A derivative-free trust-region algorithm using calculus rules to build the model function

Presenter: Gabriel Jarry-Bolduc gabjarry@alumni.ubc.ca
Presentation based on research done with Warren Hare

July 18, 2022.

The University of British Columbia- Okanagan Campus

Goals

- Compare two versions of a derivative-free trust-region algorithm:
  - One version employs a calculus approach to build the model function.
  - The second version employs a non-calculus approach to build the model function.

Establishing the context

- The optimization problem considered is

$$\min_{\ell \leq x \leq u} F(x)$$

where

Establishing the context

- The optimization problem considered is

$$\min_{\ell \le x \le u} F(x)$$

where

- $F : \mathbb{R}^n \to \mathbb{R}$ is obtained by manipulating two blackboxes with a similar degree of expensiveness,
- F is $\mathcal{C}^2$ on the box,
- the inequalities $\ell \le x \le u$ are taken component-wise ($\ell_i \le x_i \le u_i \quad \forall i \in \{1, \ldots, n\}$).

Form of F considered

In this presentation, we consider two different cases for F:

1. F is the **product** of two blackboxes f_1 and f_2 :

$$F = f_1 \cdot f_2,$$

   where $f_1 : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^2$, $f_2 : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^2$.

2. F is the **quotient** of two blackboxes f_1 and f_2 :

$$F = \frac{f_1}{f_2},$$

   where $f_1 : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^2$, $f_2 : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^2$ and $f_2(x) \neq 0$ for any x in the box.

What is a blackbox?

A **blackbox** is any process that returns an output whenever we provide an input, but the mechanism of the process is not analytically available to the optimizer.

$$\xrightarrow{\textit{input}:\ x} \boxed{?} \xrightarrow{\textit{output}:\ f(x)}$$

- Example: Computer simulations, laboratory experiments.

The main algorithm

- It is a derivative-free trust-region algorithm.

The main algorithm

- It is a derivative-free trust-region algorithm.
- It is inspired by the the pseudo-code presented in [Conn, Scheinberg and Vicente, 2009] and [Hough and Roberts, 2022].

The main algorithm

- It is a derivative-free trust-region algorithm.
- It is inspired by the the pseudo-code presented in [Conn, Scheinberg and Vicente, 2009] and [Hough and Roberts, 2022].
- It has been adapted for box constrained optimization problems by considering the projected gradient onto the box.

The main algorithm

- It is a derivative-free trust-region algorithm.
- It is inspired by the the pseudo-code presented in [Conn, Scheinberg and Vicente, 2009] and [Hough and Roberts, 2022].
- It has been adapted for box constrained optimization problems by considering the projected gradient onto the box.
- The convergence theory may be derived from the recent paper by Hough and Roberts: *Model-based derivative-free methods for convex-constrained optimization (2022).*

Details on the model function

- The model m at iteration k, denoted m^k, has the form

$$m^k(x) = F(x^k) + (g^k)^\top (x - x^k) + \frac{1}{2}(x - x^k)^\top H^k (x - x^k),$$

where

- x^k is the incumbent solution,
- g^k is an approximation of the gradient $\nabla F(x^k)$,
- H^k is a symmetric approximation of the Hessian $\nabla^2 F(x^k)$.

Model function continued

- Letting $x = x^k + s^k$, where $s^k \in \mathbb{R}^n$ is a step direction, the model can be written as

$$m^k(x^k + s^k) = F(x^k) + (g^k)^\top s^k + \frac{1}{2}(s^k)^\top H^k s^k.$$

How do we build g^k and H^k?

- Let $Q_F(x^k)$ be a quadratic interpolation function of F at x^k using the $(n+1)(n+2)/2$ distinct sample points

$$x^k, \quad x^k \oplus h\,\mathrm{Id}, \quad x^k \oplus h\,\mathrm{Id} \oplus h\,\mathrm{Id}$$

where $h \neq 0$.

