Constrained derivative-free optimization using an SQP-filter approach

Philippe L. Toint
Department of Mathematics, FUNDP,
Namur, Belgium
(with B. Colson)

Tilburg, The Netherlands
December 2001

The problem

$$\min \quad f(x)$$
$$\text{s. t.} \quad c(x) = 0$$

where

- $x \in \mathbb{R}^n, \quad f(x) \in \mathbb{R}, \quad c(x) \in \mathbb{R}^m$
- f and c are nonlinear
- f and c are smooth

Assumptions

- $m \leq n$
- $f'(x)$ and $\nabla c(x)$ not available
- $f(x)$ and $c(x)$ computable $\forall x \in \mathbb{R}^n$

Framework

1. General:
   Derivative-Free Optimization (DFO)

2. Optimization:
   Sequential Quadratic Programming (SQP)

3. Approximation:
   Newton Fundamental Polynomials (NFP)

Outline

1. Derivative-Free Optimization

2. Sequential Quadratic Programming

3. Filter method

4. Interpolation models

5. Application to constrained DFO

6. Practicalities

7. Computational experience

1. Derivative-Free Optimization

- Many applications:
  ⇒ useful for nasty functions
  typical from some real-world problems

- Idea:

  do not attempt to compute directly approximations to the unavailable derivative information ...

  ... but rather derive an improvement in the objective function from a model of this function

2. Sequential quadratic programming

- Basic idea

$$\min_x \quad f(x)$$

$$\text{s. t.} \quad c(x) = 0$$

$$\Updownarrow$$

Iteratively solve

$$\min_s \quad m_k(s)$$

$$\text{s. t.} \quad c_k + A_k s = 0$$

where

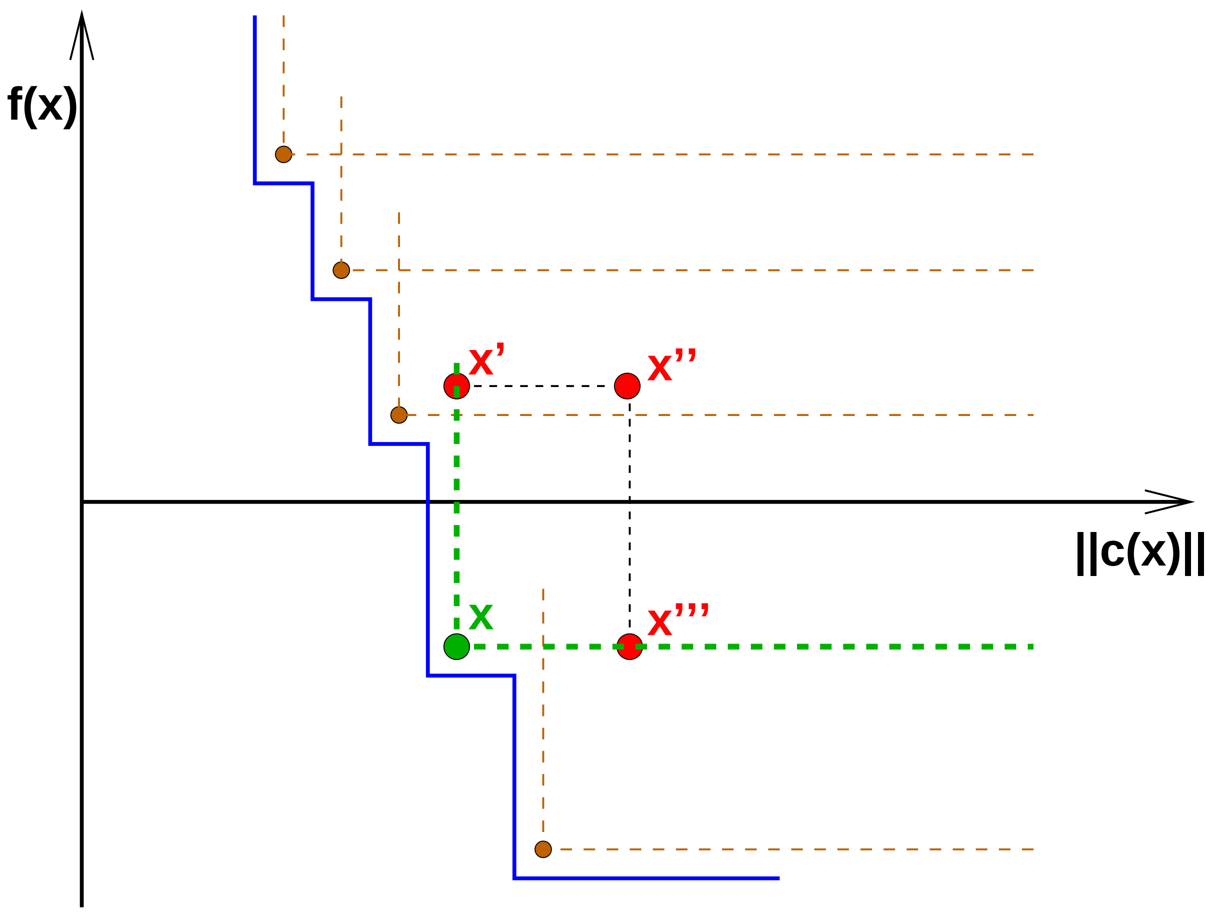
m_k $\equiv$ *quadratic* model of f at x_k
s $\equiv$ improvement step ($\Rightarrow$ *sequential*)
c_k $\equiv c(x_k)$
A_k $\equiv$ Jacobian of c at x_k

