Use of approximate derivatives in the filter-trust-region method for nonlinear unconstrained optimization

Caroline Sainvitu[1] Philippe L. Toint [1]

[1]Department of Mathematics, FUNDP, Namur, Belgium

caroline.sainvitu@fundp.ac.be philippe.toint@fundp.ac.be

Sparse Days Meeting 2006 at CERFACS, June 15th-16th

Outline

1. Introduction to filter methods
2. Unconstrained optimization
   - Multidimensional filter
   - Algorithm
   - Approximate derivatives
3. Numerical results
   - Framework of the tests
   - Finite-difference
   - Secant update
   - Comparison
4. Conclusion

Outline

1. Introduction to filter methods
2. Unconstrained optimization
   - Multidimensional filter
   - Algorithm
   - Approximate derivatives
3. Numerical results
   - Framework of the tests
   - Finite-difference
   - Secant update
   - Comparison
4. Conclusion

Outline

1. Introduction to filter methods
2. Unconstrained optimization
   - Multidimensional filter
   - Algorithm
   - Approximate derivatives
3. Numerical results
   - Framework of the tests
   - Finite-difference
   - Secant update
   - Comparison
4. Conclusion

Outline

1. Introduction to filter methods
2. Unconstrained optimization
   - Multidimensional filter
   - Algorithm
   - Approximate derivatives
3. Numerical results
   - Framework of the tests
   - Finite-difference
   - Secant update
   - Comparison
4. Conclusion

Introduction to filter methods

Constrained optimization

use the Sequential Quadratic Programming step s_k from x_k

Ideal

- reduce the objective function $f(x)$
- reduce the constraint violation $\theta(x)$

☺ two potentially conflicting aims ☹

▶ Filter method

Introduction to filter methods

Constrained optimization

use the Sequential Quadratic Programming step s_k from x_k

Ideal

- reduce the objective function $f(x)$
- reduce the constraint violation $\theta(x)$

☺ two potentially conflicting aims ☹

▶ Filter method

Introduction to filter methods

Idea of Fletcher and Leyffer

Replace the question

What is a better point ?

by

What is a worse point ?

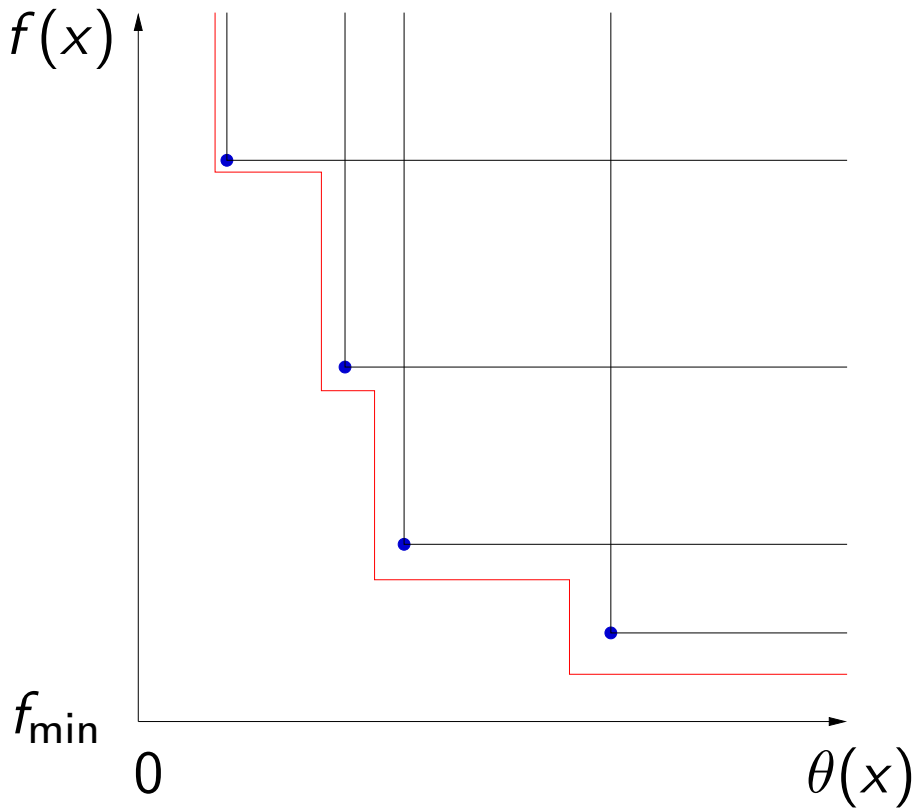
Of course, y is "worse" than x if it is dominated by x, i.e., when

$$f(x) \leq f(y) \quad \text{and} \quad \theta(x) \leq \theta(y)$$

Accept or reject a trial point ?

Introduction to filter methods

Idea: accept non-dominated points



Introduction to filter methods

Gould, Leyffer and Toint

Feasibility

Find x such that

$$c_{\mathcal{E}}(x) = 0 \quad \text{and} \quad c_{\mathcal{I}}(x) \geq 0$$

Least-squares

Find x such that

$$\min_{x \in \mathbb{R}^n} \sum_{i \in \mathcal{E} \cup \mathcal{I}} \theta_i(x)^2$$

▶ More dimensions in the filter space . . .

Introduction to filter methods

Gould, Leyffer and Toint

Feasibility

Find x such that

$$c_{\mathcal{E}}(x) = 0 \quad \text{and} \quad c_{\mathcal{I}}(x) \geq 0$$

Least-squares

Find x such that

$$\min_{x \in \mathbb{R}^n} \sum_{i \in \mathcal{E} \cup \mathcal{I}} \theta_i(x)^2$$

- More dimensions in the filter space ...

Introduction to filter methods

Gould, Sainvitu and Toint

Unconstrained optimization

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

Simple-bound constrained optimization

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & l \leq x \leq u \end{array}$$

Combined methods:

Trust-region + filter + projected gradient

- Multidimensional filter

Multidimensional filter

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

- Newton's method $\longrightarrow \min_s \quad f(x_k) + \nabla_x f(x_k)^T s + \frac{1}{2} s^T H_k s$
- Trust-region method
- Filter technique

Idea: encourage convergence to first-order critical points by driving every component of the objective's gradient

$$\nabla_x f(x) \stackrel{\text{def}}{=} g(x) = (g_1(x), \ldots, g_n(x))^T$$

to zero.

▶ Multidimensional Filter

Multidimensional filter

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

- Newton's method $\longrightarrow \min_s \quad f(x_k) + \nabla_x f(x_k)^T s + \frac{1}{2} s^T H_k s$
- Trust-region method
- Filter technique

Idea: encourage convergence to first-order critical points by driving every component of the objective's gradient

$$\nabla_x f(x) \stackrel{\text{def}}{=} g(x) = (g_1(x), \ldots, g_n(x))^T$$

to zero.