How do we build g^k and H^k?

- Let $Q_F(x^k)$ be a quadratic interpolation function of F at x^k using the $(n+1)(n+2)/2$ distinct sample points

$$x^k, \quad x^k \oplus h\,\mathrm{Id}, \quad x^k \oplus h\,\mathrm{Id} \oplus h\,\mathrm{Id}$$

where $h \neq 0$.

Non-calculus approach

H^k: It is $\nabla^2 Q_F(x^k)$, the Hessian of the quad. interpolation function Q_F.

g^k : It is $\nabla Q_F(x^k)$, the gradient of the quad. interpolation function Q_F.

Calculus approach

- When $F = f_1 \cdot f_2$,

$$H^k = f_2(x^k)\nabla^2 Q_{f_1}(x^k) + \nabla Q_{f_1}(x^k)\left(\nabla Q_{f_2}(x^k)\right)^\top + \nabla Q_{f_2}(x^k)\left(\nabla Q_{f_1}(x^k)\right)^\top + f_1(x^k)\nabla^2 Q_{f_2}(x^k),$$

and

$$g^k = f_1(x^k)\nabla Q_{f_2}(x^k) + f_2(x^k)\nabla Q_{f_1}(x^k).$$

Calculus approach

- When $F = \frac{f_1}{f_2}$,

$$
\begin{aligned}
H^k = \frac{1}{[f_2(x^k)]^3} \Big[& [f_2(x^k)]^2 \nabla^2 Q_{f_1}(x^k) - f_1(x^k) f_2(x^k) \nabla^2 Q_{f_2}(x^k) \\
& + 2 f_1(x^k) \nabla Q_{f_2}(x^k) \nabla Q_{f_2}(x^k)^\top \\
& - f_2(x^k) \left(\nabla Q_{f_1}(x^k) \nabla Q_{f_2}(x^k)^\top + \nabla Q_{f_2}(x^k) \nabla Q_{f_1}(x^k)^\top \right) \Big],
\end{aligned}
$$

and

$$
g^k = \frac{f_2(x^k) \nabla Q_{f_1}(x^k) - f_1(x^k) \nabla Q_{f_2}(x^k)}{[f_2(x^k)]^2}.
$$

Calculus approach

- When $F = \frac{f_1}{f_2}$,

$$
\begin{aligned}
H^k = \frac{1}{[f_2(x^k)]^3} \Big[& [f_2(x^k)]^2 \nabla^2 Q_{f_1}(x^k) - f_1(x^k) f_2(x^k) \nabla^2 Q_{f_2}(x^k) \\
& + 2 f_1(x^k) \nabla Q_{f_2}(x^k) \nabla Q_{f_2}(x^k)^\top \\
& - f_2(x^k) \left(\nabla Q_{f_1}(x^k) \nabla Q_{f_2}(x^k)^\top + \nabla Q_{f_2}(x^k) \nabla Q_{f_1}(x^k)^\top \right) \Big],
\end{aligned}
$$

and

$$
g^k = \frac{f_2(x^k) \nabla Q_{f_1}(x^k) - f_1(x^k) \nabla Q_{f_2}(x^k)}{[f_2(x^k)]^2}.
$$

- For both approaches, H^k and g^k are obtained with $(n+1)(n+2)/2$ function evaluations.

Accuracy of the techniques

- The Hessian of Q_F is a $\mathcal{O}(h)$ accurate approximation of the Hessian at x^k.
- The gradient of Q_F is $\mathcal{O}(h^2)$.

Accuracy of the techniques

- The Hessian of Q_F is a $\mathcal{O}(h)$ accurate approximation of the Hessian at x^k.
- The gradient of Q_F is $\mathcal{O}(h^2)$.

- The calculus approach to approximate the Hessian and the gradient are also
  - $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$ respectively [Chen, Hare, Jarry-Bolduc, 2022].

