Derivative Free Optimization: Constrained and Unconstrained

Philippe Toint
(University of Namur, Belgium)
(with A. Conn, K, Scheinberg, B. Colson,
R. Fletcher, N. Gould, S.Leyffer)

Lunteren, January 2002

Outline

Part I
unconstrained problems

- The problem and some of its history

- The basis of interpolation methods
  - poisedness
  - geometry improvement
  - a trust-region algorithm

- Geometry and fundamental polynomials
  - Lagrange and Lagrange
  - the interpolation model
  - a glimpse of convergence theory

Outline (2)

- Introducing the sparsity structure
  - choosing the adequate approximation space
  - consequences on storage
  - consequences on iteration costs
- Numerical experience
  - the test functions
  - the results
- Partially separable functions
  - how do we know about sparsity?
  - algorithmic consequences

Outline (3)

Part II
constrained problems

- Sequential Quadratic Programming (SQP)
- The Filter Method
  - the filter
  - application to DFO
  - an algorithm
- Computational experience
- Future research

Part I

Unconstrained Derivative Free Optimization

The problem

Consider the unconstrained problem

$$\min_x f(x)$$

Gradient (and Hessian) of $f(x)$ unavailable but value compurable everywhere

- physical measurement
- object code
- typically small-scale

$\Rightarrow$ "Derivative free optimization" (DFO)

The problem (2)

$f(x)$ typically very costly

> Exploit each evaluation of $f(x)$
> to the utmost possible

considerable interest of practitioners

Possible approaches

- pattern search methods

  Box ('57), Campey-Nickols ('61),

  Hooke-Jeeves ('61), Spendley-Hext-Himsworth ('62),

  Nelder-Mead ('65), Dixon ('73) [ACSIM],

  Dennis-Torczon ('91) [PDS], Buckley ('94),

  Wright ('96), Torczon ('97), Dennis-Torczon ('97),

  Alexandrov et al. ('98), Audet ('99), . . .

- line search methods

  Powell ('64) [VA04, VA24], Brent ('73) [PRAXIS],

  Toint-Callier ('77), Lucidi-Sciandrone ('96), . . .

Possible approaches (2)

- finite-difference gradients + quasi-Newton

  Powell ('65), Stewart ('67), Powell ('70),

  Gill-Murray-Wright ('81), Dennis-Schnabel ('83),

  . . .

- interpolation methods

Interpolation methods for DFO

The idea:

Until "convergence":

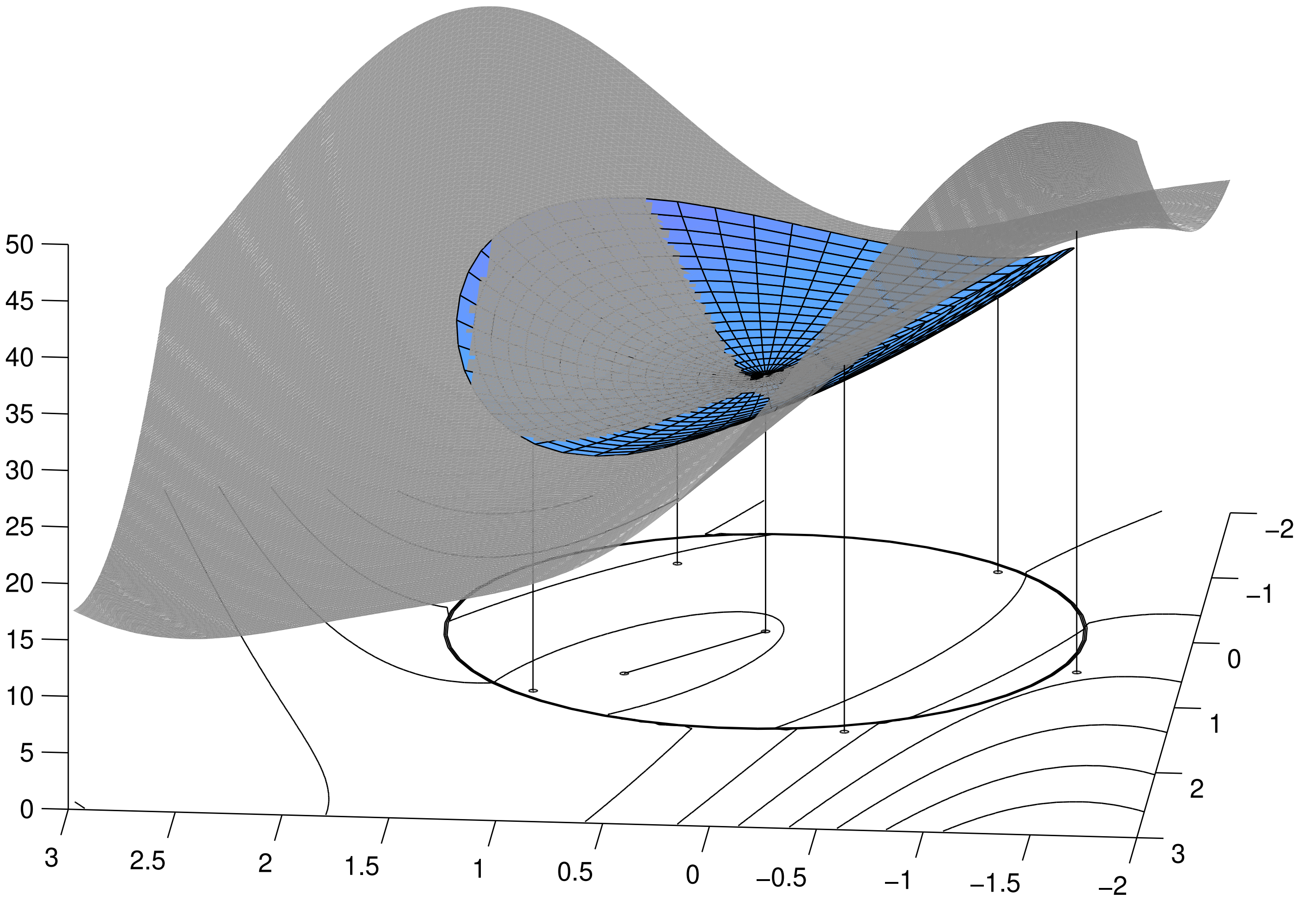
- Use the available function values to build a polynomial interpolation model m_k:

$$m_k(y_i) = f(y_i) \quad y_i \in Y;$$

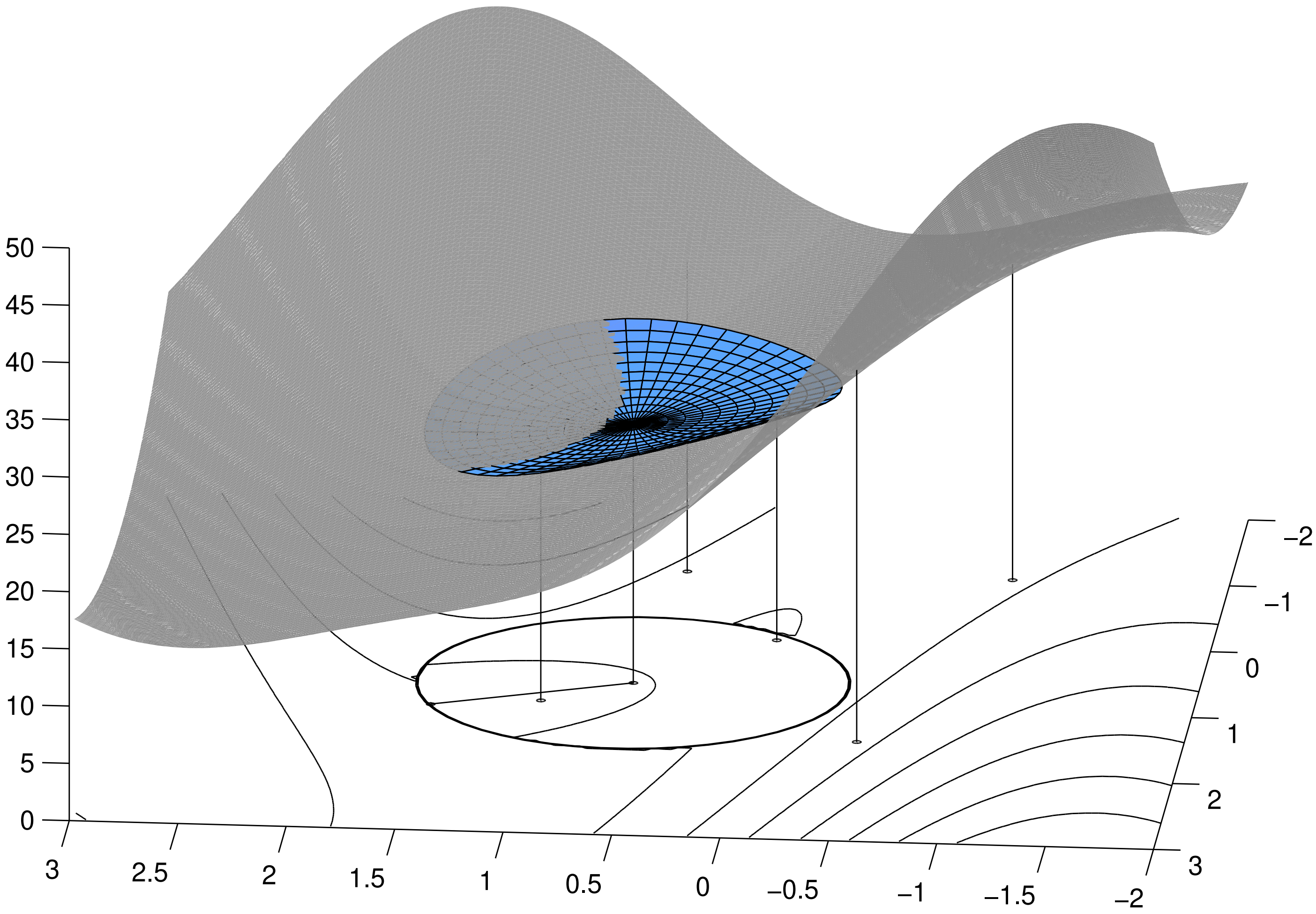
- Minimize the model in a "trust region", yielding a new potentially good point;
- Compute a new function value.