3. Filter method

(Fletcher and Leyffer, 1997, 1998)

For equality constraints:
x' is *dominated* by x when

$$f(x') \geq f(x) \text{ and } \|c(x')\| \geq \|c(x)\|$$



4. Interpolation models (1)

$h \equiv$ general smooth function
$Y \equiv$ interpolation set

$\Rightarrow Y \subset \{x_k\} \cup \{y_i$ at which $h(y_i)$ is known$\}$

Requirements :

- Interpolation models

$$m_k^h(y_i) = h(y_i) \quad y_i \in Y$$

For quadratic models

$$m_k^h(x_k + s) = h(x_k) + \langle g_k, s\rangle + \frac{1}{2}\langle s, H_k s\rangle$$
$$\Downarrow$$
$$|Y| \geq 1 + n + \tfrac{1}{2}n(n+1) = \tfrac{1}{2}(n+1)(n+2)$$

- Add geometric conditions to ensure existence and uniqueness of the interpolant

$\Rightarrow$ *poisedness*

4. Interpolation models (2)

Newton fundamental polynomials (NFP)

- Y is partitioned into $d+1$ blocks

  ($d \equiv$ degree)

$$Y^{[l]} = \{y_1^{[l]}, \ldots, y_{|Y^{[l]}|}^{[l]}\} \qquad l = 0, \ldots, d$$

- to each point $y_i^{[l]}$ corresponds a single *NFP* of degree l satisfying

$$N_i^{[l]}(y_j^{[m]}) = \delta_{ij}\delta_{lm} \; \forall y_j^{[m]} \in Y^{[m]} \; (m \leq l)$$

- computation of $N_i^{[l]}$ involves

$$N_i^{[l]}(x) := N_i^{[l]}(x)/|N_i^{[l]}(y_i^{[l]})|$$

(analogy with Gram-Schmidt)

4. Interpolation models (3)

- Y is *poised* $\Leftrightarrow$ "nonzero" pivots

  (in practice : $|N_i^{[l]}(y_i^{[l]})| \geq \theta_{thr} > 0$)

After computation of the NFPs :

- use *generalized finite differences*

$$\lambda_0(x) = h(x)$$

$$\lambda_{l+1}(x) = \lambda_l(x) - \sum_{i=1}^{|Y^{[l]}|} \lambda_l(y_i^{[l]}) N_i^{[l]}(x)$$

$$(l = 0, \ldots, d-1)$$

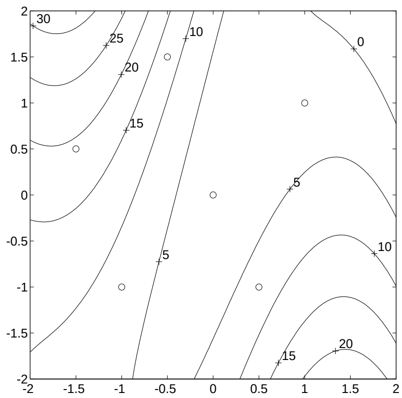
to compute interpolating polynomials

$$m_k(x) = \sum_{l=0}^{d} \sum_{i=1}^{|Y^{[l]}|} \lambda_l(y_i^{[l]}) N_i^{[l]}(x)$$

Newton fundamental polynomials (1)

