to find an approximate KKT point.
- Find the "primal-dual" variables x^*, ν^*, λ^* such that:

$g(x^*) = 0, h(x^*) \leq 0$	*Primal feasibility*
$\nu^* \geq 0$	*Dual feasibility*
$\nu_i^* h_i(x^*) = 0, i = 1, ..., m_i$	*Complementary slackness*
$\nabla \mathcal{L}(x^*, \nu^*, \lambda^*) = 0$	*Stationarity*

where:
Lagrangian function:
$\mathcal{L}(x, \nu, \lambda) = f(x) + \sum_{i=1}^{m_e} \lambda_i g_i(x) + \sum_{i=1}^{m_i} \nu_i h_i(x)$
Lagrangian multipliers: λ_i, ν_i

Numerical methods for nonlinear optimization

- Unconstrained optimization

$$\min_x \; f(x)$$

- Equality constrained optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \end{aligned}$$

- Equality and inequality constrained optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) \leq 0. \end{aligned}$$

Numerical methods for nonlinear optimization

- Unconstrained optimization

$$\min_x \ f(x)$$

- Equality constrained optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \end{aligned}$$

- Equality and inequality constrained optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) \leq 0. \end{aligned}$$

Numerical methods for nonlinear optimization

- Unconstrained optimization

$$\min_x \ f(x)$$

- Equality constrained optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \end{aligned}$$

- Equality and inequality constrained optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) \leq 0. \end{aligned}$$

Newton-type optimization

- Find x^* such that $\nabla f(x^*) = 0$
- $x_{k+1} = x_k + d_k$
- Linearizing the non-linear equations at x_k
  $\nabla f(x_k) + \nabla^2 f(x_k) d_k = 0$
- Newton step: $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$

- Hessian matrix $\nabla^2 f(x_k)$ expensive $\rightarrow B_k \approx \nabla^2 f(x_k)$

Newton-type optimization

- Find x^* such that $\nabla f(x^*) = 0$
- $x_{k+1} = x_k + d_k$
- Linearizing the non-linear equations at x_k
  $\nabla f(x_k) + \nabla^2 f(x_k) d_k = 0$
- Newton step: $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$

- Hessian matrix $\nabla^2 f(x_k)$ expensive $\rightarrow B_k \approx \nabla^2 f(x_k)$

Newton-type optimization

- Find x^* such that $\nabla f(x^*) = 0$
- $x_{k+1} = x_k + d_k$
- Linearizing the non-linear equations at x_k
  $\nabla f(x_k) + \nabla^2 f(x_k) d_k = 0$
- Newton step: $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$

- Hessian matrix $\nabla^2 f(x_k)$ expensive $\rightarrow B_k \approx \nabla^2 f(x_k)$

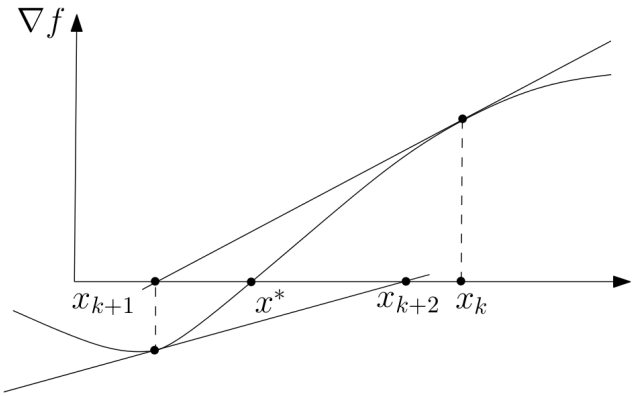
Newton-type optimization

- Find x^* such that $\nabla f(x^*) = 0$
- $x_{k+1} = x_k + d_k$
- Linearizing the non-linear equations at x_k
  $\nabla f(x_k) + \nabla^2 f(x_k) d_k = 0$
- Newton step: $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$

- Hessian matrix $\nabla^2 f(x_k)$ expensive $\rightarrow B_k \approx \nabla^2 f(x_k)$

Newton-type optimization

- Find x^* such that $\nabla f(x^*) = 0$
- $x_{k+1} = x_k + d_k$
- Linearizing the non-linear equations at x_k
  $\nabla f(x_k) + \nabla^2 f(x_k) d_k = 0$
- Newton step: $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$



- Hessian matrix $\nabla^2 f(x_k)$ expensive $\rightarrow B_k \approx \nabla^2 f(x_k)$

Steepest descent

- $B_k = \alpha_k^{-1} I$
- $d_k = -B_k^{-1} \nabla f(x_k) = -\alpha_k \nabla f(x_k)$
- How to choose α_k? $\rightarrow$ Line search
- Linear convergence when x_k is close to x^*

Steepest descent

- $B_k = \alpha_k^{-1} I$
- $d_k = -B_k^{-1} \nabla f(x_k) = -\alpha_k \nabla f(x_k)$
- How to choose α_k? $\rightarrow$ Line search
- Linear convergence when x_k is close to x^*

