PDFO: Powell's Derivative-Free Optimization Solvers with MATLAB and Python Interfaces

Zaikun Zhang

Hong Kong Polytechnic University

Joint work with Tom M. Ragonneau (Ph.D. student)

May 13, 2020, ICMSEC, AMSS, CAS, Beijing

In deep memory of late Professor M. J. D. Powell (1936–2015)

Why optimize a function without using derivatives?

I started to write computer programs in Fortran at Harwell in 1962. ... after moving to Cambridge in 1976 ... I became a consultant for IMSL. One product they received from me was the TOLMIN package for optimization ... which requires first derivatives ... Their customers, however, prefer methods that are without derivatives, so IMSL forced my software to employ difference approximations ... I was not happy ... Thus there was strong motivation to try to construct some better algorithms.

— Powell

A view of algorithms for optimization without derivatives, 2007

Derivative-free optimization (DFO)

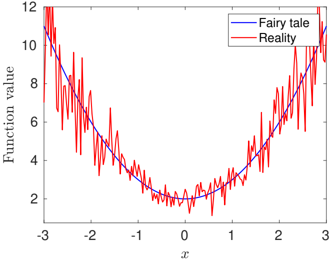
- Minimize a function f using function values but not derivatives.
- A typical case: f is a black box without an explicit formula.

$$x \longrightarrow \boxed{f} \longrightarrow f(x)$$

- Here, the reason for not using derivatives is not nonsmoothness!
- Do not use derivative-free optimization methods if any kind of (approximate) first-order information is available.
- Regarding your problem as a pure black box is generally a bad idea. It is more often a gray box. Any known structure should be explored.

DFO is no fairy-tale world

- The black box defining f can be extremely noisy.



(Yes, this is your favorite convex quadratic function)

- The function evaluation can be extremely expensive.
- The budget can be extremely low.

> (In real applications) ... one almost never reaches a solution but even 1% improvement can be extremely valuable.

— Conn
Inversion, history matching, clustering and linear algebra, 2015

About the name(s)

- Talking about optimization methods that do not use derivatives, Powell called them direct search optimization methods or optimization without derivatives, but never derivative-free optimization.
- These days, "direct search methods" refers to a special class of methods.
- Problems that only provide function values are often categorized as black-box optimization or simulation-based optimization.

Applications

- Colson, *et al.*, Optimization methods for advanced design of aircraft panels: a comparison, 2010.
- Ciccazzo, *et al.*, Derivative-free robust optimization for circuit design, 2015
- Wild, Sarich, and Schunck, Derivative-free optimization for parameter estimation in computational nuclear physics, 2015
- Campana, *et al.*, Derivative-free global ship design optimization using global/local hybridization of the direct algorithm, 2016
- Ghanbari and Scheinberg, Black-box optimization in machine learning with trust region based derivative free algorithm, 2017

No applications by Powell, because ...

> The development of algorithms for optimization has been my main field of research for 45 years, but I have given hardly any attention to applications. It is very helpful, however, to try to solve some particular problems well, in order to receive guidance from numerical results, and in order not to be misled from efficiency in practice by a desire to prove convergence theorems. ... I was told ... that the DFP algorithm (Fletcher and Powell, 1963) had assisted the moon landings of the Apollo 11 Space Mission.

— Powell

A view of algorithms for optimization without derivatives, 2007

Well-developed theory and methods

- Powell, Direct search algorithms for optimization calculations, 1998
- Powell, A view of algorithms for optimization without derivatives, 2007
- Conn, Scheinberg, and Vicente, Introduction to Derivative-Free Optimization, 2007
- Audet and Warren, Derivative-Free and Blackbox Optimization, 2017
- Larson, Menickelly, and Wild, Derivative-free optimization methods, 2019

Two classes of methods

- Trust-region methods: iterates are defined based on minimization of models of the objective function in adaptively chosen trust regions.
  - Examples: Powell's methods are trust-region method based on linear or quadratic models built by interpolation.
- Direct search methods: iterates are defined based on comparison of objective function values without building models.
  - Examples: Simplex method (Nelder and Mead, 1965), Implicit Filtering (Gilmore and Kelley, 1995), `GPS` (Torczon, 1997), `MADS` (Audet and Dennis, 2006), `BFO` (Porcelli and Toint, 2015), ...