▶ Multidimensional Filter

Multidimensional Filter

Accept x_k^+ more often

A point x dominates a point y whenever

$$|g_i(x)| \leq |g_i(y)| \quad \forall\, i = 1, \ldots, n$$

Remember non-dominated points $\Rightarrow$ FILTER

Accept a new iterate ?

if it is not dominated by any other iterate in the filter

Multidimensional Filter

A multidimensional filter is a list of elements of the form $(g_{k,1}, \ldots, g_{k,n})$ such that if g_k and $g_\ell \in \mathcal{F}$, then

$$|g_{k,j}| < |g_{\ell,j}| \quad \text{for at least one } j \in \{1, \ldots, n\}$$

▶ Margin

x_k^+ is acceptable for the filter $\mathcal{F}$ if and only if

$$\forall\, g_\ell \in \mathcal{F} \qquad \exists\, j \in \{1, \ldots, n\} \;:\; |g_j(x_k^+)| \leq |g_{\ell,j}| - \underbrace{\gamma_g \|g_\ell\|}_{\text{margin}}$$

where γ_g is a small positive constant

Multidimensional Filter

A multidimensional filter is a list of elements of the form $(g_{k,1}, \ldots, g_{k,n})$ such that if g_k and $g_\ell \in \mathcal{F}$, then

$$|g_{k,j}| < |g_{\ell,j}| \quad \text{for at least one } j \in \{1, \ldots, n\}$$

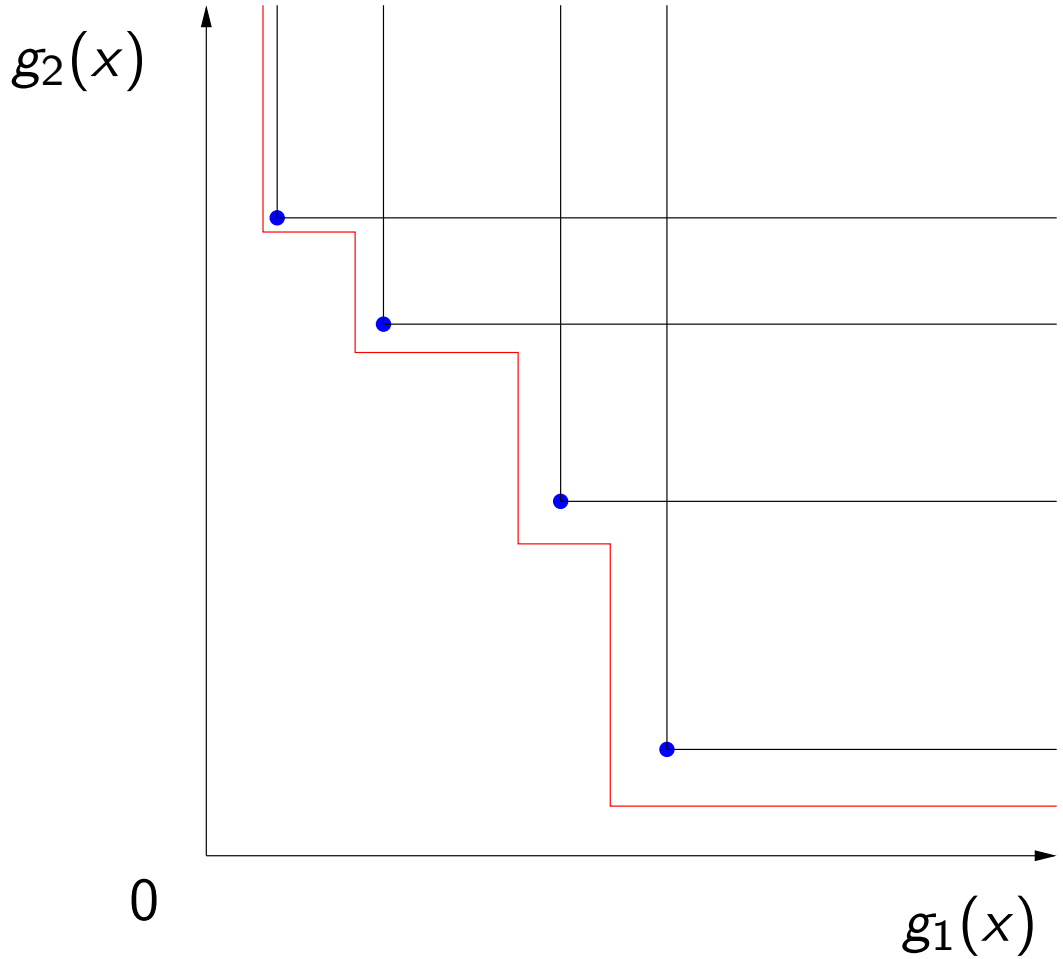
▶ Margin

x_k^+ is acceptable for the filter $\mathcal{F}$ if and only if

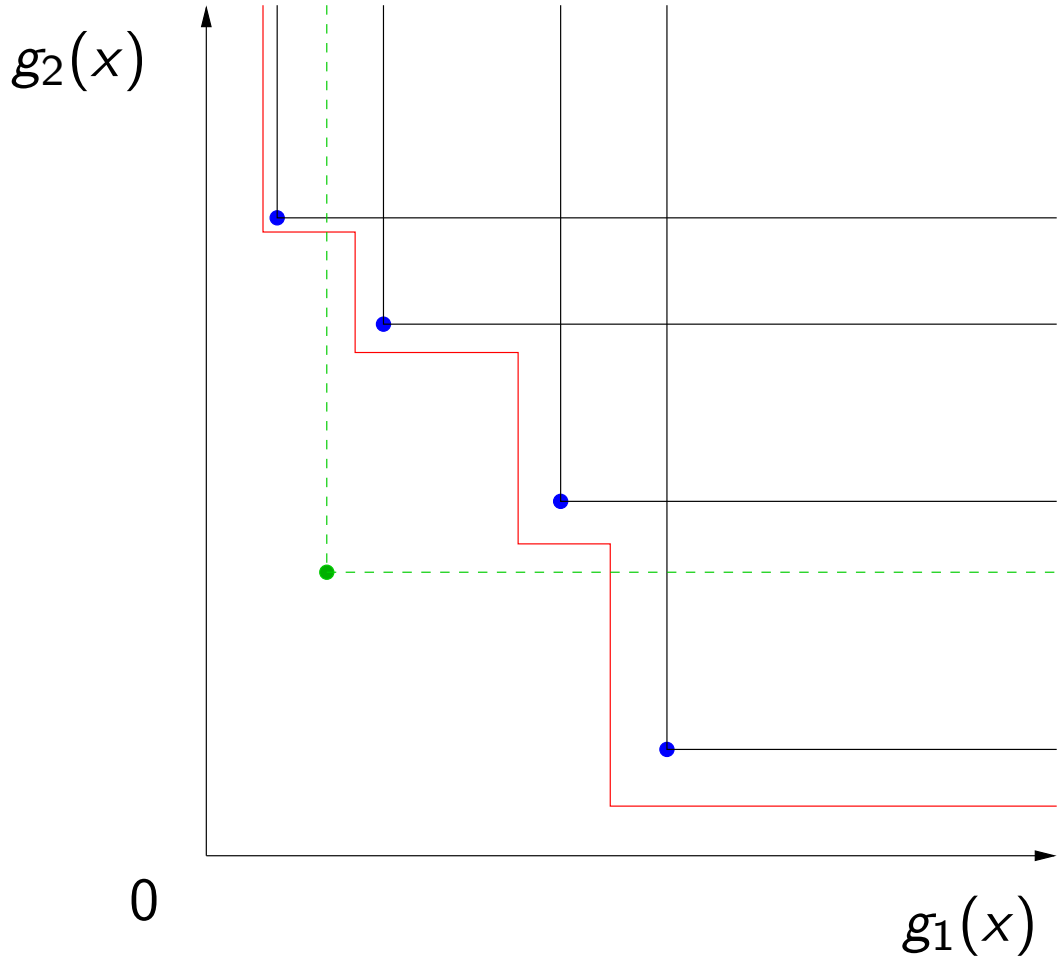
$$\forall\, g_\ell \in \mathcal{F} \qquad \exists\, j \in \{1, \ldots, n\} \;:\quad |g_j(x_k^+)| \leq |g_{\ell,j}| - \underbrace{\gamma_g \|g_\ell\|}_{\text{margin}}$$

