An (exhaustive?) overview of optimization methods

C. Dutang

Summer 2010
## Contents

1 Minimisation problems ........................................... 3
   1.1 Continuous optimisation, uncountable set $X$ ........................................... 3
      1.1.1 Unconstrained optimisation ........................................... 3
      1.1.2 Constrained optimisation ........................................... 8
      1.1.3 Saddle point ........................................... 11

2 Root problems ........................................... 13
   2.1 General equation, uncountable set $X$ ........................................... 13

3 Fixed-point problems ........................................... 17

4 Variational Inequality and Complementarity problems ........................................... 18
   4.1 Examples and problem reformulation ........................................... 18
      4.1.1 Examples ........................................... 18
      4.1.2 Problem reformulation ........................................... 19
   4.2 Algorithms for CPs ........................................... 20
   4.3 Algorithms for VIPs ........................................... 23

A Bibliography ........................................... 24

B Websites ........................................... 25

1 Minimisation problems

Minimisation problems consist in solving

$$\min_{x \in X \subset \mathbb{R}^n} f(x) \text{ such that } c_j(x) = 0, j \in E \text{ and } c_j(x) \leq 0, j \in I.$$ 

Notations:

- gradient vector $g(x) = \nabla f(x)$,
- Hessian matrix $H(x) = \nabla^2 f(x)$,
- Jacobian matrix of the function $c J = (\frac{\partial c_i}{\partial x_j})_{ij}$,
- positive and negative part of $z$, $z_+ = \max(z, 0)$ and $z_- = -\min(z, 0)$,
- $c(x)^\# = c_i(x)$ if $i \in E$ or $c_i(x)_+$ if $i \in I$,
- the superscript $^T$ denotes the transpose.

1.1 Continuous optimisation, uncountable set $X$

1.1.1 Unconstrained optimisation, $E, I = \emptyset$

1. Quadratic problems $f(x) = \frac{1}{2}x^T M x + b^T x + c$.

Prop: unique solution if matrix $M$ is symmetric positive and definite. Thus solve $M x + b = 0$.

(a) General descent scheme $(x_{k+1} = x_k + t_k d_k$ with stepsize $t_k$ and direction $d_k$ such that $f(x_{k+1}) < f(x_k)$

i. $d_k = -(M x_k + b)$: steepest descent method,

ii. $d_k = -(M x_k + b)$ and $t_k = \frac{(x_k - x_{k-1})^T (x_k - x_{k-1})}{(x_k - x_{k-1})^T M (x_k - x_{k-1})}$: Barzilai-Borwein method,

iii. relaxed descent scheme $(x_{k+1} = x_k + t_k \theta_k d_k$ with relaxation parameters $0 < \theta_k < 2$)

Assuming direction $d_k = -g(x_k)$ and optimal stepsize $t_k = \frac{g(x_k)^T g(x_k)}{g(x_k)^T M g(x_k)}$, the choice of relaxation parameters are

- $\theta_1 = 1$ steepest descent method,
- $\theta_2 = 2$ we get $f(x_{k+1}) = f(x_k)$,
- If the sequence $(\theta_k)_k$ has an accumulation point $\theta^*$ then the relaxed Cauchy method converges.
(b) Conjugate gradient methods \((x_{k+1} = x_k + t_k d_{k+1}\) given the full information at \(k\)th iteration):

i. classic CG method:
- Init: \(x_0, d_1 = -g_1\)
- Iter:
  \[
g_{k+1} = g_k + t_k M d_k \quad \text{with} \quad t_k = -\frac{|g_k|^2}{g_k^T M d_k},
  
d_{k+1} = -g_{k+1} + c_k d_k \quad \text{with} \quad c_k = \frac{|g_k|^2}{g_k^T M d_k}.
\]

NB: all directions \((d_1, \ldots, d_k)\) are conjugate w.r.t. \(M\).

ii. preconditioned conjugate gradient method,
(c) Gauss-Newton method for least square problems \((f(x) = \sum_{j=1}^p f_j^2(x))\)

Iteration is \(x_{k+1} = x_k + d_k\) with
- \(d_k = -G(x_k)^{-1} \nabla f(x)\),
- gradient \(\nabla f(x) = \sum_{j=1}^p f_j(x) \nabla f_j(x)\),
- approximate Hessian \(G(x) = \sum_{j=1}^p \nabla f_j(x) \nabla f_j(x)^T\).

2. Smooth non linear problems \(f \in C^1\)
(a) General descent scheme \((x_{k+1} = x_k + t_k d_k\) with stepsize \(t_k\) and direction \(d_k\) such that \(f(x_{k+1}) < f(x_k))\)

Direction rule:

i. \(d_k = \arg \min_{||d||<\delta} g(x_k)^T d\)
  - \(L^1\) norm, \(d_k\) is the index of the largest component. Gauss Siedel?
  - \(L^1\) norm, \(d_k = -g(x_k): \text{Steepest descent method}\).

Stepsize rule:

i. fixed: method of successive approximation,
ii. optimal \(t_k = \arg \min_{t>0} f(x_k + td_k)\) (useless in practice),
iii. Wolfe’s rule,
iv. Goldstein and Price,
v. Armijo.

Direction+Stepsize rule:

i. \(d_k = -g(x_k)\) and \(t_k = \frac{d_{k-1}^T d_{k-1}}{d_{k-1}^T (g(x_k) - g(x_{k-1}))}\) Barzilai-Borwein method,
ii. \(d_k = -g(x_k)\) and \(t_k = \arg \min_{t>0} f(x_k + td_k)\) Cauchy method,

(b) Conjugate gradient methods:
i. classic CG
   - Init: \( d_1 = -g(x_1) \)
   - Iter: \( d_k = -g(x_k) + \beta_k d_{k-1} \) for \( k > 1 \)
     
     \[ \beta_k = \frac{g(x_k)^T g(x_k)}{g(x_{k-1})^T g(x_{k-1})} : \text{Fletcher-Reeves update,} \]
     
     \[ \beta_k = \frac{g(x_k)^T (g(x_k) - g(x_{k-1}))}{g(x_{k-1})^T g(x_{k-1})} : \text{Polak-Ribi`ere update,} \]
     
   - Beale-Sorenson,
   - Hestenes-Stiefel,
   - Conjugate-Descent,
   - . . .

