

# Numerical Algorithms (MU4IN910)

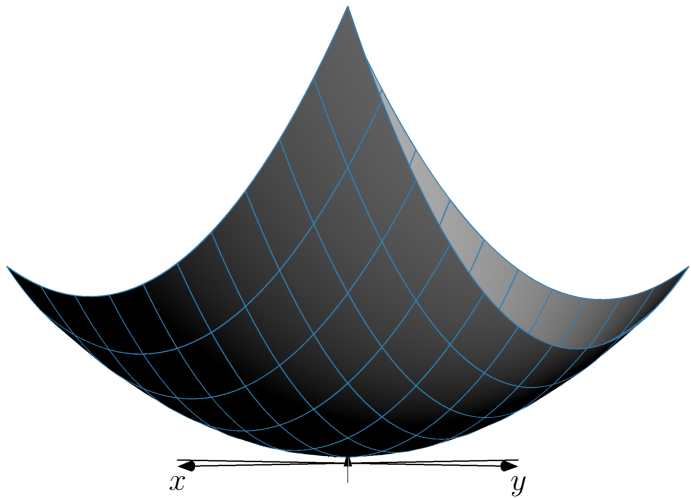
## Lecture 4: Introduction to optimization (2/2)

Stef Graillat

Sorbonne Université



Back to tutorial:  $f(x, y) = 3x^2 + 2y^2 + 2xy + x + y + 10$



# Summary of the previous lecture

- Generalities on the problems of optimization
  - Notion of critical point
  - Necessary and sufficient condition of optimality
  - Lagrange multipliers
- Optimization in dimension 1
  - Golden section search
  - Method of quadratic interpolation
  - Newton's method

- Direct search method
- Steepest descent method
- Newton's method
- Quasi-Newton's method
- Conjugate gradient method (nonlinear)

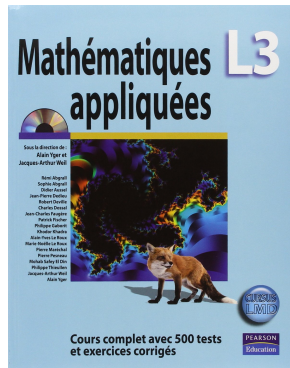
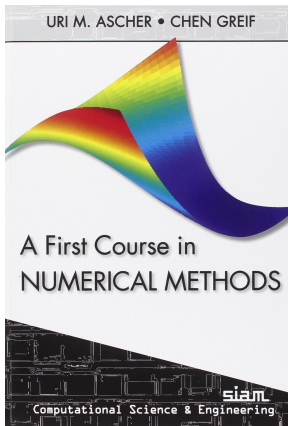
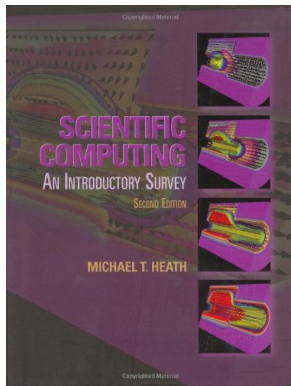
# References

- **Scientific Computing, An Introductory Survey, Michael .T. Heath, Revised Second Edition, SIAM, 2018** (the lecture is mainly based on this book and on the associated slides)
- A First Course in Numerical Methods, Uri M. Ascher, Chen Greif, SIAM, 2011
- Mathématiques appliquées L3, sous la direction de Jacques-Arthur Weil et Alain Yger, Pearson, 2009
- Scientific Computing with Case Studies, Dianne P. O'Leary, SIAM, 2009
- Numerical Optimization, Jorge Nocedal, Stephen Wright, Springer, 2006
- Introduction à l'Analyse Numérique Matricielle et à l'Optimisation, Philippe G. Ciarlet, Dunod, 1982
- Analyse numérique et optimisation, Grégoire Allaire, Éditions de l'École polytechnique, 2005
- L'optimisation, J.B. Hiriart-Urruty, PUF, 1996
- Convex Optimization, S. Boyd et L. Vandenberghe, Cambridge University Press, 2004

