Nonlinear Minimization Techniques Without Using Derivatives

Florian Jarre (Univ. Düsseldorf)
Markus Lazar (Univ. Appl. Sc. Rosenheim),
Felix Lieder (Univ. Düsseldorf)

Aug. 10, ICCOPT 2016, Tokyo

with support from:

ifm – close to you!

.

Key Points

- Some Interesting Application
- Challenges and open problems in the solution approach
- Numerically expensive derivative approximations lead to methods that would otherwise (for standard NLP's) not be competitive
- Some results and further applications
- Generalization to Constrained Minimization
- Further (preliminary) Numerical Results

A Practical Calibration Problem

(The main motivation for this talk)

Three situations where an inclination sensor is used:

Reachstacker

Escape Stair

Turbine

Sensor used:

Accuracy:

Genauigkeit	$\pm$ 0,01°
Temperaturkoeffizient [1/K]	$\leq \pm$ 0,0008°
Reproduzierbarkeit	$\leq \pm$ 0,01°
Auflösung [°]	0,001; parametrierbar

(http://www.ifm.com/products/de/ds/JN2101.htm)

The Mathematics behind this:

Based on some MEMS with inaccuracies in three locations. Further measurement errors of accelerations lead to a nonlinear least squares problem for calibrating each single sensor.

Abstract: "Lazyness": (not the engineer who was lazy)

Approximation model
– technical formulae relating unknown angles ...

Nonlinear least squares problem with 12 unknowns and no derivative information (unconstrained)

Derivative-Free Minimization Approaches (Rios, Sahinidis, 2013)

- Most elegant (here!): Automatic differentiation
  - Technical, further "investment" with unknown outcome
- Just in Matlab: Optimization toolbox (more later)
- Other public domain Matlab implementations: GLOBAL (Csendes), SID-PSM (Custodio, Vicente), cmaes (Hansen), snobfit (Huyer, Neumaier), minFunc (Schmidt), and PSwarmM (Vaz, Vicente) ...

Derivative-Free Minimization Approaches (Rios, Sahinidis, 2013)

- Most elegant (here!): Automatic differentiation
  - Technical, further "investment" with unknown outcome
- Just in Matlab: Optimization toolbox (more later)
- Other public domain Matlab implementations: GLOBAL (Csendes), SID-PSM (Custodio, Vicente), cmaes (Hansen), snobfit (Huyer, Neumaier), minFunc (Schmidt), and PSwarmM (Vaz, Vicente) ...

Derivative-Free Minimization Approaches (Rios, Sahinidis, 2013)

- Most elegant (here!): Automatic differentiation
  - Technical, further "investment" with unknown outcome
- Just in Matlab: Optimization toolbox (more later)
- Other public domain Matlab implementations: GLOBAL (Csendes), SID-PSM (Custodio, Vicente), cmaes (Hansen), snobfit (Huyer, Neumaier), minFunc (Schmidt), and PSwarmM (Vaz, Vicente) ...

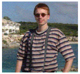
Derivative-Free Minimization Approaches (Rios, Sahinidis, 2013)

- Most elegant (here!): Automatic differentiation
  - Technical, further “investment” with unknown outcome
- Just in Matlab: Optimization toolbox (more later)
- Other public domain Matlab implementations: GLOBAL (Csendes), SID-PSM (Custodio, Vicente), cmaes (Hansen), snobfit (Huyer, Neumaier), minFunc (Schmidt), and PSwarmM (Vaz, Vicente) ...
- Natural focus: Global optimization (lack of “local” derivative information hurts least when a global minimizer is to be approximated). Expensive / noisy function evaluations.

Calibration Problems
(and a number of other situations)

- Local solution, reasonable initial guess
- High accuracy
- Smoothness
- Moderately expensive function evaluations
- Robustness / ease of use / no installation

Jos Sturm, SeDuMi



- First:
  Method with best theoretical complexity estimate
  Not-so-long steps
  (maintaining theoretical complexity estimate)
- Second:
  User-friendly, convenient input-output format
- Third:
  Decide on numerical linear algebra
  (Cholesky factorization as preconditioner for cg iterations)

Best theoretical approach here?

Problem: minimize $f(x)$ where $\text{lb} \leq \text{x} \leq ub$.