Steepest descent

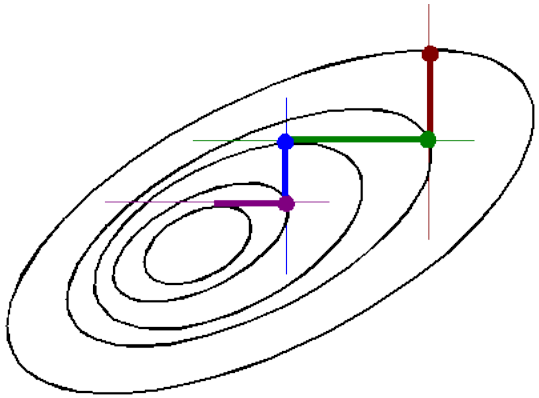
- $B_k = \alpha_k^{-1} I$
- $d_k = -B_k^{-1} \nabla f(x_k) = -\alpha_k \nabla f(x_k)$
- How to choose α_k? $\rightarrow$ Line search
- Linear convergence when x_k is close to x^*

Steepest descent

- $B_k = \alpha_k^{-1} I$
- $d_k = -B_k^{-1} \nabla f(x_k) = -\alpha_k \nabla f(x_k)$
- How to choose α_k? $\rightarrow$ Line search
- Linear convergence when x_k is close to x^*

Steepest descent

- $B_k = \alpha_k^{-1} I$
- $d_k = -B_k^{-1} \nabla f(x_k) = -\alpha_k \nabla f(x_k)$
- How to choose α_k? $\rightarrow$ Line search
- Linear convergence when x_k is close to x^*



Quasi-Newton methods

- Computing $\nabla^2 f(x)$ numerically
- BFGS

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

Secant condition: Find B_{k+1} such that $B_{k+1} s_k = y_k$

BFGS formula: $$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \quad B_0 = I$$

- Quadratic convergence when x_k is close to x^*

Quasi-Newton methods

- Computing $\nabla^2 f(x)$ numerically
- BFGS

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

Secant condition: Find B_{k+1} such that $B_{k+1} s_k = y_k$

BFGS formula: $$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \quad B_0 = I$$

- Quadratic convergence when x_k is close to x^*

Quasi-Newton methods

- Computing $\nabla^2 f(x)$ numerically
- BFGS

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

Secant condition: Find B_{k+1} such that $B_{k+1} s_k = y_k$

BFGS formula: $$\boxed{B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \quad B_0 = I}$$

- Quadratic convergence when x_k is close to x^*

Quasi-Newton methods

- Computing $\nabla^2 f(x)$ numerically
- BFGS

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

Secant condition: Find B_{k+1} such that $B_{k+1} s_k = y_k$

BFGS formula: $$\boxed{B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \quad B_0 = I}$$

- Quadratic convergence when x_k is close to x^*

Line search strategy

Problem: situation $f(x_k + d_k) > f(x_k)$ can occur

Idea: making the steps in the iteration shorter to ensure descent

$$f(x_k + d_k) < f(x_k)$$

Find the step size such that: $t = arg \min_{s \in (0,1]} f(x_k + s d_k)$

But, steps shouldn't get too short either!

Armijo's backtracking line search:

Given d_k, $0 < \alpha \leq \frac{1}{2}$ and $0 < \beta \leq 1$
$t = 1$
while $f(x_k + t d_k) \geq f(x_k) + \alpha t \nabla f(x_k)^T d_k$ **do**
$t = \beta t$
end while

Line search strategy

Problem: situation $f(x_k + d_k) > f(x_k)$ can occur

Idea: making the steps in the iteration shorter to ensure descent

$$f(x_k + d_k) < f(x_k)$$

Find the step size such that: $t = arg \min_{s \in (0,1]} f(x_k + s d_k)$

But, steps shouldn't get too short either!

Armijo's backtracking line search:

Given d_k, $0 < \alpha \leq \frac{1}{2}$ and $0 < \beta \leq 1$

$t = 1$

while $f(x_k + t d_k) \geq f(x_k) + \alpha t \nabla f(x_k)^T d_k$ **do**

$t = \beta t$

end while

Line search strategy

Problem: situation $f(x_k + d_k) > f(x_k)$ can occur

Idea: making the steps in the iteration shorter to ensure descent

$$f(x_k + d_k) < f(x_k)$$

Find the step size such that: $t = arg \min_{s \in (0,1]} f(x_k + s d_k)$

But, steps shouldn't get too short either!

Armijo's backtracking line search:

Given d_k, $0 < \alpha \leq \frac{1}{2}$ and $0 < \beta \leq 1$
$t = 1$
while $f(x_k + t d_k) \geq f(x_k) + \alpha t \nabla f(x_k)^T d_k$ **do**
$t = \beta t$
end while

Line search strategy

Problem: situation $f(x_k + d_k) > f(x_k)$ can occur

Idea: making the steps in the iteration shorter to ensure descent

$$f(x_k + d_k) < f(x_k)$$

Find the step size such that: $t = arg \min_{s \in (0,1]} f(x_k + s d_k)$

But, steps shouldn't get too short either!