Y = "interpolation set"
$\subseteq$ { points y_i at which $f(y_i)$ is known }

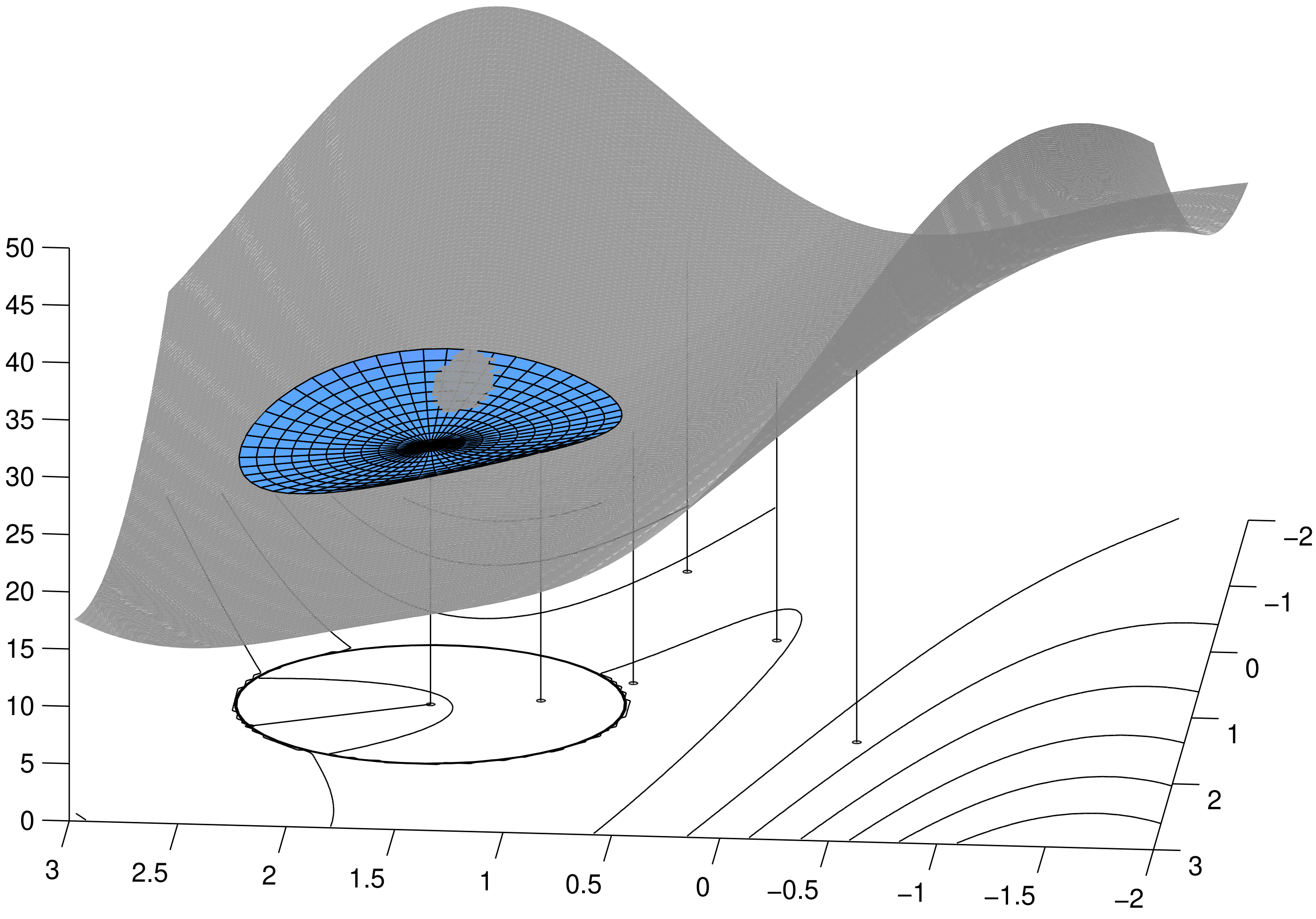
A trust-region method for DFO



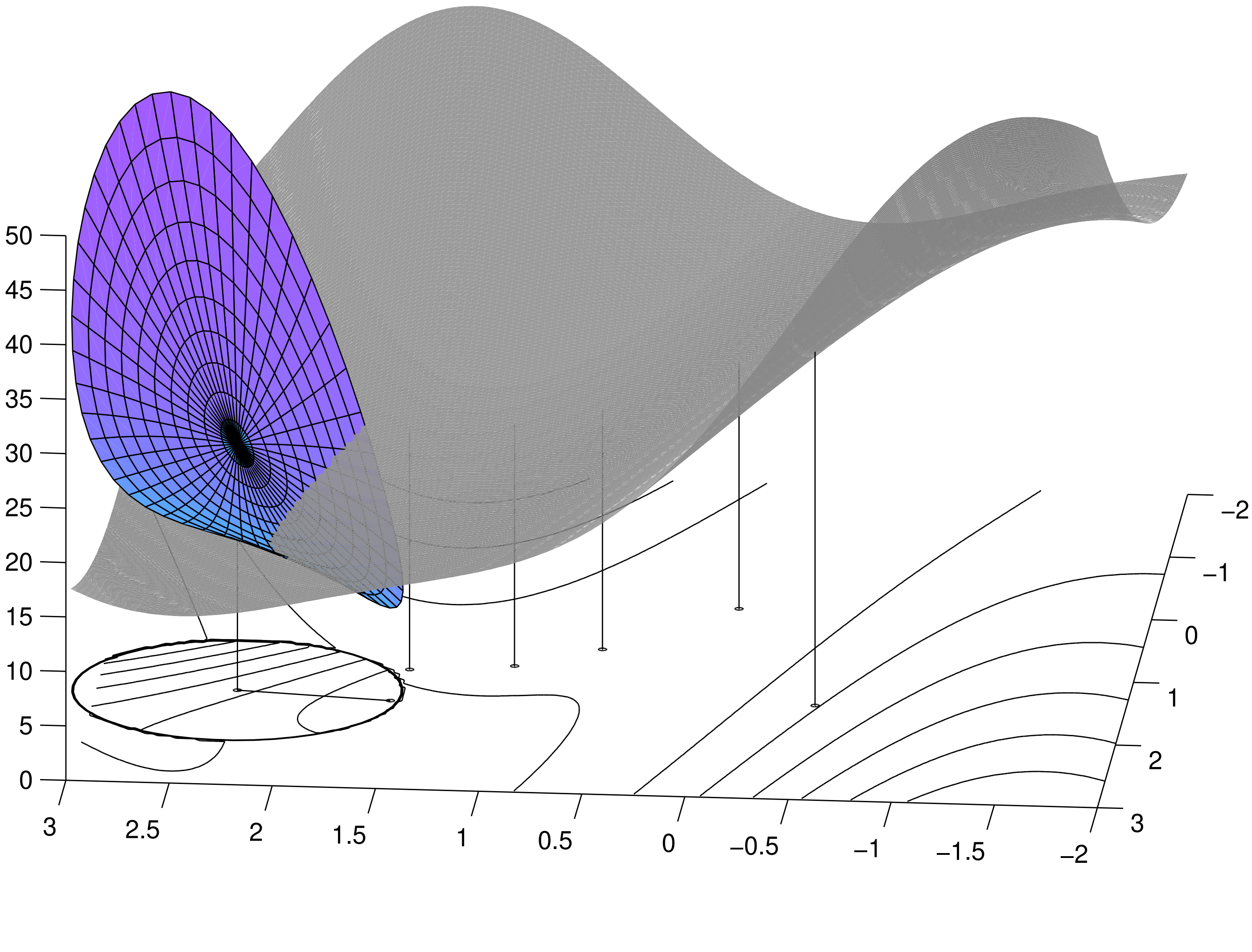
A trust-region method for DFO (2)



A trust-region method for DFO (3)



A trust-region method for DFO (4)



Interpolation methods for DFO (2)

To be considered:

- poisedness of the interpolation set Y
- choice of models
  - approximation subspace
  - choice of basis
- (convergence theory)
- numerical performance

Historical outline for interpolation methods

- early proposal:
  full quadratic models,
  poisedness ???, convergence ???
  Winfield ('69), Winfield ('73)

- revival:
  full or partial quadratic models,
  mechanism to ensure poisedness,
  proven convergence
  Powell ('94), Conn-Toint ('96), Powell ('96),
  Conn-Scheinberg-Toint ('97) ('98),
  Powell ('99), Powell ('00), Marazzi-Nocedal ('00),
  . . .

