Powell, Yuan–Stoer, and OptimIST

Model-Based and Subspace Derivative-Free Optimization Algorithms

ZHANG Zaikun

Sun Yat-sen University

MoCaO Lecture, July 29, 2026

A recent **theoretical** paper (not the topic of today)

C. Huang and Z. Zhang, Non-convergence Analysis of Probabilistic Direct Search, arXiv:2606.01320, 2026

Derivative-free optimization (DFO)

- Minimize f using function values, not 1st-order info., e.g., derivatives.
- Example 1: f is a black box given by simulations; no formula (yet).
- Example 2: 1st-order info. of f is possible to get, but too expensive.
- Even the evaluation of f is often expensive and noisy.

Derivative-free optimization (DFO)

- Minimize f using function values, not 1st-order info., e.g., derivatives.
- Example 1: f is a black box given by simulations; no formula (yet).
- Example 2: 1st-order info. of f is possible to get, but too expensive.
- Even the evaluation of f is often expensive and noisy.
- Aim: Minimize/reduce f using as few function evaluations as possible.
- We may also have constraints whose 1st-order info. is not accessible.
- Examples of methods: Nelder-Mead, Powell's algorithms, SOLNP+, ...

The late Professor M. J. D. Powell and DFO

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.

— M. J. D. Powell

A view of algorithms for optimization without derivatives, 2007

Professor Ya-xiang Yuan and DFO

袁亚湘

中国科学院院士袁亚湘：

这是年轻人学术“独立”路上的一件利器

■本报记者 韩扬眉

“非线性规划的直接方法”，30 多年前，袁亚湘认真写下青年科学基金项目的标题。

这是国家自然科学基金青年科学基金项目（以下简称青年基金）早期申请书中的一份。为了发现和培养人才，促进优秀青年科学工作者脱颖而出，1986 年国家自然科学基金委员会在成立的当年就准备设立青年基金项目，并于 1987 年开放青年基金的申请和资助。

对科研生涯的这一起点，作为早期青年基金获得者，中国科学院院士袁亚湘的感受是，“对年轻人来说，博士刚毕业那几年可能是科研生涯最重要的成长阶段。青年基金为年轻人独立、稳定做自己想做的研究提供了一份支持”。

独立、安心做自己想做的事情

1988 年 9 月，袁亚湘从剑桥大学学成归国，随后成为获青年基金资助的科研人员。1.3 万元的经费，对于当时数学研究正处于起步阶段的袁亚湘来说，“是一笔不少的费用，足够开展一些研究”。

为什么要将“非线性规划的直接方法”作为回国后最先开展的研究？

事实上，非线性规划为工业、交通运输、经济管理等领域的“最优设计”提供了重要的数学工具和方法。它形成于 20 世纪 50 年代，并于七八十年代在国际上得到了快速发展。

袁亚湘在剑桥大学的博士生导师 M.J.D. Powell 便是非线性规划的顶尖专家。那时国内关于非线性规划直接方法的研究几乎一片空白，袁亚湘希望回国之后，把国际上新兴的研究方向引到国内，同时培养一些学生，开辟一些研究领域。

“青年基金实际上培养了刚毕业研究生的‘独立’。”袁亚湘强调，要在未来成为科学家，最重要的就是独立。

回到熟悉的中科院数学与系统科学研究院，袁亚湘不再是学生，而是一名研究人员。他成为优化研究组组长、课题负责人。

青年基金作为资助之一，支持袁亚湘心无旁骛潜心钻研非线性规划的理论方法，更重要的是，帮助他很快在学术上“自立门户”，逐步成长为一名科学家。

袁亚湘直言，很多年轻人做研究“后劲不足”，主要原因正是缺乏独立性。读博士时做研究跟着导师做，题目由导师分配，一旦没有人带领便不会做研究或者做不好，就是独立性没有培养起来。

“青年基金提供资助，关键在于稳定支持年轻人学会‘脱离’导师，有独立空间自主选题，独立做自己想做的研究，这对培养一批有潜力的年轻人很重要。”袁亚湘说。

如今，袁亚湘已主持承担过 10 余项重要的基金项目，他依然觉得青年基金十分关键。“它的本质不在于最终获得一个好的成果，而是给年轻人一个长期投入、安心做研究的稳定支持的机会。”

多“养”一些年轻人，让他们都去赛跑

“青年基金就是要‘养’更多的年轻人，让他们参与竞争和赛跑，并提供一个生存的环境和条件。”袁亚湘告诉《中国科学报》。

他以数学为例指出，对于数学研究来说，首先要把队伍壮大。“数学研究更青睐年轻人。可以给每个人的资助相对少一点，但面一定要撒得广一点。多‘养’几个人，让他们都去跑。谁跑得快，最后自然就冒出来了。”

的确，数学被认为是年轻人的事业，欧拉、高斯等最杰出的数学家都是在年轻时做出极具影响力的成果。被誉为“数学界诺贝尔奖”的菲尔兹奖只授予 40 岁以下的年轻数学家。

“还要更多关注那些还没有崭露头角的年轻人。对他们来说，青年基金就是‘雪中送炭’。”袁亚湘进一步指出，支持年轻人，本质在于让他们能够在一段时间内少有压力，潜心、安心地做研究，在学术上顺利成长，而并非期望他们在短时间内做出很有突破性的成果。

究竟应该“养”什么样的年轻人？

在袁亚湘看来，对于基础研究来说，需要一大批真正热爱科学、有科学精神、真正想做科研的人。“基础研究需要研究人员甘于寂寞、执着钻研，青年基金就是要遴选出这些有兴趣、有能力的人。”