Basic idea of trust-region methods

$$x_{k+1} \;\approx\; x_k + \operatorname*{argmin}_{\|d\|\le\Delta_k} m_k(x_k + d)$$

- m_k is the trust-region model and $m_k(x) \approx f(x)$ around x_k.
  - When derivatives are available: Taylor expansion or its variants (Newton, quasi-Newton, ...)
  - When derivatives are unavailable: interpolation/regression
  - Applicable in nonsmooth case: Yuan (1983 and 1985), Grapiglia, Yuan, Yuan (2016)
- $\|d\| \le \Delta_k$ is the trust-region constraint.
- Δ_k is the adaptively chosen trust-region radius.
- x_{k+1} may equal x_k.
- I am abusing the notation argmin (in multiple ways).

Trust-region framework

Algorithm (Trust-region framework for unconstrained optimization)

Pick $x_0 \in \mathbb{R}^n$, $\Delta_0 > 0$, $0 \le \eta_1 \le \eta_2 < 1$, $\eta_2 > 0$, and $0 < \gamma_1 < 1 < \gamma_2$. $k := 0$.

Step 1. Construct a model

$$m_k(x) \;\approx\; f(x) \quad \text{around} \quad x_k.$$

Step 2. Obtain a trial step d_k by solving (inexactly)

$$\min_{\|d\| \le \Delta_k} \; m_k(x_k + d).$$

Step 3. Evaluate the reduction ratio $\rho_k = \dfrac{f(x_k) - f(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)}$, and set

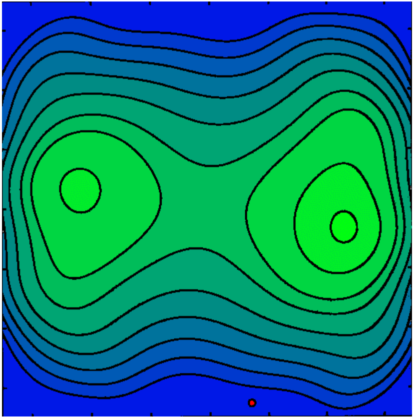
$$x_{k+1} = \begin{cases} x_k & \text{if } \rho_k \le \eta_1 \\ x_k + d_k & \text{if } \rho_k > \eta_1 \end{cases}, \qquad \Delta_{k+1} \begin{cases} = \; \gamma_1 \Delta_k & \text{if } \rho_k \le \eta_2 \\ \in \; [\Delta_k, \gamma_2 \Delta_k] & \text{if } \rho_k > \eta_2 \end{cases}.$$

Increment k by 1. Go to **Step 1**.

Typical parameters: $\eta_1 = 0$, $\eta_2 = 1/10$, $\gamma_1 = 1/2$, $\gamma_2 = 2$.

Note: The framework needs to be adapted if derivatives are unavailable.

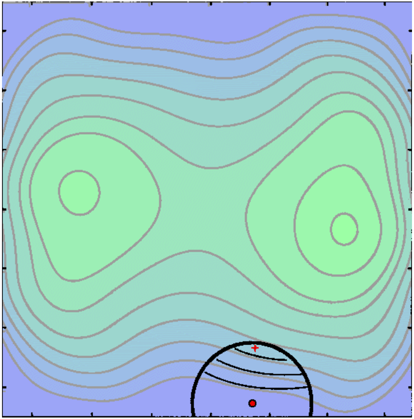
An illustration of trust-region method



An illustration of trust-region method [1]

[1]Images by Dr. F. V. Berghen from `http://www.applied-mathematics.net`.

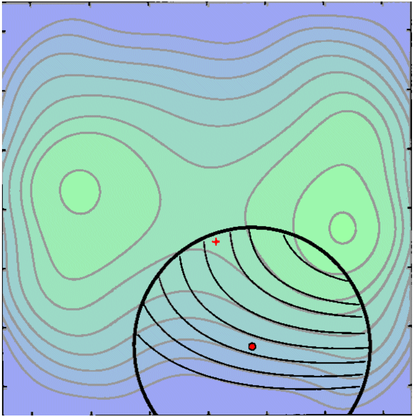
An illustration of trust-region method



An illustration of trust-region method [1]

[1] Images by Dr. F. V. Berghen from `http://www.applied-mathematics.net`.