Poisedness

Assume a quadratic model

$$m_k(x_k + s) = f_k + \langle g_k, s\rangle + \tfrac{1}{2}\langle s, H_k s\rangle$$

Thus

$$p = 1 + n + \tfrac{1}{2}n(n+1) = \tfrac{1}{2}(n+1)(n+2)$$

parameters to determine
$\Rightarrow$ need p function values ($|Y| = p$)

not sufficient!

$\Rightarrow$ need geometric conditions
for the points in Y ...

Poisedness: geometry

Counter-examples ($n = 2, p = 6$)

- 6 points on a line
- 6 points on a circle
- 6 points on a quadratic curve

$\Rightarrow$ measure of poisedness for Y?

Poisedness: geometry (2)

If $\{\phi_i(\cdot)\}_{i=1}^{p}$ = basis for approximation space, then

$$\sum_{i=1}^{p} \alpha_i \phi_i(y_j) = f(y_j) \quad j = 1, \ldots, p$$

Possible measure

$$\delta(Y) = \det \begin{pmatrix} \phi_1(y_1) & \cdots & \phi_p(y_1) \\ \vdots & & \vdots \\ \phi_1(y_p) & \cdots & \phi_p(y_p) \end{pmatrix}$$

$$\boxed{Y \text{ (well) poised} \Leftrightarrow |\delta(Y)| \geq \epsilon}$$

- scale for the spread of the y_i's
- notion of geometry improvement

  $\Rightarrow$ more convenient measures

Other algorithmic ingredients

- include a new point in the interpolation set
  - need to drop an existing interpolation point?
  - select which one to drop:
    make Y "as poised as possible"

note:
model/function minimizer may produce bad geometry!!
$\Rightarrow$ geometry improvement procedure . . .

Other algorithmic ingredients (2)

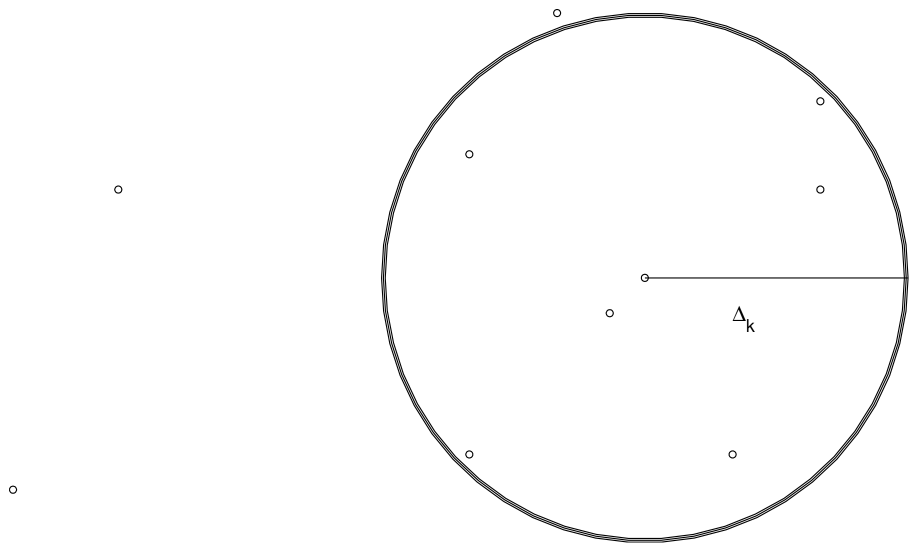
- trust-region radius management
  trust region:

$$\mathcal{B}_k = \{x_k + s \mid \|s\| \leq \Delta_k\}$$

  - classical: reduce Δ_k when "no progress"
  - DFO: more complicated!
    need to verify that Y is poised before reducing Δ_k

$\Rightarrow$ geometry improvement procedure ...

Improving the geometry in a ball



- attempt to reuse past points that are close to x_k

- attempt to replace a distant point of Y

- attempt to replace a close point of Y

if impossible:
good geometry given the current Δ_k

Trust-region algorithm for DFO (1)

criticality test: if g_k small, then
improve the geometry until Y is poised in a ball of radius $\delta \leq \mu \|g_k\|$, centered at x_k

subproblem: compute s_k from

$$\min_{s \in \mathcal{B}_k} m_k(x_k + s)$$

evaluation: compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$$

if $\rho_k \geq \eta_1$, define $\mathcal{X}_k = \{x_k + s_k\}$
else $\mathcal{X}_k = \{x_k\}$.

Trust-region algorithm for DFO (2)

model management:

if $\rho_k \geq \eta_1$, include $x_k + s_k$ in Y,
else if Y not poised, improve the geometry
(possibly enlarging $\mathcal{X}_k$)

next iterate:

compute

$$\widehat{x}_k = \arg\min_{x \in \mathcal{X}_k} f(x)$$

$$\widehat{\rho}_k = \frac{f(x_k) - f(\widehat{x}_k)}{m_k(x_k) - m_k(x_k + s_k)}$$

if $\widehat{\rho}_k \geq \eta_0$, accept $\widehat{x}_k$, $(x_{k+1} = \widehat{x}_k)$
else keep x_k $(x_{k+1} = x_k)$.

Trust-region algorithm for DFO (3)

radius update: (remember $\eta_0 < \eta_1$)

if $\widehat{\rho}_k \geq \eta_0$ or Y was poised

$$\hat{\Delta} = \Delta_k$$

if $\rho_k \geq \eta_1$

$$\Delta_{k+1} \geq \hat{\Delta}$$

else

$$\Delta_{k+1} \in [\gamma_1 \hat{\Delta}, \gamma_2 \hat{\Delta}]$$

Using the Lagrange fundamental polynomials

Remarkable: replace y_- by y_+ in Y:

$$\frac{\delta(Y_+)}{\delta(Y)} = L(y_+, y_-) \text{ is independent of the base!}$$

where

$$L(y, y_-) = \begin{cases} 1 & \text{if } y = y_- \\ 0 & \text{otherwise} \end{cases}$$

is the Lagrange fundamental polynomial

note: $L(y, y_-)$ is a quadratic polynomial

Using the Newton fundamental polynomials

Choosing a basis for the approximation space:
the Newton fundamental polynomials

- points in Y $\longleftrightarrow$
  Newton fundamental polynomials

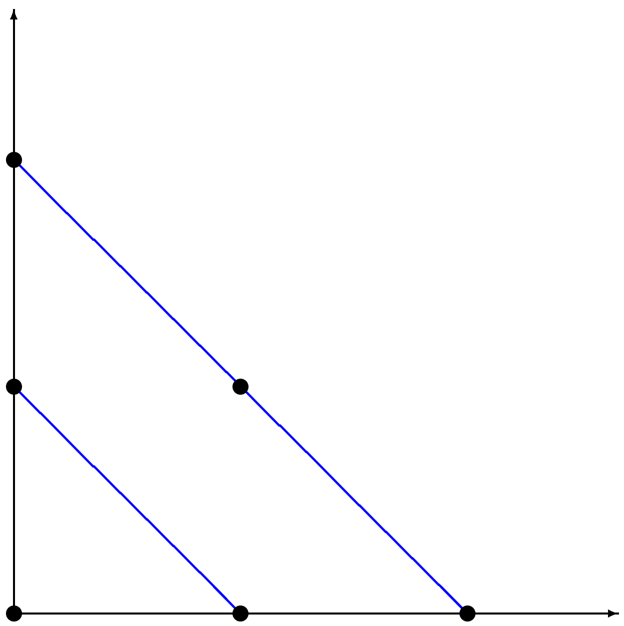
$$\boxed{y_j^{[k]} \in Y \longleftrightarrow N_i^{[\ell]}(x)}$$

- Y structured in $d+1$ blocks $Y^{[\ell]}$:
  containing polynomials of equal degree

$$\boxed{N_i^{[\ell]}(y_j^{[k]}) = \delta_{ij}\delta_{k\ell} \quad y_j^{[k]}, y_i^{[\ell]} \in Y, \quad k \leq \ell}$$

The Newton fundamental polynomials

Example: 2D interpolation on a regular grid:



$$Y^{[0]} = \{(0,0)\}, \quad Y^{[1]} = \{(1,0),(0,1)\},$$

$$Y^{[2]} = \{(2,0),(1,1),(0,2)\},$$

Initial basis of monomials :

$$\{1, \quad x_1, \quad x_2, \quad x_1^2, \quad x_1 x_2, \quad x_2^2\}$$

(for quadratic interpolation)

The Newton fundamental polynomials (2)

