About minimization

- Generalities
- Line search problem
- Newton's methods
- Conjugate gradient method
- Large size problems
- Lanczos algorithm

The problem

Let an ensemble $X \subset R^n$

And a function $f : X \rightarrow R$

We search $x^* \in X$ such as :

$\forall\, x \in X\ ,\ f(x) \geq f(x^*)$

Some exemples

- Molecular biologie
- Meteorologie (very large size problem)
- Trajectory of an undersea object (constraint)
- Optimization of electricity production

General principle

An optimization program is composed of two distinct parts :

1 ➔ Is strickly the algorithm (written by a mathematician)

2 ➔ Is the « simulator » which compute f and ∇f (written by the physicist)

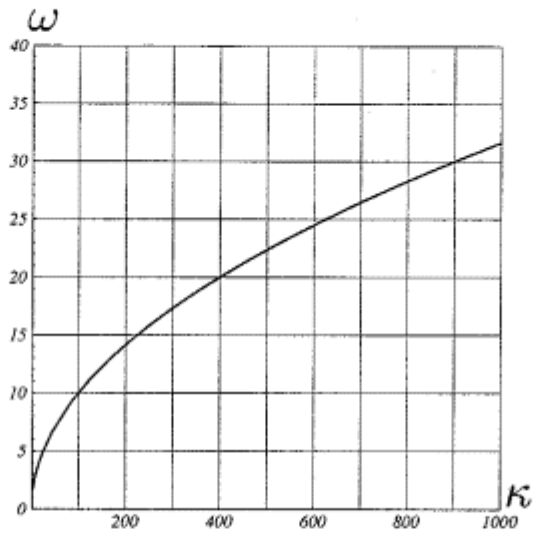
Schematic descent algorithm

- 1➔ Initialization (x_1 and $\varepsilon > 0$)
- 2➔ Stop test : if $|\nabla f(x_k)| \leq \varepsilon$ stop
- 3➔ Computation of d_k the direction of descent
- 4➔ Line search : find t_k $x_k \rightarrow x_{k+1}=x_k+t_k d_k$

Gradients methods (first order)

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

These methods are very bad because $f(x-td_k)$ decreases only if t is very small



Number of iterations of Steepest Descent required to match one iteration of CG.

Conditioning

Let a variable change $x(y) = Ay$

Consider the function $h(y)=f[x(y)]=f(Ay)$

Then : $\nabla h(y) = A^T \nabla f(Ay)$

The problem is numericaly very different

Line search

- When the direction d_k is found (see later)
- The aim is to find the best step t_k
- It's a sub-algorithm, so it must be cheap

Line search

- We search t optimal
- We have x : start point
- And d : direction of descent
- We need a cost function $t \rightarrow q(t)$, $(t \geq 0)$

$(q(t)=f(x+td))$

RQ : We want to minimize f and not q !

Schematic line search algorithm

- Step 0 : initial $t>0$, t_l (and may be t_r)
- Step 1 : t is tested

  If a) ➔ stop

  If b) ➔ $t_r=t$ go to step 2

  If c) ➔ $t_l=t$ go to step 2
- Step 2 :

  If t_r is not defined we compute $t>t_l$

  Else we compute $t\in]t_l,t_r[$

  go to step 1

Schematic line search algorithm some remarks

- a), b) and c) must be a partition of R^+
- $\exists$ t* such as $\forall$ t$\geq$t* a) or b) is verified
- Each $[t_l, t_r]$ contains an interval where a) is satisfied

Schematic line search algorithm computation of a new t

- t_r is not defined :

  We need an extrapolation $t^+ = at$, with $a>0$

- t_r is defined :

  We need an interpolation $t^+ = (t_l+t_r)/2$

Line search of Wolfe

- a) $q(t) \leq q(0)+m_1 t q'(0)$ and $q'(t) \geq m_2 q'(0)$ ➔ then stop
- b) $q(t) > q(0)+m_1 t q'(0)$ ➔ then $t_r = t$
- c) $q(t) \leq q(0)+m_1 t q'(0)$ and $q'(t) < m_2 q'(0)$ ➔ then $t_l = t$