An example:
interpolate

$$(x_2 - 3\sin x_1)^2 - x_1^2 + \frac{1}{100}(x_1 - 4)^4 + e^{2x_1 - 8}$$



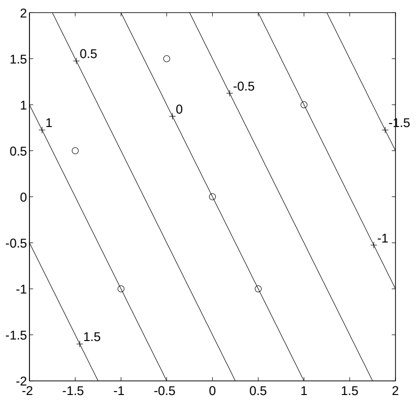
with the monomials basis on

$$Y^{[0]} = \{(0,0)\},$$

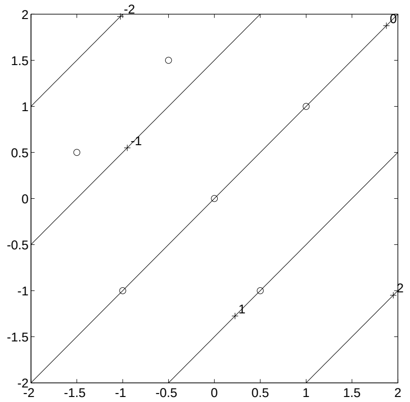
$$Y^{[1]} = \{(-1,-1),\ (\frac{1}{2},-1)\},$$

$$Y^{[2]} = \{(-\frac{1}{2},\frac{3}{2}),\ (1,1),\ (-\frac{3}{2},\frac{1}{2})\}.$$

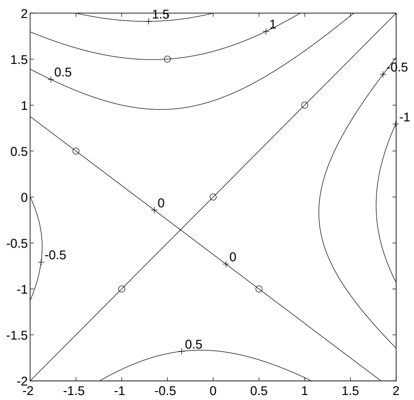
Newton fundamental polynomials (2)



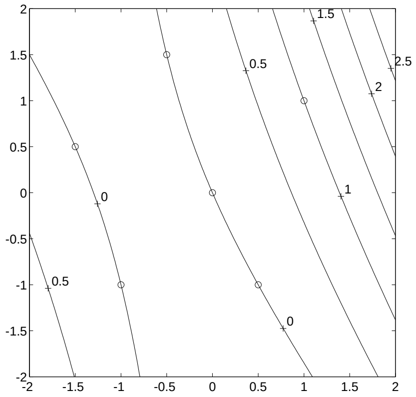
$N_1^{[1]}$



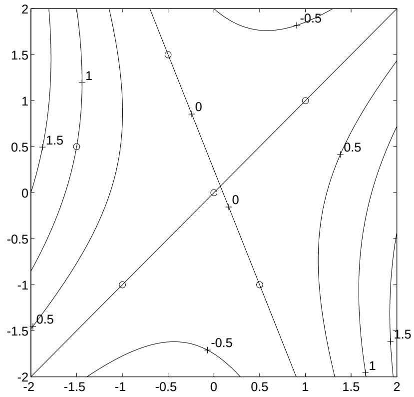
$N_2^{[1]}$



$N_1^{[2]}$



$N_2^{[2]}$



$N_3^{[2]}$

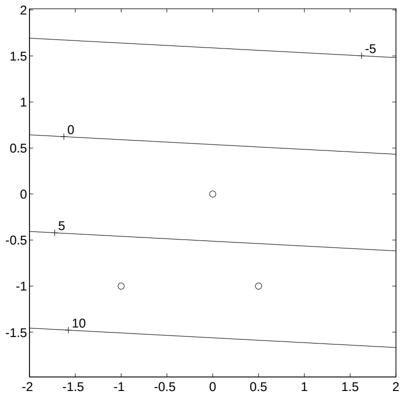
Newton fundamental polynomials (3)

$$m(x) = \sum_{y_i^{[\ell]} \in Y} \lambda_i^{[\ell]}(Y, f) N_i^{[\ell]}(x)$$

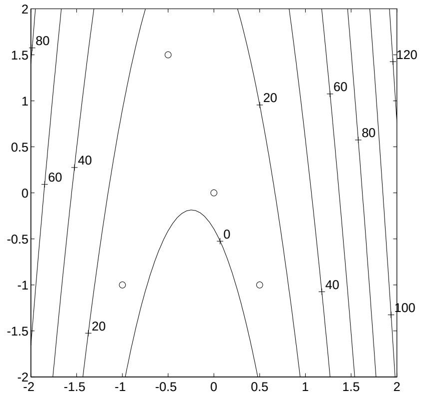
- $\lambda_i^{[\ell]}(Y, f)$ are generalized divided differences

- the sum may be incomplete if Y not poised: consider the subsets
  (1) $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1)\}$,
  (2) $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1),\ (-\frac{1}{2},\frac{3}{2})\}$,
  (3) $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1),\ (-\frac{1}{2},\frac{3}{2}),\ (1,1)\}$
  (4) $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1),\ (-\frac{1}{2},\frac{3}{2}), (\ 1,1),\ (-\frac{3}{2},\frac{1}{2})\}$.