$N_j^{[k]}$ obtained by "orthogonalization" of the initial basis

$\Rightarrow$ pivots: $N_i^{[\ell]}$

$\boxed{Y \text{ not poised} \Leftrightarrow \text{zero pivot}}$

Here:

$$N_1^{[0]} = 1, \quad N_1^{[1]} = x_1, \quad N_2^{[1]} = x_2,$$

$$N_1^{[2]} = \tfrac{1}{2}(x_1^2 - x_1), \quad N_2^{[2]} = x_1 x_2, \quad N_3^{[2]} = \tfrac{1}{2}(x_2^2 - x_2),$$

The Newton fundamental polynomials (3)

Another example:
interpolate

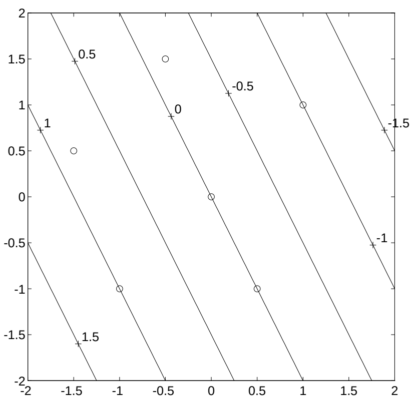
$$(x_2 - 3\sin x_1)^2 - x_1^2 + \frac{1}{100}(x_1 - 4)^4 + e^{2x_1 - 8}$$

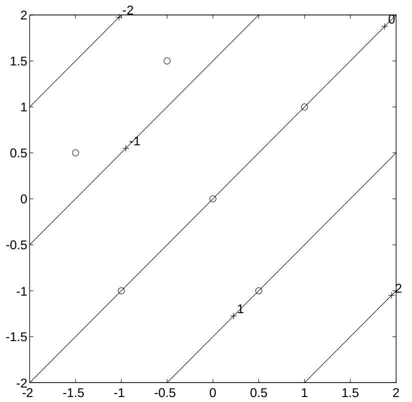
with the monomials basis on

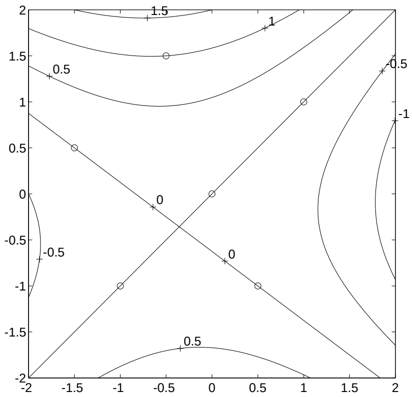
$$Y^{[0]} = \{(0,0)\},$$

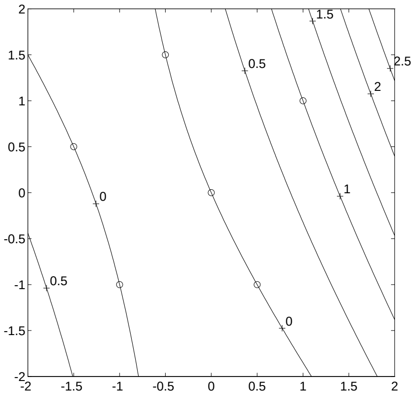
$$Y^{[1]} = \{(-1,-1),\ (\tfrac{1}{2},-1)\},$$

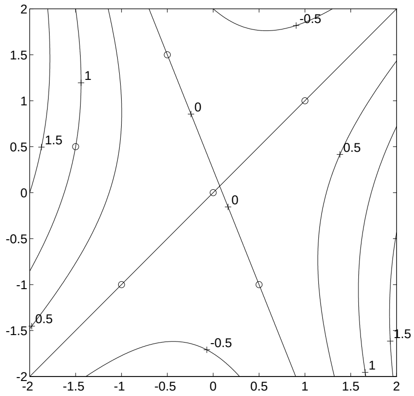
$$Y^{[2]} = \{(-\tfrac{1}{2},\tfrac{3}{2}),\ (1,1),\ (-\tfrac{3}{2},\tfrac{1}{2})\}.$$


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
0.5
-0.5
0
1
-1.5
-1
1.5


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
-2
0
-1
2
1


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
1.5
1
0.5
-0.5
-1
0
-0.5
0
0.5


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
1.5
0.5
2.5
2
0
1
0.5
0


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2
-0.5
1
0
1.5
0.5
0
0.5
-0.5
1.5
1

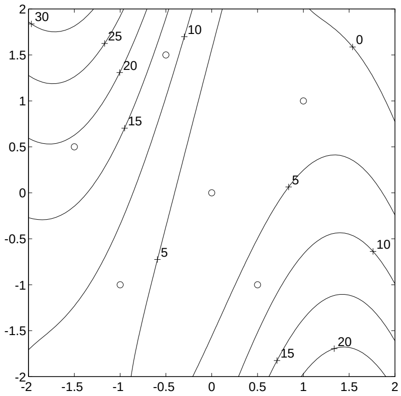
The Newton fundamental polynomials (5)

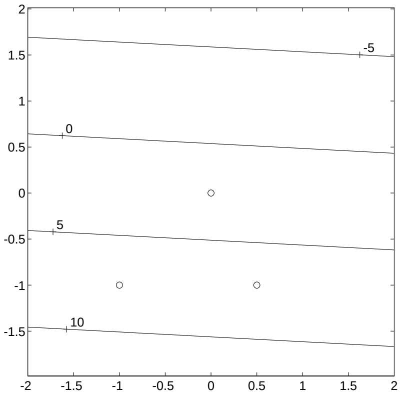
$$m(x) = \sum_{y_i^{[\ell]} \in Y} \lambda_i^{[\ell]}(Y, f) N_i^{[\ell]}(x)$$

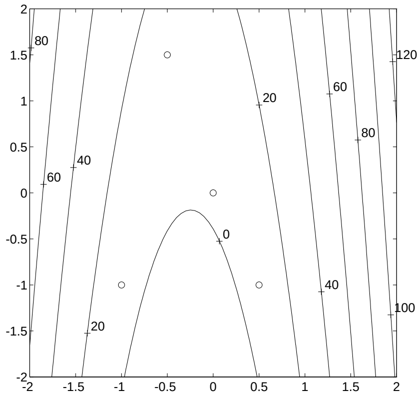
- $\lambda_i^{[\ell]}(Y, f)$ are generalized divided differences

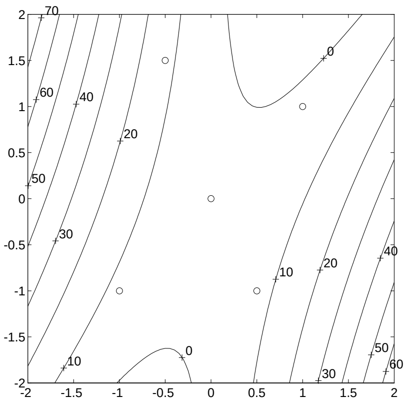
- the sum may be incomplete if Y not poised: consider the subsets
  $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1)\}$,
  $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1),\ (-\frac{1}{2},\frac{3}{2})\}$,
  $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1),\ (-\frac{1}{2},\frac{3}{2}),\ (1,1)\}$
  $\{(0,0),\ (-1,-1),\ (\frac{1}{2},-1),\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

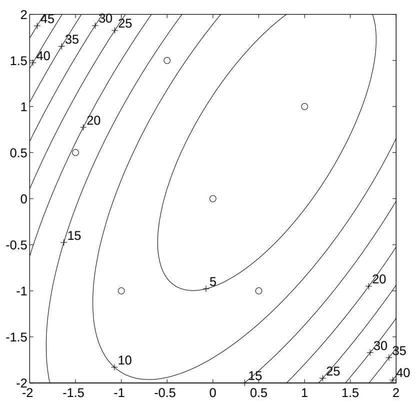
The Newton fundamental polynomials (6)











A glimpse of the convergence theory

Newton fundamental polynomials

$\Rightarrow$ Bound on the interpolation error

(Sauer-Xu ('95))

$$\boxed{|f(x) - m_k(x)| \le \kappa_2 \Delta^3 \max_{Y,x} |N_i^{[\ell]}(y_j^{[\ell+1]})|}$$

$\Rightarrow$ trust-region convergence theory!

model at least fully linear
interpolation set well poised
$\Rightarrow$ convergence to 1rst-order critical points

model at least fully quadratic
interpolation set well poised
$\Rightarrow$ convergence to 2nd-order critical points

Geometry and Newton polynomials

idea: use orthogonalization pivots

- Strategy for including y_+: replace

  $y_- = y_i^{[\ell]}$ associated with smallest pivot

  ($\Rightarrow$ decrease $|N_i^{[\ell]}|$ and thus $|N_i^{[\ell]}(y_j^{[\ell+1]})|$)

- strategy for improving the geometry:
  replace y_- by

  $$y_+ = \arg\max_{y \in \mathcal{B}} |N_i^{[\ell]}(y)|$$

  if $|N_i^{[\ell]}(y_+)| \geq 1 + \epsilon$
  (otherwise the geometry is fine)

Introducing sparsity

Assume $\nabla_{xx} f(x)$ is known to be $\boxed{\text{sparse}}$ (independently of x)