Theoretical advantages of the calculus approach

The non-calculus approach:
If f_1, f_2 are linear functions, then g^k and H^k are perfectly accurate.

Theoretical advantages of the calculus approach

The non-calculus approach:
If f_1, f_2 are linear functions, then g^k and H^k are perfectly accurate.

We can do better than that with the calculus based approach!

The calculus approach:
If f_1, f_2 are quadratic functions, then H^k and g^k are perfectly accurate.

(A calculus approach also allows to use different approximation techniques depending on the sub-function).

Theoretical advantages of the calculus approach

The non-calculus approach:
If f_1, f_2 are linear functions, then g^k and H^k are perfectly accurate.

We can do better than that with the calculus based approach!

The calculus approach:
If f_1, f_2 are quadratic functions, then H^k and g^k are perfectly accurate.

(A calculus approach also allows to use different approximation techniques depending on the sub-function).

- Will it make a significant difference in an algorithm?

Numerical experiments

Implementation

- Two versions of a derivative-free trust-region algorithm have been implemented in Matlab2021b.
- The initial values for the parameters have been influenced by preliminary numerical results and the values proposed in *Trust region methods*, Chapter 6.

The values of the parameters

$$
\begin{aligned}
\Delta_t^0 &= 1 && \text{(initial trust-region radius)},\\
\Delta_{t\ \max} &= 1e+03 && \text{(maximal trust-region radius)},\\
\Delta_s^0 &= 1 && \text{(Initial sampling radius)},\\
\Delta_{s\ \min} &= 1e-03 && \text{(minimal sampling radius)},\\
\Delta_{s\ \max} &= 1 && \text{(maximal sampling radius)},\\
\eta_1 &= 0.1 && \text{(parameter for accepting the trial point)},\\
\eta_2 &= 0.9 && \text{(parameter for the trust-region radius update)},\\
\gamma &= 0.5 && \text{(parameter to decrease trust-region radius)},\\
\gamma_{inc} &= 2 && \text{(parameter to increase the trust-region radius)},\\
\epsilon_{stop} &= 1e-05 && \text{(parameter to verify optimality)},\\
\mu &= 1 && \text{(parameter to verify the size of the trust-region radius)}.
\end{aligned}
$$

More details on the model function

- Note: the sampling points are allowed to be taken out of the box constraint.
- Every time the incumbent solution x^k is updated, H^k and g^k are computed again so that the the model is always *fully linear* on the trust region ball.
- This requires $(n+1)(n+2)/2$ function evaluations.

Trust-region subproblem

- To solve the trust-region subproblem in Matlab, we use the quadprog with the algorithm trust-region reflective.

Comparison

Using data profiles with $\tau = 1e-01, 1e-03, 1e-05$, we compare two versions of our derivative-free trust-region algorithm:

- Version 1 builds the model with a non-calculus approach.
- Version 2 builds the model with a calculus approach.

- To check if our algorithms are not that bad compared to well-established algorithms, we include fmincon in the comparisons.