NB: all directions \((d_1, \ldots, d_k)\) are conjugate to the Hessian matrix \( H(x_k) \)?

ii. preconditionned CG

(c) Newton method \( x_{k+1} = x_k + d_k \) with direction \( d_k \) minimizes the quadratic function \( q_f(d) = f(x_k) + g(x_k)^T d + \frac{1}{2} d^T \nabla g(x_k) d \) .

i. exact Newton method, \( d_k \) solves \( g(x_k) + \nabla g(x_k) d = 0 \) i.e. minimizer of the local quadratic approximation \( q_f \).

ii. Quasi-Newton methods, \( d_k \) approximates the exact minimizer of \( q_f \)

   Scheme:
   - Init: \( x_0, W_0 \)
   - Iter: while \( |g(x_k)| > \epsilon \)
     
     approximate Hessian inverse \( W_k = W_{k-1} + B_k \)
     
     compute direction \( d_k = -W_k g(x_k) \)
     
     line search for \( t_k \)

   Choice of \( W \):
   - Constraints: symmetric, positive, definite and verified the quasi-Newton equation \( W_k (g_{k+1} - g_k) = x_{k+1} - x_k \),
   - known methods for \( W \): Davidon-Fletcher-Powell (DFP) or Broyden-Fletcher-Goldfarb-Shanno (BFGS).

iii. inexact Newton or truncated Newton methods:

   It requires that \( d_k \) decreases the linear residual, \( \|H(x_k) d_k + g(x_k)\| \leq \eta_c |g(x_k)| \), where \( 0 < \eta_c < 1 \) is the forcing term.

NB: univariate case \((n = 1)\), quasi-Newton has a unique equation: secant method or regula falsi method.

(d) Trust region algorithms \( x_{k+1} = x_k + h_k \) with \( h_k \) a local minimizer

   Scheme:
   - \( h_k(\Delta_k) = \arg \min_{||h|| < \Delta_k} f(x_k) + g(x_k)^T h + \frac{1}{2} h^T \tilde{H}_k h \) with \( \tilde{H}_k \) is positive definite matrix,
   - if \( f(x_k + h_k(\Delta_k)) \leq f(x_k) - m h_k(\Delta_k) \) then terminates iteration,
   - otherwise decrease \( \Delta_k \).

NB: \( 0 < m < 1 \) a fixed coefficient.

Choice of matrix \( \tilde{H}_k \):
- $\tilde{H}_k$ is the Hessian $H(x_k)$: positive-definiteness is not guaranteed,
- $\tilde{H}_k = H(x_k) + \lambda I_n$ with $\lambda > 0$: Levenberg-Marquardt algorithm (derived with KKT conditions),
- quasi-Newton approximation,
- Gauss-Newton approximation
NB: $\tilde{H}_k = 0$ gives the steepest descent.

3. Non smooth problems

(a) direct search (derivative free) methods
   i. Nelder-Mead algorithm
   ii. Hooke-Jeeves algorithm
   iii. multi-directional algorithm
   iv. ... 

(b) metaheuristics
   i. evolutionary algorithms
   ii. stochastic algorithms
      - simulated annealing
      - Monte-Carlo method
      - ant colony

(c) regularizing techniques
   i. filter algorithms
   ii. noisy algorithms

(d) NSO methods
   i. subgradient methods
   ii. bundle methods
   iii. gradient sampling methods
   iv. hybrid methods

(e) special problems
   i. piecewise linear
   ii. minmax
   iii. partially separable
### 1.1 Continuous Optimisation, Uncountable Set $\mathbb{X}$

<table>
<thead>
<tr>
<th>Function</th>
<th>Method type</th>
<th>Algorithms type</th>
<th>Method</th>
<th>R function - package</th>
</tr>
</thead>
<tbody>
<tr>
<td>quadratic</td>
<td>descent scheme</td>
<td>steepest descent</td>
<td>Barzilai-Borwein Cauchy</td>
<td>dfsane BB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>relaxed descent scheme</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gauss-Newton method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>conjugate gradient (CG)</td>
<td>CG methods preconditionned CG</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td>smooth</td>
<td>descent scheme</td>
<td>steepest descent (BB, Cauchy)</td>
<td>Gauss-Siedel</td>
<td>dfsane BB</td>
</tr>
<tr>
<td></td>
<td>conjugate gradient</td>
<td>CG methods preconditionned CG</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td>Newton methods</td>
<td>exact</td>
<td>Gauss-Siedel</td>
<td>nlm stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quasi-Newton</td>
<td>Gauss-Siedel</td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>truncated Newton</td>
<td>Gauss-Siedel</td>
<td>trust trust</td>
</tr>
<tr>
<td></td>
<td>trust-region</td>
<td>direct Hessian</td>
<td>trust-region</td>
<td>trust trust</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Levenberg-Marquardt</td>
<td>trust-region</td>
<td>trust trust</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quasi-Newton</td>
<td>trust-region</td>
<td>trust trust</td>
</tr>
<tr>
<td>non smooth</td>
<td>direct search methods</td>
<td>Nelder-Mead algorithm</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>multidirectional algorithm</td>
<td></td>
<td>genoud rgenoud, mco cmaes</td>
</tr>
<tr>
<td></td>
<td>metaheuristics</td>
<td>evolutionary algorithms</td>
<td>genetic algorithm</td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>covariance matrix adaptation evo. strat.</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stochastic algorithms</td>
<td>simulated annealing</td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monte-Carlo method</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ant colony</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td>regularizing techniques</td>
<td>filter algorithms</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>noisy algorithms</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td>NSO methods</td>
<td>subgradient methods</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bundle methods</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gradient sampling methods</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td>large scale</td>
<td>limited-memory algorithms</td>
<td>L-BFGS</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td>Hessian free methods</td>
<td>conjugate gradient (CG)</td>
<td></td>
<td>optim stats</td>
</tr>
<tr>
<td></td>
<td></td>
<td>preconditionned CG</td>
<td></td>
<td>optim stats</td>
</tr>
</tbody>
</table>
1.1.2 Constrained optimisation \((E, I \neq \emptyset)\), also known as nonlinear programming