$\Rightarrow$ exists $\mathcal{S} = \{(i,j) | 1 \leq i,j \leq n\}$ (symmetric)

with coefficient of $x_i x_j$ identically zero in Taylor's expansion of f, i.e.

$$\forall x \quad \forall (i,j) \in \mathcal{S} \quad \langle e_i, \nabla_{xx} f(x) e_j \rangle = 0 \qquad (*)$$

Simple idea:

> reduce the approximation space to the set of quadratic polynomials satisfying (*)

Introducing sparsity (2)

$\Rightarrow N_j^{[k]}$ obtained by orthogonalization of a partial basis (excluding $x_i x_j$ for $(i,j) \in \mathcal{S}$)

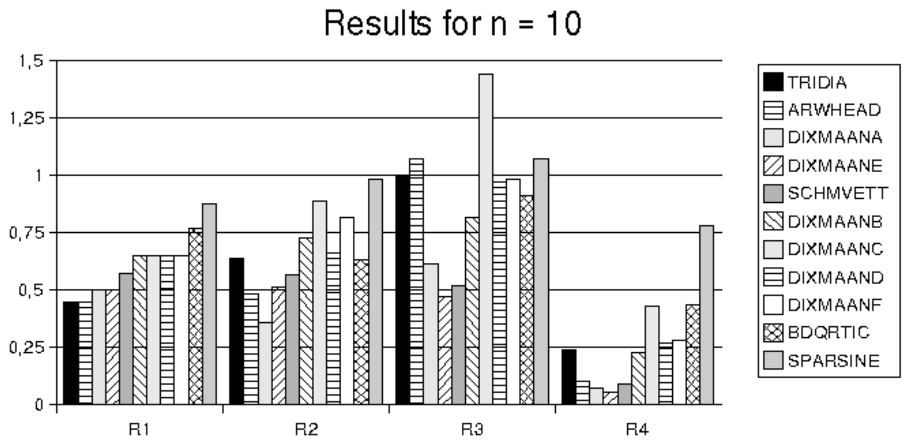
$\Rightarrow$ fewer interpolation points needed (Y smaller)

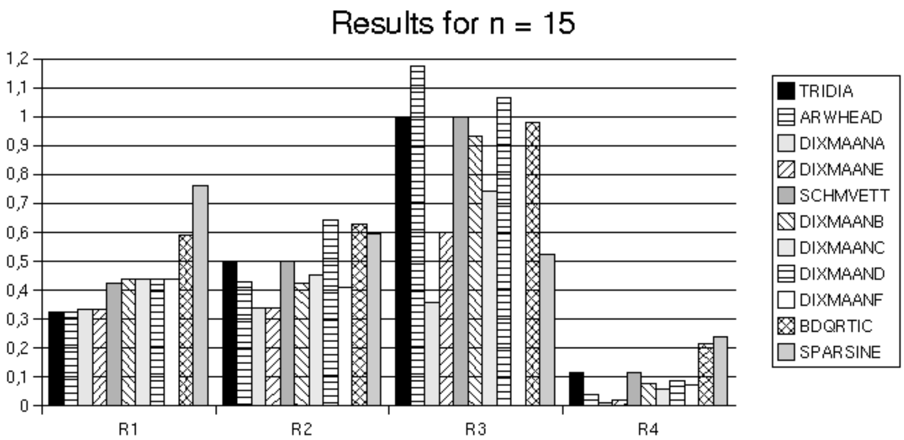
$$m_k(x_k + s) = \sum_{y_i^{[\ell]} \in Y} \lambda_i^{[\ell]}(Y, f) N_i^{[\ell]}(s)$$

Triple impact:

- less data points needed for complete interpolation
- less polynomials need being built and stored
- each polynomial smaller

Numerical experience for sparse problems





Some (unconstrained) conclusions

- very useful technique when sparsity is known
  - storage
  - speed
  - dominant parts of the Hessian can be used

- introduce "artificial sparsity" ?

Partially separable structure

Q: How do we know that $\nabla_{xx} f(x)$ is sparse?
A: (Nearly) always when

$$f(x) = \sum_{i=1}^{p} f_i(\tilde{x}_i)$$

where $\tilde{x}_i$ is a small subset of the variables.

$$\boxed{f \text{ is partially separable}}$$

$\tilde{x}_i$ defines dense principal submatrices in the Hessian.

Example:

$$f(x_1, \ldots, x_n) =$$

$$f_1(x_1, x_2) + f_2(x_2, x_3) + \cdots + f_{n-1}(x_{n-1}, x_n)$$

gives a tridiagonal Hessian

Structure often know from application!

Algorithmic consequences

Main idea:
build an interpolation model for each $f_i(\tilde{x}_i)$!

Some issues:

- p different interpolation sets
- use maximal decomposition
  (no $\tilde{x}_i$ included in another)
- geometry improvement may be local
- evaluation method for f may be
  - for individual f_i
  - for all f_i at once
    $\Rightarrow$ use CPR grouping of partial list!

$\Rightarrow$ substantially more complex data structures,
but algorithm eseentially similar

Efficiency

$\Rightarrow$ much fewer evaluations are needed to build a complete model!

Example: for a trididiagonal quadratic

Dense	$(n+1)(n+2)/2$
Sparse	$3n$
Partially separable	6

Numerical experience in progress . . .

. . . more next time!

Part II

Constrained Derivative Free Optimization using the Filter Method

The problem

$$\begin{array}{ll} \min & f(x) \\ \text{s. t.} & c(x) = 0 \end{array}$$

(m equality constraints)

where

- f and c are nonlinear
- f and c are smooth

Sequential quadratic programming

(for general constrained problems)

- Basic idea

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

$$\Updownarrow$$

Iterativeley solve

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

where

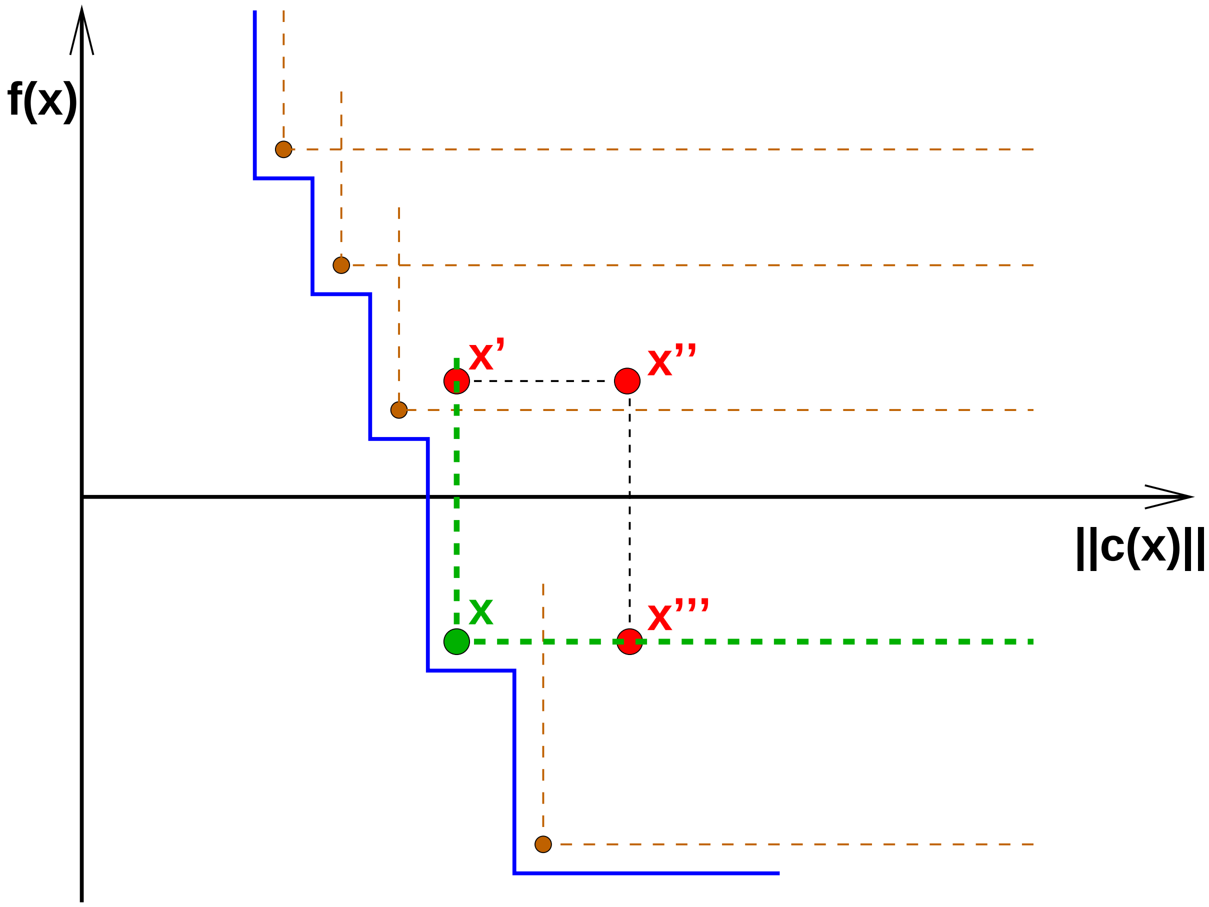
m_k	$\equiv$ *quadratic* model of f at x_k
s	$\equiv$ improvement step ($\Rightarrow$ *sequential*)
c_k	$\equiv c(x_k)$
A_k	$\equiv$ Jacobian of c at x_k