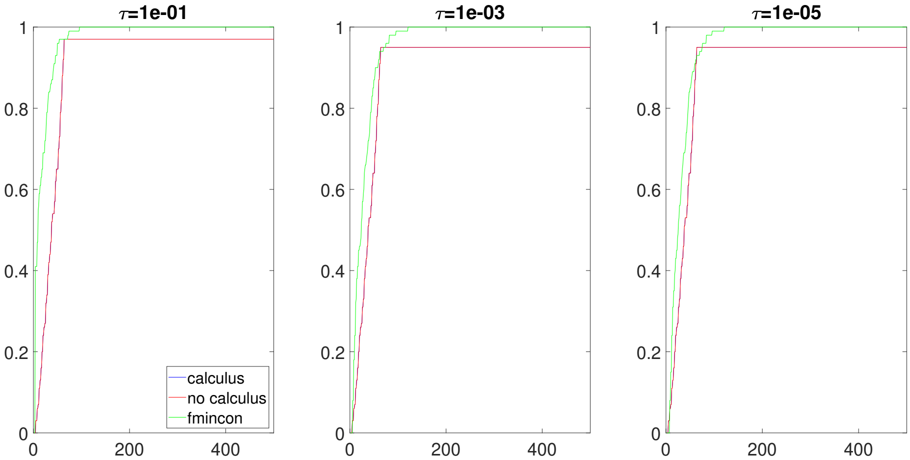
Details on the experiments

- f_1 and f_2 are taken to be linear functions or quadratic functions with random dimensions n between 1 and 30.
- The coefficients in f_1 and f_2 are generated randomly with **randi** (integers in $[-10, 10]$).
- The starting point $x^0 \in \mathbb{R}^n$ is generated with **randi** (each component is in $[-5, 5]$)
- The lower bound ℓ is set to $\ell_i = x_i^0 - 1$ for all $i \in \{1, \ldots, n\}$
- The upper bound u is set to $u_i = x_i^0 + 1$ for all i.
- We repeat 100 times each experiment.

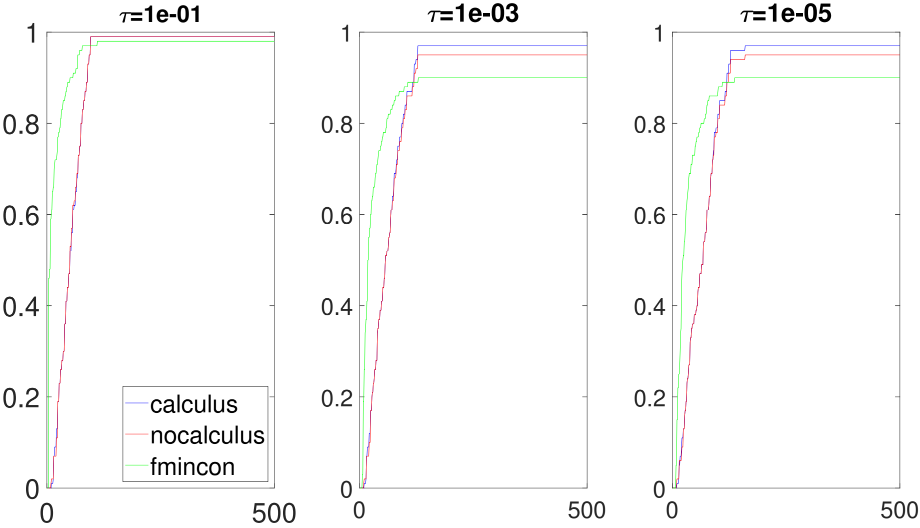
Experiment 1: product

- First, we investigate the case $F = f_1 \cdot f_2$ for the 3 following situations:
  - f_1: linear, f_2: linear,
  - f_1: quadratic, f_2: linear,
  - f_1: quadratic, f_2: quadratic.

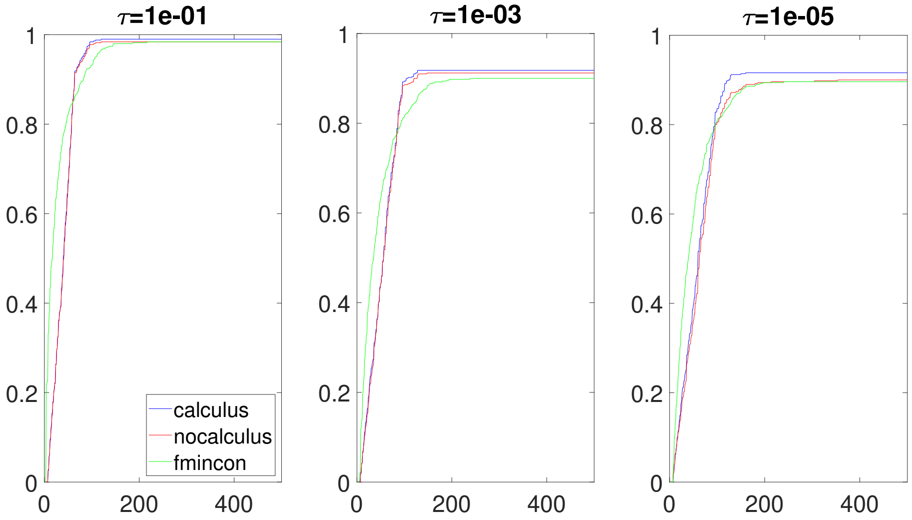
Data profiles, $F = f_1 \cdot f_2$, f_1 linear, f_2 linear