Avec $0 < m_1 < m_2$

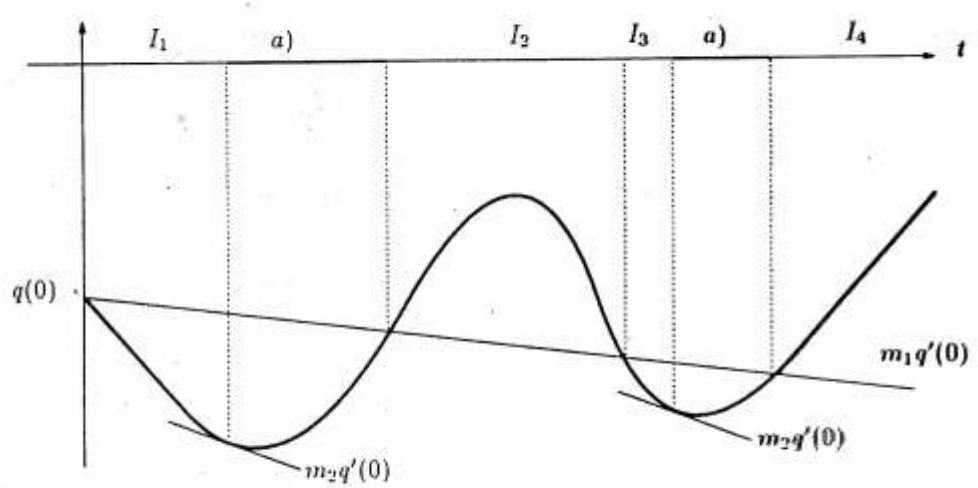


FIG. 2.2 – La règle de Wolfe

Rule of Wolfe, some remarks

- The rule of Wolfe need $q'(t)$ ($\nabla f(x_k+td_k)$) at each call to the simulator, therefore at each step during the line search.

- In some problems, the computation of the gradient is more expensive than the computation of the cost function.

Rule of Goldstein and Price

- a) $m_2 q'(0) \leq (q(t)-q(0))/t \leq m_1 q'(0)$
- b) $m_1 q'(0) < (q(t)-q(0))/t$
- c) $(q(t)-q(0))/t < m_2 q'(0)$

Avec $0 < m_1 < m_2$

Newtonian's method

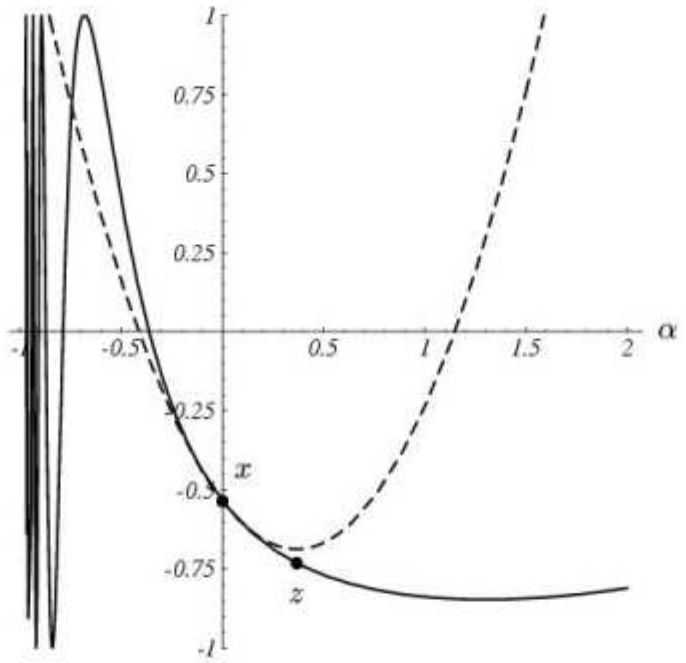
- We want to solve the optimal condition $f'(x)=g(x)=0$

- Starting at x_k, we replace g by its linear approximation :

$$g(x_k+d) = g(x_k) + g'(x_k)d + 0(|d|)$$

Our problem is to find d such as $g(x_k+d)=0$

$$\rightarrow d = -[g'(x_k)]^{-1} g(x_k)$$



The Newton-Raphson method.