“不是为了‘谋生’而去申请科学基金，而是真正想解决一些科学问题。”袁亚湘叮嘱年轻人，在成长过程中，要注重提高自己的学术水平、学术造诣和学术视野。

评价回归适度，优秀人才、科学问题考核不出来

采访结束后，袁亚湘找出 28 年前他的青年基金项目申请书发给《中国科学报》，其中“拟开展的工作研究设想”仅有一页纸。

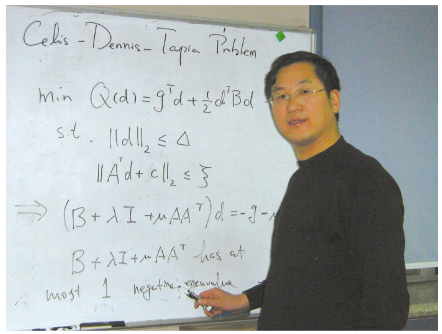
青年基金的申请给袁亚湘留下了“短平快”的印象。虽然其最终呈现在纸上的文字寥寥，但背后凝结着他对科学问题长久而不间断的思考。

“有时看起来是‘灵光闪现’，但实际上‘灵光’可能是一个人经过长期不受干扰地思考、不断认真积累才出现的。”袁亚湘说。

然而，现实却时常让他感到担忧——袁亚湘发现，现在的青年学把准备基金申请书当作一个“大工程”，花费大时间准备“本子”，而非思考科学问题。“大家都样做。这给人一种感觉，好像写得越多就越好委就会越看重。”

造成这种现象的原因来自青年人才的竞竞争和单位的考核压力等多方面。袁亚湘建过于简单和过于复杂都不可取，要回归到“的申请”，让年轻人在准备申请书上不耗费精力，而应该把重心放在科学问题的凝练上。实现的关键之一，在于科学基金不断地深革，让“懂行”的专家评审，评委们需要在平多地发现并了解值得支持的年轻人。

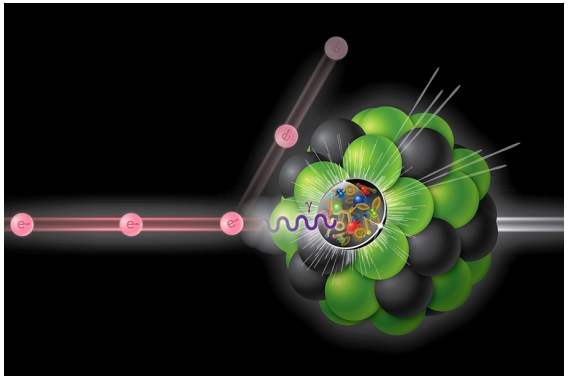
“考核需适度。对于真正的科学家来说，做出好的成果从来不是靠考核评估‘逼’出来更多的是来自内心驱动。”袁亚湘强调，对于人来说，有了稳定支持，减少后顾之忧，心情了，自然能做出好成果。



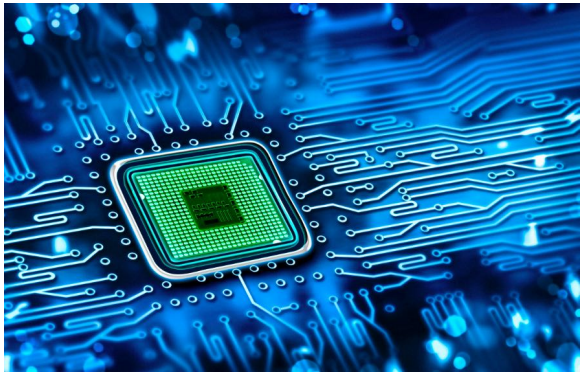
“Direct Methods for Nonlinear Programming”, more than 30 years ago, Ya-xiang Yuan wrote down carefully the title for his NSFC early-career grant proposal.

— *China Science Daily*, April 25, 2022

Applications



Nuclear physics



Circuit design

Artificial intelligence

- Wang *et al.*, Zeroth-order federated fine-tuning for large AI models in resource-constrained wireless networks, IEEE Trans. Wirel. Commun., 2026
- Zeng, Liu, Zhang *et al.*, BBGP-sDFO: Batch Bayesian and Gaussian Process enhanced subspace Derivative Free Optimization for high-dimensional analog circuit synthesis, *IEEE TCAD*, 2024
- Chen *et al.*, Enhancing zeroth-order fine-tuning for language models with low-rank structures, arXiv:2410.07698, 2024
- Eldred, Larson, Padidar, Stern, and Wild, Derivative-free optimization of a rapid-cycling synchrotron, *Optim. Eng.*, 2022

Monographs and surveys

- Roberts, Introduction to Model-Based Derivative-Free Optimization, arXiv:2510.04473, 2025
- Ragonneau, Model-Based Derivative-Free Optimization Methods and Software, PhD thesis, The Hong Kong Polytechnic University, Hong Kong, China, 2022
- Larson, Menickelly, and Wild, Derivative-free optimization methods, Acta Numer., 28:287–404, 2019
- Audet and Hare, Derivative-Free and Blackbox Optimization, Springer, 2017
- Custódio, Scheinberg, and Vicente, Methodologies and software for derivative-free optimization, In: Advances and Trends in Optimization with Engineering Applications, pages 495–506, SIAM, 2017
- Rios and Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global Optim., 56:1247–1293, 2013.
- Conn, Scheinberg, and Vicente, Introduction to Derivative-Free Optimization, SIAM, 2009
- Powell, A view of algorithms for optimization without derivatives, Technical Report DAMTP2007/NA03, DAMTP, University of Cambridge, 2007

Words of caution

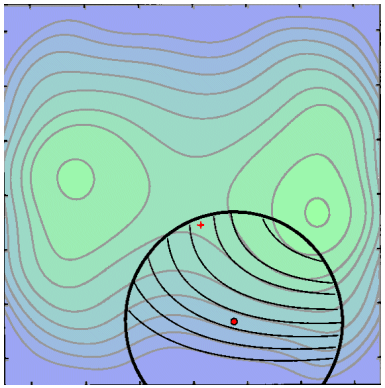
- The reason for not using derivatives is not nonsmoothness but rather the unavailability of the first-order information.
- It is most likely a bad idea to use derivative-free optimization methods if any kind of first-order information is available.
- A real problem is often a gray box instead of a black box. Any known structure should be exploited.