Data profiles, $F = f_1 \cdot f_2$, f_1 quadratic, f_2 linear



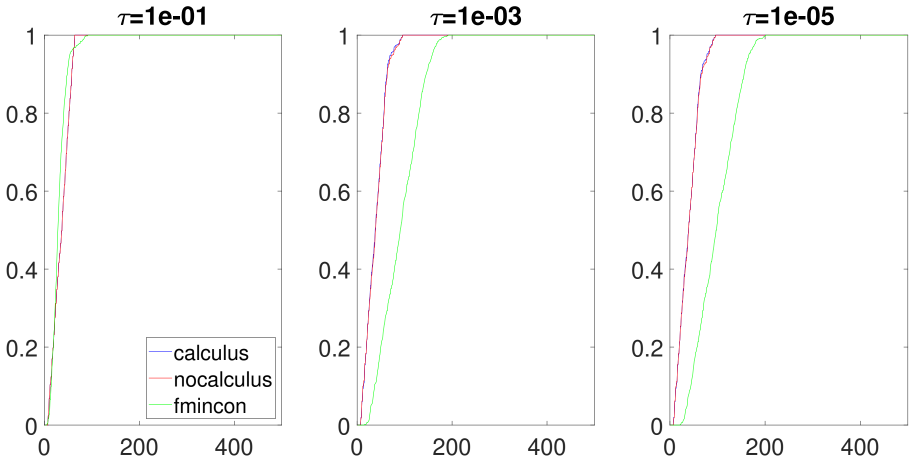
Data profiles, $F = f_1 \cdot f_2$, f_1 quadratic, f_2 quadratic



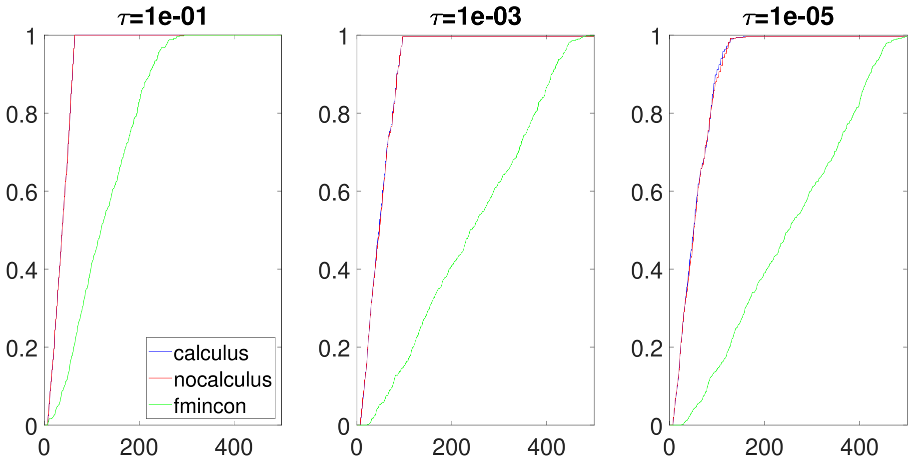
Experiment 2: easy quotient

- Second, we investigate the case $F = \frac{f_1}{f_2}$ for the following 4 situations:
  - f_1: linear, f_2: linear,
  - f_1: quadratic, f_2: linear,
  - f_1: linear, f_2: quadratic,
  - f_1: quadratic, f_2: quadratic.
- f_2 and the box are built so that there are no roots of f_2 close to the box.