Armijo's backtracking line search:

Given d_k, $0 < \alpha \leq \frac{1}{2}$ and $0 < \beta \leq 1$

$t = 1$

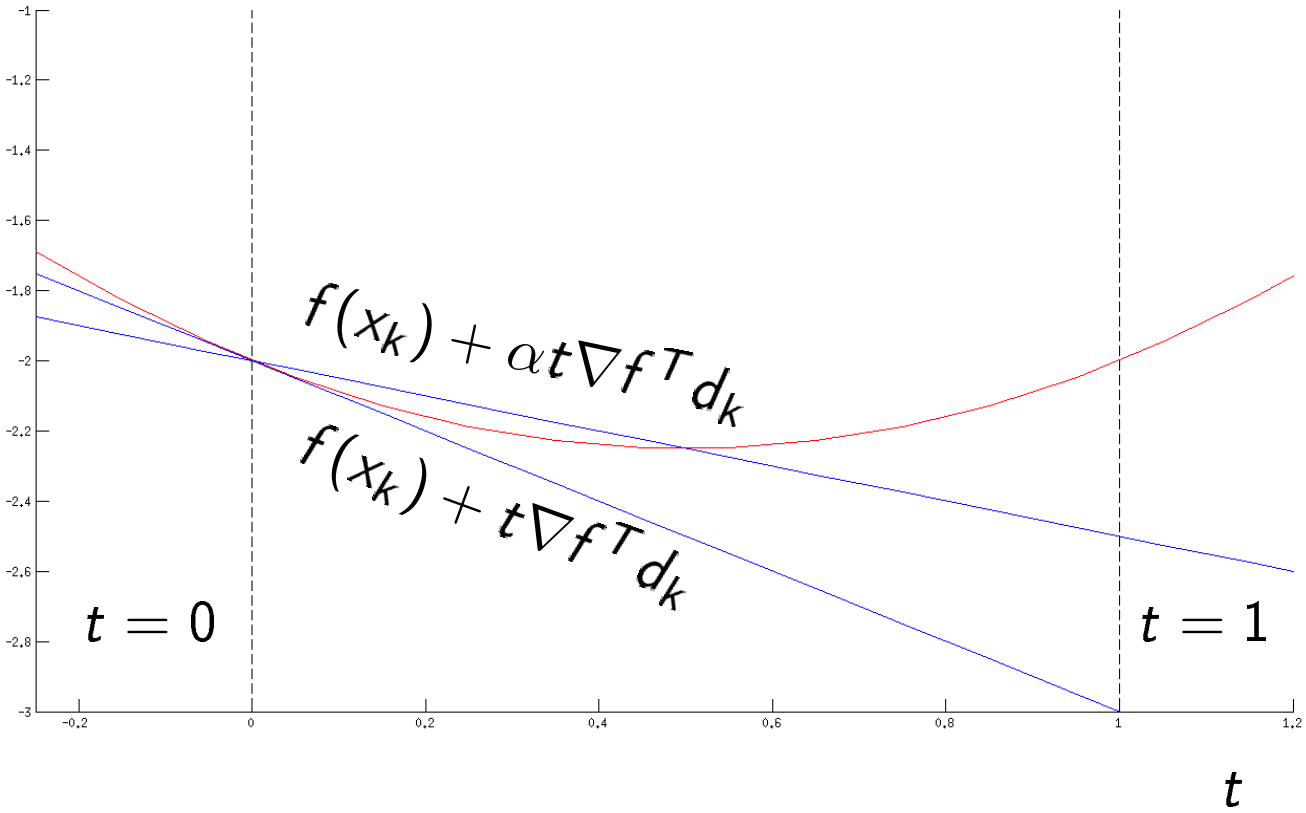
while $f(x_k + t d_k) \geq f(x_k) + \alpha t \nabla f(x_k)^T d_k$ **do**

$t = \beta t$

end while

Line search strategy

$f(x_k + t\nabla f^T d_k)$



Newton method for equality constrained problems

Find the "primal-dual" variables x^*, λ^* such that:

$$\boxed{\begin{aligned} \nabla \mathcal{L}(x^*, \lambda^*) &= 0 \\ g(x^*) &= 0 \end{aligned}}$$

Applying a Newton search, we get Newton recursion for solving the KKT conditions:

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} x_{k+1} - x_k \\ -(\lambda_{k+1} - \lambda_k) \end{pmatrix} + \begin{pmatrix} \nabla \mathcal{L}(x_k, \lambda_k) \\ g(x_k) \end{pmatrix} = 0$$

replacing $x_{k+1} = x_k + d_k$ and $\nabla \mathcal{L}(x_k, \lambda_k) = \nabla f(x_k) + \nabla g(x_k) \lambda_k$

Newton method for equality constrained problems

Find the "primal-dual" variables x^*, λ^* such that:

$$\boxed{\begin{aligned} \nabla \mathcal{L}(x^*, \lambda^*) &= 0 \\ g(x^*) &= 0 \end{aligned}}$$

Applying a Newton search, we get Newton recursion for solving the KKT conditions:

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} x_{k+1} - x_k \\ -(\lambda_{k+1} - \lambda_k) \end{pmatrix} + \begin{pmatrix} \nabla \mathcal{L}(x_k, \lambda_k) \\ g(x_k) \end{pmatrix} = 0$$

replacing $x_{k+1} = x_k + d_k$ and $\nabla \mathcal{L}(x_k, \lambda_k) = \nabla f(x_k) + \nabla g(x_k)\lambda_k$

Newton method for equality constrained problems

Find the "primal-dual" variables x^*, λ^* such that:

$$\boxed{\begin{aligned} \nabla \mathcal{L}(x^*, \lambda^*) &= 0 \\ g(x^*) &= 0 \end{aligned}}$$

Applying a Newton search, we get Newton recursion for solving the KKT conditions:

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} x_{k+1} - x_k \\ -(\lambda_{k+1} - \lambda_k) \end{pmatrix} + \begin{pmatrix} \nabla \mathcal{L}(x_k, \lambda_k) \\ g(x_k) \end{pmatrix} = 0$$

replacing $x_{k+1} = x_k + d_k$ and $\nabla \mathcal{L}(x_k, \lambda_k) = \nabla f(x_k) + \nabla g(x_k)\lambda_k$

Newton method for equality constrained problems

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} d_k \\ -\lambda_{k+1} \end{pmatrix} + \begin{pmatrix} \nabla f(x_k) \\ g(x_k) \end{pmatrix} = 0$$

KKT system is equivalent to the KKT of Qudratic Program (QP)

$$\begin{aligned} \min_{d_k} \quad & \tfrac{1}{2} d_k^T \nabla^2 \mathcal{L}(x_k, \lambda_k) d_k + \nabla f(x_k)^T d_k \\ \text{s.t.} \quad & g(x_k) + \nabla g(x_k)^T d_k = 0 \end{aligned}$$

Newton method for equality constrained problems

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x_k, \lambda_k) & \nabla g(x_k) \\ \nabla g(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} d_k \\ -\lambda_{k+1} \end{pmatrix} + \begin{pmatrix} \nabla f(x_k) \\ g(x_k) \end{pmatrix} = 0$$

KKT system is equivalent to the KKT of Qudratic Program (QP)

$$\begin{aligned} \min_{d_k} \quad & \tfrac{1}{2} d_k^T \nabla^2 \mathcal{L}(x_k, \lambda_k) d_k + \nabla f(x_k)^T d_k \\ \text{s.t.} \quad & g(x_k) + \nabla g(x_k)^T d_k = 0 \end{aligned}$$

Equality and inequality constrained problems

Find the "primal-dual" variables x^*, ν^*, λ^* such that:

$$
\begin{aligned}
& g(x^*) = 0, h(x^*) \leq 0 \\
& \nu^* \geq 0 \\
& \nu_i^* h_i(x^*) = 0, i = 1, ..., m_i \\
& \nabla \mathcal{L}(x^*, \nu^*, \lambda^*) = 0
\end{aligned}
$$

- Active set method
- Sequential Quadratic Programming
- Primal-dual interior point method

Equality and inequality constrained problems

Find the "primal-dual" variables x^*, ν^*, λ^* such that:

$$
\begin{aligned}
& g(x^*) = 0, h(x^*) \leq 0 \\
& \nu^* \geq 0 \\
& \nu_i^* h_i(x^*) = 0, i = 1, ..., m_i \\
& \nabla \mathcal{L}(x^*, \nu^*, \lambda^*) = 0
\end{aligned}
$$

- Active set method
- Sequential Quadratic Programming
- Primal-dual interior point method

Active set method

Guess the active set $\mathbb{A}$

Solve:

$$\begin{aligned} h_i(x^*) &= 0, \ i \in \mathbb{A} \\ g_i(x^*) &= 0, \ \ \nabla \mathcal{L}(x^*, \nu^*, \lambda^*) = 0 \end{aligned}$$

If $\nu^* \geq 0$ and $h_i(x^*) \leq 0, \ i \in \mathbb{A}^C$ **then**

Solution found

else

Add or remove constraints indices of $\mathbb{A}$

end if

Efficent only for Quadratic programs!

Active set method

Guess the active set $\mathbb{A}$

Solve:

$$\begin{aligned} h_i(x^*) &= 0, \ i \in \mathbb{A} \\ g_i(x^*) &= 0, \ \ \nabla \mathcal{L}(x^*, \nu^*, \lambda^*) = 0 \end{aligned}$$

If $\nu^* \geq 0$ and $h_i(x^*) \leq 0, \ i \in \mathbb{A}^C$ **then**
  Solution found
else
  Add or remove constraints indices of $\mathbb{A}$
end if

Efficent only for Quadratic programs!