Outline

Powell's trust-region DFO solvers and software

- COBYLA: trust-region method with linear models, nonlinearly constrained problems, code released in 1992, paper published in 1994
- UOBYQA: trust-region method with quadratic models, unconstrained problems, code released in 2000, paper published in 2002
- NEWUOA: trust-region method with quadratic models, unconstrained problems, code released in 2004, paper published in 2006
- BOBYQA: trust-region method with quadratic models, bound-constrained problems, code released and paper written in 2009
- LINCOA: trust-region method with quadratic models, linearly constrained problems, code released in 2013 but no paper written

Trust-region DFO methods based on interpolation models

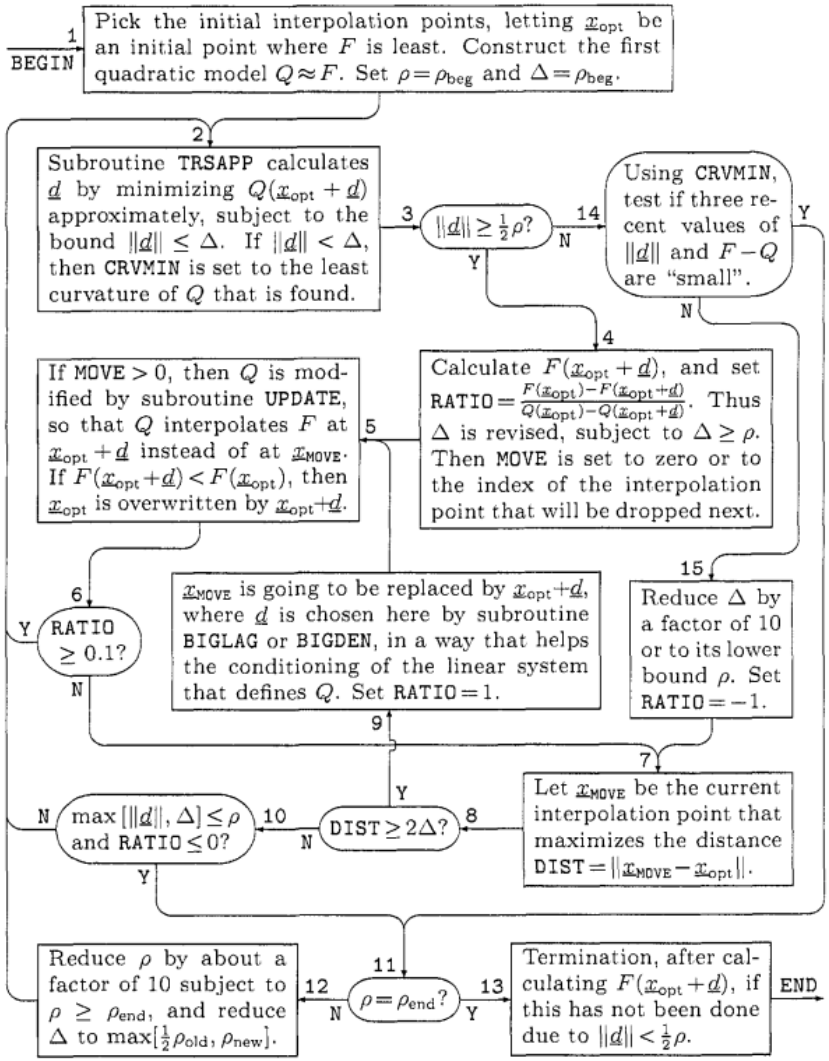


$$x_{k+1} \approx x_k + \underset{\|d\| \leq \Delta_k}{\arg\min} M_k(x_k + d)$$

- M_k is the trust-region model (surrogate)
  - $M_k(x) \approx f(x)$ around x_k
  - M_k interpolates f on a set $\mathcal{X}_k$ consisting of previous iterates
- $\|d\| \leq \Delta_k$ is the trust-region constraint
  - If "things work well", increase Δ_k
  - Otherwise, decrease Δ_k

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

What do these algorithms look like?



NEWUOA

Implementation of these methods is HARD

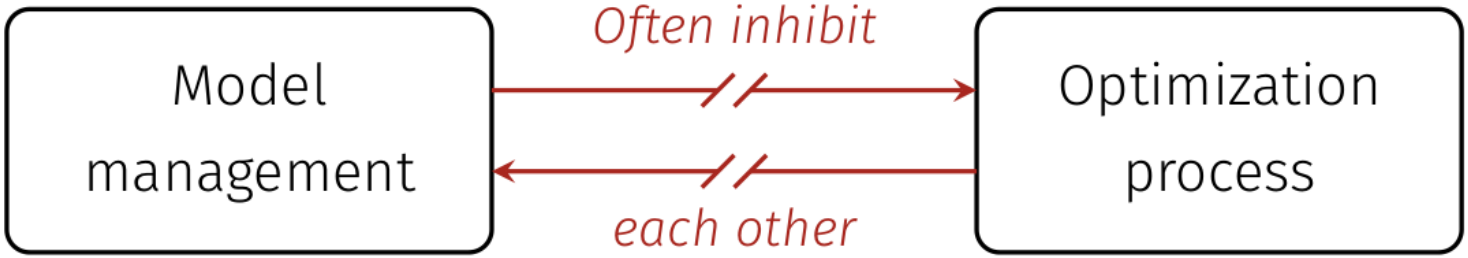
The development of NEWUOA has taken nearly three years. The work was very frustrating ...

— M. J. D. Powell

The NEWUOA software for unconstrained optimization without derivatives, 2006

The intrinsic difficulty

- Gradient-based methods: 1st order information is provided by an oracle independent of the optimization process.
- Model-based DFO methods: 1st order information is extracted from a model constructed from past iterates.