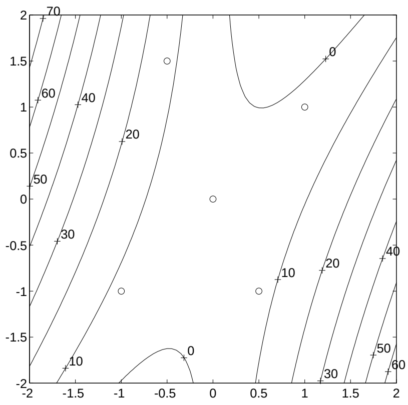
Newton fundamental polynomials (4)



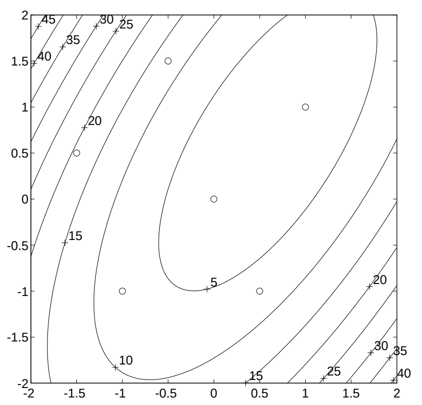
(1)



(2)



(3)



(4)

5. Application to constrained DFO (1)

Problem QP(x_k) :

$$\begin{aligned} &\text{min} \quad m_k(s) \\ &\text{s. t.} \quad c_k + A_k s = 0 \end{aligned}$$

Trust-region framework

to ensure global convergence

$$\Downarrow$$

Problem TRQP(x_k, δ_k) :

$$\begin{aligned} &\text{min} \quad m_k(s) \\ &\text{s. t.} \quad c_k + A_k s = 0 \\ &\text{and} \quad \|s\| \le \delta_k \end{aligned}$$

5. Application to constrained DFO (2)

- Step selection

  Idea : find a step $s_k = n_k + t_k$, where

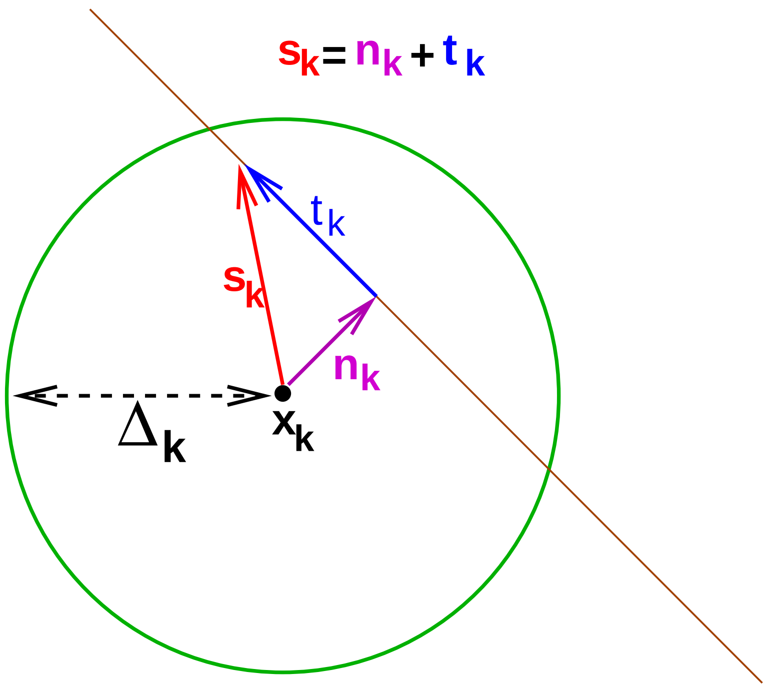
  - $n_k \equiv$ normal step
    $\rightarrow$ satisfies the linearized constraints
    (or sufficiently reduces infeasibility)

  - $t_k \equiv$ tangential step
    $\rightarrow$ reduces m_k
    (without deteriorating
    linear feasibility)

5. Application to constrained DFO (3)

Problem TRQP(x_k, δ_k)

$$\begin{array}{ll} \text{min}_s & m_k(s) \\ \text{s. t.} & c_k + A_k s = 0 \\ \text{and} & \|s\| \le \delta_k \end{array}$$



Trust-region methods (1)

(*e.g.* Conn, Gould, Toint, SIAM, 2000)

$$\min_{x \in \mathbb{R}^n} f(x)$$

k-th iteration:

Given $x^{(k)} \in \mathbb{R}^n$,
$\delta_k > 0$ (*trust-region radius*),
$m_k : \mathbb{R}^n \to \mathbb{R}$ (*model*),

1. Determine a vector $s^{(k)}$ with

$$\|s^{(k)}\| \leq \delta_k$$

and

$$m_k(s^{(k)}) < m_k(0).$$

If no such vector exists, then Stop.