where γ_g is a small positive constant

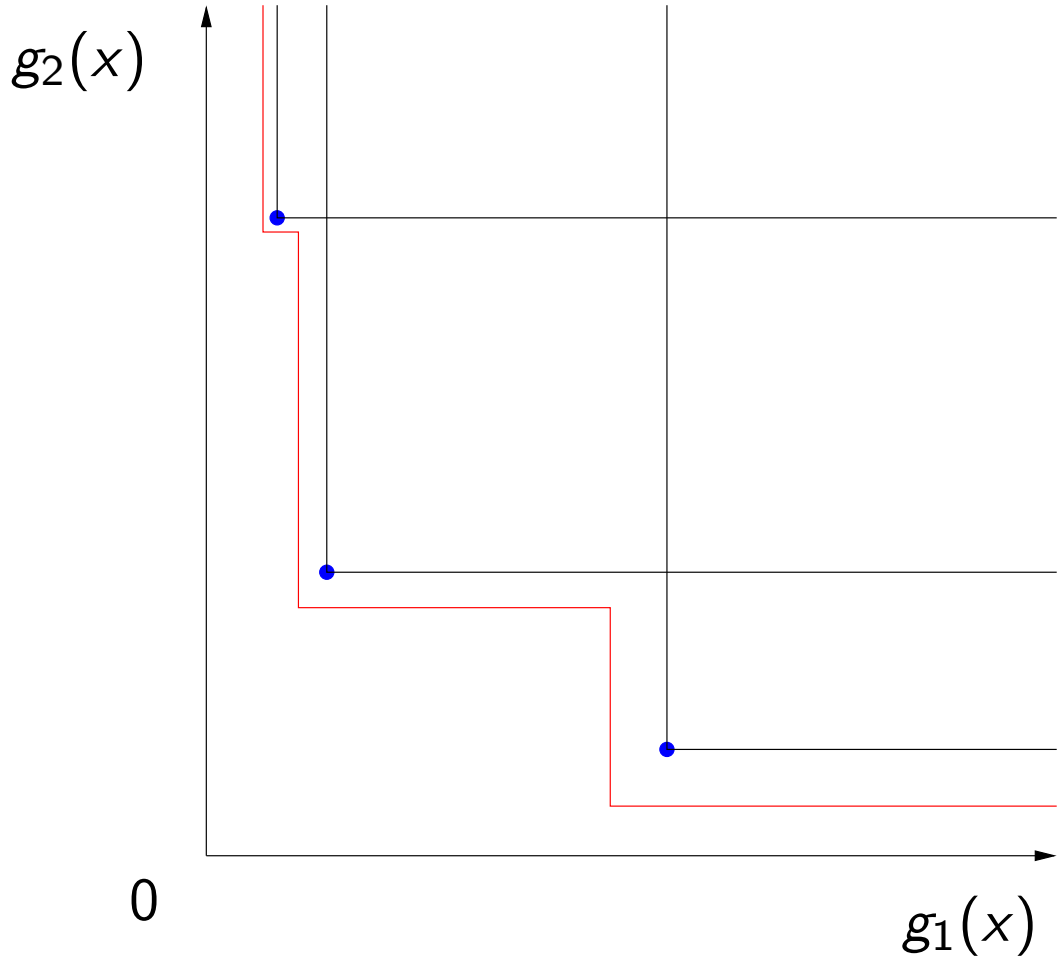
Multidimensional filter



Multidimensional filter



Multidimensional filter



Algorithm

☺ **Adequate for convex problems**

But ...

☹ $\nabla_x f(x) = 0$ **not sufficient for nonconvex problems !**

- always reject trial point if f is larger than reference value f_{sup}
- use filter as primary acceptance mechanism when locally convex
- if locally nonconvex or unacceptable for the filter, use sufficient decrease of f as secondary acceptance mechanism

Algorithm

☺ **Adequate for convex problems**

But ...

☹ $\nabla_x f(x) = 0$ **not sufficient for nonconvex problems !**

- always reject trial point if f is larger than reference value f_{sup}
- use filter as primary acceptance mechanism when locally convex
- if locally nonconvex or unacceptable for the filter, use sufficient decrease of f as secondary acceptance mechanism

Algorithm

- we do not require that $\|s_k\| \leq \Delta_k$ at every iteration
  - only restrict steps when locally nonconvex or not acceptable
    - ▶ traditional TR algorithm
- trust-region updates as normal if $\|s_k\| \leq \Delta_k$
- add a point to filter when convex if
  - insufficient decrease in f
  - outside the trust-region

Detection of a negative curvature:

- after each iteration giving sufficient decrease of f
  - reset the filter to the empty set
  - reset reference upper bound f_{sup} on the acceptable objective function values

Algorithm

- we do not require that $\|s_k\| \leq \Delta_k$ at every iteration
  - only restrict steps when locally nonconvex or not acceptable
    - ▶ traditional TR algorithm
- trust-region updates as normal if $\|s_k\| \leq \Delta_k$
- add a point to filter when convex if
  - insufficient decrease in f
  - outside the trust-region

Detection of a negative curvature:

- after each iteration giving sufficient decrease of f
  - reset the filter to the empty set
  - reset reference upper bound f_{sup} on the acceptable objective function values

Algorithm

General idea of the algorithm

- the filter plays the major role in ensuring global convergence within "convex basins"
- use the usual trust-region method only if things do not go well or if negative curvature is encountered

Important characteristics of the algorithm

- non-monotonicity in the objective function values
- use of unrestricted steps

Approximate derivatives

- filter-trust-region algorithm requires knowledge of first and second derivatives
- derivatives calculated analytically and supplied by the user
- unavailable or computationally expensive

The question

Is the behaviour of the algorithm directly related to the use of exact derivatives ?

Approximate derivatives

- filter-trust-region algorithm requires knowledge of first and second derivatives
- derivatives calculated analytically and supplied by the user
- unavailable or computationally expensive

The question

Is the behaviour of the algorithm directly related to the use of exact derivatives ?

Approximate derivatives

Where ?