Powell's implementation

- Powell implemented these five methods into publicly available solvers.
- The solvers are widely used in science and engineering (e.g., Airbus).
- They are often used as benchmarks when designing new algorithms.
- However, the implementation was in Fortran 77, with plenty of GOTOs: in total, 7939 lines of code with 249 GOTOs!

A modernized implementation is greatly needed.

Why should I work on a modernized implementation

It would be a relief to me if you would kindly continue to look after my optimisation software (NEWUOA, BOBYQA and LINCOA). Also I would like you to add COBYLA and TOLMIN if you do not have them already.

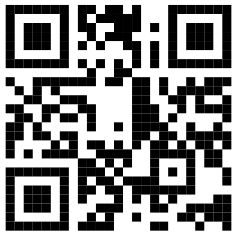
— Powell's email to me, April 6, 2015

A perfect project for an engineer, a student, or AI?

- Where can I find a genius engineer who can learn all the 5 algorithms from scratch and implement them in a reasonable amount of time?
- How can I persuade a student to be fully devoted to a project for 3 years (as I did) without producing a single publication?
- AI is a good tool to translate code from one language to another, but this needs the original code to be structured and modularized, which is exactly PRIMA endeavors to do. In 2020 (when I started PRIMA), AI could do little on a maze of 249 GOTOs.

With a modernized reference implementation, all the above impossibilities **will** become possible. This is exactly why we need PRIMA.

PRIMA



https://www.libprima.net

PRIMA is an acronym for

"Reference Implementation for Powell's Methods
with Modernization and Amelioration",

"P" for Powell.

- Powell's classical implementation: Fortran 77 with GOTOs.
- PRIMA: Fortran 2008, C, C++, Python, Julia, MATLAB ...
- Number of lines: $>$ 100,000.
- The total time I spent on PRIMA:

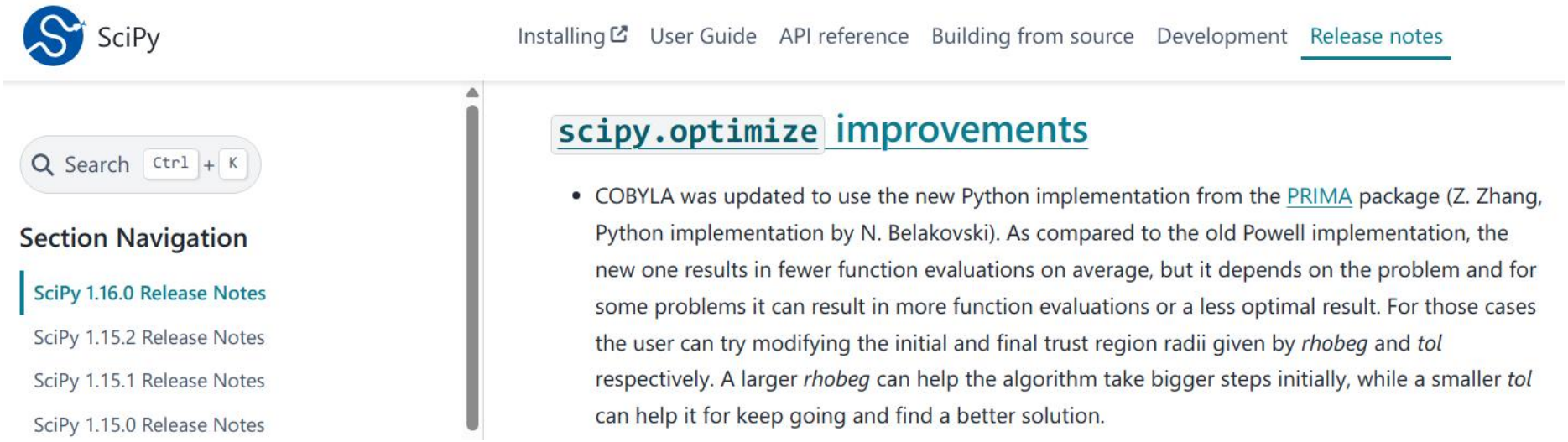
$$\geq 3 \text{ years} \times 300 \text{ days per year} \times 10 \text{ hours per day} = 9,000 \text{ hours}.$$

A "fun" fact ...

- Working on PRIMA, I have spotted more than 40 bugs in reputable Fortran compilers and 4 bugs in MATLAB.
- Each of them represents days of bitter debugging.
- From an unusual angle, they reflect how intensive the coding is.
- The bitterness behind this fun fact is exactly why I work on PRIMA:

  I hope I am the last one in the world to decode a maze of 249 GOTOs in 7939 lines of Fortran 77 code — I did this for 3 years and I do not want anyone else to do it again.

PRIMA integrated into SciPy



SciPy 1.16.0: PRIMA replaces Powell's Fortran 77 version of COBYLA.