Trust-region methods (2)

2. Compute

$$\rho_k := \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{m_k(x^{(k)}) - m_{(k)}(x^{(k)} + s^{(k)})} = \frac{\texttt{ared}}{\texttt{pred}}$$

3. If $\rho_k < \eta_1 < 1$, then

$$x^{(k+1)} := x^{(k)},$$
$$\delta_{k+1} := \texttt{Reduce}(\delta_k),$$

else if $\eta_1 \leq \rho_k < \eta_2 \quad (\eta_2 > 1)$ then

$$x^{(k+1)} := x^{(k)} + s^{(k)},$$
$$\delta_{k+1} := \delta_k,$$

else (*i.e.* $\rho_k \geq \eta 2$)

$$x^{(k+1)} := x^{(k)} + s^{(k)},$$
$$\delta_{k+1} := \texttt{Increase}(\delta_k).$$

4. Determine m_{k+1}.

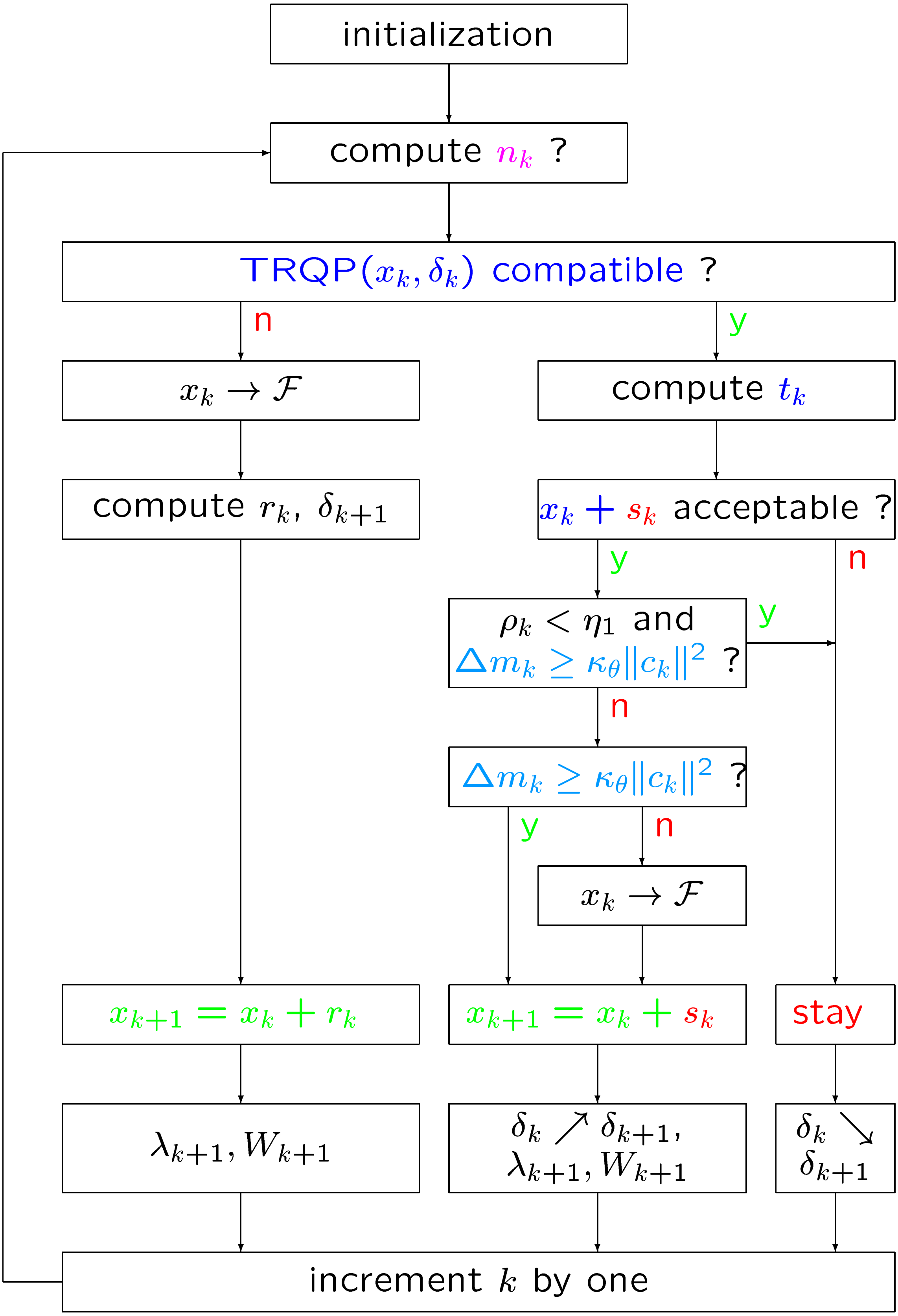
5. Application to constrained DFO (4)