## References (cont'd)

- Numerical optimization: Theoretical and practical aspects, F. Bonnans, J.C. Gilbert, C. Lemaréchal et C. Sagastizábal, Springer-Verlag, 2006
- Optimisation continue, F. Bonnans, Dunod, 2006
- Optimisation et contrôle des systèmes linéaires, M. Bergounioux, Dunod, 2001.
- Convex Analysis and Nonlinear Optimization, J.M. Borwein et A.S. Lewis, Springer, 2000
- Lectures on Modern Convex Optimization - Analysis, Algorithms and Engineering Applications, A. Ben-Tal, A. Nemirovski, SIAM, 2001
- Numerical Computing with MATLAB, Cleve Moler, SIAM, 2004
- Linear and Nonlinear Optimization, Igor Griva, Stephen G. Nash, Ariela Sofer, SIAM, 2009
- Practical Methods of Optimization, Roger Fletcher, Wiley, 1987
- A Mathematical View of Interior-Point Methods in Convex Optimization, James Renegar, SIAM, 2001
- Numerical Recipes. The Art of Scientific Computing, William Press, Saul Teukolsky, William Vetterling, Brian Flannery, 3rd Edition, Cambridge University Press, 2007

# References (cont'd)



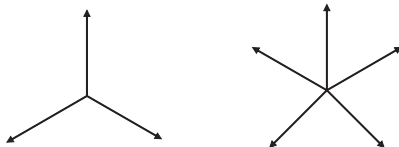
# Optimization in dimension $n \geq 2$



# Pattern search method

## Idea:

- Suppose we are given an initial guess  $x$  at the solution and a set of at least  $n + 1$  directions  $v_i$ ,  $i = 1, \dots, n + 1$  that form a positive basis for  $\mathbb{R}^n$ : this means that any vector can be expressed as a linear combination of these vectors, where the coefficients in the combination are positive numbers.

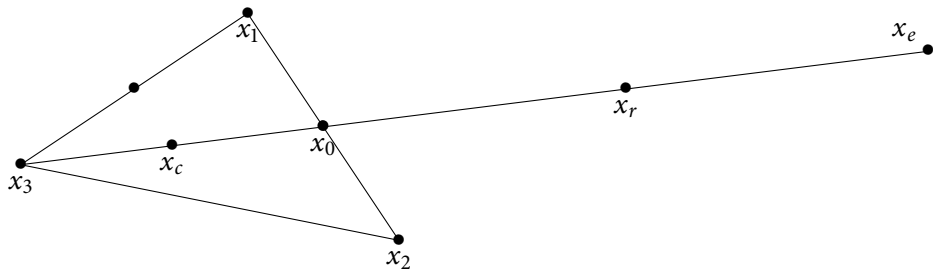


- At each step, we do a line search in each of the directions to obtain  $f(x + \alpha_i v_i)$  and replace  $x$  by the point with the smallest function value.
- This is a remarkably simple algorithm, but works well in practice and is provably convergent!
- Another desirable property is that it is easy to parallelize, and this is crucial to making a no-derivative algorithm effective when  $n$  is large.

# Nelder-Meade algorithm

- 1 Choice of  $N + 1$  points of the  $N$ -dimensional space of the unknowns, forming a simplex:  $\{x_1, x_2, \dots, x_{N+1}\}$ ,
- 2 Compute the values of the function  $f$  at these points, sort the points so as to have  $f(x_1) \leq f(x_2) \leq \dots \leq f(x_{N+1})$ . In fact, it is enough to know the first and the last two.
- 3 Compute  $x_0$ , center of gravity of all points except  $x_{N+1}$ .
- 4 Compute  $x_r = x_0 + (x_0 - x_{N+1})$  (reflection of  $x_{N+1}$  from  $x_0$ ).
- 5 If  $f(x_r) < f(x_N)$ , compute  $x_e = x_0 + 2(x_0 - x_{N+1})$  (simplex expansion). If  $f(x_e) < f(x_r)$ , replace  $x_{N+1}$  by  $x_e$ , otherwise, replace  $x_{N+1}$  by  $x_r$ . Return to step 2.
- 6 If  $f(x_N) < f(x_r)$ , compute  $x_c = x_{N+1} + 1/2(x_0 - x_{N+1})$  (simplex contraction). If  $f(x_c) \leq f(x_N)$ , replace  $x_{N+1}$  by  $x_c$  and return to step 2, otherwise go to step 7.
- 7 Shrink toward  $x_1$ : replace  $x_i$  by  $x_1 + 1/2(x_i - x_1)$  for  $i \geq 2$ . Return to step 2.