The Filter Method

For equality constraints:

x' is *dominated* by x when

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



Application to constrained DFO (1)

Problem QP(x_k) :

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

Trust-region framework

to ensure global convergence

$$\Downarrow$$

Problem TRQP(x_k, δ_k) :

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

Application to constrained DFO (2)

- Step selection

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

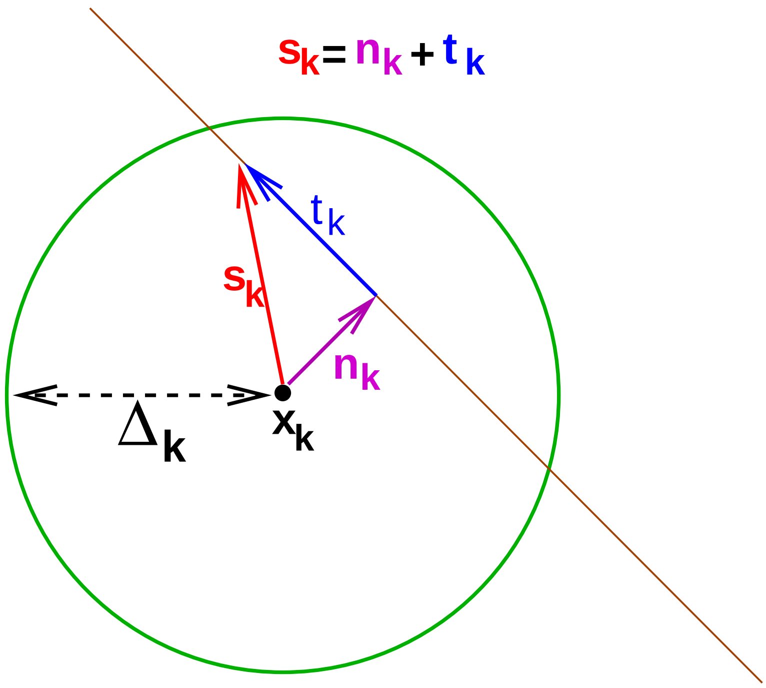
  - $n_k \equiv$ normal step
    $\rightarrow$ satisfies the linearized constraints
    (or sufficiently reduces infeasibility)

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

Application to constrained DFO (3)

Problem TRQP(x_k, δ_k)

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



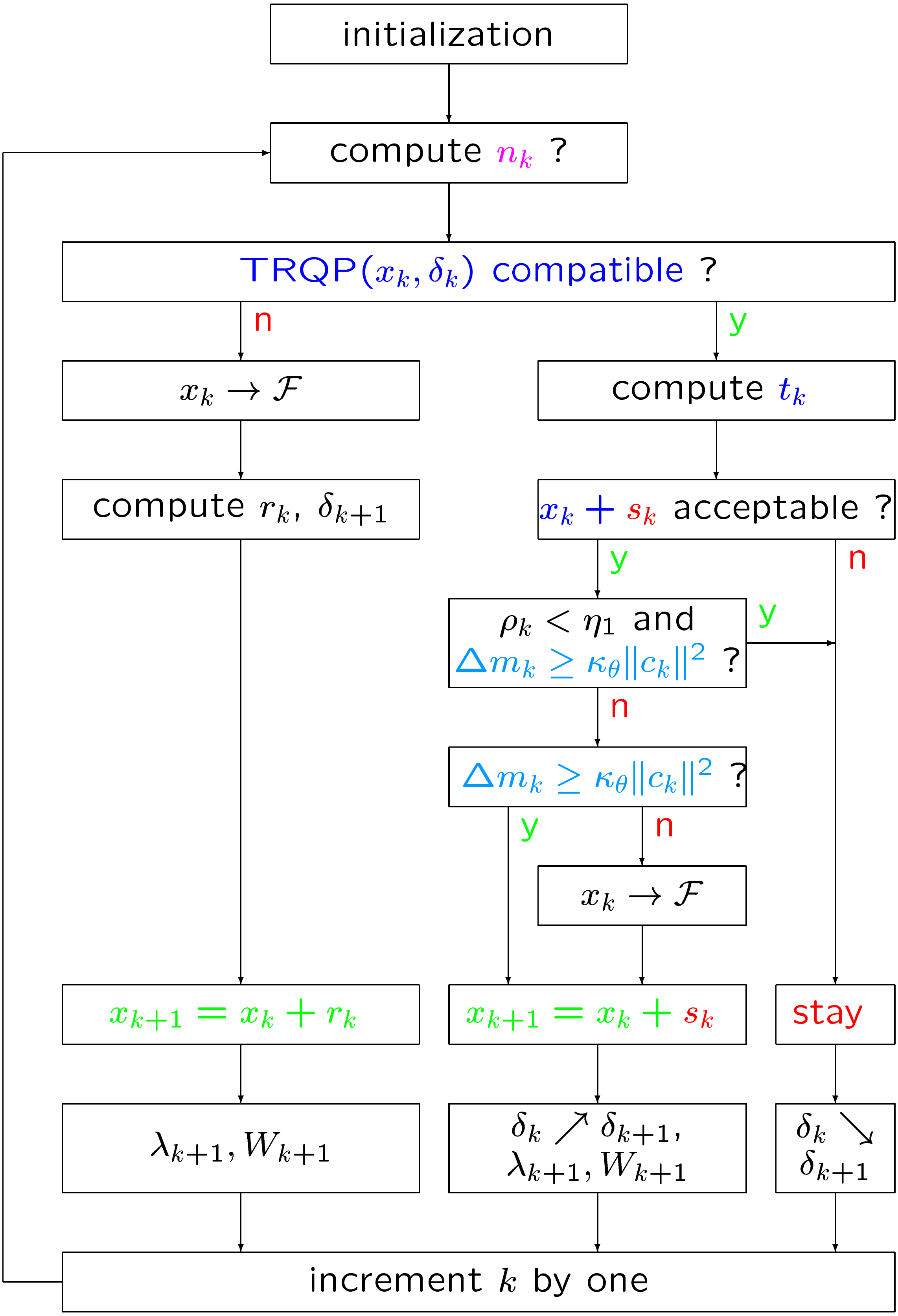
Application to constrained DFO (4)

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}
$\delta_k \searrow$
δ_{k+1}
increment k by one

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"

DFO Filter Algorithm (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!)

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

DFO Filter Algorithm (2)

5. Tangential step (t_k) :

   a.

$$t_k = N_k w,$$

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

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

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

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

DFO Filter Algorithm (3)

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

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

- Δ_k^c (input) governed by

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

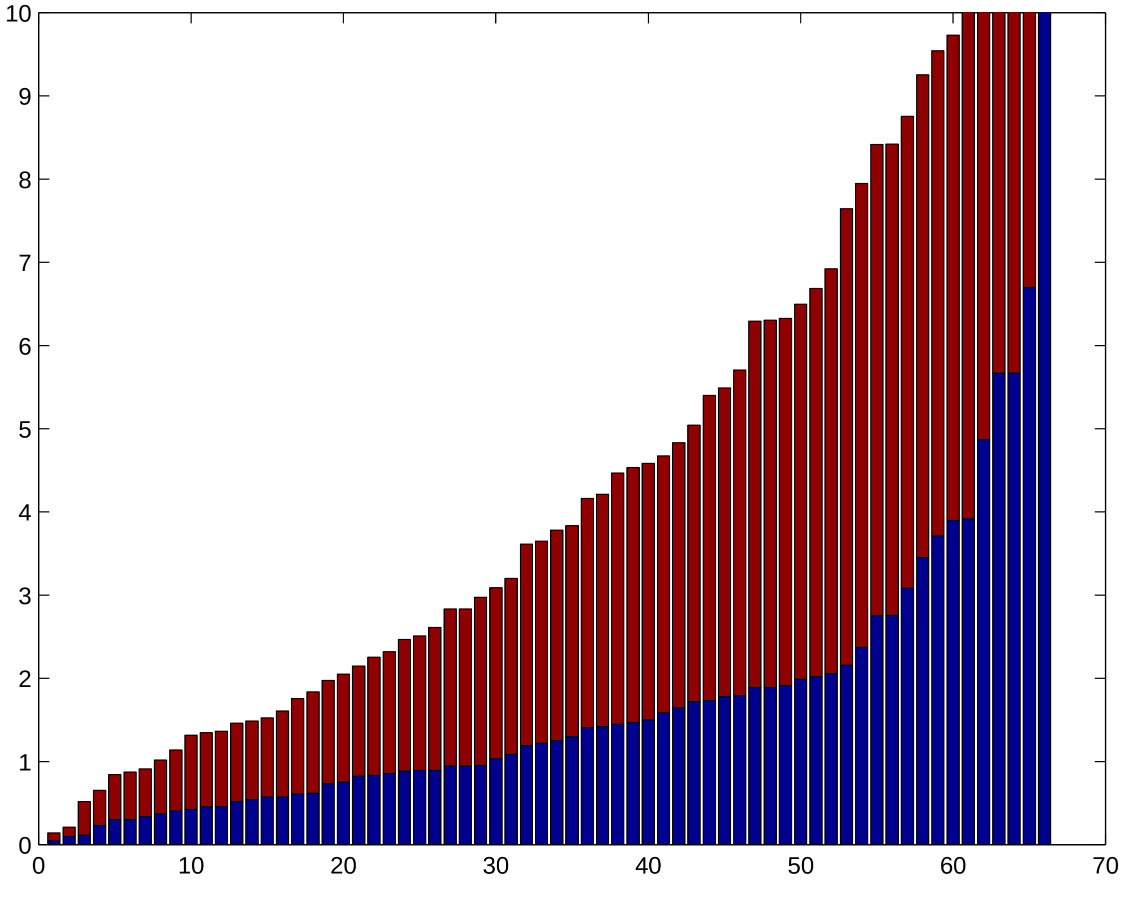
- lower bound on $\|c(x)\|$:
10^{-3} is generally sufficient ($\Rightarrow$ less work!)