If TRQP(x_k, δ_k) is not compatible :

- forget the objective function
- minimize $\|c(x)\|$

$\Rightarrow$ compute a *restoration step* r_k such that

- $x_k + r_k$ is non-dominated
- TRQP$(x_k + r_k, \delta_{k+1})$ is compatible


initialization
compute n_k ?
TRQP(x_k, δ_k) compatible ?
n
y
$x_k \to \mathcal{F}$
compute t_k
compute r_k, δ_{k+1}
$x_k + s_k$ acceptable ?
y
n
$\rho_k < \eta_1$ and $\Delta m_k \geq \kappa_\theta \|c_k\|^2$?
y
n
$\Delta m_k \geq \kappa_\theta \|c_k\|^2$?
y
n
$x_k \to \mathcal{F}$
$x_{k+1} = x_k + r_k$
$x_{k+1} = x_k + s_k$
stay
λ_{k+1}, W_{k+1}
$\delta_k \nearrow \delta_{k+1}$, λ_{k+1}, W_{k+1}
$\delta_k \searrow \delta_{k+1}$
increment k by one

5. Application to constrained DFO (5)

Resulting algorithm

Repeat

- use the available function values to build polynomial models m_k^f and $m_k^{c_i}$

- apply a filter algorithm yielding a new (potentially good) point

- update function and constraint values

until “convergence”

6. Practicalities (1)

1. Initial trust-region radius
   (1st iteration: $\Delta = \max\{1.0, 1.5\|n_1\|\}$)

2. Compute interpolation polynomials
   Current version:
   same set of interpolation points
   for approximating both f and c

3. Normal step (n_k) :

$$\min_{n} \frac{1}{2}\|c_k + A_k n\|^2$$

4. Lagrange multipliers

$$\nabla_{xx} f(x_k) - \sum_{i=1}^{m} [\lambda_k]_i \nabla_{xx} c_i(x_k)$$

$\Rightarrow$ least-squares (again!)

$$\min_{n} \frac{1}{2}\|\nabla_x f(x_k) - A_k^T \lambda_k\|^2$$

6. Practicalities (2)

5. Tangential step (t_k) :

(a)

$$t_k = N_k w,$$

where $\mathrm{Ker}(A_k) = \mathrm{Span}\{N_k(:,i)\}$
$(\Rightarrow A_k t_k = 0)$

(b) exact model minimization using Moré-Sorensen algorithm (DGQT)

$$\begin{array}{ll} \text{min} & b^T x + \frac{1}{2} x^T A x \\ \text{s. t.} & \|x\| \le \delta. \end{array}$$

$$\begin{array}{ll} \text{min} & \langle N_k^T (g_k + H_k n_k), w\rangle + \frac{1}{2}\langle w, N_k^T H_k N_k w\rangle \\ \text{s. t.} & \|w\| \le \sqrt{\Delta_k^2 - \|n_k\|^2}. \end{array}$$

6. Practicalities (3)

6. Restoration step (r_k) : $\min \|c(x)\|$

- unconstrained DFO algorithm (`UDFO`, sparse version available)

- Δ_k^c (input) governed by

$$\rho_k^c = \frac{\|c(x_k)\| - \|c(x_k + s_k)\|}{\|c(x_k)\| - \|c_m(x_k + s_k)\|}$$