- definition of the model of the objective-function

$$m_k(s) = f(x_k) + g_k{}^T s + \frac{1}{2} s^T H_k s$$

  - computation of the next trial point

- Gradient
  - definition of the filter
  - filter test acceptance mechanism
  - stopping criteria

- Hessian
  - decision to use the filter technique or not
  - restricted steps

Approximate derivatives

Two different ways to approximate
$\nabla_x f(x)$ and $\nabla_{xx} f(x)$

finite-difference approximations
to the gradient
and/or the Hessian

secant approximations
to the Hessian
(BFGS, SR1 updates)

Finite-difference approximations

Finite-difference gradients

- Forward

$$(\nabla_x f(x))_j \approx \frac{f(x + h_j e_j) - f(x)}{h_j}, \quad j = 1, \ldots, n$$

$\hookrightarrow$ n additional function evaluations

- Central

$$(\nabla_x f(x))_j \approx \frac{f(x + h_j e_j) - f(x - h_j e_j)}{2h_j}, \quad j = 1, \ldots, n$$

$\hookrightarrow$ $2n$ additional function evaluations

  - different tested stepsizes h_j

Finite-difference approximations

Finite-difference Hessians

- $\nabla_x f(x)$ analytically available
  - Forward

$$(\nabla_{xx} f(x))_{.j} \approx B_{.j} = \frac{\nabla_x f(x + h_j e_j) - \nabla_x f(x)}{h_j}, \quad j = 1, \ldots, n$$

$\hookrightarrow$ n additional gradient evaluations

  - Central

$$(\nabla_{xx} f(x))_{.j} \approx B_{.j} = \frac{\nabla_x f(x + h_j e_j) - \nabla_x f(x - h_j e_j)}{2h_j}, \quad j = 1, \ldots, n$$

$\hookrightarrow$ $2n$ additional gradient evaluations

Finite-difference approximations

- $\nabla_x f(x)$ not available
  - $(\nabla_x f(x))_j$ approximates by one of the finite-difference formulae
  - $$(\nabla_{xx} f(x))_{ij} \approx B_{ij} \qquad 1 \leq i \leq j \leq n$$
    $$= \frac{f(x + h_i e_i + h_j e_j) - f(x + h_i e_i) - f(x + h_j e_j) + f(x)}{h_i h_j}$$
    $\hookrightarrow \frac{1}{2}(n^2 + 3n)$ additional function evaluations

▶ DFO techniques

Secant approximations

Update approximations of the Hessian matrices in some computational cheap ways

- Broyden-Fletcher-Goldfarb-Shanno (BFGS)

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

- Symmetric rank-one (SR1)

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

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla_x f(x_{k+1}) - \nabla_x f(x_k)$

  - different initial Hessian approximations B_0

Numerical results

Study the influence of approximate derivatives on the robustness and the efficiency of the filter-trust-region algorithm

Two variants were tested on 66 unconstrained small-scaled problems from the CUTEr collection :

- "Default"
  - algorithm described above
  - generalized Lanczos trust-region method without preconditioning (`GLTR` from the GALAHAD Library) to solve the TR subproblem
- "Pure trust-region"
  - no trial point is ever accepted in the filter
  - step always restricted to the trust-region

Numerical results

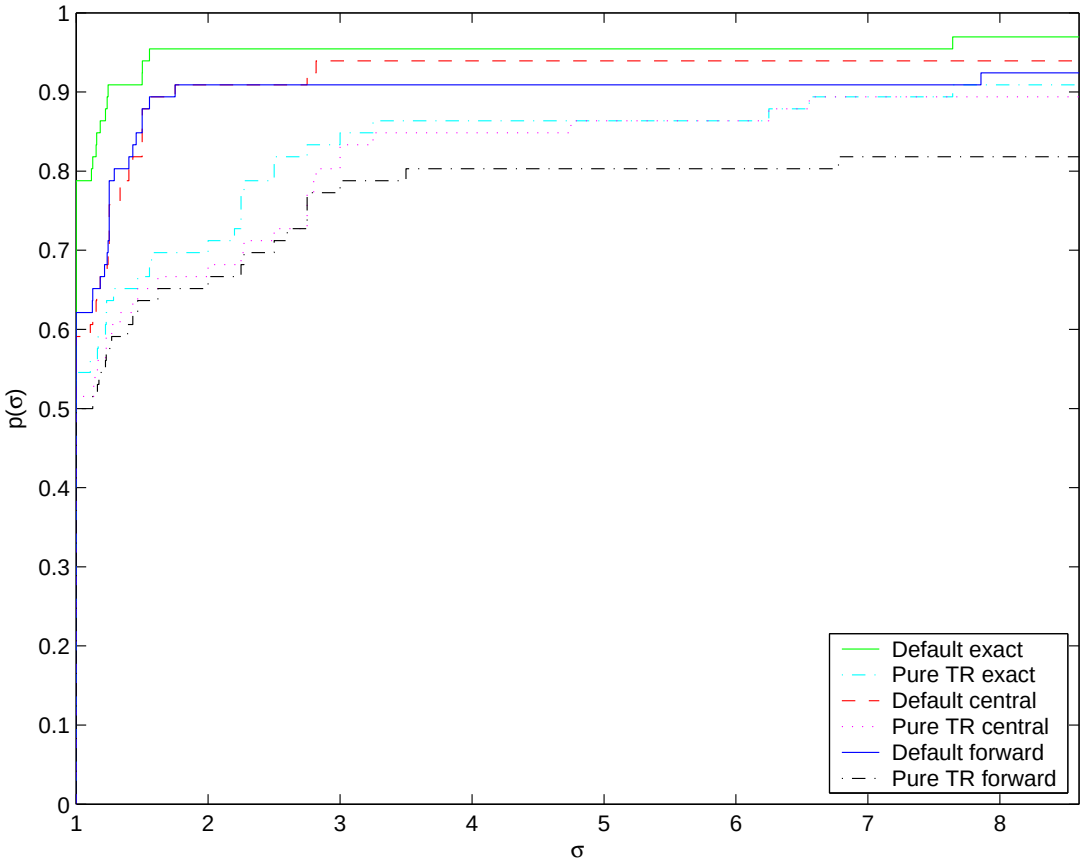
For these two variants :

- exact derivatives
- finite-difference approximations to the gradient and/or the Hessian (stepsizes, central, forward, gradient available, ...)
- secant approximations to the Hessian (BFGS, SR1, B_0, gradient available,...)