- This application: local minimization, smoothness
- Symmetric finite differences for first derivative (minFunc, Matlab toolbox)
  Gradients more expensive than in standard nonlinear minimization
- Quasi-Newton updates for second derivative
- Trust region approach

Euclidean norm trust region

- Euclidean norm least squares symmetric update of Hessian (PSB)
- Additional least squares update using central finite differences?
- Gradient: $2n$ function evaluations: $\Longrightarrow$
  Line search for optimal trust region radius at each step

Underestimated detail: Line search
(minimizing some scalar function $f : [a, b] \to \mathbb{R}$)

(open interval, when $a = -\infty$ or $b = \infty$.)

Underestimated detail: Line search
(minimizing some scalar function $f : [a, b] \to \mathbb{R}$)

(open interval, when $a = -\infty$ or $b = \infty$.)

- fminbnd of Matlab does not guarantee value at least as low as end points
  (Example, log barrier, set to *Inf* whenever undefined)

Underestimated detail: Line search (minimizing some scalar function $f : [a, b] \to \mathbb{R}$)

(open interval, when $a = -\infty$ or $b = \infty$.)

- fminbnd of Matlab does not guarantee value at least as low as end points
  (Example, log barrier, set to *Inf* whenever undefined)
- New code using golden mean search and spline interpolation

  Rather long and technical but simple idea:
  Minimize interpolating spline, decide whether to use minimizer as next point to evaluate f.
  (Also gives approximation for first and second derivative.)

Cubic spline interpolation

- Choice: Natural spline, Not-a-knot-spline, ... ?
- Natural spline minimizes some measure of curvature of f
  - Suitable for CAD
  - No reason to assume $f'' = 0$ at end points,
  - Approximation quality?
- Not-a-knot-spline
  - Interpolation error generally higher near end points $\Longrightarrow$
  - Higher degree of interpolation near end points
  - But minimizer by construction in the middle (5 points).
- Least squares spline:

General cubic spline through $(x_0, f_0), \ldots, (x_n, f_n)$

- First, fix $s'(x_0) = s''(x_0) = 0$.
- Using $f(x_0), f(x_1)$ determine $s\Big|_{[x_0,x_1]}$ (2×2 linear system).
- This gives you $s(x_1),\ s'(x_1)$, and $s''(x_1)$.
- Repeat determining $s\Big|_{[x_i,x_{i+1}]}$ for $1 \leq i \leq n-1$.
- Same way interpolate the zero function with initial values $\hat{s}'(x_0) = 1,\ \hat{s}''(x_0) = 0$ and with $\bar{s}'(x_0) = 0,\ \bar{s}''(x_0) = 1$.
- For any $\alpha, \beta \in \mathbb{R}$ the function $s + \alpha\hat{s} + \beta\bar{s}$ interpolates f.

Least squares cubic spline

- Could fix α, β
  such that $s''(x_0) = s''(x_n) = 0$ (natural spline), or
  such that $s'''(x_{1_-}) = s'''(x_{1_+})$ and $s'''(x_{n-1_-}) = s'''(x_{n-1_+})$
  (not-a-knot spline)
- Again, system of 2 linear equations for 2 unknowns.
- Minimize
  $$\sum_{i=1}^{n-1} w_i (s'''(x_{i_-}) - s'''(x_{i_+}))^2$$
  for some weights $w_i \geq 0$.
  Here, $w_i := 1/(x_{i+1} - x_{i-1})$
  (to favor smaller jumps between short intervals).

Conditioning

Note: Above conditions define a scalar product on the space of zero-interpolating functions.

- In spite of
  $\hat{s}'(x_0) = 1,\ \hat{s}''(x_0) = 0$ and $\bar{s}'(x_0) = 0,\ \bar{s}''(x_0) = 1,$
  $\hat{s}$ and $\bar{s}$ are nearly linearly dependent when x_i accumulate in the middle.
- Orthogonalize $\hat{s}$ and $\bar{s}$ w.r.t. above scalar product.
- Get new initial values for $\hat{s}$ and $\bar{s}$ (at x_0).
- Recompute $\hat{s}$ and $\bar{s}$ for the new initial values.
- Stable solution of 2×2 system.