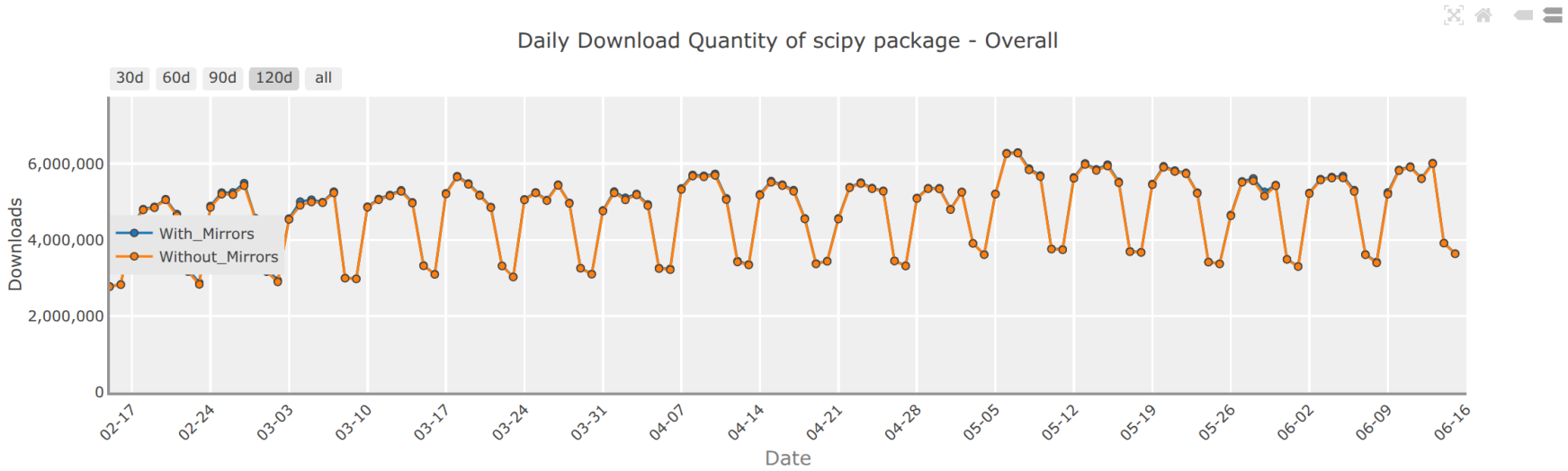
SciPy: Number of downloads

/packages/scipy

scipy

PyPI page
Home page
Author: None
License: Copyright (c) 2001-2002 Enthought, Inc. 2003-2024, SciPy Developers. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that ...
Summary: Fundamental algorithms for scientific computing in Python
Latest version: 1.15.3
Required dependencies: ninja | numpy
Optional dependencies: array-api-strict | asv | cython | cython-lint | doit | gmpy2 | hypothesis | intersphinx_registry | jupyterlite-pyodide-kernel | jupyterlite-sphinx | jupytext | matplotlib | meson | mpmath | mypy | myst-nb | numpydoc | pooch | pycodestyle | pydata-sphinx-theme | pydevtool | pytest | pytest-cov | pytest-timeout | pytest-xdist | rich-click | ruff | scikit-umfpack | sphinx | sphinx-copybutton | sphinx-design | threadpoolctl | types-psutil | typing_extensions

Downloads last day: 3,636,063
Downloads last week: 36,074,511
Downloads last month: 145,692,466



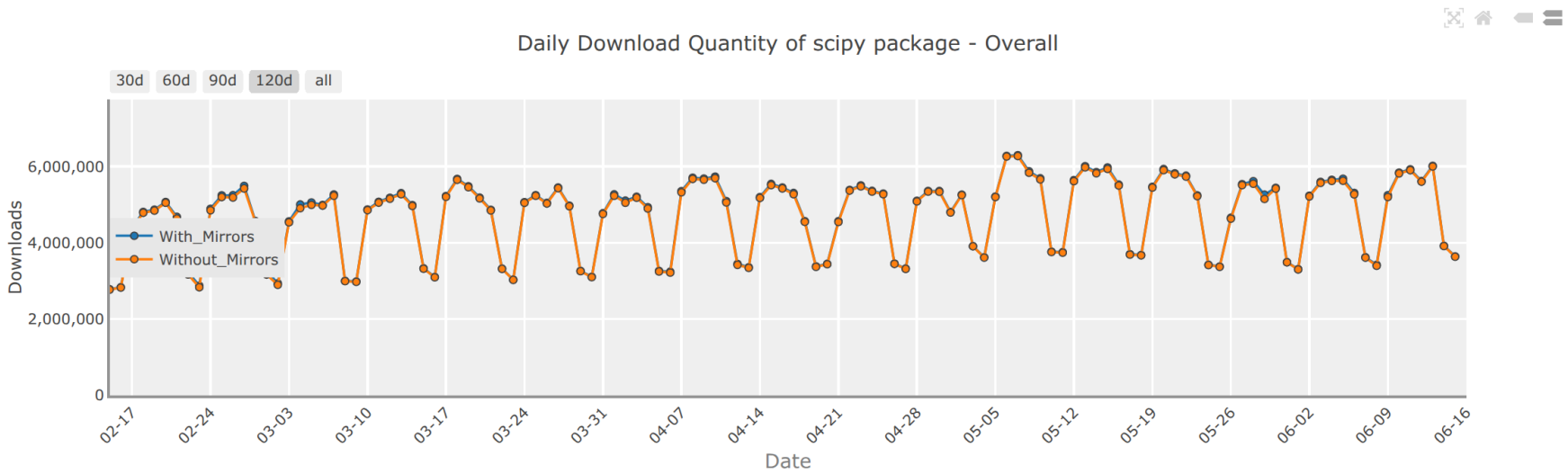
SciPy: Number of downloads

/packages/scipy

scipy

PyPI page
Home page
Author: None
License: Copyright (c) 2001-2002 Enthought, Inc. 2003-2024, SciPy Developers. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that ...
Summary: Fundamental algorithms for scientific computing in Python
Latest version: 1.15.3
Required dependencies: ninja | numpy
Optional dependencies: array-api-strict | asv | cython | cython-lint | doit | gmpy2 | hypothesis | intersphinx_registry | jupyterlite-pyodide-kernel | jupyterlite-sphinx | jupytext | matplotlib | meson | mpmath | mypy | myst-nb | numpydoc | pooch | pycodestyle | pydata-sphinx-theme | pydevtool | pytest | pytest-cov | pytest-timeout | pytest-xdist | rich-click | ruff | scikit-umfpack | sphinx | sphinx-copybutton | sphinx-design | threadpoolctl | types-psutil | typing_extensions

Downloads last day: 3,636,063
Downloads last week: 36,074,511
Downloads last month: 145,692,466



Feedback from industry: IRT Saint Exupéry (Airbus)

IRT Saint Exupéry
Toulouse, France
21th September 2023

Dr. Zaikun Zhang
Department of Applied Mathematics
The Hong Kong Polytechnic University
Hong Kong, China

Dear Dr. Zhang,

This letter is to provide some feedback on the optimisation software packages your team have developed, in particular PRIMA, PDFO, and COBYQA. Your packages are of major importance for industry that involves optimal design problems based on numerical simulations, which is particularly the case for the MDO (Multidisciplinary Design Optimisation) projects at IRT Saint Exupéry, France. IRT is an industrial research institute focused on advanced manufacturing technologies, greener technologies, smart technologies and methods & tools for the development of complex systems. We are co-founded by major industry players including Airbus, Safran, Thales, CNES (French national space agency), and the French state (see https://www.irt-saintexupery.com/our-members/ for a full list).

