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
Sensor used:
Accuracy:

<table>
<thead>
<tr>
<th></th>
<th>± 0,01°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genauigkeit</td>
<td>± 0,01°</td>
</tr>
<tr>
<td>Temperaturkoeffizient [1/K]</td>
<td>≤ ± 0,0008°</td>
</tr>
<tr>
<td>Reproduzierbarkeit</td>
<td>≤ ± 0,01°</td>
</tr>
<tr>
<td>Auflösung [°]</td>
<td>0,001; parametrierbar</td>
</tr>
</tbody>
</table>

(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) ...
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 $lb \leq x \leq ub$.

- 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: $\Rightarrow$
  - 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 $f : [a, b] \rightarrow \mathbb{R}$)
(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 $\text{Inf}$ 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 \( \infty \) 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 $\Rightarrow$
  - 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 \times 2 linear system).
- This gives you \(s(x_1), s'(x_1), \text{ and } s''(x_1)\).
- 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 $\alpha, \beta$ 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 
  \[ \hat{s}'(x_0) = 1, \quad \hat{s}''(x_0) = 0 \]  
  and 
  \[ \bar{s}'(x_0) = 0, \quad \bar{s}''(x_0) = 1, \]  
  \( \hat{s} \) 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 \( \times \) 2 system.
Comparison with quadratic interpolation

- 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

<table>
<thead>
<tr>
<th>Update</th>
<th>$f_{STmod}$</th>
<th>$f_{CR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero Hessian</td>
<td>3.4e-4 / 461</td>
<td>4.8e-3 / 10000</td>
</tr>
<tr>
<td>curv. only, $U=I$</td>
<td>4.4e-5 / 123</td>
<td>3.8e-3 / 10000</td>
</tr>
<tr>
<td>curv. only, $U=\text{rand}_w$</td>
<td>1.1e-11 / 73</td>
<td>1.7e-14 / 1853</td>
</tr>
<tr>
<td>PSB only</td>
<td>4.5e-13 / 23</td>
<td>6.4e-16 / 738</td>
</tr>
<tr>
<td>PSB &amp; curv., $U=I$</td>
<td>1.9e-8 / 34</td>
<td>6.2e-15 / 739</td>
</tr>
<tr>
<td>PSB &amp; curv., $U=\text{rand}_w$</td>
<td>2.3e-11 / 39</td>
<td>1.8e-16 / 824</td>
</tr>
</tbody>
</table>

($n = 10$, 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 \( \lambda \geq \lambda_{\text{min}} := \max\{0, -\lambda_{\text{min}}(H)\} \).
- Moré-Sorensen-type reparameterization:
  \[
  \lambda(t) := \lambda_{\text{min}} + \epsilon + \frac{1 - \lambda}{\epsilon + \lambda} \quad \text{for} \quad t \in [0, 1],
  \]
  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:

\[(\text{Dennis, Echebest, Guardarucci, Martinez, Scolnik, and Vacchino, 1991})\]

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

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

- 120 measurement values for fixing 12 parameters.
- Nonlinear least squares problem.
- Lousy result.
- ????
- Subcontractor guaranteed certain high accuracy 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 Test Results

- Actual Value
- Competitor #1
- Competitor #2
- MWD-Sensor

Graph showing the comparison of accuracy with varying Euler angles.
Other applications (Thanks to Roland Freund)

- Worst case performance of an algorithm?
- Example (with known answer): $k$ steps of cg algorithm.

\[ f : x \mapsto x^T A x - 2 b^T x + b^T A^{-1} b \equiv \|x - A^{-1} b\|_A^2 \]

- 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 $\text{cond}(A) \leq \kappa$.

(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 $\kappa = 10$:

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

(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 $\kappa = 100$.

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

($n = 200$ 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 $I \cdot D$ s.t. $D$ diagonal, $D \geq 0$, and $X - D \succeq 0$.
Factor $X - D = \hat{F} \hat{F}^T$, possibly omit some columns of $\hat{F}$.

• $20 \times 20$ 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 $\leq 300$
- 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:

\[
\begin{align*}
\text{minimize} & \quad f(x) \mid f_{E_1}(x) = 0, \quad A_{E_2}x = b_{E_2}, \\
& \quad f_{I_1}(x) \leq 0, \quad A_{I_2}x \leq b_{I_2}, \quad lb \leq x \leq ub,
\end{align*}
\]

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:

\[
\text{minimize } f(x) \mid f_{E_1}(x) = 0, \quad A_{E_2}x = b_{E_2}, \\
\quad f_{I_1}(x) \leq 0, \quad A_{I_2}x \leq b_{I_2}, \quad lb \leq x \leq ub,
\]

where all functions are assumed to be differentiable.

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

\[
\text{minimize } f(x) \mid f_E(x) = 0, \quad f_I(x) \leq 0, \quad Ax \leq b.
\]

(Lower dimension, loss of sparsity)

Preprocessing: \( b \geq 0 \).
“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.

\[ \begin{align*}
\text{minimize} & \quad g^T \Delta x + \frac{1}{2} \Delta x^T \tilde{H} \Delta x \\
\text{s.t.} & \quad Df_E \Delta x - s_E = -f_E, \\
& \quad Df_{\hat{i}} \Delta x - s_{\hat{i}} \leq -f_{\hat{i}}, \\
& \quad -s_{\hat{i}} \leq 0, \\
& \quad \hat{A} \Delta x \leq \hat{b}, \\
& \quad \| \Delta x \|^2 \leq \delta^2, \\
& \quad \| s_E \|^2 + \| s_{\hat{i}} \|^2 \leq 0.9 r_v^2 + 0.1(\| f_E \|^2 + \| f_{\hat{i}} \|^2). 
\end{align*} \]

- Three SOC constraints
Use Second Order Cone Programming?

- Two 2-norm cones (seem to be o.k.)
- Cholesky factor of $\tilde{\mathbf{H}} \succeq 0$ 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 reformulation?
- 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 initial 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)
  - (Smooth) minimization without using derivatives
- (Constrained version upon request)