Comparison with quadratic interpoation

- Better bounds; numerical examples give significantly better approximation quality than quadratic interpolation
- Challenge: When a (very good – or not so good?) candidate for a minimizer is given, how to choose the next point to evaluate f.
- If fminbnd works it is hard to beat quadratic interpolation of fminbnd.
- Jumps in s''' may give some error bound that can be used for the selection of the next iterate?
- Least squares approach for (low!) noise in the function evaluation?

Gradient and Hessian Approximation

- Gradient: central finite difference
  (ad hoc determination of step size)
- Hessian H: PSB
- followed by central finite difference for diagonal of H
  (this is also a Euclidean least squares update) ???
- using other orthogonal bases?

Comparison of Hessian Approximations

Update	f_{STmod}	f_{CR}
zero Hessian	3.4e-4 / 461	4.8e-3 /10000
curv. only, U=I	4.4e-5 / 123	3.8e-3 / 10000
curv. only, U=rand$_{worst}$	1.1e-11 / 73	1.7e-14/ 1853
PSB only	4.5e-13 / 23	6.4e-16 / 738
PSB & curv., U=I	1.9e-8 / 34	6.2e-15 / 739
PSB & curv., U=rand$_{worst}$	2.3e-11 / 39	1.8e-16 / 824

(n = 10, random test: 100 test runs)

Trust Region Step (ignoring the "hard case")

- $\Delta x(\lambda) := -(H + \lambda I)^{-1} g$ where $g \approx \nabla f(x)$ and $\lambda \geq \lambda_- := \max\{0, -\lambda_{min}(H)\}$.
- Moré-Sorensen-type reparameterization:

$$\lambda(t) := \lambda_- + \epsilon + \frac{1 - \lambda}{\epsilon + \lambda} \quad \text{for} \quad t \in [0, 1],$$

with $\Delta x(\lambda(0)) \approx 0$,

and $\Delta x(\lambda(1)) \approx \begin{cases} \text{Newton step} \\ \text{large step along negative curvature} \end{cases}$

Line search:

(Dennis, Echebest, Guardarucci, Martinez, Scolnik, and Vacchino, 1991)

$$\text{minimize} \quad f(x(\lambda(t))) \qquad \text{for} \quad t \in [0,1].$$

For some examples a rather high accuracy in t is needed.

Given an eigenvalue decomposition of H, the computation of $x(\lambda(t))$ is cheap (Nevertheless, line search matters).

Usage

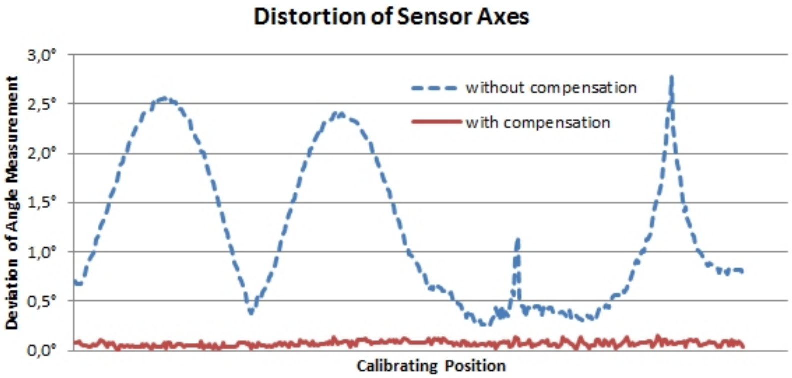
- Download min_f.m (one file)
- simplest call: min_$f(@f)$ for $n = 1$
- simplest call: min_$f(@f, zeros(n, 1))$ for $n > 1$
  Dimension of the starting point (e.g. zeros(n,1)), used to initialize the algorithm.
- or: [x,y,g,H]=min_$f(@f, zeros(n, 1), options)$ with bounds...