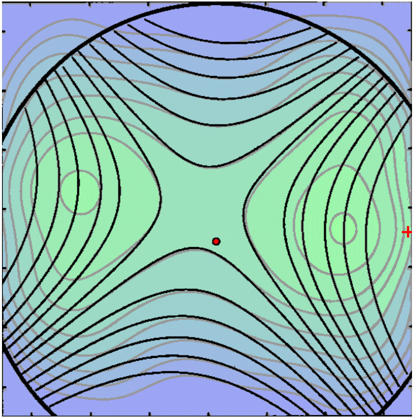
An illustration of trust-region method



An illustration of trust-region method [1]

[1] Images by Dr. F. V. Berghen from `http://www.applied-mathematics.net`.

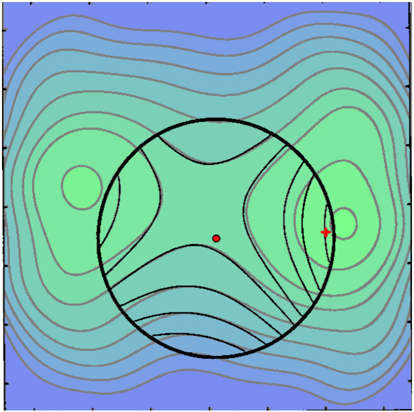
An illustration of trust-region method



An illustration of trust-region method [1]

[1] Images by Dr. F. V. Berghen from `http://www.applied-mathematics.net`.

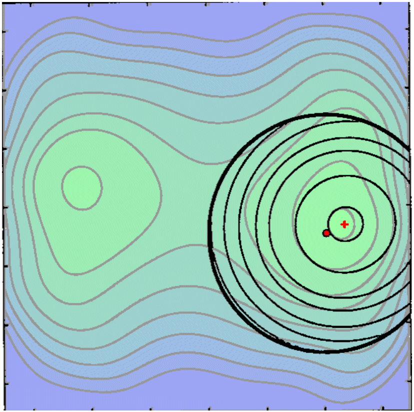
An illustration of trust-region method



An illustration of trust-region method [1]

[1] Images by Dr. F. V. Berghen from `http://www.applied-mathematics.net`.

An illustration of trust-region method



An illustration of trust-region method [1]

[1] Images by Dr. F. V. Berghen from `http://www.applied-mathematics.net`.

Powell's algorithms and Fortran solvers (I)

Powell's second paper on optimization:

Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, 1964

- It is also Powell's second most cited paper (4991 on Google Scholar as of May 13, 2020).
- The method is known as Powell's conjugate direction method.
- Powell did not release his own implementation.

Powell's algorithms and Fortran solvers (II)

- COBYLA: solving general nonlinearly constrained problems using linear models; code released in 1992; paper published in 1994
- UOBYQA: solving unconstrained problems using quadratic models; code released in 2000; paper published in 2002
- NEWUOA: solving unconstrained problems using quadratic models; code released in 2004; paper published in 2006
- BOBYQA: solving bound constrained problems using quadratic models; code released and paper written in 2009
- LINCOA: solving linearly constrained problems using quadratic models; code released in 2013; no paper written
- Maybe COBYQA in heaven ...

Quadratic models in UOBYQA

- UOBYQA maintains an interpolation set Y_k and decides m_k by

$$m_k(y) = f(y), \quad y \in Y_k.$$

- The above condition is a linear system of the coefficients of m_k.
- In $\mathbb{R}^n$, Y_k consists of $\mathcal{O}(n^2)$ points. (Why?)
- Most points in Y_k are recycled. Y_{k+1} differs from Y_k by 2 points.
- It is crucial to make sure that Y_k has "good geometry".
- Normally, UOBYQA cannot solve large problems.
- UOBYQA may solve large problems by parallel function evaluations.

Quadratic models in NEWUOA, BOBYQA, and LINCOA

Underdetermined interpolation with much less function evaluations:

$$\min \|\nabla^2 m_k - \nabla^2 m_{k-1}\|_{\mathsf{F}}$$
$$\text{s.t. } m_k(y) = f(y), \quad y \in Y_k.$$

- In general, $|Y_k| = \mathcal{O}(n)$.
- The idea originates from the least change properties of quasi-Newton methods, of which DFP was the first one.
- The objective can be generalized to a functional $\mathcal{F}$ measuring the regularity of a model m. For instance:

$$\mathcal{F}(m) = \|\nabla^2 m - \nabla^2 m_{k-1}\|_{\mathsf{F}}^2 + \sigma_k \|\nabla m(x_k) - \nabla m_{k-1}(x_k)\|_2^2.$$

See Powell (2012) and Z. (2014).