Optimisation Based on Simulation (also known as Black-box Optimisation) is a key technology in every field of science and engineering and a key element of the ongoing digital transformation of industry. However, most open source packages are based on unmaintained Fortran code from the 1980s. For example, the NLopt and SciPy packages (more than 1 million downloads per day) are largely based on obsolete implementations of L-BFGS-B, COBYLA and SLSQP, which haven't been significantly improved since their creation. These codes have known bugs and limitations that have very practical implications in terms of the solutions obtained and their robustness, and thus their potential for use by industry. This has also been the case for the widely used derivative-free optimisation (DFO) solvers in industry (COBYLA, UOBYQA, NEWUOA, BOBYQA, and LINCOA), but your PRIMA package changes the game. Thanks to your very large effort in modernizing the code, the PRIMA package now provides most reliable implementations of these solvers. Moreover, the PRIMA solvers work evidently more efficient than the classical implementation by the late Professor M.J.D. Powell, which is quite remarkable.

The CUTEst benchmark suite, which is the reference in the field, have shown that PRIMA achieves significant improvements in performance and reliability. The benchmark we use internally to compare these packages shows a similar trend for the COBYLA algorithm, which is useful to us. The modernized code is easy to modify and will allow further research into the optimisation algorithms, whereas the old implementation prevented this due to its architectural limitations: mostly a single file with many GOTO statements, which prevented important modifications.

The reliability of the PRIMA solvers is of particular importance for us. We have been using COBYLA in critical applications including optimisation of aircraft components and architecture, and design electrical systems such as car power electronics for car engines. Obviously, such applications require a high level of reliability, as the optimisation results are used to make important decisions. It is also noted that the PRIMA solvers are quite robust in the presence of noise in function evaluations, which is always the case in the aforementioned applications. PRIMA's capability of doing so is great news for the industry.

Your work on PDFO is also very important for us. The PDFO package provides a unified and user-friendly interface for the above-mentioned DFO solvers, which is very convenient for us. We note

IRT Saint Exupéry
Fondation de Coopération Scientifique

Bâtiment B612
3 rue Tarfaya • CS 34436
31405 Toulouse Cedex 4

www.irt-saintexupery.com
+ 33 (0)5 61 00 67 50
contact@irt-saintexupery.com

SIREN 793 007 048
TVA Intracom : FR29 793 007 048

IRT Saint Exupéry (Airbus)

- "Optimisation Based on Simulation is a key technology in every field of science and engineering."
- "Most open source packages are based on unmaintained Fortran code from the 1980s ... but your PRIMA package changes the game. ... the PRIMA package now provides most reliable implementations of these solvers."
- "PRIMA solvers work evidently more efficient than the classical implementation by the late Professor M.J.D. Powell, which is quite remarkable."
- "PRIMA solvers are quite robust in the presence of noise."

COBYQA: A derivative-free trust-region SQP algorithm

We design a method named **COBYQA** for solving

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0, \cancel{h(x) = 0}, \\ & l \leq x \leq u, \end{aligned}$$

when derivatives of f, g, and h are **unavailable**.

- We omit the equality constraints for simplicity.
- COBYQA aims at being a **successor** to COBYLA (Powell 1994).
- We **implement** COBYQA into a Python solver.
- The bound constraints are **unrelaxable**:
  - they often represent **inalienable** restrictions; and
  - f, g, or h may not be well-defined outside the bounds.