▶ more than 70 tested variants

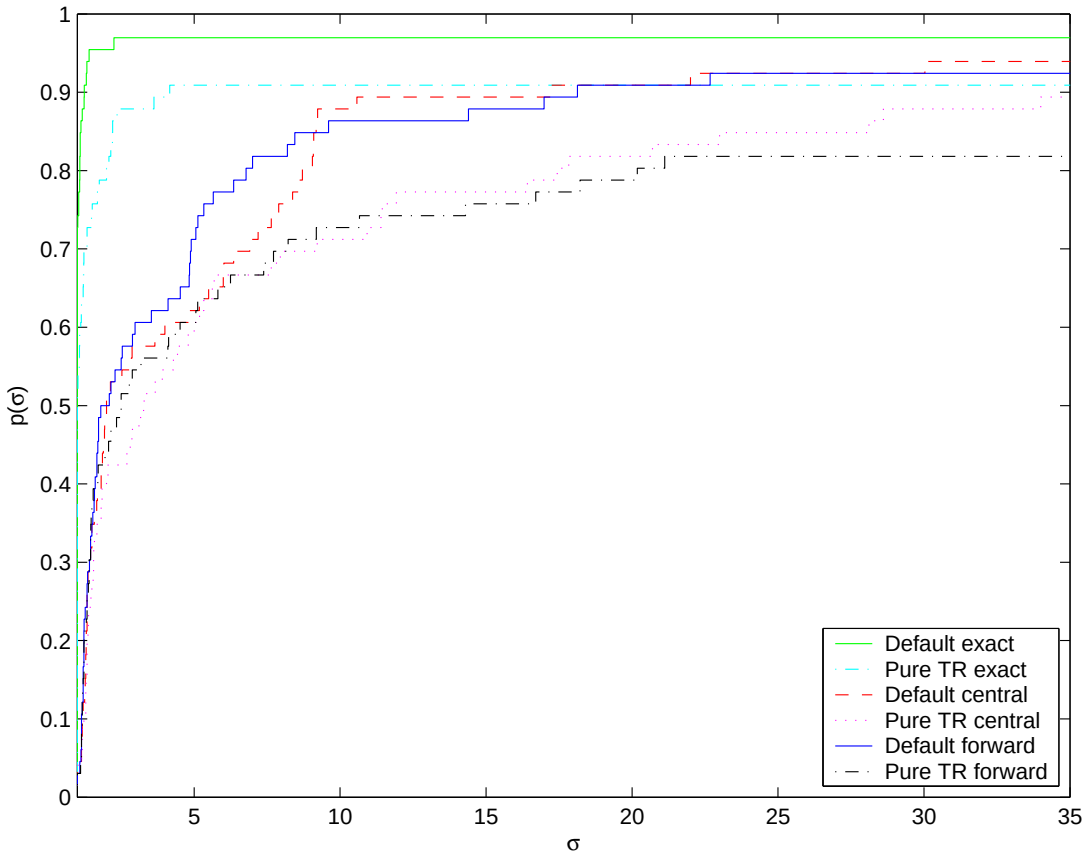
Numerical results : finite-difference - $\nabla_x f(x)$ available

Iterations performance profile



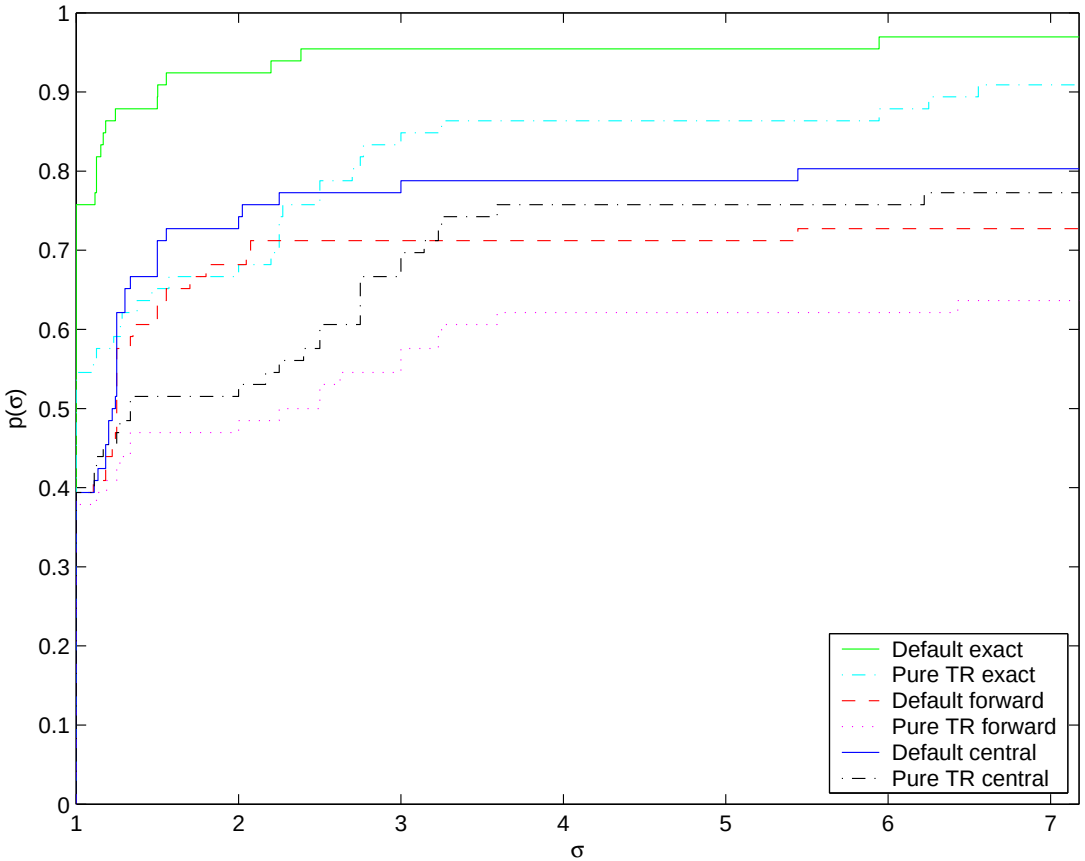
Numerical results : finite-difference - $\nabla_x f(x)$ available

CPU performance profile



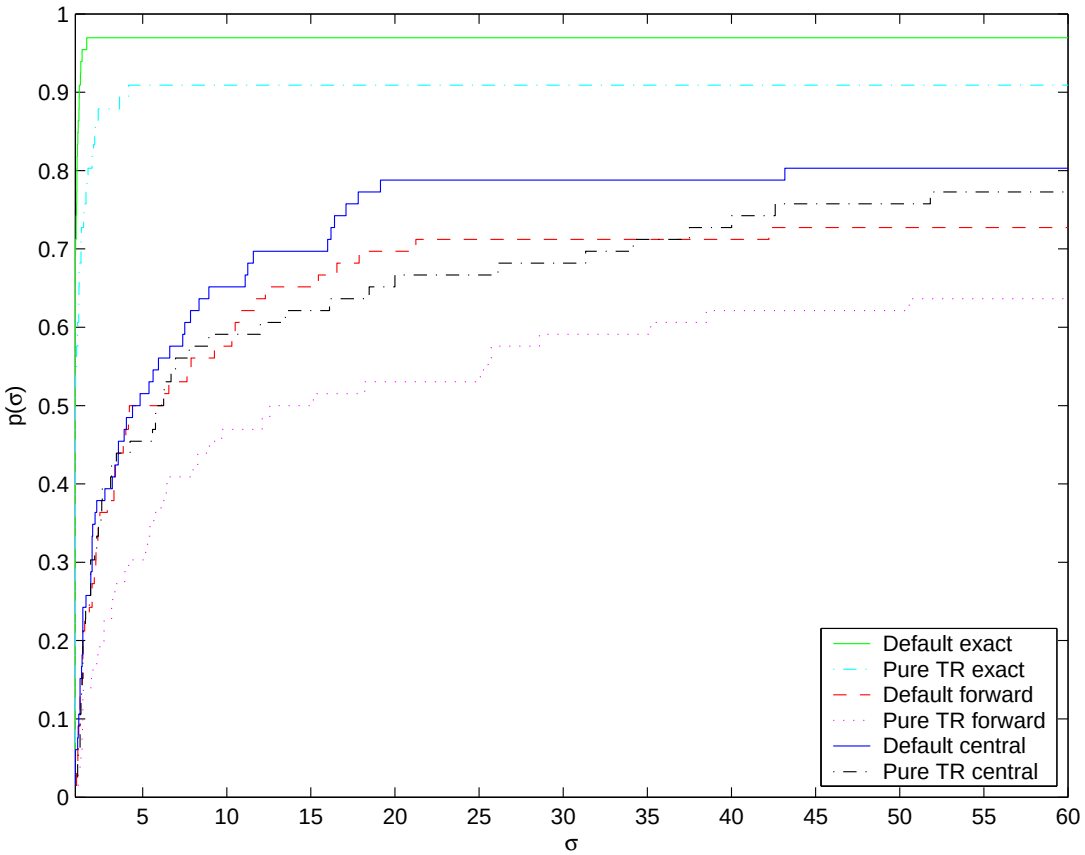
Numerical results : finite-difference - $\nabla_x f(x)$ not available