Capability of Powell's solvers

> Perhaps foremost among the limitations of derivative-free methods is that, on a serial machine, it is usually not reasonable to try and optimize problems with more than a few tens of variables, although some of the most recent techniques (NEWUOA) can handle unconstrained problems in hundreds of variables.

— Conn, Scheinberg, and Vicente
Introduction to Derivative-Free Optimization, 2007

> LINCOA is not suitable for very large numbers of variables because no attention is given to any sparsity. A few calculations with 1000 variables, however, have been run successfully overnight ...

— Powell
Comments in the Fortran code of LINCOA, 2013

PDFO: MATLAB/Python interfaces for Powell's solvers

- Powell's Fortran solvers are artworks. They are robust and efficient.
- Not everyone can (or has the chance to) appreciate artworks.
- Less and less people can use Fortran, let alone Fortran 77.
- PDFO provides user-friendly interfaces for calling Powell's solvers.
- PDFO supports currently MATLAB and Python. More will come.
- PDFO supports various platforms: Linux, Mac, and even Windows.
- PDFO is not MATLAB/Python implementations of Powell's solvers.

PDFO homepage: `www.pdfo.net`

Bayesian optimization

- Regard f as a Gaussian process (i.e., it is a considered as a function that returns random values).
- Start with a prior model (aka, surrogate) f_0 of f.
- At iteration k, using the data $\mathcal{D}_k$ up to now to update the model of f, obtaining a posterior model f_k of f :

$$f_k \;=\; \text{posterior of } f \text{ given prior } f_{k-1} \text{ and information } \mathcal{D}_k.$$

- Based on f_k, define an acquisition function $u(\cdot \mid \mathcal{D}_k)$ (e.g., expected improvement). Let

$$x_{k+1} \;=\; \operatorname*{argmin}_x u(x \mid \mathcal{D}_k).$$

- Observe (i.e., evaluate) f at x_{k+1}, obtaining y_{k+1}, and update

$$\mathcal{D}_{k+1} \;=\; \mathcal{D}_k \cup \{(x_{k+1},\, y_{k+1})\}.$$

- Iterate the above procedure.

Many advantages of Bayesian optimization

- Little assumption on the objective function (if any).
- Can handle general variables (continuous, integer, categorical, ...).
- Designed for global optimization (in theory ...).
- The idea is easy to understand and attractive (this is important!).
- Popular among engineers (being popular is surely an advantage!).

Comparison I: A synthetic noisy smooth problem

Chained Rosenbrock function (Powell, 2006):

$$f(x) = \sum_{i=1}^{n-1} [4(x_{i+1} - x_i^2)^2 + (1 - x_i)^2].$$

Observed value:

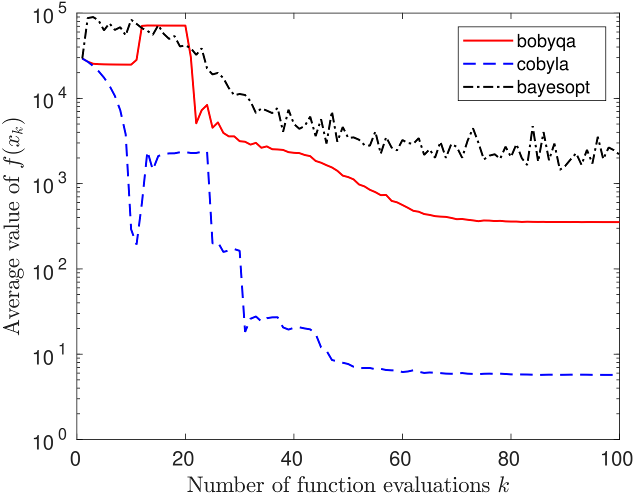
$$F(x) = f(x)[1 + \sigma\, e(x)],$$

where $e(x)$ is a random variable that follows either $\mathrm{U}([-1, 1])$ or $\mathrm{N}(0, 1)$.