Solid curve: The function to minimize.

Dashed curve: Parabolic approximation to the function, based on first and second derivatives at x.

z is chosen at the base of the parabola.

Generals remarks

- g is the gradient of f so g'=f'' is its hessian
- The convergence is very fast
- But it's necessary to compute the hessian and to inverse it
- The propriety $f(x_{k+1}) < f(x_k)$ (descent) is not sure
- The method can diverge very violently

How to force global convergence

- The main idea is to consider d (or d_k) as a direction of descent and to add a line search to minimize the function $q(t) = f(x_k+td_k)$
- Line search is possible only if $q'(0)<0$ that is $<g(x_k),f''(x_k)\ g(x_k)> > 0$
- Therefore, this method works well only if $f''(x_k)$ is definite positive.
- RQ: $d_k = -[f''(x_k)]^{-1}\ g(x_k)$ gives not only a direction but also a step along this direction, that is $t=1$ is a good approximation of the optimal step.

Schematic quasi-newton algorithm

- 0 ➔ Initialization (x_1, $\varepsilon>0$ and also a matrix W definite positive) then the gradient $g=g(x)$ is computed
- 1 ➔ If $|g| \leq \varepsilon$ stop
- 2 ➔ We compute $d = -W\ g$
- 3 ➔ We do a line search, starting at $t=1$, to compute $x^+=x+t_d$ and its gradient $g^+=g(x^+)$
- 4 ➔ We compute the new matrix W^+ and go to step 1

Schematic quasi-newton algorithm we need :

- A line search ➔ the rule of Wolfe
- An initial matrix W ➔ identity
- To compute W^+ at step 4

Quasi-newton algorithm: computation of W^+

- Notation : $s = s_k = x_{k+1} - x_k \qquad y = y_k = g_{k+1} - g_k$
- Davidon-Fletcher-Powell formula :

$$W^+ = W + (ss^t)/\langle y,s\rangle - (Wyy^tW)/\langle y,Wy\rangle$$

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula :

$$W^+ = W - (sy^tW+Wys^t)/\langle y,s\rangle + [1+\langle y,Wy\rangle/\langle y,s\rangle](ss^t)/\langle y,s\rangle$$

Conjugate gradient method

- The aim of this method is to resolve a linear system, that is to minimize a quadratic function.
- Linear system : $Ax + b = 0$ with A symetric
- Minimize the function : $f(x) = \frac{1}{2}\langle Ax,x\rangle + \langle b,x\rangle + a$

Conjugate gradient method

- We search d_k in $U_k := \{v = \Sigma \alpha_i g_i + \Sigma \beta_j d_j\}$
- One can show that :

$$\langle g_k, g_i \rangle = 0 \ (i=1, \ldots, k-1)$$
$$\langle A d_k, d_i \rangle = 0 \ (i=1, \ldots, k-1)$$
$$\langle g_k, d_i \rangle = 0 \ (i=1, \ldots, k-1)$$

The d_i are conjugate in relation to matrix A

Conjugate gradient method : direction and step

- The directions are :

$$d_1 = -g_1 \text{ , then } d_{k+1} = -g_{k+1} + c_k d_k \text{ with } c_k = |g_{k+1}|^2 / |g_k|^2$$

- For the steps we need a line search, but the function is quadratic so :

$$0 = \langle g(x_k + t d_k), d_k \rangle = \langle g_k + t A d_k, d_k \rangle = \langle g_k, d_k \rangle + t \langle A d_k, d_k \rangle$$