Calibration

Calibration

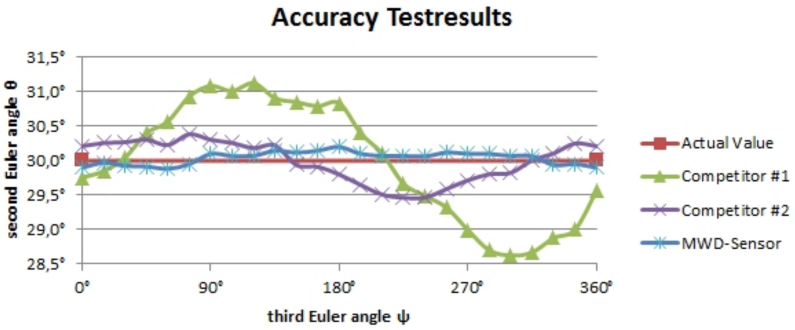
- 120 measurement values for fixing 12 parameters.
- Nonlinear least squares problem.
- Lousy result.
- ???
- Subcontractor guaranteed certain high accurcy of “his” parts.
- Treat accuracy of “his” parts as a variable
  (not a given parameter).
- No new programming necessary just add 3 further variables.

Results



(Subcontractor admitted wrong specifications)

Results


Accuracy Testresults
31,5°
31,0°
30,5°
30,0°
29,5°
29,0°
28,5°
second Euler angle θ
0°
90°
180°
270°
360°
third Euler angle ψ
Actual Value
Competitor #1
Competitor #2
MWD-Sensor

Other applications (Thanks to Roland Freund)

- Worst case performance of an algorithm?
- Example (with known answer): k steps of cg algorithm.
- $$f: \ x \mapsto x^T A x - 2b^T x + b^T A^{-1} b \equiv \|x - A^{-1}b\|_A^2$$
- Find eigenvalue distribution of $A \succ 0$ and initial point x^0 such that the k-th cg-iterate x^k maximizes $\left(\frac{f(x^k)}{f(x^0)}\right)^{1/2}$ subject to the constraint $\mathrm{cond}(A) \leq \kappa$.

  (Worst case performance of cg method)
- Known:

$$\max \left(\frac{f(x^k)}{f(x^0)}\right)^{1/2} = 2\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k\right]^{-1}.$$

Worst case performance

W.l.o.g. A is diagonal. Initially, eigenvalues evenly distributed in $[1, \kappa]$, x^0 all ones. Here, $n = 10$ and $\kappa = 10$:

k	red_0	red_{max}	$dist$	# it.	# f-eval.	time
1	0.6158	0.8182	1.7e-13	11	700	0.70
3	0.2257	0.2750	0	22	1242	0.54
5	0.0607	0.0756	1.2e-14	25	1426	0.70
7	0.0098	0.0204	4.0e-15	28	1619	0.87
9	0.0006	0.0055	8.0e-15	45	2688	1.60

(Solution not unique, some b_i are zero, multiple eigenvalues)
For each k, a different matrix A and a different starting vector x^0 lead to the worst case.

Worst case performance

Here, $k = 20$ and $\kappa = 100$.

n	red_0	red_{max}	$dist$	# it.	# f-eval.	time
25	0.0002	0.0361	6.0e-15	68	8205	7.34
50	0.0095	0.0361	1.4e-14	79	17486	14.48
100	0.0235	0.0361	3.6e-12	194	8.2213	67.92
200	0.0250	0.0361	4.5e-12	222	182701	150.23

($n = 200$ means 397 variables with bounds).

Principal Components and Factor Analysis

Given a symmetric positive semidefinite $n \times n$ matrix X find a nonnegative diagonal matrix D and a $n \times k$ matrix F such that $\|X - FF^T - D\|_F$ is small.
(More constraints to be taken into account.)

(SDP)
Maximize $I \bullet D$ s.t. D diagonal, $D \geq 0$, and $X - D \succeq 0$.
Factor $X - D = \hat{F}\hat{F}^T$, possibly omitt some columns of $\hat{F}$.

• 20×20 example from the literature.
After fixing k minimizing $\|X - FF^T - D\|_F^2$ yields another 20 % improvement. (Just to find out whether it is worth continuing optimization on a given factorization)

CUTE-ST Test Problems (Gould, Orban, Toint)

- Replace NaN with Inf
- Increase step length accuracy from 10^{-6} to 10^{-10}
  (for smooth functions about 0.5 extra fn eval. in line search)
- 170 unconstrained problems of dimension ≤ 300
- 29 times iteration limit reached (100*n)
  (all but 2 problems had 4 digits accuracy by then)
- 16 times stop after 2 iterations
- 49 times norm of gradient $\leq 10^{-8}$ (twice fn value -Inf)
- 107 times norm of gradient $\leq 10^{-4}$
- 120 times norm of gradient $\leq 10^{-2}$
  (some others seem to be close to a local opt. as well)

Constrained Minimization (with open questions)

Input format:

$$\text{minimize} \quad f(x) \quad | \quad \begin{array}{l} f_{E_1}(x) = 0, \quad A_{E_2}x = b_{E_2}, \\ f_{I_1}(x) \leq 0, \quad A_{I_2}x \leq b_{I_2}, \quad lb \leq x \leq ub, \end{array}$$

where all functions are assumed to be differentiable.