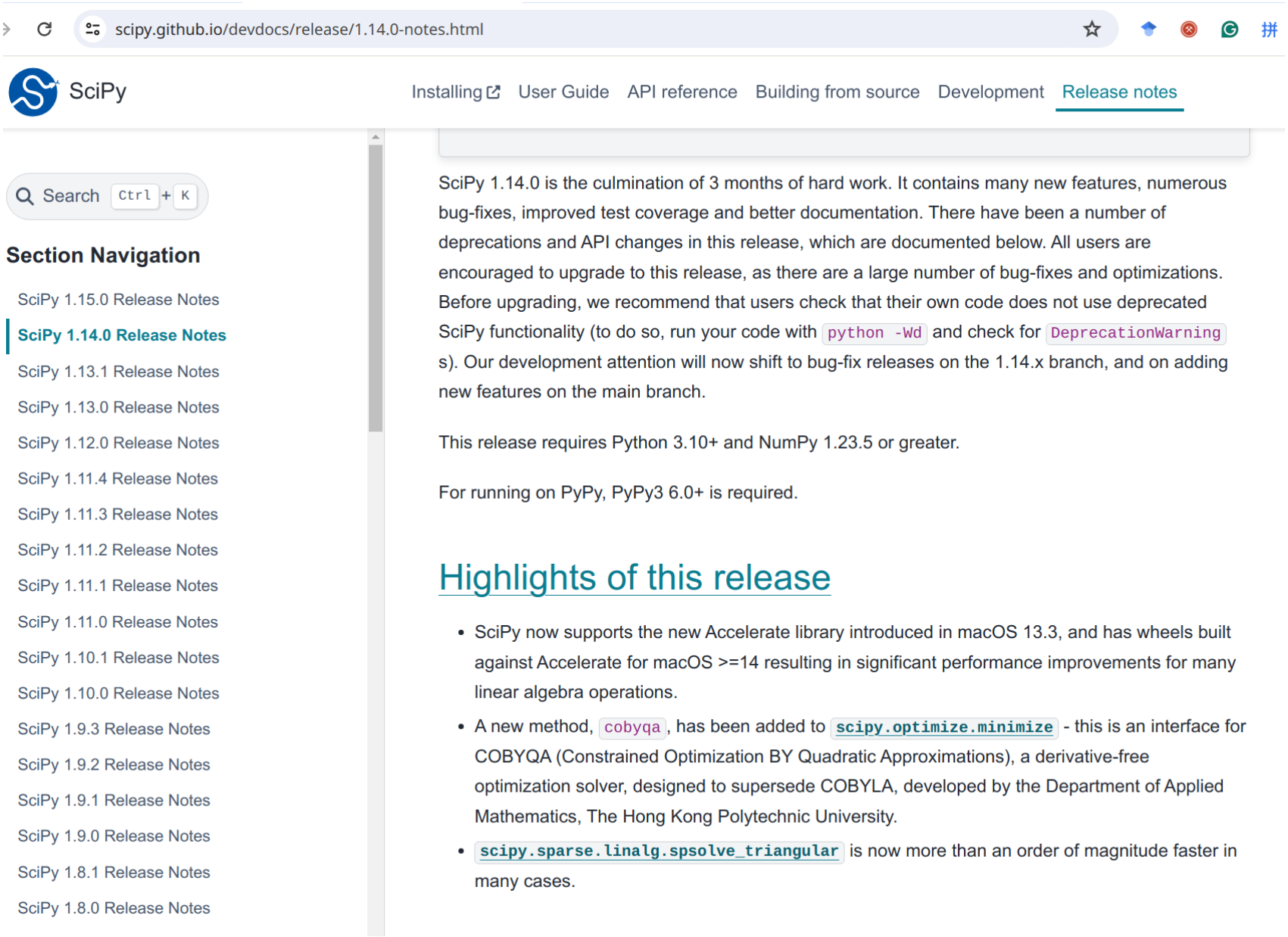
COBYQA: A derivative-free trust-region SQP algorithm

COBYQA iteratively solves the trust-region SQP subproblem

$$\begin{aligned} \min_{s\in\mathbb{R}^n} \quad & \nabla \hat{f}_k(x_k)^\mathsf{T} s + \frac{1}{2} s^\mathsf{T} \nabla^2_{x,x} \widehat{\mathcal{L}}_k(x_k, \lambda_k) s \\ \text{s.t.} \quad & \hat{g}_k(x_k) + \nabla \hat{g}_k(x_k) s \leq 0, \\ & l \leq x_k + s \leq u, \\ & \|s\| \leq \Delta_k, \end{aligned}$$

with $\widehat{\mathcal{L}}_k(x, \lambda) = \hat{f}_k(x) + \lambda^\mathsf{T} \hat{g}_k(x)$, given some models $\hat{f}_k$ and $\hat{g}_k$.

COBYQA: A highlight of SciPy 1.14.0



scipy.github.io/devdocs/release/1.14.0-notes.html

SciPy | Installing | User Guide | API reference | Building from source | Development | Release notes

Search Ctrl + K

Section Navigation

- SciPy 1.15.0 Release Notes
- SciPy 1.14.0 Release Notes
- SciPy 1.13.1 Release Notes
- SciPy 1.13.0 Release Notes
- SciPy 1.12.0 Release Notes
- SciPy 1.11.4 Release Notes
- SciPy 1.11.3 Release Notes
- SciPy 1.11.2 Release Notes
- SciPy 1.11.1 Release Notes
- SciPy 1.11.0 Release Notes
- SciPy 1.10.1 Release Notes
- SciPy 1.10.0 Release Notes
- SciPy 1.9.3 Release Notes
- SciPy 1.9.2 Release Notes
- SciPy 1.9.1 Release Notes
- SciPy 1.9.0 Release Notes
- SciPy 1.8.1 Release Notes
- SciPy 1.8.0 Release Notes

SciPy 1.14.0 is the culmination of 3 months of hard work. It contains many new features, numerous bug-fixes, improved test coverage and better documentation. There have been a number of deprecations and API changes in this release, which are documented below. All users are encouraged to upgrade to this release, as there are a large number of bug-fixes and optimizations. Before upgrading, we recommend that users check that their own code does not use deprecated SciPy functionality (to do so, run your code with `python -Wd` and check for `DeprecationWarning` s). Our development attention will now shift to bug-fix releases on the 1.14.x branch, and on adding new features on the main branch.

This release requires Python 3.10+ and NumPy 1.23.5 or greater.

For running on PyPy, PyPy3 6.0+ is required.

Highlights of this release

- SciPy now supports the new Accelerate library introduced in macOS 13.3, and has wheels built against Accelerate for macOS >=14 resulting in significant performance improvements for many linear algebra operations.
- A new method, `cobyqa`, has been added to `scipy.optimize.minimize` - this is an interface for COBYQA (Constrained Optimization BY Quadratic Approximations), a derivative-free optimization solver, designed to supersede COBYLA, developed by the Department of Applied Mathematics, The Hong Kong Polytechnic University.
- `scipy.sparse.linalg.spsolve_triangular` is now more than an order of magnitude faster in many cases.

Thanks

1. Thank the late Professor M. J. D. Powell FRS for the inspiration and for the wealth he left to us.
2. Thank Professor Ya-xiang Yuan for his everlasting support.
3. Thank my students Dr. Tom M. Ragonneau, Haitian Li, and Cunxin Huang for working with me on software development, which is not an easy choice for students (and professors).