The Lagrangian function is defined as
\[
L(x, \lambda) = f(x) + \lambda^T c(x),
\]
with \(\lambda\) the Lagrange multiplier. The Karush-Kuhn-Tucker (KKT) conditions\(^*\) are
- \(\nabla_x L(x, \lambda) = 0\),
- equality constraints \(\forall j \in E, c_j(x) = 0\) and \(\lambda_j \in \mathbb{R}\),
- active inequality constraints \(j \in I, c_j(x) = 0\) and \(\lambda_j > 0\),
- inactive inequality constraints \(j \in I, c_j(x) < 0\) and \(\lambda_j = 0\).

1. box constraints
   (a) projected Newton method,
   (b) limited-memory box-constraint BFGS method: L-BFGS-B,
2. linear constraints
   (a) linear programming \((f \text{ is linear})\)
      i. simplex method,
      ii. dual simplex method,
      iii. Karmarkar algorithm,
   (b) Interior point methods for linear constraints
      Consider the problem \(\min_x f(x)\) such that \(Ax = b\) and \(x \geq 0\). Let a log potential be \(\pi(x) = -\sum_i \log(x_i)\). We want to minimize the penalised function \(f(x) + \mu \pi(x)\). A central path or path following algorithm is a sequence of points \((x_{\mu_n}^*, \lambda_{\mu_n}^*, s_{\mu_n}^*)\) solution of the problem

\[
\begin{aligned}
  x.s = \mu_n \mathbb{I} \\
  \nabla f(x) + A^T \lambda = s & \quad \text{such that } x \geq 0, s \geq 0, \\
  Ax = b & \quad \text{for a sequence of decreasing } (\mu_n)_n \text{ to } 0.
\end{aligned}
\]

Main algorithms are
i. potential reduction algorithm,
ii. primal-dual symmetric algorithm,

\(^*\). first order optimality conditions
iii. generic predictor-corrector algorithms or adaptive path-following algorithm,

Generic predictor-corrector algorithms solve a linear complementarity reformulation of this problem with an iterative method:
- a small-neighborhood algorithm,
- a predictor-corrector algorithm with modified field,
- a large-neighborhood algorithm.

3. general constraints

(a) Sequential linear programming
linearize both objective and constraint functions using Taylor series expansion.

(b) Sequential quadratic programming

i. Local methods for equality constraints
KKT conditions reduce to \( \nabla f(x) + J_c(x)^T \lambda = 0 \) and \( c(x) = 0 \):
- Newton method is an iterative method
\[
\begin{align*}
x_{k+1} &= x_k + d_k \\
\lambda_{k+1} &= \lambda_k + \mu_k
\end{align*}
\]
computed from \( \begin{pmatrix} \nabla^2_{xx} L(x_k, \lambda_k) & J_c(x_k)^T \\ J_c(x_k) & 0 \end{pmatrix} \begin{pmatrix} d_k \\ \lambda_k + \mu_k \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ c(x_k) \end{pmatrix} \)
- full Hessian approximation: replace \( \nabla^2_{xx} L \) with an approximated Hessian \( B_k \) (possible scheme PSB, BFGS),
- augmented Lagrangian (add a constraint-penalty term) to guarantee positiveness,
- reduced Hessian matrix: positiveness is only guaranteed on a subspace,

ii. Local methods for (in)equality constraints
The SQP algorithm consists in solving a sequence of quadratic (Taylor) approximations of the Lagrangian.

Scheme:
- Init: \( x_0, \lambda_0 \)
- Iterate while a termination criterion
\[
x_{k+1} = x_k + d_k
\]
solution of \( \min_d \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2_{xx} L(x_k, \lambda_k) d \)
such that \( c_E(x_k) + J_{c_E}(x_k) d = 0 \) and \( c_I(x_k) + J_{c_I}(x_k) d \leq 0 \)

Implementations:
- active-set strategies,
- interior-point algorithms,
- dual approaches.

iii. Globalization of SQP method

A. trust region method:
Iterative method of a sequence of quadratic bounded sub-problem \( \min_{||d|| < \Delta_k} \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \), where the trust region radius \( \Delta_k \) is updated at each stage.
B. line search method:

Iterative method $x_{k+1} = x_k + t_k d_k$, where direction $d_k$ is approximated by a solution of quadratic sub-problem and $t_k$ chosen to ensure a reasonable decrease of a merit function (e.g. $f(x) + \sigma ||c(x)||_p$). NB: the penalty parameter has to be updated ($\sigma_k$).

NB: for both those techniques, one particular approximation of the Hessian has to be chosen: quasi-Newton, reduced quasi-Newton, . . .

(c) Sequential unconstrained methods

i. Exterior point methods

The general idea is to minimize the merit function $f(x) + p(x)$ where $p$ is a. During the optimization process, an extertior-point method does not guarantee that all points are in the feasible region (hence the name).

Possible penalty terms are:
- quadratic penalty: $p(x) = \sigma/2 ||c(x)||^2$,
- Lagrangian: $p(x) = \mu^T c(x)$,
- augmented Lagrangian: $p(x) = \mu^T c(x) + \sigma/2 ||\tilde{c}(x)||_2$ with $\tilde{c}(x)_j = c_j(x)$ if $j \in E$ and $\max(-\mu/j, c_j(x))$ if $j \in I$,
- non differentiable (exact) function: $p(x) = \sigma ||c(x)||_p$

Then we can either minimize of the associated Lagrangian function is carried out by an iterative method, or solve the dual problem $\arg \max_{u \geq 0} \theta(u)$ with $\theta(u) = \min_x f(x) + up(x)$.

ii. Interior point methods (or adaptive barrier methods):

Barrier methods operates in the interior of the feasible region: adapting iteratively the barrier permits the convergence to a boundary by lowering the strength of the corresponding barrier.