- matlab fmincon
- Sampaio and Toint (OMS, 2016): Derivative-free trust-funnel method

Constrained Minimization (with open questions)

Input format:

$$\text{minimize} \quad f(x) \quad | \quad \begin{array}{l} f_{E_1}(x) = 0, \quad A_{E_2}x = b_{E_2}, \\ f_{I_1}(x) \le 0, \quad A_{I_2}x \le b_{I_2}, \quad lb \le x \le ub, \end{array}$$

where all functions are assumed to be differentiable.

Eliminate the equations $A_{E_2}x = b_{E_2}$ (QR factorization):

$$\text{minimize} \quad f(x) \quad | \quad f_E(x) = 0, \quad f_I(x) \le 0, \quad Ax \le b.$$

(Lower dimension, loss of sparsity)

Preprocessing: $b \ge 0$.

“Best” approach?

- Central finite differences
- SQP subproblems with Euclidean norm trust region constraint
- Second order correction
- Project Hessian H of Lagrangian to orthogonal complement of active constraint gradients $\rightarrow \tilde{H}$.
- Regularize with a multiple of the identity
- Evaluate linearized feasibility within trust region radius: “r_v”

SQP Subproblem

- $\hat{I}$: violated inequalites at current iterate, (all nonlinear)
  $\hat{A}$: remaining inequalities.
- $$\begin{array}{llcccc}
\text{minimize} & & & g^T\Delta x + \frac{1}{2}\Delta x^T \tilde{H} \Delta x & & \\
 & Df_E \Delta x & - & s_E & = & -f_E, \\
 & Df_{\hat{I}} \Delta x & - & s_{\hat{I}} & \leq & -f_{\hat{I}}, \\
 & & - & s_{\hat{I}} & \leq & 0, \\
\text{s.t.} & \hat{A}\Delta x & & & \leq & \hat{b}, \\
 & \|\Delta x\|^2 & & & \leq & \delta^2, \\
 & \|s_E\|^2 & + & \|s_{\hat{I}}\|^2 & \leq & 0.9 r_v^2 + 0.1(\|f_E\|^2 + \|f_{\hat{I}}\|^2).
\end{array}$$
- Three SOC constraints

Use Second Order Cone Programming?

- Two 2-norm cones (seem to be o.k.)
- Cholesky factor of $\tilde{H} \succ 0$ defining the third cone is typically poorly conditioned
- Solution very poorly scaled.
  Inactive parts vary by 10^5 for a well conditioned problem of 3 variables and a feasible iterate with 10^{-4} accuracy.
  (SOCP to 10 digits accuracy)
- Search direction is an ascent direction.
- Trying to scale the problem — even after knowing the solution – did not help.
- Conditioning also was the reason for using a 2-stage approach, first determining r_v and then solving the SQP problem, rather than using a "big-M" approach.
- SOCP formulation inherently / unnecessarily ill-conditioned?

To Do

- Other IPM approaches without using SOCP refomulation?
- Active Set methods?

CUTE-ST Test Problems (Gould, Orban, Toint)

- 651 constrained problems with $m + n \leq 500$, where m is the number of inequality constraints
- 221 times $\geq$ 6 digits accuracy reached (local sol.)
- 131 times infeasible stationary point reached (6 digits)
- 94 times 3-6 digits accuracy reached
- 54 times iteration limit reached
- 123 times stop due to slow progress
- 4 times return inital point.

Conclusion

- Many interesting applications
- Expensive finite differences lead to a wider selection of solution approaches
  allowing more expensive linear algebra without dominating the overall computational effort.
- (and a new line search)
- Cautious new insight on SOCP solver
- Unconstrained version available from home page
  - Florian Jarre (page in English) $\longrightarrow$
  - (Smooth) minimization without using derivatives
- (Constrained version upon request)