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

Computational experience (2)

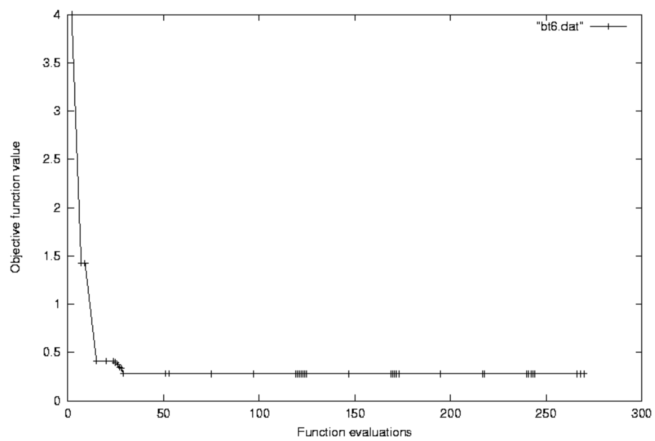
Comparison with LANCELOT (1/3):

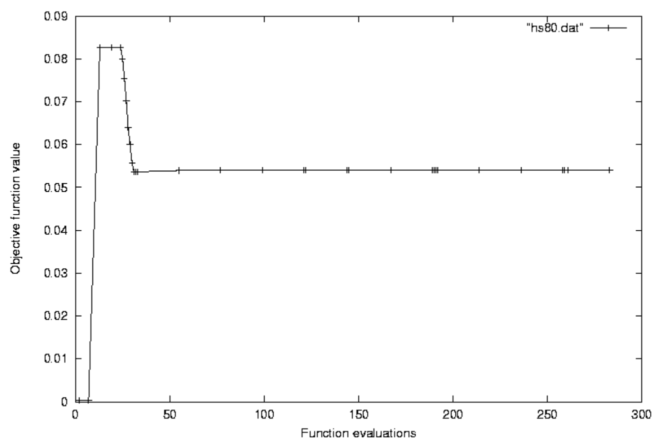


Computational experience (3)

Comparison with `LANCELOT` (2/3):

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

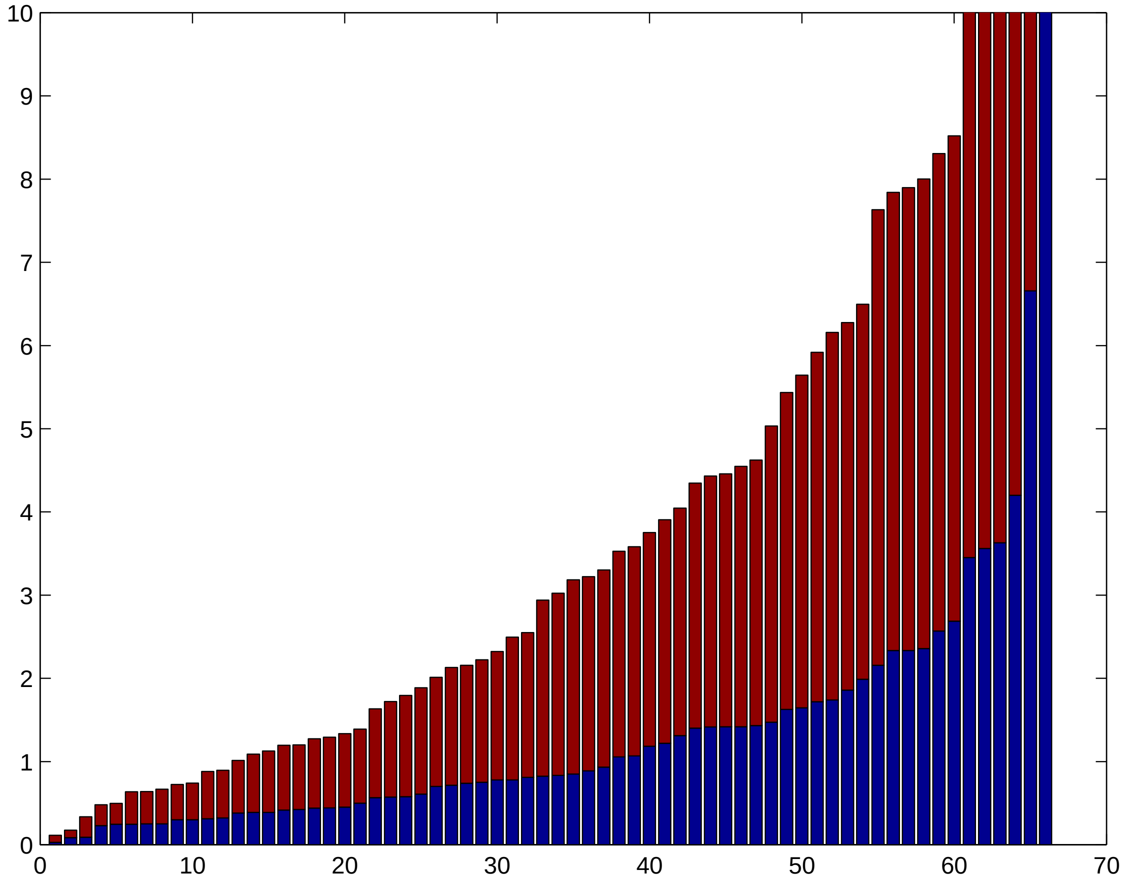




$\Rightarrow$ work is completed earlier!

Computational experience (4)

Comparison with `LANCELOT` (3/3):



Computational experience (5)

Other issues:

- Filter :
  - usually 'small' (usually 1-10 points) at the end of the process

- Restoration steps :
  - used for 60% of the test problems
  - may be time-consuming ... and involve constraint evaluations !

Future research for DFO Filter

- convergence issues
- restoration steps
- ...

Presolving for linear and quadratic programming

Ph. Toint
(University of Namur, Belgium)

(with N. Gould)

Lunteren, January 2002

The software context

- a successor to LANCELOT:
  $\longrightarrow$ $\boxed{\text{GALAHAD}}$

- necessity to handle linear constraints better (than LANCELOT)

- availability of efficient large-scale QP solvers (interior and active-set based)

- good experience of colleagues (packages by Gill, Nocedal, . . .)

$\Rightarrow$ Sequential Quadratic Programming

(see above!)

GALAHAD general features (1)

Our current intentions:

- general smooth (un)constrained nonconvex optimization
  + nonlinear equations
  and nonlinear least-squares

- SQP approach ($S\ell_1QP$ or filter)

- SIF (CUTE) and AMPL input format

- code aimed at both "small-dense" and "large-sparse" problems

- coordinate or sparse row-wise storage

GALAHAD general features (2)

- includes preprocessing of QPs for
  - analysis of linear constraints
  - a priori variable reordering
    for efficient QP solution
    (fast matrix-vector products, . . .)

Of course, . . .