Let a log potential be $\pi(x) = - \sum_i \log(x_i)$. Using the convex property on $f(x) + \mu \pi(x)$, two majorants of $f(x)$ can be found
- log surrogate function: $g_l(x, y, \mu) = f(x) - \mu \sum_{j \in I} c_j(y) \log c_j(x) + \mu \sum_{j \in E} c_j'(y)(x-y)$,
- power surrogate function: $g_p(x, y, \mu) = f(x) + \mu \sum_{j \in I} c_j(y)^{\alpha} \log c_j(x)^{-\alpha} + \mu \sum_{j \in I} c_j(y)^{\beta} c_j'(y)(x-y)$

Then a MM algorithm can be used with an adaptive barrier constant $\mu_k$:
- Init: $x_0, \mu_0$,
- Iterate while a termination criterion
  $x_{k+1} = \arg \min_x g_l(x, x_k, \mu_k)$ or $\arg \min_x g_p(x, x_k, \mu_k)$,
  adapt $\mu_{k+1}$.
## 1.1 Continuous optimisation, uncountable set $\mathcal{X}$

<table>
<thead>
<tr>
<th>Constraint type</th>
<th>Method type</th>
<th>Algorithm type</th>
<th>Algorithms</th>
<th>R function - package</th>
</tr>
</thead>
<tbody>
<tr>
<td>box constraints</td>
<td>projected Newton method</td>
<td></td>
<td></td>
<td>optim stats, Rvmmin</td>
</tr>
<tr>
<td></td>
<td>L-BFGS-B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linear constraints</td>
<td>simplex method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dual simplex method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Karmarkar algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>potential reduction algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>primal-dual symmetric method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>adaptive path-following algo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>small-neighborhood algo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>predictor-corrector algo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>large-neighborhood algo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>general constraints</td>
<td>sequential linear program</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sequential quadratic program</td>
<td>trust-region SQP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>line-search SQP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unconstrained program</td>
<td>exterior-point algorithms</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>interior-point algorithms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 1.1.3 Min-max saddle point

A saddle point $x^*$ satisfies the following inequality

$$\forall u, v, \phi(u^*, v) \leq \phi(u^*, v^*) \leq \phi(u, v^*),$$

with $x = (u^T v^T)^T$. Grantham (2005) provide an unified way of most algorithms solving saddle points. If we see the iterative algorithm as a function of time, then iterative algorithm can be defined as the solution of partial differential equation (PDE). Let $g$ be the gradient of $\phi$: $g(x) = \frac{\partial \phi}{\partial x} = \begin{pmatrix} g_u(x) \\ g_v(x) \end{pmatrix}$. Then a general algorithm solve the PDE

$$\frac{\partial x}{\partial t} = -P(x)g(x),$$
where \( P(x) \) denotes a matrix. Let \( H(x) \) be the Hessian matrix

\[
H(x) = \begin{pmatrix}
H_{uu} & H_{u}v \\
H_{v}^T & H_{vv}
\end{pmatrix}(x).
\]

We have the following algorithms

1. steepest descent \( P(x) = \text{diag}(I_u, -I_v) \),
2. Newton method \( P(x) = -H(x)^{-1} \),
3. gradient enhanced min-max

\[
P(x) = \begin{pmatrix}
H_{uu} + \alpha_u I_u & H_{uv} \\
H_{v}^T & H_{vv} - \alpha_v I_v
\end{pmatrix}(x)^{-1}.
\]
2 Root problems

Root problems consist in solving

\[ f(x) = 0, x \in X \subset \mathbb{R}^n. \]

2.1 General equation, uncountable set \( X \)

1. \( f \) linear: \( Ax = b \)
   
   The solution is unique if \( A \) is invertible, i.e. \( \det(A) \neq 0. \)
   
   (a) direct inversion
   
   Compute \( A^{-1} \), then \( x = A^{-1}b. \)
   
   (b) Gaussian elimination
   
   If \( A \) is not upper triangular, transform \( A \) into a upper triangular matrix, and then compute ascendently the solution (if it exists). Note that it corresponds to a factorization \( PLU \) where \( L \) is a lower triangular matrix, \( U \) upper triangular and \( P \) permutation matrix.

   i. Gauss pivot method
   
   ii. Gauss Jordan method
   
   iii. Grassman algorithm

   (c) Decomposition
   
   Decompose \( A \) into three matrix \( D - E - F \), with \( D \) the diagonal part, \( E \) the opposite lower part and \( F \) the opposite upper part.

   i. Jacobi iterations: \( x_{k+1} = D^{-1}(E + F)x_k + D^{-1}b, \)
   
   ii. Gauss-Siedel iterations: \( x_{k+1} = (D - E)^{-1}Fx_k + (D - E)^{-1}b, \)
   
   iii. Successive Over Relaxation: \( (D - \omega E)x_{k+1} = (\omega F + (1- \omega)D)x_k + \omega b \) for \( \omega \in [0, 1] \).

   (d) projection methods for large scale problems

2. \( f \) univariate

   (a) Newton methods

   i. Newton-Raphson method
   
   Assuming, we have the gradient \( f' \), the algorithm is
   
   - Init: \( x_0 \)
   
   - Iter: \( x_{k+1} \) root of \( f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0. \)
ii. the secant method
it consists in replacing $f'(x_k)$ by a finite-difference approximation $\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$ in the Newton-Raphson method.

iii. the Muller method
It consists in approximating the function by the quadratic function
\[ q(x) = f(x_k) + (x - x_k)f[x_k, x_{k-1}] + (x - x_k)(x - x_{k-1})f[x_k, x_{k-1}, x_{k-2}], \]
where $f[.,.]$ denotes divided differences. Of course, analytical solution exists to find the next iterate $x_{k+1}$.