# Nelder-Mead algorithm



# Steepest Descent Method

- Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be real-valued function of  $n$  real variables
- At any point  $x$  where gradient vector is nonzero, negative gradient,  $-\nabla f(x)$ , points downhill toward lower values of  $f$
- In fact,  $-\nabla f(x)$  is **locally** direction of steepest descent:  $f$  decreases more rapidly along direction of negative gradient than along any other
- **Steepest descent method**: starting from initial guess  $x_0$ , successive approximate solutions given by

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

where  $\alpha_k$  is a **line search** parameter that determines how far to go in given direction  $-\nabla f(x_k)$

# Steepest Descent Method (cont'd)

- Given descent direction, such as negative gradient, determining appropriate value for  $\alpha_k$  at each iteration is one-dimensional minimization problem

$$\min_{\alpha_k \geq 0} f(x_k - \alpha_k \nabla f(x_k))$$

that can be solved by methods already discussed in dimension 1

- Steepest descent method is very reliable: it can always make progress provided gradient is nonzero
- But method is myopic in its view of function's behavior, and resulting iterates can zigzag back and forth, making very slow progress toward solution
- In general, convergence rate of steepest descent is only linear, with constant factor that can be arbitrarily close to 1

# Steepest Descent Method: example

- Use steepest descent method to minimize

$$f(x) = 0.5x_1^2 + 2.5x_2^2$$

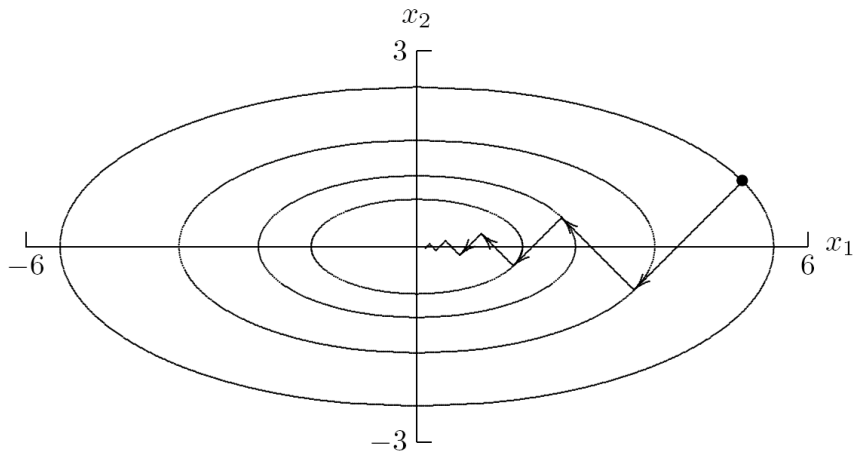
- Gradient is given by  $\nabla f(x) = \begin{pmatrix} x_1 \\ 5x_2 \end{pmatrix}$
- Taking  $x_0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$ , we have  $\nabla f(x_0) = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$
- Perform line search along negative gradient direction

$$\min_{\alpha_0} f(x_0 - \alpha_0 \nabla f(x_0))$$

Exact minimum along line is given by  $\alpha_0 = 1/3$  so next approximation is

$$x_1 = \begin{pmatrix} 3.333 \\ -0.667 \end{pmatrix}$$

# Steepest Descent Method: example (cont'd)



## Principle:

- We want to approximate the gradient by finite difference

$$(\nabla f(x))_i \approx \frac{f(x + te_i) - f(x)}{t}$$

for small  $t$  and  $e_i$  the  $i$ th standard unit vector

- Small accuracy on the gradient
- It is better to use automatic differentiation methods: it is a set of techniques to evaluate the derivative of a function specified by a computer program.



# Newton's method

- We know that a minimum  $x^*$  satisfies  $\nabla f(x^*) = 0$ . We can then solve the equation  $\nabla f(x) = 0$  with Newton's method.
- We can also approximate  $f$  by

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

and minimize the quadratic approximation with respect to  $h$

- In the two cases, we obtain the following iteration

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

## Newton's method (cont'd)

- We do not explicitly invert Hessian matrix, but instead we solve the linear system

$$H_f(x_k)s_k = -\nabla f(x_k)$$

and we compute

$$x_{k+1} = x_k + s_k$$

- Convergence rate of Newton's method for minimization is normally quadratic
- As usual, Newton's method is unreliable unless started close enough to solution to converge