Iterations performance profile



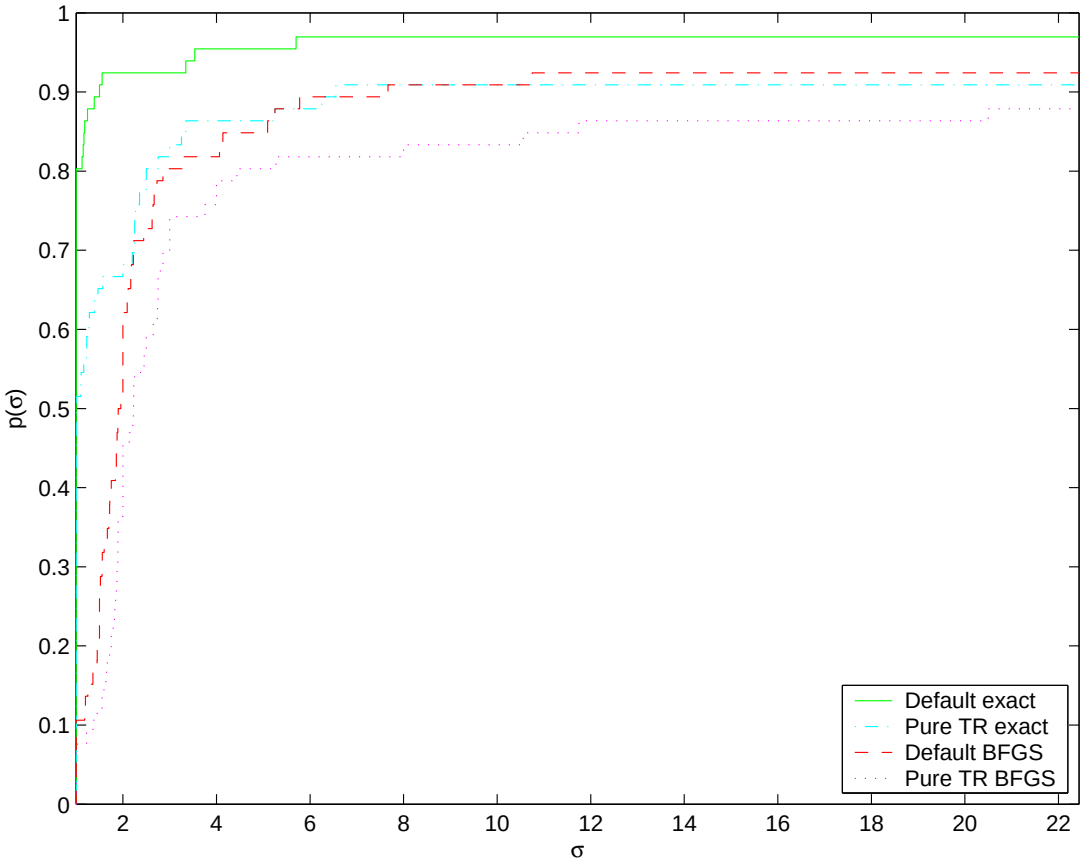
Numerical results : finite-difference - $\nabla_x f(x)$ not available