(b) dichotomic search

i. bisection method
Assuming $f$ is continuous, the procedure is
- Init: $a_0$ and $b_0$ such that $f(a_0)f(b_0) < 0$,
- Iter: let $c = \frac{a_k+b_k}{2}$.
  If $f(c)f(a_k) > 0$ then $a_{k+1} = c$, $b_{k+1} = b_k$,
  otherwise $b_{k+1} = c$, $a_{k+1} = a_k$.

ii. regula falsi or false position method
a hybrid combining dichotomic search and the secant method. Replace the $c$ of the bissection method (middle of $[a_k, b_k]$) by $c_k$ root of $f(b_k) + \frac{f(b_k)-f(a_k)}{b_k-a_k}(c_k - b_k) = 0$.

3. $f$ polynomial $p$

(a) Bairstow method:
Consider a polynomial with real coefficients. It consists in dividing successively the polynomial by a quadratic polynomial. Thus we get pairs of conjugate zeros.

(b) Bernoulli method:
Iterative method that use the characteristic polynomial to compute zero one after another, and deflate the polynomial at each stage.

(c) Muller method:
Use a quadratic approximation of the polynomial to find zeros.

(d) Newton method:
Iterates the Newton method on the polynomial. If combines with Muller method, it can be powerful to find zeros successively.

(e) Laguerre method:
It use the derivatives of log $p$, used in an iterative method.

(f) Durand-Kerner or Weierstrass method:
Use a fixed-point iteration procedure to compute all roots at a time.

4. $f$ smooth
(a) Newton-Raphson method

Assuming, we have the gradient $\nabla f$, the algorithm is

- Init: $x_0$
- Iter: $x_{k+1}$ root of $f(x_k) + \nabla f(x_k)(x_{k+1} - x_k) = 0$.

(b) quasi-Newton methods

Approximate the gradient by $G_k$ with an update scheme:

- Init: $x_0$
- Iter: $x_{k+1}$ root of $f(x_k) + G_k(x_{k+1} - x_k) = 0$.

Update schemes

- Broyden method: $G_k = G_{k-1} + \frac{(y_{k-1} - G_{k-1} s_{k-1}) s_{k-1}^T}{s_{k-1}^T s_{k-1}}$ with $y_{k-1} = f(x_k) - f(x_{k-1})$ and $s_k = x_k - x_{k-1}$.
- DFP or BFGS direct approximation of $G_k^{-1}$. 
<table>
<thead>
<tr>
<th>Function</th>
<th>Methods</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>direct inversion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gaussian elimination</td>
<td>Gauss pivot method</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gauss-Jordan method</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Grassman method</td>
</tr>
<tr>
<td></td>
<td>decomposition</td>
<td>Jacobi iterations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gauss-Siedel iterations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Successive over relaxations</td>
</tr>
<tr>
<td></td>
<td>projection methods</td>
<td></td>
</tr>
<tr>
<td>univariate</td>
<td>Newton method</td>
<td>Newton-Raphson method</td>
</tr>
<tr>
<td></td>
<td></td>
<td>secant method</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Muller method</td>
</tr>
<tr>
<td></td>
<td>dichotomic search</td>
<td>bissection method</td>
</tr>
<tr>
<td></td>
<td></td>
<td>regula falsi</td>
</tr>
<tr>
<td>polynomial</td>
<td>Bairstow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bernoulli</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Muller</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Laguerre</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weierstrass</td>
<td></td>
</tr>
<tr>
<td>smooth</td>
<td>Newton-Raphson method</td>
<td></td>
</tr>
<tr>
<td></td>
<td>quasi-Newton</td>
<td>Broyden</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BFGS</td>
</tr>
</tbody>
</table>
3 Fixed-point problems

Root problems consist in solving
\[ F(x) = 0, x \in \mathbb{X} \subset \mathbb{R}^n. \]

1. Direct method: \( x_{k+1} = F(x_k) \),
2. Polynomials methods: \( x_{k+1} = \sum_{i=0}^{d} \gamma_{i,k} F^i(x_k) \) with \( F^i \) the \( i \)th composition of \( F \).
   Let \( u_k^i \) be \( F^i(x_k) \), one iterate is \( x_k \rightarrow u_k^0, \ldots, u_k^d \rightarrow x_{k+1} \). The sequence \( u_k^i \)'s is called a cycle or a restart.
   (a) 1st order method: \( x_{k+1} \) can be rewritten as \( x_k - \alpha_k r_k \) with \( r_k = F(x_k) - x_k \),
      i. relaxation method: \( \alpha_k \) independent of \( x_k \) such as \( \frac{1}{k+1} \) or a random uniform number in \( ]0,2[ \),
      ii. Lemaréchal method or RRE1: \( \alpha_k = \frac{\langle v_k, r_k \rangle}{\langle v_k, v_k \rangle} \) where \( v_k = F(F(x_k)) - 2F(x_k) + x_k \),
      iii. Brezinski method or MPE1: \( \alpha_k = \frac{\langle r_k, r_k \rangle}{\langle r_k, v_k \rangle} \),
   (b) \( d \)th order method:
      The coefficients \( \gamma_{i,k} \) must satisfy the constraints
      - \( \sum_{i=0}^{d} \gamma_{i,k} = 1 \),
      - \( \sum_{i=0}^{d} \gamma_{i,k} \beta_{i,j,k} = 0 \).
   i. Reduced Rank Extrapolation (RREd): \( \beta_{i,j,k} = \langle \Delta_{i,1} x_k, \Delta_{j,2} x_k \rangle \),
      ii. Minimal Polynomial Extrapolation (MPEd): \( \beta_{i,j,k} = \langle \Delta_{i,1} x_k, \Delta_{j,1} x_k \rangle \),
   with \( \Delta_{i,j} x_k = \sum_{l=1}^{j} (-1)^{l-j} C_{l-j}^i F^l(x_k) \) and \( C_{l-j}^i \) the binomial coefficients.
   Squaring methods consist in applying twice a cycle step to get the next iterate. So we have \( x_{k+1} = \sum_{i=0}^{d} \sum_{j=0}^{d} \gamma_{i,k} \gamma_{j,k} F^{i+j}(x_k) \).
   (a) squaring 1st order method: \( x_{k+1} \) can be rewritten as \( x_k - 2\alpha_k r_k + \alpha_k^2 v_k \),
      i. SqRRE1: \( \alpha_k = \frac{\langle v_k, r_k \rangle}{\langle v_k, v_k \rangle} \),
      ii. SqMPE1: \( \alpha_k = \frac{\langle r_k, r_k \rangle}{\langle r_k, v_k \rangle} \),
   (b) squaring \( d \)th order method:
      i. SqRREd: \( \beta_{i,j,k} = \langle \Delta_{i,1} x_k, \Delta_{j,2} x_k \rangle \),
      ii. SqMPEd: \( \beta_{i,j,k} = \langle \Delta_{i,1} x_k, \Delta_{j,1} x_k \rangle \),
   NB: the RRE1 method is (sometimes) called the Richardson method when \( F \) is linear and Lemaréchal method otherwise. The SqRRE1 method is called the Cauchy Barzilai Borwein method when \( F \) is linear. The MPE1 method is called the Cauchy method when \( F \) is linear and Brezinski method otherwise.
3. Epsilon algorithms:
   (a) Scalar \( \epsilon \) algorithm SEA
   (b) Vector \( \epsilon \) algorithm VEA
   (c) Topological \( \epsilon \) algorithm TEA
### 4 Variational Inequality and Complementarity problems