# Newton's method: example

- Use Newton's method to minimize

$$f(x) = 0.5x_1^2 + 2.5x_2^2$$

- Gradient and Hessian are given by

$$\nabla f(x) = \begin{pmatrix} x_1 \\ 5x_2 \end{pmatrix} \quad \text{and} \quad H_f(x) = \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}$$

- Taking  $x_0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$ , we have  $\nabla f(x_0) = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$
- The linear system for Newton step is

$$\begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix} s_0 = \begin{pmatrix} -5 \\ -5 \end{pmatrix}$$

and so  $x_1 = x_0 + s_0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix} + \begin{pmatrix} -5 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$  which is exact solution for this problem (as expected for quadratic function)

## Newton's method (cont'd)

- In principle, line search parameter is unnecessary with Newton's method, since quadratic model determines length, as well as direction, of step to next approximate solution
- When started far from solution, however, it may still be advisable to perform line search along direction of Newton step  $s_k$  to make method more robust
- Once iterates are near solution, then  $\alpha_k = 1$  should suffice for subsequent iterations

## Newton's method (cont'd)

- If objective function  $f$  has continuous second partial derivatives, then Hessian matrix  $H_f$  is symmetric, and near minimum it is positive definite
- Thus, linear system for step to next iterate can be solved in only about half of work required for LU factorization using Cholesky factorization
- Far from minimum,  $H_f(x_k)$  may not be positive definite, so Newton step  $s_k$  may not be descent direction for function, i.e., we may not have

$$\nabla f(x_k)^T s_k < 0$$

- In this case, alternative descent direction can be computed, such as negative gradient or direction of negative curvature, and then perform line search

# Trust regions: an alternative to linesearch

- We approach  $f$  by

$$f(x + h) \approx q(h) := f(x) + \nabla f(x)h + \frac{1}{2}h^T H_f(x)h$$

- We try to minimize  $q(h)$  but we restrict the search to  $\|h\| < r$  with  $r$  small enough
- We choose  $x_{\text{new}} = x + h^*$  with  $h^*$  solution of

$$\min_{\|h\| \leq r} q(h)$$

- This allows us to search for a solution only in a domain where the quadratic approximation is sufficiently accurate

- Newton's method is very fast (if it converges) but it requires the gradient and the Hessian matrix to be evaluated at each iteration. So each iteration is quite expensive!
- Can we solve

$$\min_x f(x)$$

when  $\nabla f(x)$  can be computed but not  $H_f(x)$ ?

Two main options::

- Estimate  $H_f(x)$  using finite differences: discrete Newton method
- Approximate  $H_f(x)$  using Quasi-Newton methods

# Quasi-Newton Methods (cont'd)

- The Newton step:  $x_{k+1} = x_k + s_k$  with  $s_k = H_f(x_k)^{-1} \nabla f(x_k)$
- Quasi-Newton step: accumulate an approximation  $B_k \approx H_f(x_k)$  using free information!
- At step  $k$ , we know  $\nabla f(x_k)$  and we compute  $\nabla f(x_{k+1})$  where  $x_{k+1} = x_k + s_k$
- $H_f(x_k)$  satisfies

$$H_f(x_k)s_k = \lim_{t \rightarrow 0} \frac{\nabla f(x_k + ts_k) - \nabla f(x_k)}{t}$$

Let us denote  $g_k = \nabla f(x_k)$ . If  $f$  is quadratic then

$$H_f(x_k)s_k = g_{k+1} - g_k$$

- We will ask the same property of the approximation  $B_{k+1}$  and call this the **secant equation**:

$$B_{k+1}s_k = g_{k+1} - g_k$$



# Quasi-Newton Methods (cont'd)

- So we know how we want  $B_{k+1}$  to behave in the direction  $g_{k+1} - g_k$ , and we have no new information in any other direction, so we could require

$$B_{k+1}v = B_k v \quad \text{if} \quad v^T s_k = 0$$

- There is a unique matrix  $B_{k+1}$  that satisfies the secant equation and the no-change conditions. It is called **Broyden's method**:

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

where  $s_k = x_{k+1} - x_k$  and  $y_k = g_{k+1} - g_k$

- $B_{k+1}$  is formed from  $B_k$  by adding a rank-one matrix

# BFGS (Broyden-Fletcher-Goldfarb-Shanno) method

- $B_{k+1}$  is not necessarily symmetric, even if  $B_k$  is
- In order to regain symmetry, we need to sacrifice the no-change conditions. Instead, we formulate the problem in a least change sense:

$$\min_{B_{k+1}} \|B_{k+1} - B_k\|$$

satisfying  $B_{k+1}$  symmetric and the secant condition  $B_{k+1}s_k = y_k$

- The solution depends on the choice of norm
- For a certain norm, we obtain the BFGS formula

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

# BFGS method (cont'd)

```
 $x_0 =$  initial guess  
 $B_0 =$  initial Hessian approximation  
for  $k = 0, 1, 2, \dots$   
    Solve  $B_k s_k = -\nabla f(x_k)$   
     $x_{k+1} = x_k + s_k$   
     $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$   
     $B_{k+1} = B_k - (B_k s_k s_k^T B_k) / (s_k^T B_k s_k) + (y_k y_k^T) / (y_k^T s_k)$   
end
```

## BFGS method (cont'd)

- In practice, factorization of  $B_k$  is updated rather than  $B_k$  itself, so linear system for  $s_k$  can be solved at cost of  $\mathcal{O}(n^2)$  rather than  $\mathcal{O}(n^3)$  work
- Unlike Newton's method for minimization, no second derivatives are required
- Can start with  $B_0 = I$ , so initial step is along negative gradient, and then second derivative information is gradually built up in approximate Hessian matrix over successive iterations
- BFGS normally has superlinear convergence rate, even though approximate Hessian does not necessarily converge to true Hessian

# Nonlinear conjugate gradient method

- This is a method which does not require the calculation of the of the Hessian and does not require either the storage of a matrix approximating the Hessian
- The algorithm generates a sequence of conjugate directions accumulating implicitly information on the Hessian
- For some quadratic functions, the algorithm of the conjugate gradient algorithm converges after at most  $n$  iterations, where  $n$  is the dimension of the problem

# Nonlinear conjugate gradient method (cont'd)

$x_0 =$  initial approximation

$g_0 = \nabla f(x_0)$

$s_0 = -g_0$

for  $k = 0, 1, 2, \dots$

    choose  $\alpha_k$  minimizing  $f(x_k + \alpha_k s_k)$

$x_{k+1} = x_k + \alpha_k s_k$

$g_{k+1} = \nabla f(x_{k+1})$

$\beta_{k+1} = (g_{k+1}^T g_{k+1}) / (g_k^T g_k)$

$s_{k+1} = -g_{k+1} + \beta_{k+1} s_k$

end

→ the algorithm will be studied in detail in the course on iterative methods for solving linear systems

# Truncated Newton's method

- One can reduce the amount of computation of a Newton type method by solving the linear system by an iterative method
- A small number of iterations then allows to obtain in general a solution
- The iterative method used to solve the system is often the conjugate gradient method
- These algorithms only require matrix-vector products. It is then possible to avoid the construction of the Hessian by using a finite difference method on the gradient

# Nonlinear least squares

- Given data  $(t_i, y_i)$ , find vector  $x$  of parameters that gives “best fit” in least squares sense to model function  $f(t, x)$ , where  $f$  is nonlinear function of  $x$
- We define components of residual function

$$r_i(x) = y_i - f(t_i, x) \text{ pour } i = 1, \dots, m$$

so we want to minimize  $\phi(x) = \frac{1}{2} r(x)^T r(x)$

- Gradient vector is  $\nabla \phi(x) = J(x)^T r(x)$  and Hessian matrix is

$$H_\phi(x) = J(x)^T J(x) + \sum_{i=1}^m r_i(x) H_i(x)$$

where  $J(x)$  is Jacobian of  $r(x)$  and  $H_i(x)$  is Hessian of  $r_i(x)$



# Gauss-Newton method

- Linear system for Newton step is

$$(J(x_k)^T J(x_k) + \sum_{i=1}^m r_i(x_k) H_i(x_k)) s_k = -J(x_k)^T r(x_k)$$

- $m$  Hessian matrices  $H_i$  are usually inconvenient and expensive to compute
- Moreover, each  $H_i$  is multiplied by residual component  $r_i$ , which is small at solution if fit of model function to data is good
- This motivates Gauss-Newton method for nonlinear least squares, in which second-order term is dropped and linear system

$$J(x_k)^T J(x_k) s_k = -J(x_k)^T r(x_k)$$

is solved for approximate Newton step  $s_k$  at each iteration

- This is system of normal equations for linear least squares problem

$$J(x_k) s_k \approx -r(x_k)$$

- Gauss-Newton method replaces nonlinear least squares problem by sequence of linear least squares problems whose solutions converge to solution of original nonlinear problem
- If residual at solution is large, then second-order term omitted from Hessian is not negligible, and Gauss-Newton method may converge slowly or fail to converge
- In such "large-residual" cases, it may be best to use general nonlinear minimization method that takes into account true full Hessian matrix