Sequential Quadratic Programming

NLP

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) \leq 0. \end{aligned}$$

QP

$$\begin{aligned} \min_{d_k} \quad & \tfrac{1}{2} d_k^T B_k d_k + \nabla f(x_k)^T d_k \\ \text{s.t.} \quad & g(x_k) + \nabla g(x_k)^T d_k = 0 \\ & h(x_k) + \nabla h(x_k)^T d_k \leq 0 \end{aligned}$$

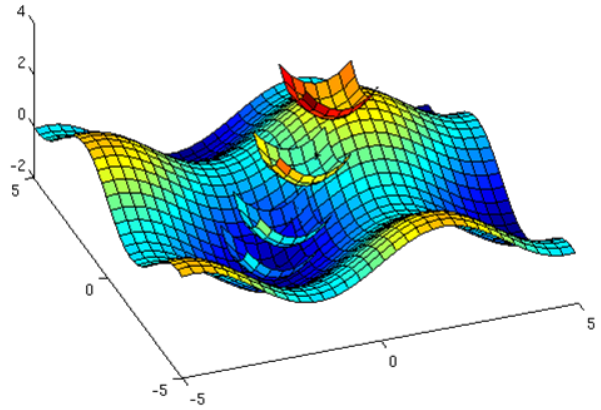
Sequential Quadratic Programming

NLP

$$\begin{array}{lrcl} \min\limits_{x} & f(x) & & \\ \text{s.t.} & g(x) & = & 0, \\ & h(x) & \leq & 0. \end{array}$$

QP

$$\begin{array}{ll} \min\limits_{d_k} & \frac{1}{2} d_k^T B_k d_k + \nabla f(x_k)^T d_k \\ \text{s.t.} & g(x_k) + \nabla g(x_k)^T d_k = 0 \\ & h(x_k) + \nabla h(x_k)^T d_k \leq 0 \end{array}$$



Sequential Quadratic Programming

L_1 merit function:

$T_1(x_k) = f(x_k) + \mu \|g(x_k)\|_1 + \mu \sum_{i=1}^{m_i} |min(0, h(x_k))|$

Line-search SQP algorithm:

while $T_1(x_k) > TOL$ **do**
get $\nabla f(x_k), \nabla g(x_k), \nabla h(x_k), B_k \approx \nabla^2 \mathcal{L}(x_k, \nu_k, \lambda_k)$
solve the QP, get $d_k, \lambda_{QP}, \nu_{QP}$
perform line search on $T_1(x_k + d_k)$, get step lenght t_k
update: $x_{k+1} = x_k + t_k d_k$
update: $\lambda_{k+1} = \lambda_k + t_k \lambda_{QP}, \nu_{k+1} = \nu_k + t_k \nu_{QP}$
end while

Sequential Quadratic Programming

L_1 merit function:

$T_1(x_k) = f(x_k) + \mu \|g(x_k)\|_1 + \mu \sum_{i=1}^{m_i} |min(0, h(x_k))|$

Line-search SQP algorithm:

while $T_1(x_k) > TOL$ **do**
get $\nabla f(x_k), \nabla g(x_k), \nabla h(x_k), B_k \approx \nabla^2 \mathcal{L}(x_k, \nu_k, \lambda_k)$
solve the QP, get $d_k, \lambda_{QP}, \nu_{QP}$
perform line search on $T_1(x_k + d_k)$, get step lenght t_k
update: $x_{k+1} = x_k + t_k d_k$
update: $\lambda_{k+1} = \lambda_k + t_k \lambda_{QP}, \nu_{k+1} = \nu_k + t_k \nu_{QP}$
end while

Primal-dual interior point method

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) \leq 0 \end{aligned}$$

Primal-dual interior point method

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) + s = 0 \\ & s \geq 0 \end{aligned}$$

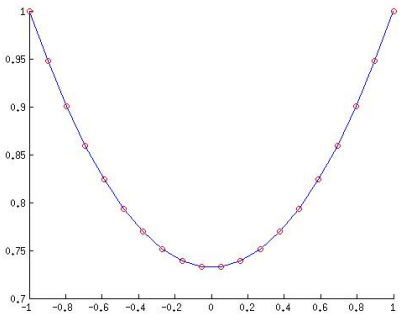
Primal-dual interior point method

$$\begin{aligned} \min_{x,s} \quad & f(x) - \tau \sum_{i=1}^{m_i} \log s_i \\ \text{s.t.} \quad & g(x) = 0, \\ & h(x) + s = 0 \end{aligned}$$

Barrier function $f(x) - \tau \sum_{i=1}^{m_i} \log s_i$ prevents the iterates from leaving the feasible region defined by the inequalities.

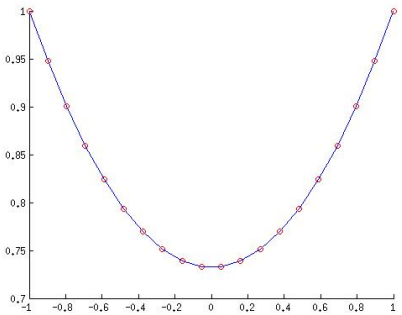
Hanging Chain

Hanging Chain



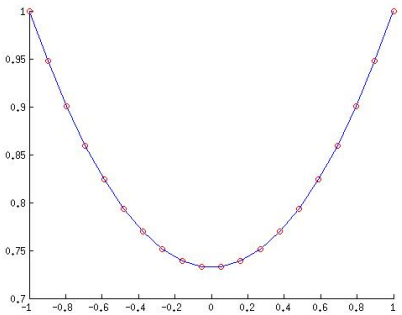
- mass points
  $(x_i, y_i), i = 1, ..., Np$
- potential energy of each spring
  $V_{el}^i = \frac{1}{2} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2)$
- gravitational potential energy of each mass
  $V_g^i = m_i g y_i$
- total potential energy
  $V_{chain}(x, y) =$
  $\frac{1}{2} \sum_{i=1}^{Np-1} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2) + g \sum_{i=1}^{N_p} m_i y_i$
- Equilibrium point? $\rightarrow \min_{x,y} V_{chain}(x, y)$