Variational inequality problems $\text{VI}(K, F)$ consist in finding $x \in K$ such that
\[\forall y \in K, (y - x)^TF(x) \geq 0,\]
where $F : K \mapsto \mathbb{R}^n$. We talk about quasi-variational inequality problems for the problem
\[\forall y \in K, (y - x)^TF(x) \geq 0.\]

Complementarity problems $\text{CP}(K, F)$ consist in finding $x \in K$ such that
\[x \in K, F(x) \in K^*, x^TF(x) = 0\]
where $K^*$ denotes the dual cone of $K$, i.e. $K^* = \{d \in \mathbb{R}^n, \forall k \in K, k^Td = 0\}$. Let us note $x^Ty = 0$ is equivalent to $x$ is orthogonal to $y$, usually noted by $x \perp y$. Furthermore if $K$ is a cone, then $\text{CP}(K, F)$ is equivalent to $\text{VI}(K, F)$.

### 4.1 Examples and problem reformulation

#### 4.1.1 Examples

Here are few examples of VI problems.
- classic complementary problems: when $K = \mathbb{R}^n_+$, $\text{CP}(K, F)$ reduces to $x \geq 0 \perp F(x) \geq 0$, i.e. $\forall i, x_i \geq 0, F(x_i) \geq 0, x_iF_i(x) = 0^*$.  
- mixed complementary problems:  
  if $K = \mathbb{R}^m \times \mathbb{R}_+^{n-m}$ and $F(u, v) = (G(u, v)^T, H(u, v)^T)^T$, then $G(u, v) = 0, v \geq 0 \perp H(u, v) \geq 0$.  
- linear variational inequality problems:  
  $F(x) = q + Mx$ and $K$ be a polyhedral set, a closed rectangle or the positive orthant.  
- link with optimization problem:  
  if we consider $\min_{x \in K} \theta(x)$ with $K$ convex, then local minimizer must satisfy $\forall y \in K, (y - x)^T\nabla \theta(x) \geq 0$, i.e. $\text{VI}(K, \nabla \theta)$. If $\theta(x) = q^Tx + \frac{1}{2}x^TMx$, then the VIP and the optimization are equivalent if $M$ is symmetric and $K$ is polyhedron.
- extended KKT system:  
  Let $K$ be $\{x \in \mathbb{R}^n, h(x) = 0, g(x) \leq 0\}$ for $h : \mathbb{R}^n \mapsto \mathbb{R}^l$ and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$. If $x$ solves $\text{VI}(K, F)$ then there exist $\mu \in \mathbb{R}^l, \lambda \in \mathbb{R}^m$ such that

---

* Each composant of $x$ and $F(x)$ are complement.
4.1 Examples and problem reformulation

- \( L(x, \mu, \lambda) = F(x) + \sum_j \mu_j \nabla h_j(x) + \sum_i \lambda_i \nabla g_i(x) = 0, \)
- \( h(x) = 0, \)
- \( \lambda \geq 0 \perp g(x) \leq 0. \)

4.1.2 Problem reformulation

A necessary tool for VIP and CP is complementarity function. We say \( \psi : \mathbb{R}^2 \rightarrow \mathbb{R} \) is a complementarity function if \( \forall a, b \in \mathbb{R}^2, \psi(a, b) = 0 \iff a \geq 0, b \geq 0, ab = 0. \) Here are some examples \( \psi_{\min}(x, y) = \min(x, y), \psi_{FB}(x, y) = \sqrt{x^2 + y^2} - (x + y). \)

A class of compl. function is the Mangasarian functions: \( \psi_{\text{Man}}(a, b) = \varphi(|a - b|) - \varphi(a) - \varphi(b) \) where \( \varphi \) is a strictly increasing function from \( \mathbb{R} \) to \( \mathbb{R} \). Typically, we use \( \varphi(t) = t \) and \( \varphi(t) = t^3. \)

Using this tool, we can reformulate the above extended KKT system as

\[
\begin{pmatrix}
L(x, \mu, \lambda) \\
h(x) \\
\psi_{\min}(\lambda, -g(x))
\end{pmatrix} = 0.
\]