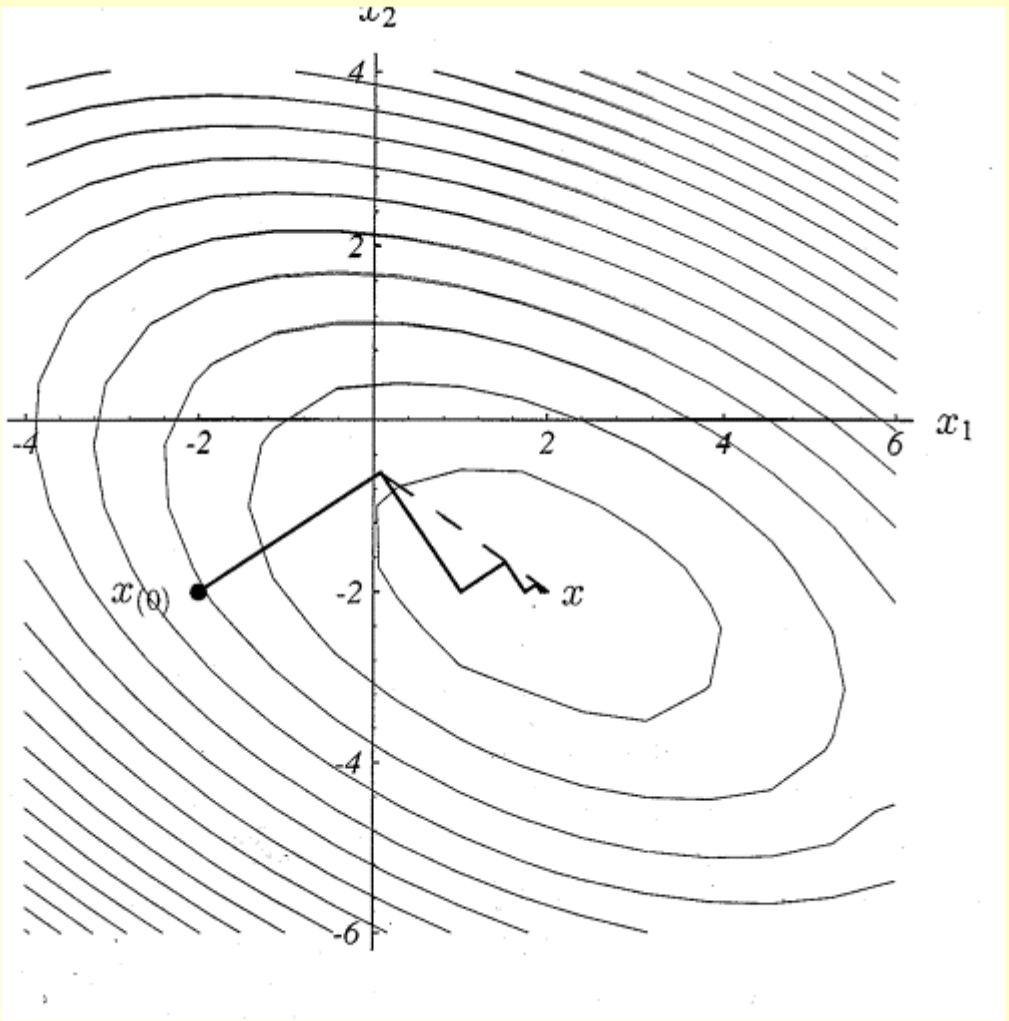
$$t_k = -\langle g_k, d_k \rangle / \langle A d_k, d_k \rangle$$

Conjugate gradient : in practice

- We want $\langle d_i, g_{i+1} \rangle = 0$ $\quad (g_{i+1}=g(x_i+t_i d_i))$
- We choose t^+ and we compute $g^+ = g(x_i + t^+ d_i)$
- Then : $\quad t^+(g_{i+1}-g_i) = t_i(g^+-g_i)$

$$\langle d_i, g_{i+1} \rangle = 0 = \langle d_i, g_i \rangle + \langle d_i, t_i/t^+(g^+-g_i) \rangle$$