Hanging Chain



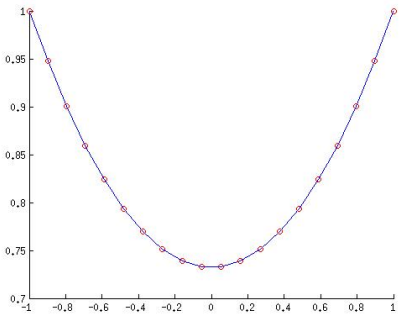
- mass points
  $(x_i, y_i), i = 1, ..., Np$
- potential energy of each spring
  $V_{el}^i = \frac{1}{2} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2)$
- gravitational potential energy of each mass
  $V_g^i = m_i g y_i$
- total potential energy
  $V_{chain}(x, y) =$
  $\frac{1}{2} \sum_{i=1}^{Np-1} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2) + g \sum_{i=1}^{N_p} m_i y_i$
- Equilibrium point? $\rightarrow \min_{x,y} V_{chain}(x, y)$

Hanging Chain



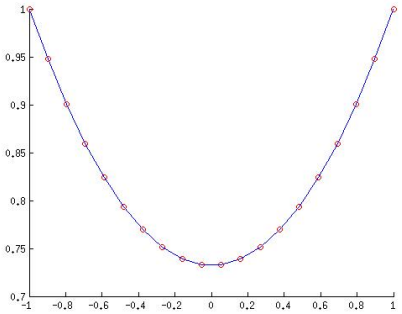
- mass points
  $(x_i, y_i), i = 1, ..., Np$
- potential energy of each spring
  $V_{el}^i = \frac{1}{2} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2)$
- gravitational potential energy of each mass
  $V_g^i = m_i g y_i$
- total potential energy
  $V_{chain}(x, y) =$
  $\frac{1}{2} \sum_{i=1}^{Np-1} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2) + g \sum_{i=1}^{N_p} m_i y_i$
- Equilibrium point? $\rightarrow \boxed{\min_{x,y} V_{chain}(x, y)}$

Hanging Chain



- mass points
  $(x_i, y_i), i = 1, ..., Np$
- potential energy of each spring
  $V_{el}^i = \frac{1}{2} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2)$
- gravitational potential energy of each mass
  $V_g^i = m_i g y_i$
- total potential energy
  $V_{chain}(x, y) =$
  $\frac{1}{2} \sum_{i=1}^{Np-1} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2) + g \sum_{i=1}^{N_p} m_i y_i$
- Equilibrium point? $\rightarrow \min_{x,y} V_{chain}(x, y)$

Hanging Chain



- mass points
  $(x_i, y_i), i = 1, ..., Np$
- potential energy of each spring
  $V_{el}^i = \frac{1}{2} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2)$
- gravitational potential energy of each mass
  $V_g^i = m_i g y_i$
- total potential energy
  $V_{chain}(x, y) =$
  $\frac{1}{2} \sum_{i=1}^{Np-1} D_i((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2) + g \sum_{i=1}^{N_p} m_i y_i$
- Equilibrium point? $\rightarrow$ $\boxed{\min_{x,y} V_{chain}(x, y)}$

$$V_{chain}(x,y) =$$
$$\sum_{i=1}^{Np-1} D_i(\sqrt{((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} - \frac{L}{N_p})^2 + g\sum_{i=1}^{N_p} m_i y_i$$

$$N_p = 20, m_i = 20/N_p kg, D_i = 75 N_p N/m, g = 9.81 m/s^2$$

$$q = (x,y)$$

$$\min_q \sum_{i=1}^{Np-1} D_i(\|q_{i+1} - q_i\| - \frac{L}{N_p})^2 + mg\sum_{i=1}^{N_p} \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T q_i$$

Unconstrained case

$$V_{chain}(x,y) = \sum_{i=1}^{Np-1} D_i(\sqrt{((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} - \frac{L}{N_p})^2 + g\sum_{i=1}^{N_p} m_i y_i$$

$$N_p = 20, m_i = 20/N_p kg, D_i = 75 N_p N/m, g = 9.81 m/s^2$$

$$q = (x,y)$$

$$\min_q \sum_{i=1}^{Np-1} D_i(\|q_{i+1} - q_i\| - \frac{L}{N_p})^2 + mg\sum_{i=1}^{N_p} \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T q_i$$