CPU performance profile



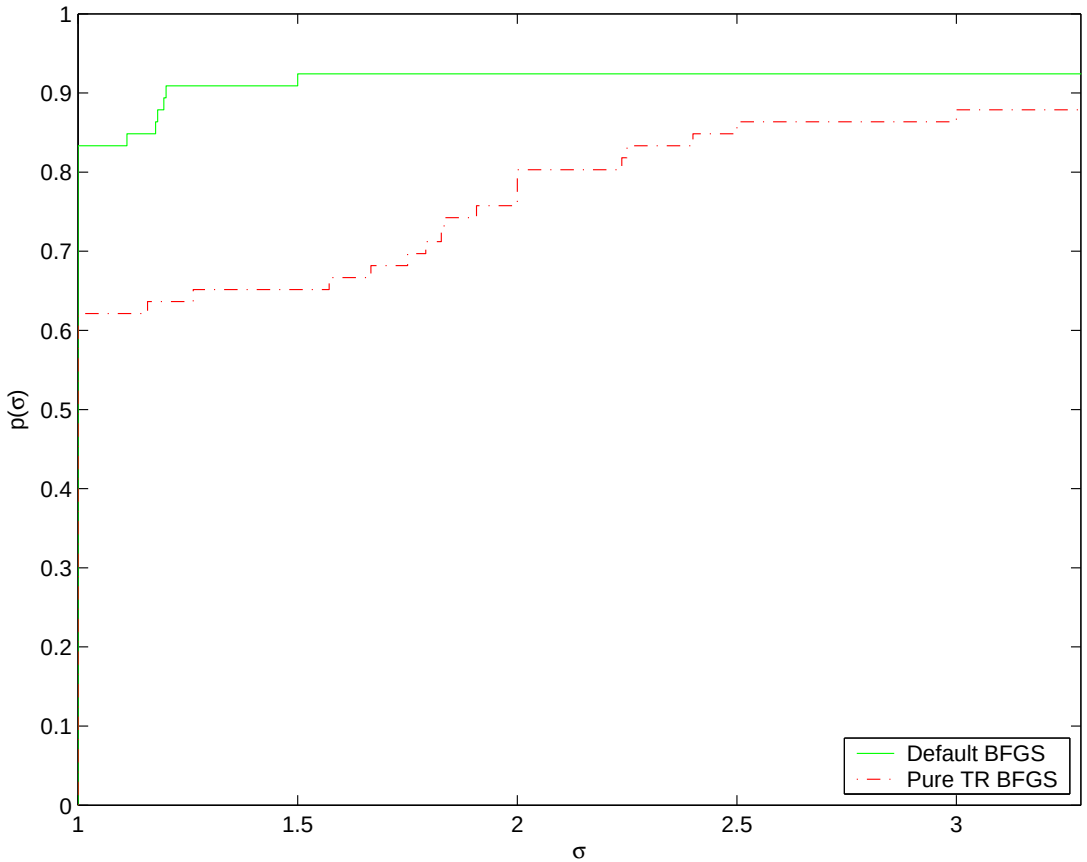
Numerical results : BFGS update

Iterations performance profile



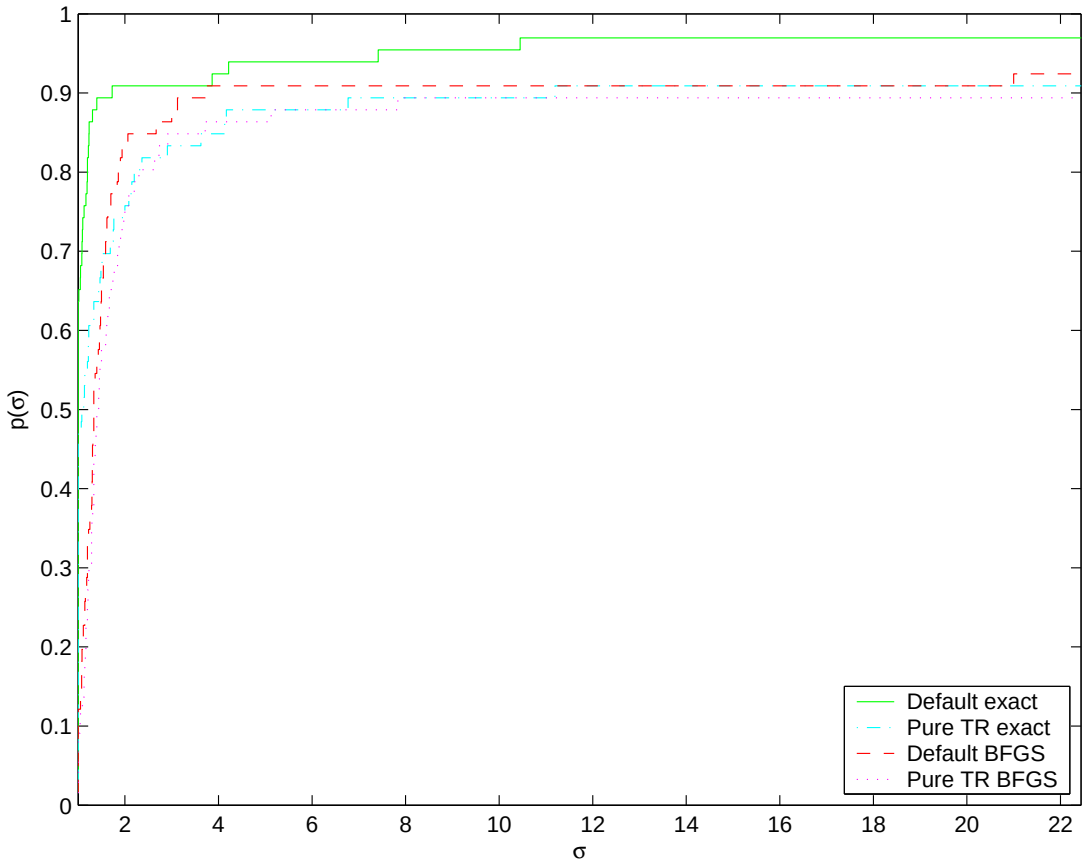
Numerical results : BFGS update

Iterations performance profile



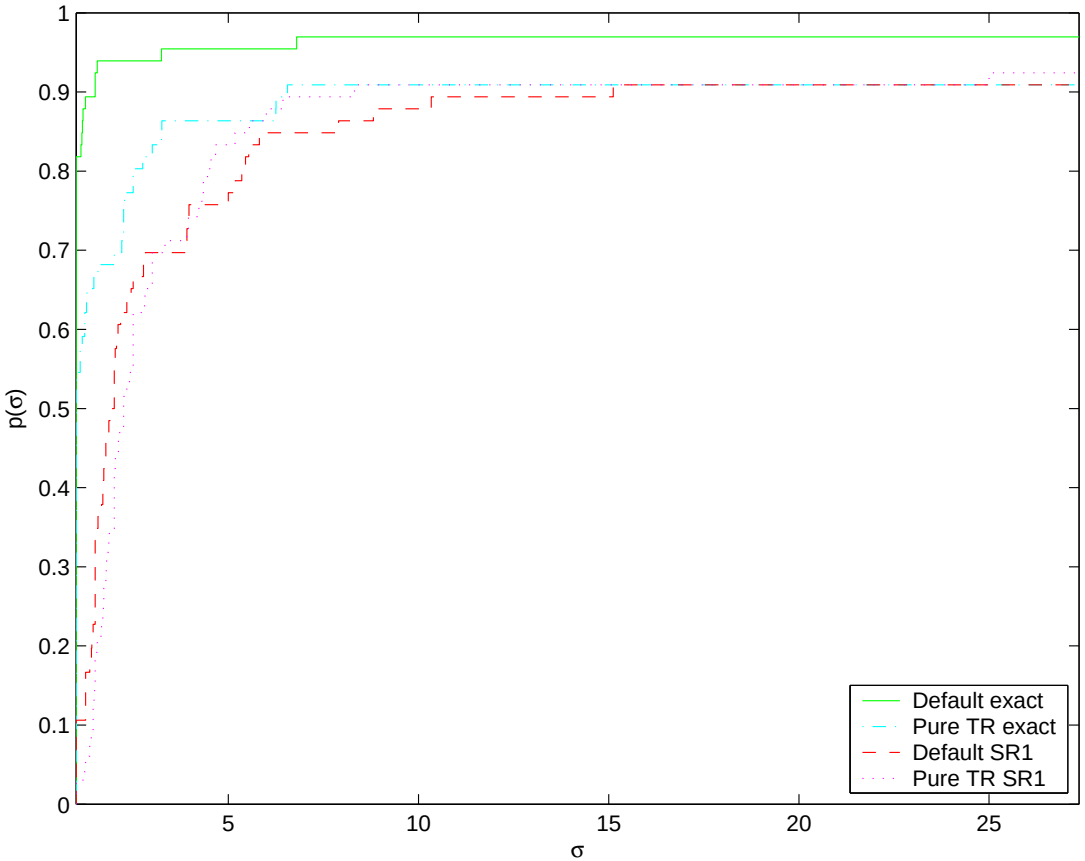
Numerical results : BFGS update

CPU performance profile



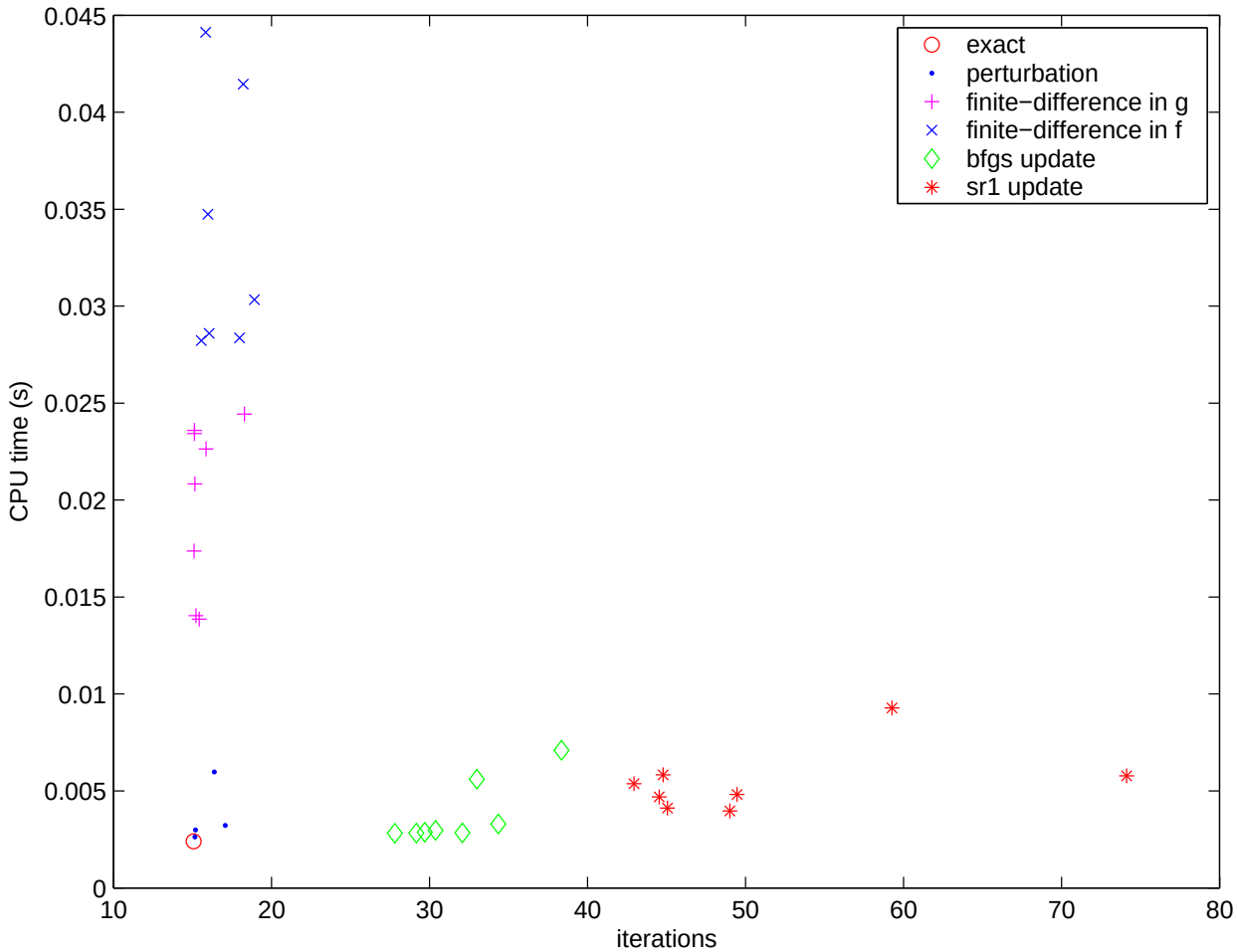