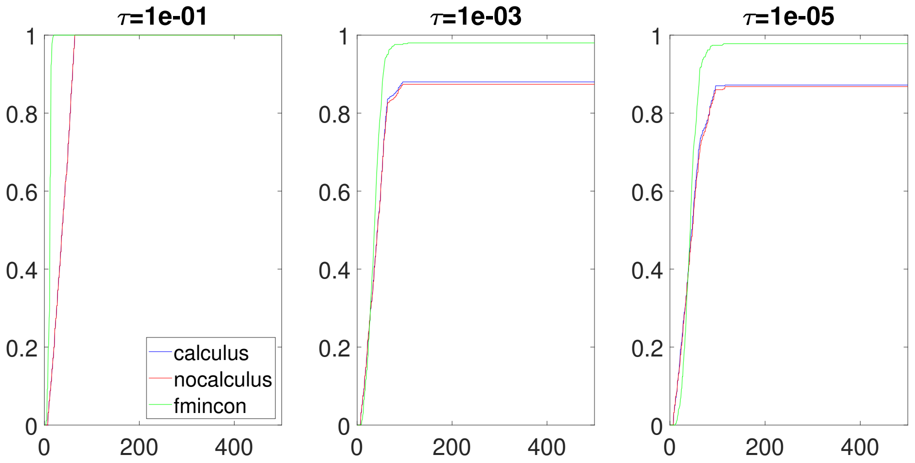
Data profiles, $F = \frac{f_1}{f_2}$, f_1 linear, f_2 linear



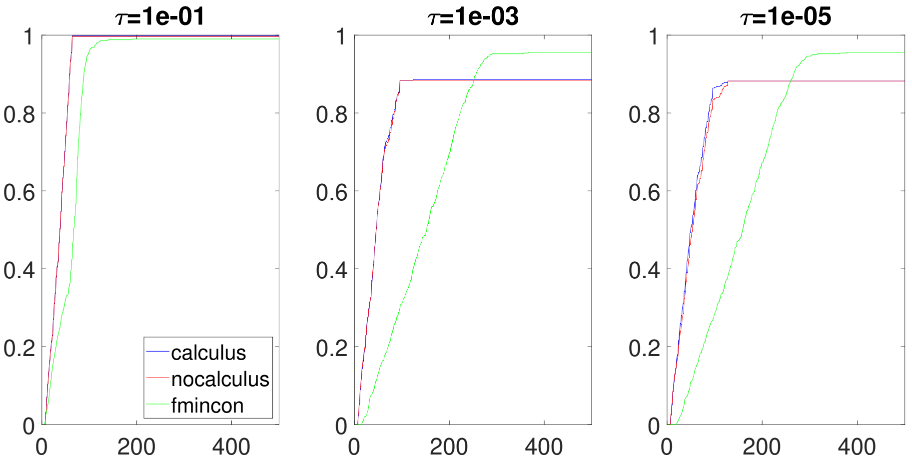
Data profiles, $F = \frac{f_1}{f_2}$, f_1 linear, f_2 quadratic