Unconstrained case

$$V_{chain}(x,y) = \sum_{i=1}^{Np-1} D_i(\sqrt{((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} - \frac{L}{N_p})^2 + g\sum_{i=1}^{N_p} m_i y_i$$

$$N_p = 20, m_i = 20/N_p kg, D_i = 75 N_p N/m, g = 9.81 m/s^2$$

$$q = (x,y)$$

$$\min_q \sum_{i=1}^{Np-1} D_i(\|q_{i+1} - q_i\| - \frac{L}{N_p})^2 + mg\sum_{i=1}^{N_p} \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T q_i$$

Unconstrained case

$$V_{chain}(x,y) = \sum_{i=1}^{Np-1} D_i(\sqrt{((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} - \frac{L}{N_p})^2 + g\sum_{i=1}^{N_p} m_i y_i$$

$$N_p = 20, m_i = 20/N_p kg, D_i = 75 N_p N/m, g = 9.81 m/s^2$$

$$q = (x,y)$$

$$\min_q \sum_{i=1}^{Np-1} D_i(\|q_{i+1} - q_i\| - \frac{L}{N_p})^2 + mg\sum_{i=1}^{N_p} \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T q_i$$

Unconstrained case

Unconstrained case

$$q_{k+1} = q_k - t_k {B_k}^{-1} \nabla f(q_k)$$

$t_k \rightarrow$ backtracking line search using the Armijo condition

$\nabla f(q_k) \rightarrow$ finite differences

- Steepest descent method
  - $B_k \rightarrow$ indentity matrix
  - 1323 iterations
- Quasi-Newton method
  - $B_k \rightarrow$ BFGS formula
  - 42 iterations

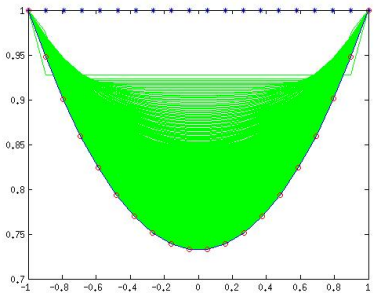
Unconstrained case

$$q_{k+1} = q_k - t_k {B_k}^{-1} \nabla f(q_k)$$

$t_k \rightarrow$ backtracking line search using the Armijo condition

$\nabla f(q_k) \rightarrow$ finite differences

- Steepest descent method
  - $B_k \rightarrow$ indentity matrix
  - 1323 iterations



- Quasi-Newton method
  - $B_k \rightarrow$ BFGS formula
  - 42 iterations

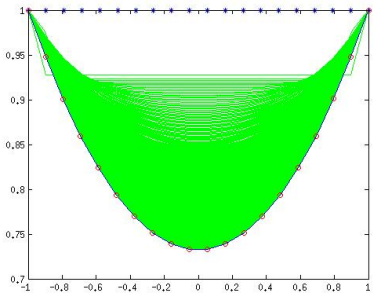
Unconstrained case

$$q_{k+1} = q_k - t_k {B_k}^{-1} \nabla f(q_k)$$

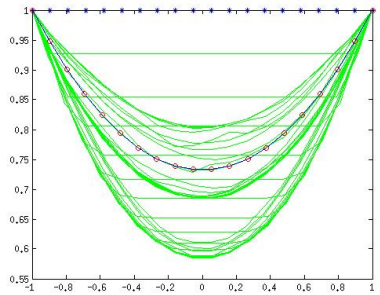
$t_k \rightarrow$ backtracking line search using the Armijo condition

$\nabla f(q_k) \rightarrow$ finite differences

- Steepest descent method
  - $B_k \rightarrow$ indentity matrix
  - 1323 iterations



- Quasi-Newton method
  - $B_k \rightarrow$ BFGS formula
  - 42 iterations

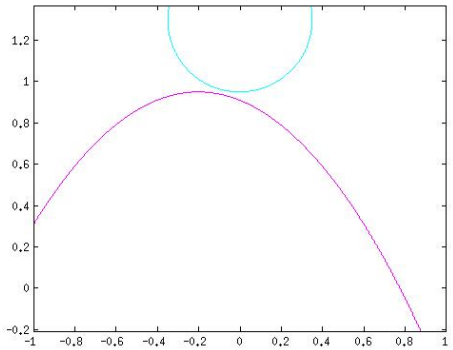


Constrained Optimization

$$\min_q \sum_{i=1}^{Np-1} D_i(\|q_{i+1} - q_i\| - \frac{L}{N_p})^2 + mg \sum_{i=1}^{N_p} \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T q_i$$