$$t_i=(t^+\langle d_i, g_i \rangle)/(\langle d_i, g^+ \rangle - \langle d_i, g_i \rangle)$$


x_2
4
2
-2
-4
-6
-4
-2
2
4
6
x_1
$x_{(0)}$
x

Conjugate gradient if f is not quadratic

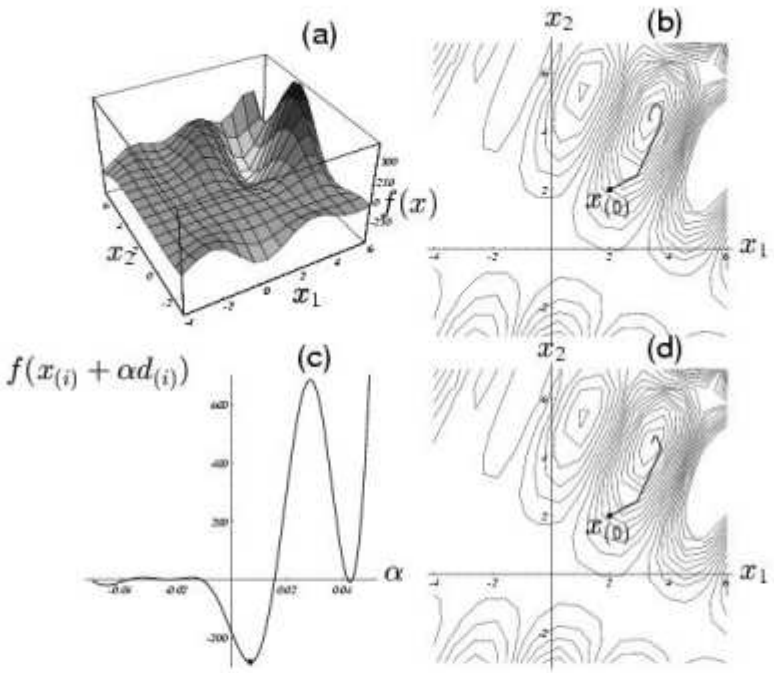
- Fletcher-Reeves : $c_k = |g_{k+1}|^2 / |g_k|^2$
- Polac-Ribière : $c_k = \langle g_{k+1} - g_k , g_{k+1} \rangle / |g_k|^2$

Remarks :

(i) One can show that Fletcher-Reeves converges

(ii) There is a known case where Polac-Ribière does not converge

(iii) But, Polac-Ribière converges more rapidly than Fletcher-Reeves.

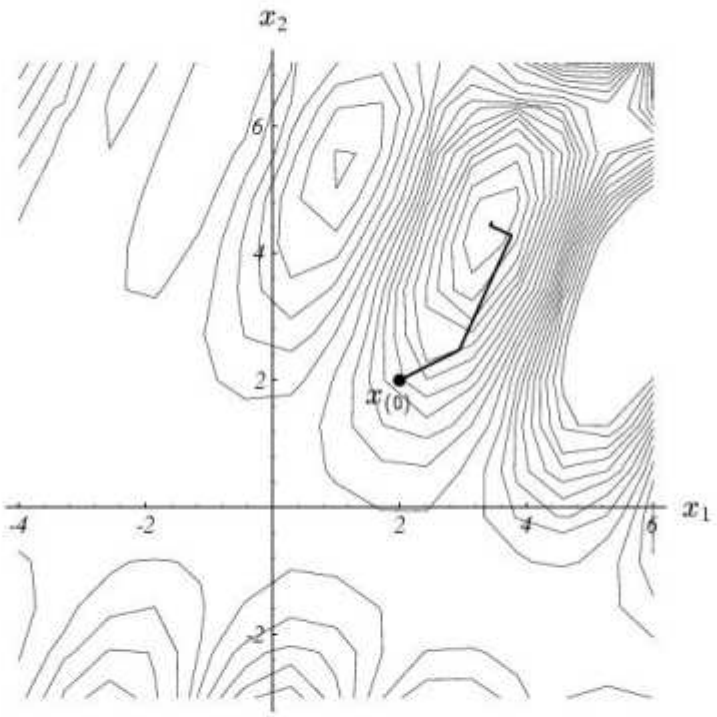


The nonlinear Conjugate Gradient Method.