Data profiles, $F = \frac{f_1}{f_2}$, f_1 quadratic, f_2 linear



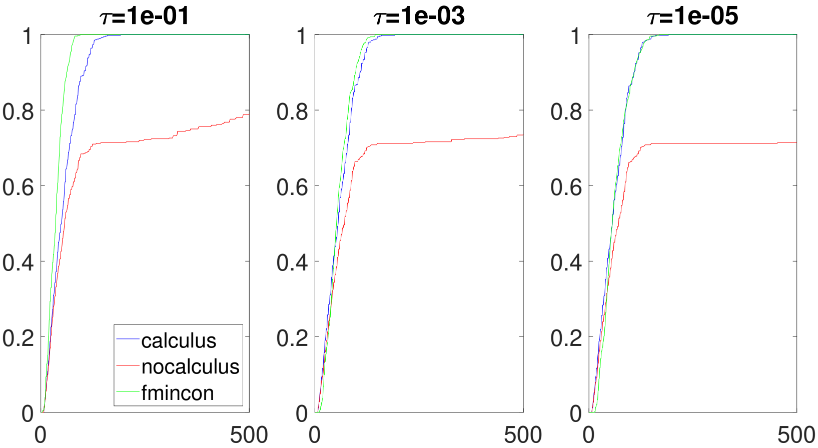
Data profiles, $F = \frac{f_1}{f_2}$, f_1 quadratic, f_2 quadratic



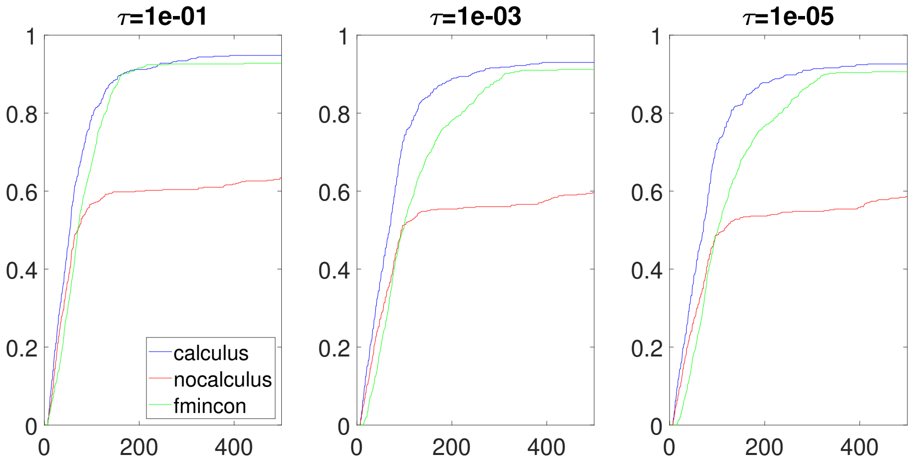
Experiment 3: hard quotient

- We repeat the experiments for $F = \frac{f_1}{f_2}$, but this time, we let a root of f_2 be near the box constraint.

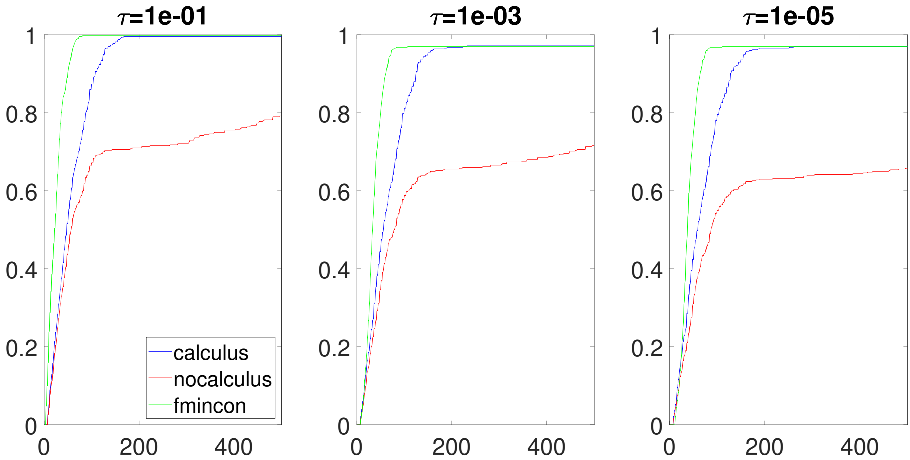
Data profiles, $F = \frac{f_1}{f_2}$, f_1 linear, f_2 linear