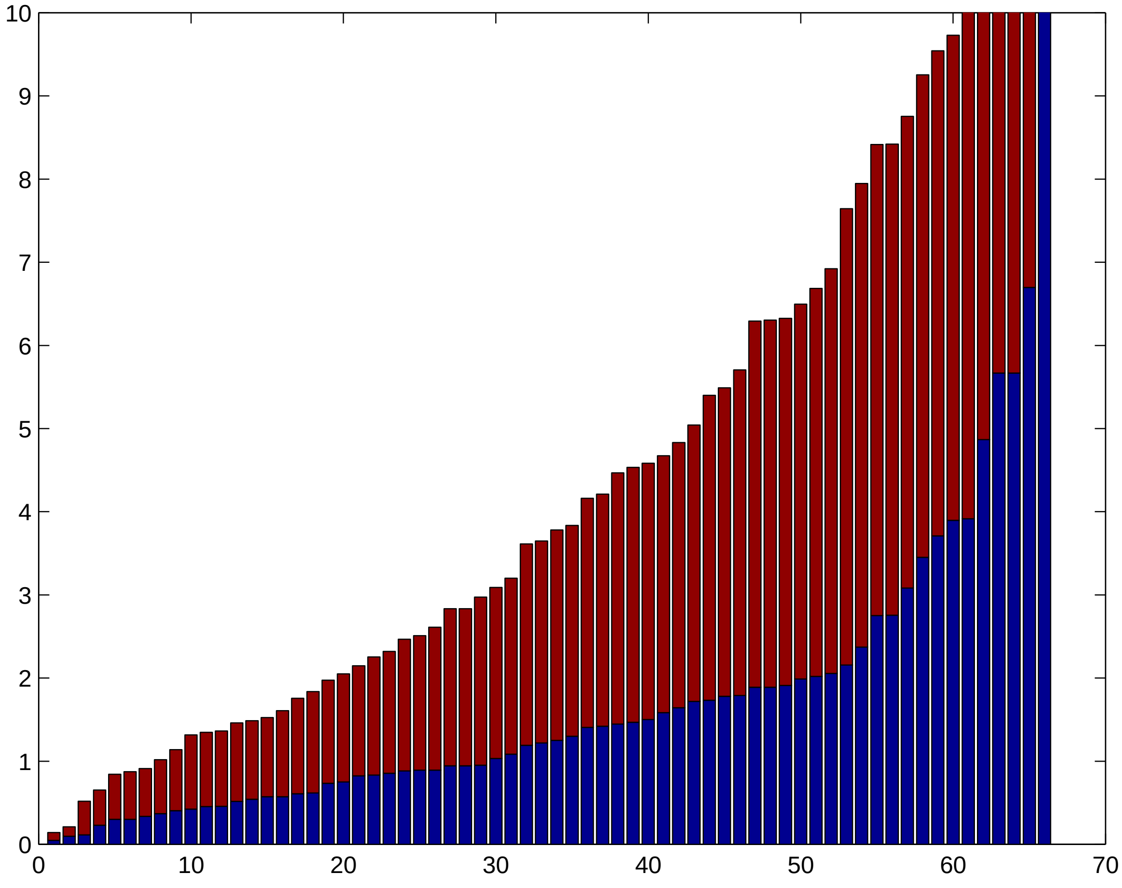
- lower bound on $\|c(x)\|$:
  10^{-3} is generally sufficient ($\Rightarrow$ less work!)

7. Computational experience (1)

- Code : `CDFO` (Fortran)
- Test problems : `CUTE`
  - 71 problems
  - $2 \leq n \leq 30$ variables
  - $m(\leq n)$ constraints
- Results :
  - conv. to optimal solution for 65 problems
  - conv. to a 'bad' solution for 2 problems
  - 4 problems with restoration process stalling
- Focus on function evaluations

7. Computational experience (2)

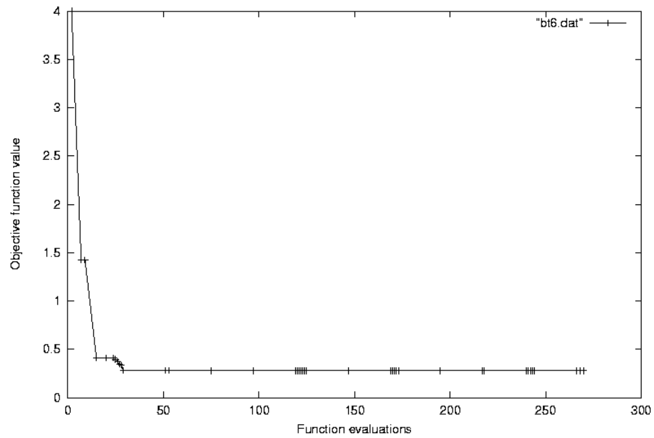
Comparison with LANCELOT (1/3):