(b) Fletcher-Reeves CG.

(c) Cross-section of the surface corresponding to the first step of Fletcher-Reeves.

(d) Polak-Ribière CG.



Nonlinear CG can be more effective with periodic restarts.

Conjugate gradient and rounding error

- Rounding error ➔ loss or orthogonality of the gradient vectors
- Loss of orthogonality prevent the super-linear convergence
- An explicit orthogonalization (by a Gram-Schmidt algorithm) is sufficient to allow super-linear convergence to occur

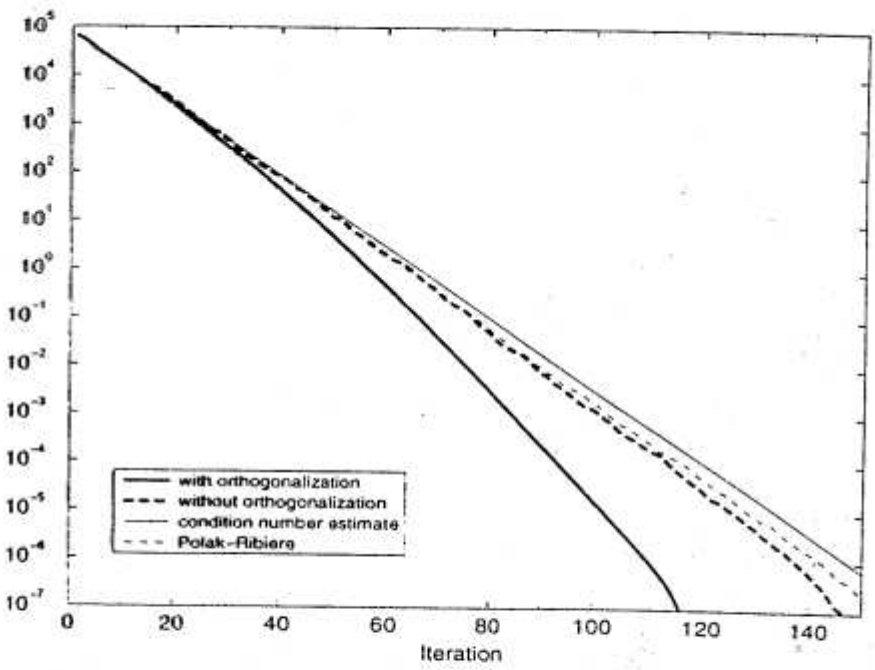


Figure 1: Convergence of conjugate gradients as a function of iteration for a 4dVar cost function. The abscissa is the square of the Hessian norm of the difference between the control vector at a given iteration and its value at the 118th iteration of the orthogonalized algorithm. The thick dashed curve is for the standard conjugate gradient algorithm. The thick solid curve is for the version of the algorithm in which each gradient is explicitly orthogonalized against its predecessors. The thin dashed curve uses the Polak-Ribiere method to determine the descent direction. The thin solid line is the upper bound on the convergence rate defined by equation 12.

Conjugate gradient and quasi-newton

- It's possible to show that the conjugate gradient method is a « poor man » quasi-newton :

- If we suppose that the line search is exact : $\langle g^+,d\rangle=0$

- And if we define the BFGS matrix with W=I :

$$W^+=I-(sy^t+ys^t)/\langle y,s\rangle + [1+|y|^2/\langle y,s\rangle](ss^t/\langle y,s\rangle)$$

Then, $$d^+=-W^+g^+ = -g^+ + (\langle y,g^+\rangle/\langle y,d\rangle)d$$

Large size problems

- In the case of large size problems it's not possible to compute and store the matrix W
- The only usefull method is the conjugate gradient
- We have seen that the conjugate gradient is a « poor man » quasi-newton
- Following this idea it's possible to construct a method witch compute and store only a litle part of the matrix