QP vs LP preprocessing (1)

The problem:

$$\min_x f + g^T x + \tfrac{1}{2} x^T H x$$

subject to

$$x_l \leq x \leq x_u \quad \rightarrow z \text{ dual variables}$$

$$c_l \leq Ax \leq c_u \quad \rightarrow y \text{ multipliers}$$

but also (when available)

$$z_l \leq z \leq z_u$$

$$y_l \leq y \leq y_u$$

- bounds may be $-\infty$ or ∞
- lower and upper bounds may be equal

QP vs LP preprocessing (2)

(Local) optimality conditions:

$$g + Hx_* - A^T y_* - z_* = 0$$

$$x_l \leq x_* \leq x_u$$

$$c_l \leq Ax_* \leq c_u$$

$$[z_*]_j \begin{cases} \geq 0 & \text{when } [x_{l,*}]_j = [x_*]_j \\ = 0 & \text{when } [x_{l,*}]_j < [x_*]_j < [x_{u,*}]_j \\ \leq 0 & \text{when } [x_*]_j = [x_{u,*}]_j \end{cases}$$

$$[y_*]_i \begin{cases} \geq 0 & \text{when } [c_{l,*}]_i = [Ax_*]_i \\ = 0 & \text{when } [c_{l,*}]_i < [Ax_*]_i < [c_{u,*}]_i \\ \leq 0 & \text{when } [Ax_*]_i = [c_{u,*}]_i \end{cases}$$

$\Rightarrow$ preprocessing depends on H

QP vs LP preprocessing (3)

- much in common with LP case (Andersen and Andersen, Gondzio, . . .)
  - implications between variables and bounds
  - analysis of primal constraints
    - empty, singleton, doubleton and linearly dependent rows
    - constraint bound tightening/fixing
    - row sparsification

QP vs LP preprocessing (4)

- but substantial differences too
  - linearly unconstrained variables (linear, quadratic)
  - analysis of dual constraints
    - empty and singleton columns
    - occurence of Hx in implied bounds
    - tightening and fixing of primal, dual variables and multipliers mixed up

An example: nonlinear singleton columns

Singleton column in A corresponding to a diagonal column in H

$$g_j + H_{jj}x_j - A_{ij}y_i = z_j$$

If $[x_u]_j = +\infty$, then $z_j \geq 0$

$$g_j + H_{jj}x_j - A_{ij}y_i \geq 0$$

for x_j	$A_{ij} > 0$	$A_{ij} < 0$
$H_{jj} > 0$	$x_j \geq -\dfrac{g_j - A_{ij}[y_l]_i}{H_{jj}}$	$x_j \geq -\dfrac{g_j - A_{ij}[y_u]_i}{H_{jj}}$
$H_{jj} < 0$	$x_j \leq -\dfrac{g_j - A_{ij}[y_l]_i}{H_{jj}}$	$x_j \leq -\dfrac{g_j - A_{ij}[y_u]_i}{H_{jj}}$

for y_i	$A_{ij} > 0$	$A_{ij} < 0$
$H_{jj} > 0$	$y_i \leq \dfrac{g_j + H_{jj}[x_u]_j}{A_{ij}}$	$y_i \geq \dfrac{g_j + H_{jj}[x_u]_j}{A_{ij}}$
$H_{jj} < 0$	$y_i \leq \dfrac{g_j + H_{jj}[x_l]_j}{A_{ij}}$	$y_i \geq \dfrac{g_j + H_{jj}[x_l]_j}{A_{ij}}$

+ symmetric bounds when $[x_l]_j = -\infty$ ...

Numerical experience (1)

- 178 problems (NETLIB + CUTE)
  - 8 purely quadratic (unconstrained)
  - 70 quadratic
  - 100 linear

Summary of results:

- very marginal processing time
- average problem reduction:

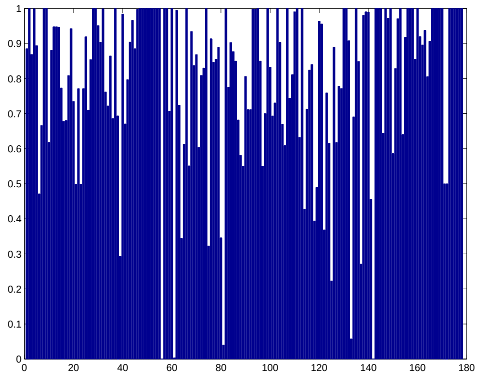
n: 19%	m: 21%
$\|A\|$: 21%	$\|H\|$: 25%

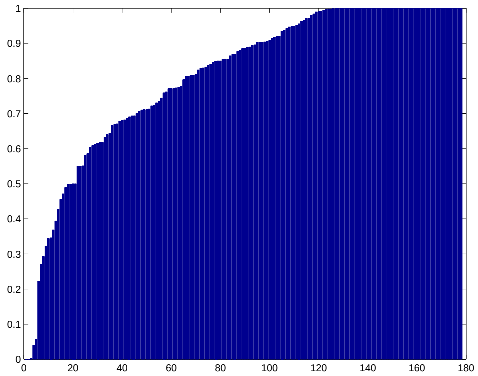
- average gain on solution time : 10%

Numerical experience (2)

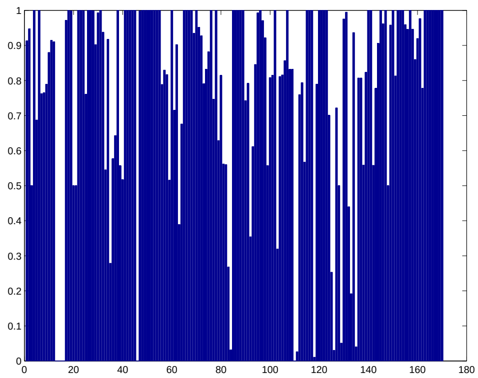
- problem size

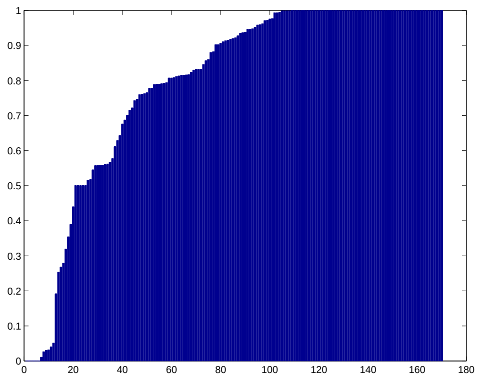
number of variables





number of constraints

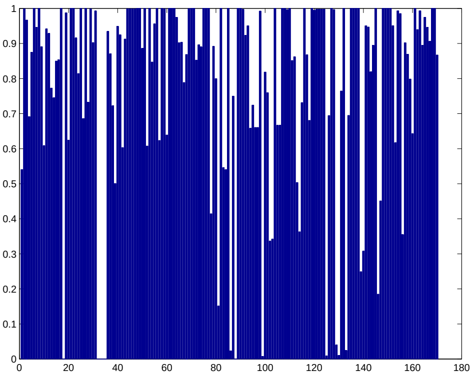


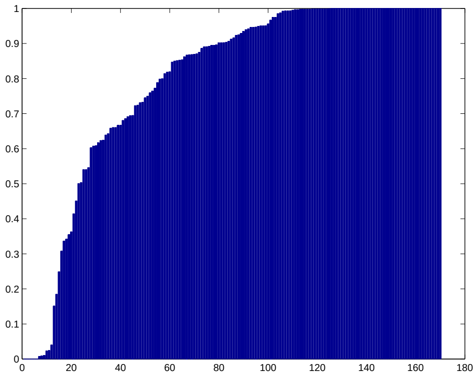


Numerical experience (3)

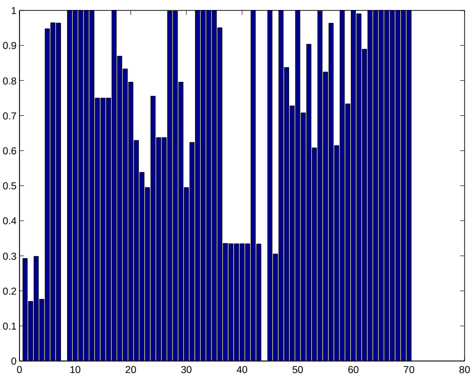
- problem size

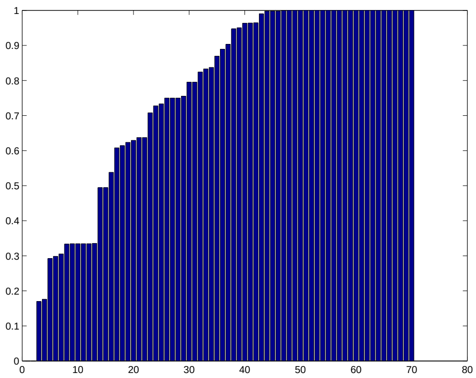
number of nonzeros in A





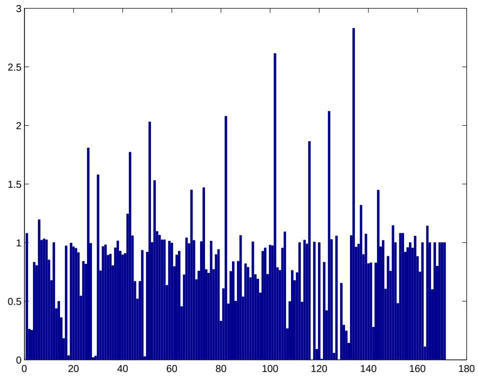
number of nonzeros in H

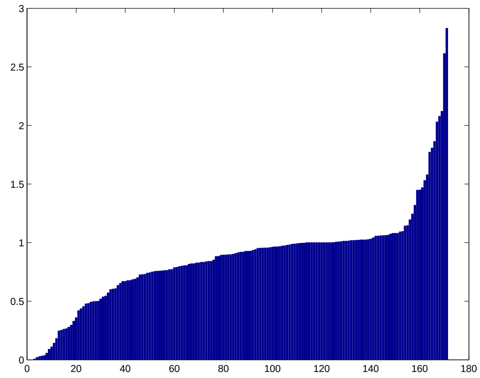




Numerical experience (4)

- reduction of the solution time





Some conclusions

- significant gain in efficiency (size, ordering)

- useful mostly on OR applications

- strong influence of sparsity pattern of H

Thank you for your attention