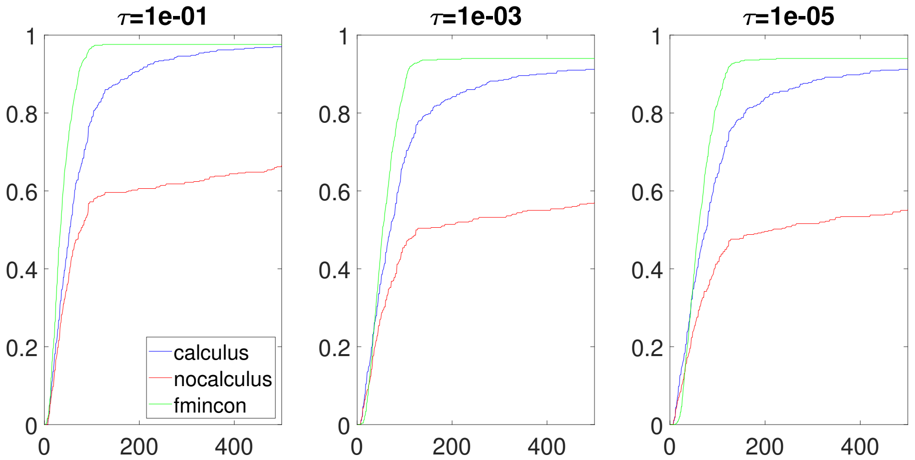
Data profiles, $F = \frac{f_1}{f_2}$, f_1 linear, f_2 quadratic



Data profiles, $F = \frac{f_1}{f_2}$, f_1 quadratic, f_2 linear



Data profiles, $F = \frac{f_1}{f_2}$, f_1 quadratic, f_2 quadratic



Analyzing the results

- The calculus approach is as good or better than the non-calculus approach on all experiments.

- The calculus approach is significantly better when $F = \frac{f_1}{f_2}$ and f_2 has a root near the box constraint.

Conclusion

- A calculus approach seems to improve the efficiency and robustness of our derivative-free trust-region algorithm.

Conclusion

- A calculus approach seems to improve the efficiency and robustness of our derivative-free trust-region algorithm.
- A calculus approach is not more difficult to implement than a non-calculus approach.

Conclusion

- A calculus approach seems to improve the efficiency and robustness of our derivative-free trust-region algorithm.
- A calculus approach is not more difficult to implement than a non-calculus approach.
- Another advantage of a calculus approach is that it allows to use different approximation techniques depending on the sub-function and/or different sample points.

Future research directions

- Consider other test sets.
- Integrate techniques to reuse sampling points.
- Find and solve a real-world problem that has this structure (product of two blackboxes or quotient of two blackboxes).

Papers related to this talk

[CHJ21] Y. Chen, W. Hare, and G. Jarry-Bolduc. "Error Analysis of Surrogate Models Constructed through Operations on Sub-models". In: *arXiv preprint arXiv:2112.08411* (2021).

[HJP20] W. Hare, G. Jarry-Bolduc, and C. Planiden. "Hessian approximations". In: *arXiv preprint arXiv:2011.02584* (2020).

Thank you!

Details on the sampling radius

- Each time a model m^k is built, it is *fully linear* on the trust region ball $B(x^k; \Delta_t^k)$ since the sampling radius to build g^k and H^k is set to

$$\Delta_s^k \leftarrow \min\{\Delta_s^k, \Delta_t^k\}.$$

- To ensure that the sampling radius is not too big, we then set

$$\Delta_s^k \leftarrow \min\{\Delta_s^k, \Delta_{s\,\max}\}.$$

- To decrease the risk of numerical errors, we finally set

$$\Delta_s^k \leftarrow \max\{\Delta_s^k, \Delta_{s\,\min}\}.$$