$$\text{s.t. } \|q_{(N_p+1)/2} - (0, 1.3)^T\|^2 - 0.35^2 = 0,$$

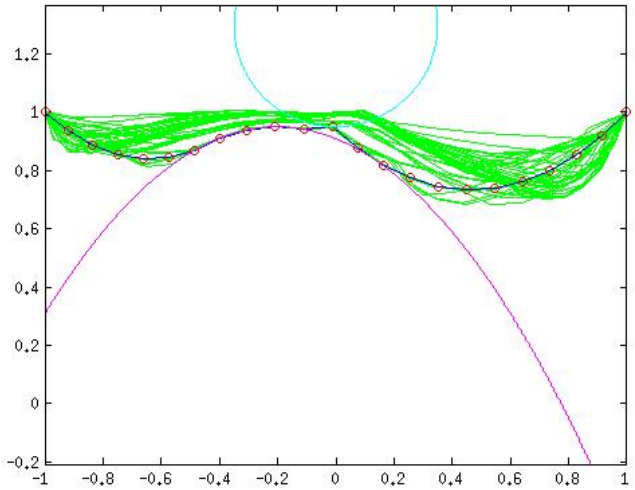
$$0.95 - \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T q_i - (\begin{pmatrix} 1 \\ 0 \end{pmatrix}^T q_i + 0.2)^2 \leq 0, \quad i = 1, ..., N_p.$$



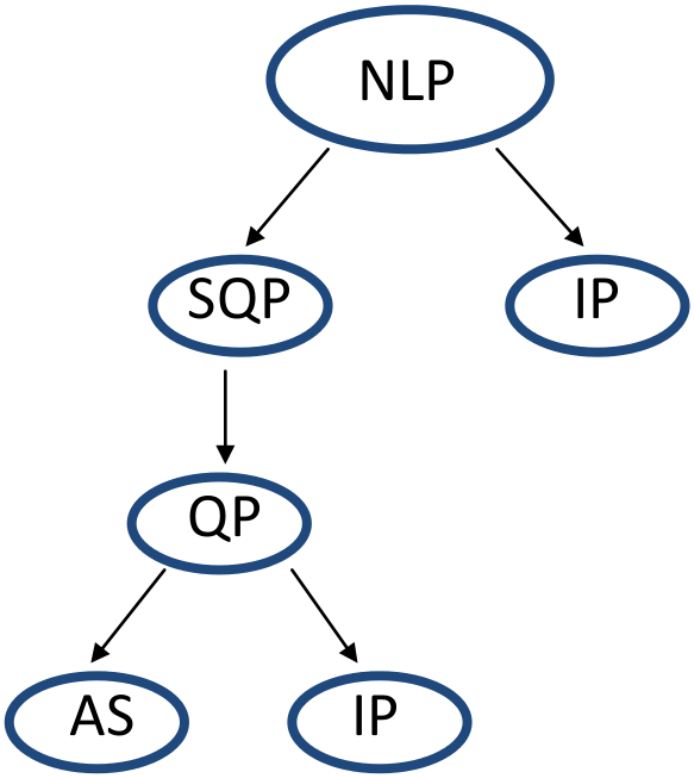
Constrained Optimization

SQP method

- $q_{k+1} = q_k - t_k d_k$
- $t_k \rightarrow$ line search with Armijo's condition using L_1 merit function
- $d_k \rightarrow$ solution of QP problem
- $B_k \rightarrow$ BFGS formula
- 54 iterations

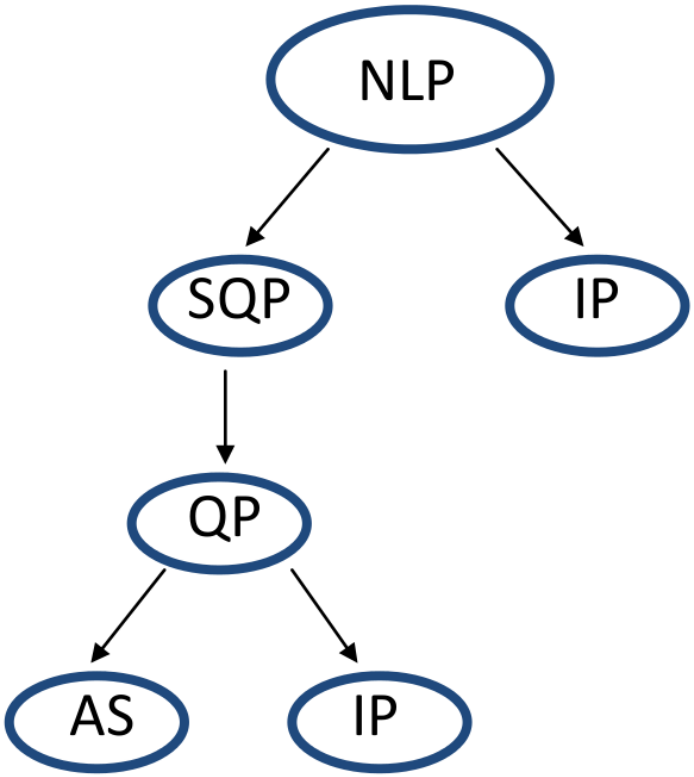


Summary



- Sequential Quadratic Programming(SQP):
  very powerful for parametric optimization problems, warm start, optimal control
- Interior Point methods(IP):
  broadly used for convex and non-convex problems, well suited for large-scale problems

Summary



- Sequential Quadratic Programming(SQP):
  very powerful for parametric optimization problems, warm start, optimal control
- Interior Point methods(IP):
  broadly used for convex and non-convex problems, well suited for large-scale problems

Thank you for your attention!