Numerical results : SR1 update

Iterations performance profile

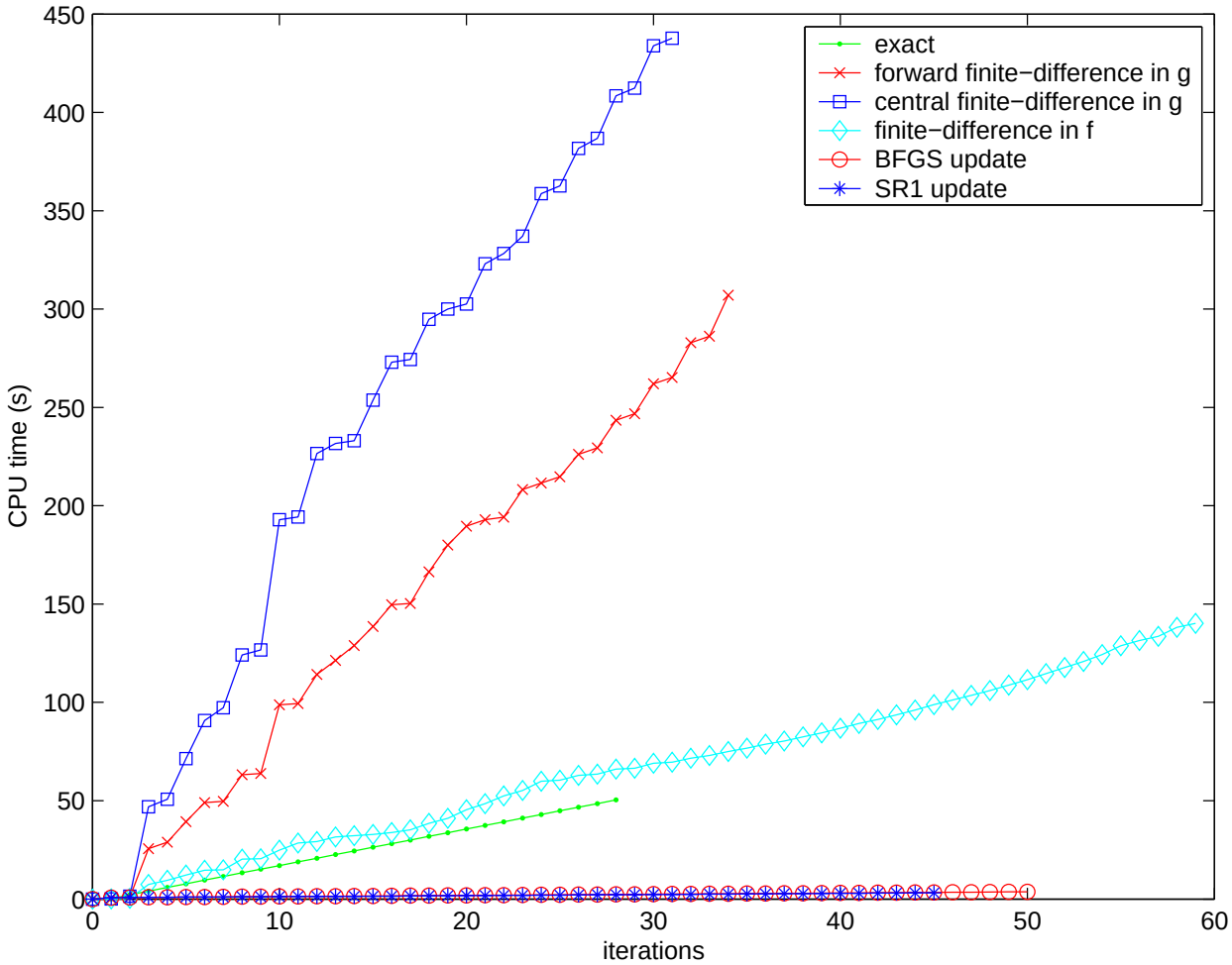


Numerical results : Comparison

Combined performance



Numerical results : STRATEC (nested logit model)



Conclusion

- the filter-trust-region algorithm does not suffer more than the classical trust-region method to the use of approximate derivatives

Thank you for your attention !