Large size problems (2)

- A quasi-newton matrix W_k is definite by the initial matrix W_1 and the 2(k-1) vectors $s_1,y_1, \ldots,s_{k-1},y_{k-1}$

- So it's necessarily possible to compute $d_k=-W_kg_k$ using these 2(k-1) vectors, that is 2(k-1)n numbers instead of n(n+1)/2 numbers

Limited memory quasi-newton algorithm

- Let an initial matrix W_0, a vector g and m couples of vectors $\{s_i,y_i\}$, then $d=-h_m$ with:

1 ➔ $q_m=g$ and for i=m to 1 we do:

$\alpha_i = \langle q_i,s_i\rangle / \langle y_i,s_i\rangle$ and $q_{i-1} = q_i - \alpha_i y_i$

2 ➔ $h_0 = W_0 q_0$; for i=1 to m we do

$\beta_i = \langle y_i,h_{i-1}\rangle / \langle y_i,s_i\rangle$ and $h_i = h_{i-1} + (\alpha_i - \beta_i)s_i$

Limited memory quasi-newton algorithm some remarks

- The algorithm is very good
- In a quasi-newton algorithm the choice of W_0 is not very important, but in the limited memory algorithm $W_0=I$ is not a good choice.
- There is an irritating question : it's seems (empiricaly) that $m \approx 20$ is an optimum independently of n. There is no explanation of that.

Truncated Newton algorithm

- The essence is to approximate the function at each iteration by the quadratic function :

$$\phi(x) = f(x_k) + \langle \nabla f(x_k), (x-x_k) \rangle + \tfrac{1}{2} \langle (x-x_k), [f''(x_k)](x-x_k) \rangle$$

Then a conjugate gradient is used to give an estimate of the minimum. Since $\phi(x)$ is only an approximation of $f(x)$ the minimization of $\phi(x)$ is « truncated ».

The Hessain-vector product is approximated with finite-differences:

$$f''d = 1/\varepsilon(\nabla f(x_k - \varepsilon d) - \nabla f(x_k))$$

Truncated Newton algorithm

- Truncated Newton methods frequently display impressive convergence rates when expressed as error reduction per ietration.
- However, each iteration is computationally expensive, since it requires the evaluation of several Hessian-vector products.
- Truncated Newton methods are not, in general, markedly better than quasi-Newton methods.

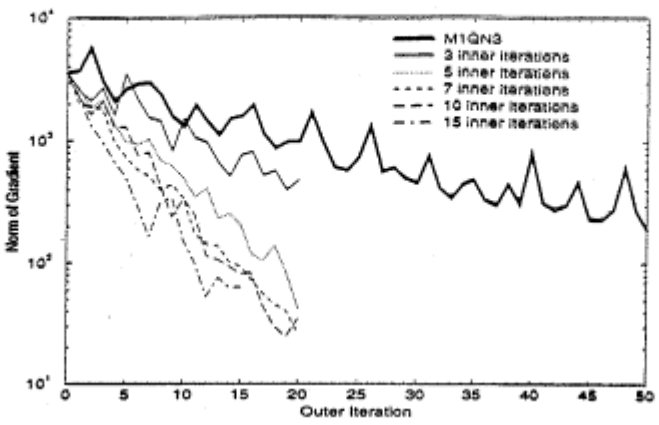


Figure 2: Convergence of the Generalized truncated Newton algorithm as a function of outer iteration.

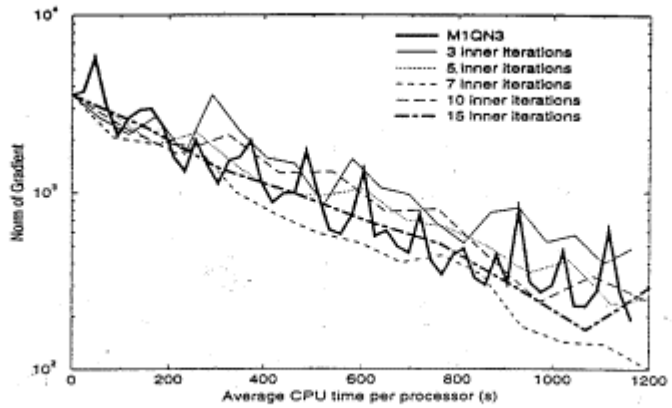


Figure 3: Convergence of the Generalized truncated Newton algorithm as a function of CPU time.