# Levenberg-Marquardt Method

- It is possible that the matrix  $J(x_k)^T J(x_k)$  is singular (but it is symmetric)
- The Levenberg-Marquardt method consists in perturbing the matrix to make it positive definite
- We solve at each step the system

$$(J(x_k)^T J(x_k) + \mu_k I) s_k = -J(x_k)^T r(x_k)$$

- The parameter  $\mu_k$  is chosen so that  $(J(x_k)^T J(x_k) + \mu_k I)$  is symmetric positive definite

# Constrained Optimization

- We seek to solve the problem

$$\min_x f(x) \text{ under } g(x) = 0$$

with  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$  with  $m \leq n$

- We look for a critical point of the Lagrangian  $\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)$
- By applying Newton's method to the system nonlinear system

$$\nabla \mathcal{L}(x, \lambda) = \begin{pmatrix} \nabla f(x) + J_g(x)^T \lambda \\ g(x) \end{pmatrix} = 0$$

we obtain the linear system

$$\begin{pmatrix} B(x, \lambda) & J_g(x)^T \\ J_g(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ \lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + J_g(x)^T \lambda \\ g(x) \end{pmatrix}$$

This technique is called sequential quadratic programming

# Linear programming

- This is a problem of minimizing a linear function under linear constraints
- It can be written in the form

$$\min f(x) = c^T x \quad \text{under} \quad Ax = b \text{ and } x \geq 0$$

with  $m < n$ ,  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$  and  $c, x \in \mathbb{R}^n$

- The set of constraints forms a polyhedron of  $\mathbb{R}^n$  and the minimum is reached at a vertex of this polyhedron
- The simplex method consists in going from vertices to vertices until we find the minimum

# Linear programming (continued)

- The simplex method is reliable and efficient in general but is exponential in the size of the problem in the worst cases
- Recently developed interior point methods allow to solve linear programming problems with polynomial complexity in the worst case
- The interior point methods navigate inside the admissible regions admissible regions by not restricting themselves only to the vertices
- Although interior point methods are of better complexity, the simplex method complexity, the simplex method remains the most commonly used method used in practice

# Example

- Consider the problem

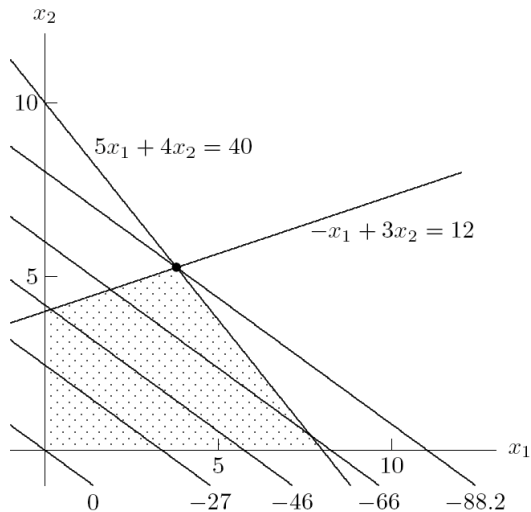
$$\min_x -8x_1 - 11x_2$$

under the conditions

$$5x_1 + 4x_2 \leq 40, \quad -x_1 + 3x_2 \leq 12, \quad x_1 \geq 0, \quad x_2 \geq 0$$

- The minimum is obtained in  $x_1 = 3,79$  and  $x_2 = 5,26$  where the objective function is  $-88.2$

## Example (continued)





- Newton's method
- Alternative methods not requiring the calculation of the hessian
  - Quasi-Newton's method
  - Newton's method by finite difference
- Alternative methods not requiring the computation of the hessian nor the storage of matrix
  - Steepest descent method
  - Nonlinear conjugate gradient method
- Alternative methods that do not require the computation of derivatives
  - Finite difference method
  - Nelder-Meade method
  - Pattern search method