Another useful tool is the Euclidean projector on \( K \), which for a point \( x \) find nearest on \( K \) according to the Euclidean distance. That is to say

\[
\Pi_K : x \mapsto \arg \min_{y \in K} \frac{1}{2} (y - x)^T (y - x).
\]

There are two possibilities to reformulate VI(\( K,F \)) problems:
- natural mapping: \( x \) solves VI(\( K,F \)) \( \iff \) \( F_{\text{nat}}^K(x) = 0 \) with \( F_{\text{nat}}^K(x) = x - \Pi_K(x - F(x)), \)
- normal mapping: \( x \) solves VI(\( K,F \)) \( \iff \exists z \in \mathbb{R}^n, x = \Pi_K(z), F_{\text{nor}}^K(z) = 0 \) with \( F_{\text{nor}}^K(z) = F(\Pi_K(z)) + z - \Pi_K(z). \)

Example for CP(\( \mathbb{R}^n_+ \), \( F \)), we have \( F_{\text{nor}}^K(z) = F(z_+) - z_-. \)

The last tool we introduce is the merit functions. A merit function for VI(\( K,F \)) is a function \( \theta : X \subset K \mapsto \mathbb{R} \) such that \( x \) solves VI(\( K,F \)) \( \iff \) \( x \in X, \theta(x) = 0 \iff x = \arg \min_{y \in X} \theta(y) \) for a closed set \( X \) and \( \min_{y \in X} \theta(y) = 0. \)

Examples:
- for VI(\( \mathbb{R}^n_+ \), \( F \)) we have \( \theta_{\psi}(x) = \sum_i \psi(x_i, F_i(x))^2 \) with \( \psi \) a compl. function,
- for VI(\( K,F \)), we have \( \theta_{\text{gap}}(x) = \sup_{y \in K} F(x)^T (x - y) \). If \( K \) is a cone, then \( \theta_{\text{gap}}(x) \) becomes \( x^T F(x) \).
4.2 Algorithms for CPs

Problem type:
1. non linear CPs: CP\((\mathbb{R}^n, F)\):

Complementarity functions

(a) FB based methods:

The equation is \(F_{FB}(x) = 0\) and the merit function is \(\theta_{FB}(x) = F_{FB}(x)^T F_{FB}(x)\) where

\[
F_{FB}(x) = \begin{pmatrix}
\psi_{FB}(x_1, F_1(x)) \\
\vdots \\
\psi_{FB}(x_n, F_n(x))
\end{pmatrix}.
\]

Tools:

- for linear Newton approximation scheme \(T\), we choose a matrix \(H\) in \(JacF_{FB}\),
- Set the set \(B\) to \(\{i \in \{1, \ldots, n\}, x_i = 0 = F_i(x)\}\),
- Choose \(z \in \mathbb{R}^n\) such that \(z_i \neq 0\) for \(i \in B\),
- For all columns \(c\) of \(H^T\),

\[
(H^T)_c = \begin{cases}
\left(\frac{x_i}{\sqrt{x_i^2 + F_i(x)^2}} - 1\right) e_i + \left(\frac{F_i(x)}{\sqrt{x_i^2 + F_i(x)^2}} - 1\right) \nabla F_i(x) & \text{if } c \notin B \\
\left(\frac{z_i}{\sqrt{z_i^2 + (\nabla F_i(x))^T z}} - 1\right) e_i + \left(\frac{\nabla F_i(x)^T z}{\sqrt{z_i^2 + (\nabla F_i(x))^T z}} - 1\right) \nabla F_i(x) & \text{if } c \in B
\end{cases}
\]

Where \(e_i\) is the vector with 1 at the \(i\)th position.

- linear CP\((K, x \mapsto q + Mx)\)

Algorithms

i. line-search methods:

Algo 9.1.20\((F_{FB}, \theta_{FB}, T)\)

- Init: \(x_0 \in \mathbb{R}^n\), \(\rho > 0, p > 1\) and \(\gamma \in ]0, 1[\),
- Iter while \(x_k\) is not a stationary point of \(\theta_{FB}\):
  - select \(H_k\) in \(T(x_k)\)
  - find \(d_k\) root of \(F_{FB}(x_k) + H_k d = 0\)
  - if the equation is not solvable or \(\nabla \theta_{FB}^T(x_k) d_k > -\rho ||d_k||^p\) then
    - \(d_k = -\nabla \theta_{FB}(x_k)\)
    - find the smallest \(i_k \in \mathbb{N}\) such that \(\theta_{FB}(x_k + 2^{-i} d_k) \leq \theta_{FB}(x_k) + \gamma 2^{-i} \nabla \theta_{FB}^T(x_k) d_k\)
    - \(x_{k+1} = x_k + t_k d_k\) with \(t_k = 2^{-i_k}\)

NB: some variants include further checks on the direction \(d_k\).
4.2 Algorithms for CPs

ii. trust region approach

Algo 9.1.35($F_{FB}, \theta_{FB}, T$)

- Init: $x_0 \in \mathbb{R}^n$, $0 < \gamma_1 < \gamma_2 < 1$, $\Delta_0, \Delta_{min} > 0$,
- Iter:
  - select $H_k$ in $T(x_k)$
  - find $d_k = \arg\min_{||d||<\Delta_k} q_{FB}(d, x_k)$ with $q_{FB}(d, x_k) = \nabla \theta_{FB}^T(x_k)d + \frac{1}{2}d^T H_k T H_k d$
  - if $q_{FB}(d_k, x_k) = 0$ then stops
  - if $\theta_{FB}(x_k + d_k) \leq \theta_{FB}(x_k) + \gamma_1 q_{FB}(d_k, x_k)$ then
    
    \[
    x_{k+1} = x_k + d_k \quad \text{and} \quad \Delta_k = \begin{cases} \max(2\Delta_k, \Delta_{min}) & \text{if } \theta_{FB}(x_k + d_k) \leq \theta_{FB}(x_k) + 2 \gamma q_{FB}(d_k, x_k) \\ \max(\Delta_k, \Delta_{min}) & \text{otherwise} \end{cases}
    \]
    
    else $x_{k+1} = x_k$ and $\Delta_k = \Delta_k / 2$

NB: a variant includes an additional constraint on the direction $d_k$.