The Lanczos method

- Lanczos algorithm computes a few of the extreme eigenvalues, and the corresponding eigenvectors, of a symetric matrix.
- The Lanczos method is a technique applicable to large, sparse, symetric eigen-problems.
- The Lanczos algorithm does not access the elements of the matrix directly (there is no need for conventional storage of the matrix)
- It is very useful when only a part of the spectrum is required (less than n/100)

The Lanczos method

- It computes a partial transformation of the matrix A(n,n), that is to find two matrices Q(n,j)(unitary matrix) and T(j,j) such that :

$$Q^t A Q = T$$

Where the matrix T is tridiagonal :

$$\begin{pmatrix}
\alpha_1 & \beta_2 & 0 & 0 & \bullet & \bullet & \bullet \\
\beta_2 & \alpha_2 & \beta_3 & 0 & \bullet & \bullet & \bullet \\
0 & \beta_3 & \alpha_3 & \bullet & \bullet & \bullet & \bullet \\
0 & 0 & \bullet & \bullet & \bullet & 0 & 0 \\
\bullet & \bullet & \bullet & \bullet & \bullet & \beta_{j-1} & 0 \\
\bullet & \bullet & \bullet & 0 & \beta_{j-1} & \alpha_{j-1} & \beta_{j-1} \\
\bullet & \bullet & \bullet & 0 & 0 & \beta_j & \alpha_j
\end{pmatrix}$$

The Lanczos method

$$Q^t A Q = T \Leftrightarrow A Q = Q T$$

Equating columns we have :

$$A q_i = \beta_{i-1} q_{i-1} + \alpha_i q_i + \beta_i q_{i+1}$$

The orthogonality of the vectors q_i implies that :

$$\alpha_i = q_i^t A q_i$$

Then :

$$q_{i+1} = ((A - \alpha_i I) q_i - \beta_{i-1} q_{i-1}) / \beta_i$$

With :

$$\beta_i = \|(A - \alpha_i I) q_i - \beta_{i-1} q_{i-1}\|$$

ARPEGE

- N1GC2 : Conjugate gradient method
- M1QN3 : Limited memory quasi-newton algorithm with line search of Wolf
- CONGRAD : Conjugate gradient with Lanczos and orthogonalization

I don’t speek about :

- Trust region method

- Constraint optimization

References

- J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, C. Sagastizabal (1997). Optimisation Numérique, Aspect théorique et pratique. *Springer*
- R. Fletcher (1993) An Overview of Unconstrained Optimization. *Numerical Analysis Report NA/149*
- R. Buizza (1992) Unstable perturbations computed using the adjoint technique. *ECMWF, Technical Memorandum No. 189*

References : Conjugate gradient

- M. Al-Baali (1985). Descent property and global convegence of the Fletcher-Reeves methods with inexact line search. *IMA Journal of Numerical Analysis, 5, 121-124*
- M.R. Hestenes (1980). Conjugate gradient methods. *Journal of Optimization Theory and Applications, 4, 303-320*
- J.Ch. Gilbert, J. Nocedal (1992). Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal of Optimization, 2, 21-42*

References : Quasi-Newton

- W.C. Davidon (1959). Variable metric method for minimization. AEC Research and Development Report ANL-5990, Argonne National Laboratory, Argone, Illinois.
- J.E. Dennis, J/J. Moré (1977). Quasi-Newton methods, motivation and theory. *SIAM Review, 19, 46-89*
- R. Fletcher, M.J.D. Powell (1963). A rapidly convergent descent method for minimzation. *The Computer Journal, 6, 163-168.*