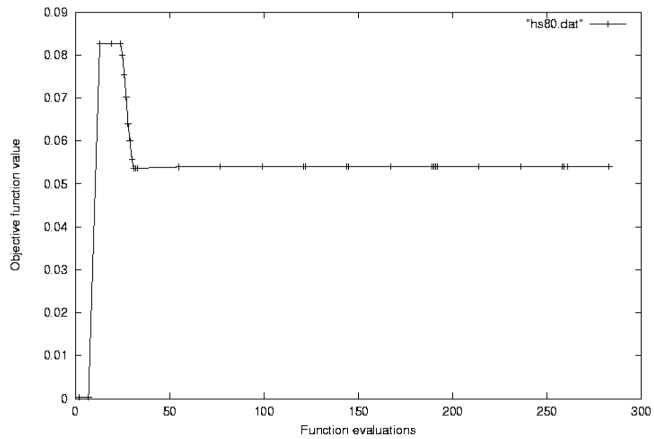


7. Computational experience (3)

Comparison with **LANCELOT** (2/3):

But ... two examples (**BT6** and **HS80**):

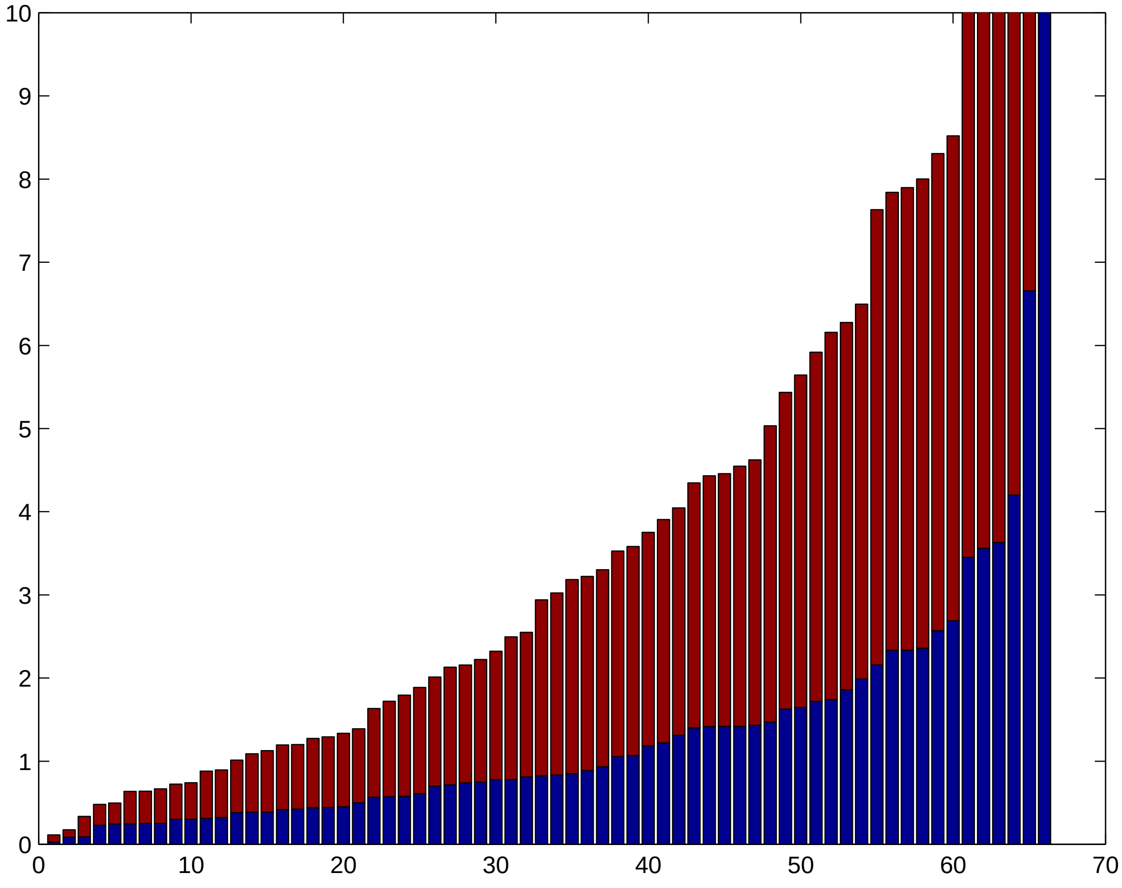




⇒ work is completed earlier!

7. Computational experience (4)

Comparison with LANCELOT (3/3):



7. Computational experience (5)

Other issues:

- Filter :
  - usually 'small' (usually 1-10 points) at the end of the process
  - impact of the small margin around the border?
- Restoration steps :
  - used for 60% test problems
  - may be time-consuming ... and involve function evaluations !

Future research

- convergence issues
- restoration steps
- introducing artificial sparsity
- partially separable functions