iii. constrained methods

Algo 9.1.39($F_{FB}, \theta_{FB}$)

- Init: $x_0 \in \mathbb{R}^n$, $\rho > 0$, $p > 1$ and $\gamma \in [0, 1[$
- Iter while $x_k$ is not a stationary point of $\theta_{FB}$:
  - find $y_{k+1}$ solution of linear CP($q_k, \text{Jac} F(x_k)$) and $d_k = y_{k+1} - x_k$ with $q_k = F(x_k) - \text{Jac} F(x_k)x_k$,
  - if the equation is not solvable or $\nabla \theta_{FB}^T(x_k)d_k > -\rho ||d_k||^p$ then
    
    \[
    d_k = -\min(x_k, \nabla \theta_{FB}(x_k))
    \]
  - find the smallest $i_k \in \mathbb{N}$ such that $\theta_{FB}(x_k + 2^{-i_k}d_k) \leq \theta_{FB}(x_k) + \gamma 2^{-i_k} \nabla \theta_{FB}^T(x_k)d_k$
  - $x_{k+1} = x_k + t_k d_k$ with $t_k = 2^{-i_k}$

NB: a variant consists in replacing the linear CP by a convex subprogram solved by a Levenberg-Marquardt method.

(b) min based methods

The equation is $F_{min}(x) = 0$ and the merit function is $\theta_{min}(x) = F_{min}(x)^T F_{min}(x)$ where

\[
F_{min}(x) = \begin{pmatrix} \min(x_1, F_1(x)) \\ \vdots \\ \min(x_n, F_n(x)) \end{pmatrix}.
\]

i. line-search method

In the following, we use

\[
\phi(x, d) = \sum_{i, x_i > F_i(x)} (F_i(x) + \nabla F_i(x)^T d)^2 + \sum_{i, x_i \leq F_i(x)} (x_i + d_i)^2 + \frac{\rho(\theta_{min}(x))}{2} d^T d
\]
and

\[ \sigma(x, d) = \sum_{i, x_i > F_i(x)} F_i(x) \nabla F_i(x)^T d + \sum_{i, x_i \leq F_i(x)} x_i d_i. \]

Algorithm 9.2.2(\( F_{\min}, \theta_{\min}, \phi, \sigma \))
- Init: \( x_0 \in \mathbb{R}^n \) and \( \gamma \in ]0, 1[ \)
- Iter:
  - find \( d_k \) solution of \( \arg \min_{x_k + d \geq 0} \phi(x_k, d) \)
  - if \( d_k = 0 \) then stops
  - find the smallest \( i_k \in \mathbb{N} \) such that \( \theta_{\min}(x_k + 2^{-i} d_k) \leq \theta_{\min}(x_k) - \gamma 2^{-i} \sigma(x_k, d_k) \)
  - \( x_{k+1} = x_k + t_k d_k \) with \( t_k = 2^{-i_k} \)

ii. trust region approach

not possible since min is not everywhere differentiable

iii. mixed compl. func. method

Algorithm 9.2.3(\( F_{FB}, \theta_{FB}, F_{\min} \))
- Init: \( x_0 \in \mathbb{R}^n, \epsilon > 0, p > 1, \rho > 0 \) and \( \gamma \in ]0, 1[ \)
- Iter while \( x_k \) is not a stationary point of \( \theta_{FB} \):
  - select \( H_k \) in \( T_{\min}(x_k) \)
  - \( d_k \) solves \( F_{\min}(x_k) + H_k d = 0 \)
  - if the system is solvable and \( ||F_{\min}(x_k + d_k)|| \leq ||F_{\min}(x_k)|| \) then
    \[ x_{k+1} = x_k + t_k d_k \] with \( t_k = 1 \)
  - else
    - if \( \nabla \theta_{FB}^T(x_k)d_k > -\rho ||d_k||^p \) then
      \[ d_k = -\nabla \theta_{FB}(x_k) \]
    - find the smallest \( i_k \in \mathbb{N} \) such that \( \theta_{FB}(x_k + 2^{-i} d_k) \leq \theta_{FB}(x_k) + \gamma 2^{-i} \nabla \theta_{FB}^T(x_k)d_k \)
    - \( x_{k+1} = x_k + t_k d_k \) with \( t_k = 2^{-i_k} \)

(c) extension to other compl. functions

FB based methods can use with other complementarity functions. Here are some examples.

- \( \psi_{LT}(a, b) = ||(a, b)||_q - (a + b) \) with \( q \geq 1 \)
- \( \psi_{KK}(a, b) = \sqrt{(a-b)^2 + 2qab - (a+b)} \) with \( 0 \leq q < 2 \)
- \( \psi_{CCK}(a, b) = \psi_{FB}(a, b) - qa + b \) with \( q \geq 0 \).

2. finite lower VIs: \( \text{CP}(\mathbb{R}^n_+, F) \):
3. finite upper VIs: \( \text{CP}(\mathbb{R}^n_+, F) \):
4. mixed CPs
5. box constrained VIs
4.3 Algorithms for VIPs
A Bibliography

References


Grantham, W. J. (2005), Gradient transformation trajectory following algorithms for determining stationary min-max saddle points. working paper.


Varadhan, R. (2004), Squared extrapolation methods (squarem): a new class of simple and efficient numerical schemes for accelerating the convergence of the em algorithm. working paper.

B Websites

- Applied mathematics: http://www.applied-mathematics.net,
- Decision tree for optimization software: http://plato.asu.edu/guide.html
- Optimization online: http://www.optimization-online.org/cgi-bin/search.cgi
- Interior-point algorithms: http://www-user.tu-chemnitz.de/~helmberg/sdp_ip.html