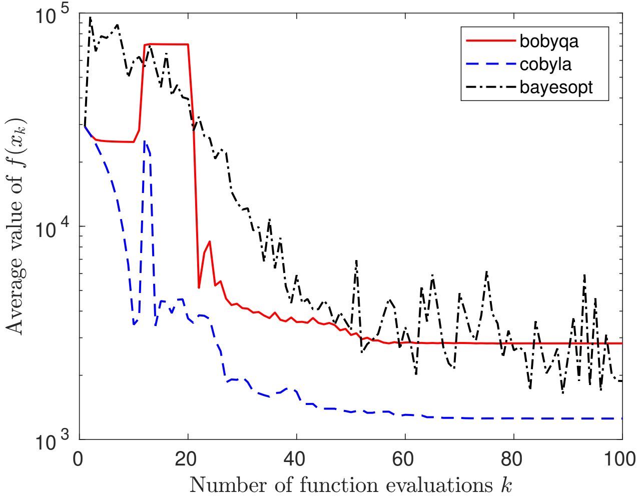
In our experiments:

- dimension: $n = 10$
- constraints: $-10 \le x_i \le \dfrac{i}{n}$, $i = 1, 2, \ldots, n$
- noise level: $\sigma = 0.1$
- starting point: mid point between lower and upper bounds
- budget: 100 function evaluations
- Bayesian optimizer: function `bayesopt` in MATLAB
- number of random experiments: 20

Uniform noise



Gaussian noise



Comparison II: A synthetic noisy nonsmooth problem

A Rosenbrock-like nonsmooth function:

$$f(x) \ = \ \sum_{i=1}^{n-1} (4|x_{i+1} - x_i^2| + |1 - x_i|).$$

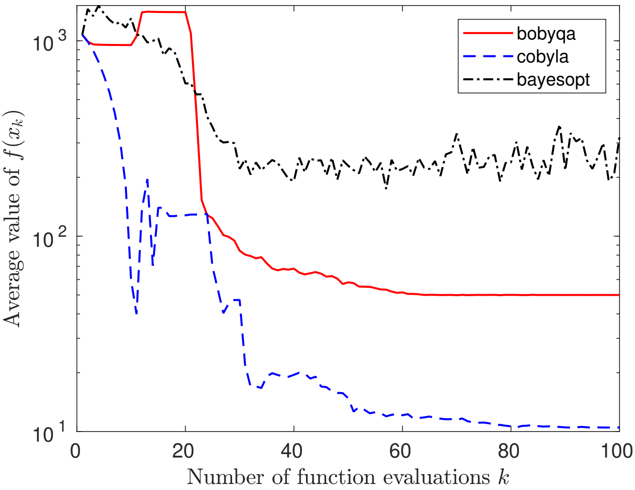
Observed value:

$$F(x) \ = \ f(x)[1 + \sigma\, e(x)],$$

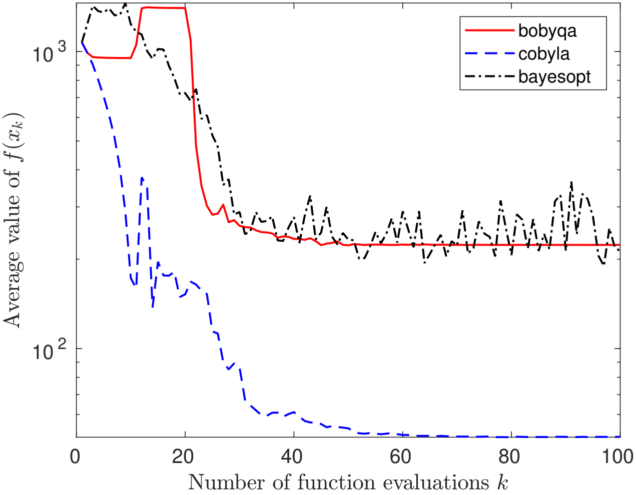
where $e(x)$ is a random variable that follows either $\mathrm{U}([-1,1])$ or $\mathrm{N}(0,1)$. The settings of the experiment is the same as the smooth case.

Note:
Powell's solvers are not (particularly) designed for nonsmooth problems. No theoretical guarantee about the behavior of the solvers in such a scenario.

Uniform noise



Gaussian noise



Summary

- Basic ideas of Powell's derivative-free optimization solvers
- PDFO: MATLAB/Python interfaces for Powell's Fortran solvers
- A brief comparison with Bayesian optimization

Thank you!

zaikun.zhang@polyu.edu.hk

PDFO homepage: www.pdfo.net