During the years working on software development (especially PRIMA), due to the gap in my publication record, I needed lots of support from the research community. Thank you for your help, known or unknown to me, explicit or implicit, without which I would not have survived.

Thank you very much!

Outline

PRIMA is not the end, but the beginning

- Powell's algorithms are masterpieces, but they are old.
- PRIMA takes me a long time, but it does not represent my research.
- PRIMA paves the way for developing new and better DFO methods based on the treasure left by Powell.
- The following is such an example.

"DFO algorithms cannot solve big problems"

"[...] the scale of the problems that can currently be efficiently solved by derivative-free methods is still relatively small and does not exceed a few hundred variables even in easy cases."

— Conn, Scheinberg, and Vicente, 2009
Introduction to Derivative-free Optimization (2015 SIAM-MOS Lagrange Prize)

Since DFO algorithms cannot solve big problems, engineers tend to limit the number of variables when designing models if they are to use DFO.

— Pauwels (Engineer and Researcher, IRT Saint Exupéry), 2024

"DFO algorithms cannot solve big problems"

"[...] the scale of the problems that can currently be efficiently solved by derivative-free methods is still relatively small and does not exceed a few hundred variables even in easy cases."

— Conn, Scheinberg, and Vicente, 2009
Introduction to Derivative-free Optimization (2015 SIAM-MOS Lagrange Prize)

Since DFO algorithms cannot solve big problems, engineers tend to limit the number of variables when designing models if they are to use DFO.

— Pauwels (Engineer and Researcher, IRT Saint Exupéry), 2024

"DFO algorithms cannot solve big problems": myth or truth?

Solving 10,000-dimensional problems using function values

		NEWUOAs		fminunc	
Problem	$f(x_0)$	$f(x_{\text{fin}})$	NF	$f(x_{\text{fin}})$	NF
ARWHEAD	2.99E+04	0.00E+00	90331	2.99E+04	170017
BRYBND	3.60E+05	4.50E−15	370895	1.61E+04	270027
CHROSEN	1.99E+05	8.80E−14	851736	1.99E+05	140014
CRAGGLVY	5.49E+06	3.40E+03	110483	5.49E+06	140014
DIXMAANE	7.36E+04	1.02E+00	170658	1.23E+04	90018
ENGVAL1	5.89E+05	1.10E+04	230880	5.89E+05	140014
LIARWHD	5.85E+06	7.89E−14	130464	3.71E+05	240024
NONDIA	1.01E+08	1.97E+00	90242	1.01E+08	90009
POWER	3.33E+11	1.64E+06	270951	3.33E+11	20002
SPARSQUR	1.40E+07	1.12E−18	410989	3.98E+06	90009
WOODS	4.79E+07	1.97E+04	90339	4.79E+07	120012

x_0: starting point; x_{fin}: final iterate; NF: number of function evaluations

N.B.: The function evaluations have only 3 correct digits in the experiment.

NEWUOAs: an almost forgotten algorithm from 2012

算法 5.18. (NEWUOAs)

步 *1.* 取正数序列 $\{h_k\}$、$\{p_k\}$ 和常数 $\varepsilon \geq 0$. 确定初始点 x_1; $s_0 := 0$; $k := 1$.

步 *2.* 选取整数 $m_k \in [n+1, (n+1)(n+2)/2]$, 调用 MODEL$(x_k, h_k, m_k)$, 获得 x_k 处的近似梯度 $\tilde{g}_k$. 若 $h_k < \varepsilon$ 且 $\|\tilde{g}_k\| < \varepsilon$, 终止. 令

$$\mathcal{S}_k = \mathrm{span}\{\tilde{g}_k, s_{k-1}\}. \tag{5.59}$$

步 *3.* 设置 RHOEND $= p_k$, 调用 NEWUOA 求解子问题

$$\min_{d \in \mathcal{S}_k} f(x_k + d) \tag{5.60}$$

得到 d_k.

步 *4.* 若 $f(x_k + d_k) < f(x_k)$, 则 $x_{k+1} := x_k + d_k$, $s_k := d_k$; 否则 $x_{k+1} := x_k$, $s_k := s_{k-1}$. $k := k+1$. 转步 *2.*

NEWUOAs (screenshot taken from Sec. 5.3.3 of my PhD thesis, 2012)

A prototypical framework of subspace algorithms

Algorithm (Conn, Toint, Sartenaer, and Gould, 1996)

Step 1. Pick the starting point x_0. $k := 0$.

Step 2. Pick a subspace $\mathcal{S}_k$ of $\mathbb{R}^n$.

Step 3. Solve the subspace subproblem

$$\min_{d \in \mathcal{S}_k} \ f(x_k + d)$$

exactly or approximately, obtaining d_k.

Step 4. $x_{k+1} := x_k + d_k$ and $k := k + 1$. Go to Step 2.

A simple convergence theorem

Theorem

Suppose that

- $\mathrm{dist}(\nabla f(x_k), \mathcal{S}_k)$ is sufficiently small, and
- d_k is sufficiently exact,

then the subspace algorithm will converge (sufficiently fast).

How to satisfy the above-mentioned requirements on $\mathcal{S}_k$ and d_k?

- Find a good model $\hat{f}_k$ of f around x_k, and define

$$\mathcal{S}_k \ni \nabla \hat{f}_k(x_k).$$

- Call a good solver to solve (up to a sufficient extent)

$$\min_{d \in \mathcal{S}_k} f(x_k + d).$$

We do not necessarily need derivatives.

The subspace: The Yuan–Stoer method (1992)

Josef Stoer

Ya-xiang Yuan

A Subspace Study on Conjugate Gradient Algorithms*

J. Stoer
Institut für Angewandte Mathematik, Universität Würzburg
D-97074 Würzburg, Germany

Ya-xiang Yuan†
Computing Center, Academia Sinica
P.O. Box 2719, Beijing 10080, China

Abstract

In this paper, we analyse techniques of computing a search direction by minimizing the approximate quadratic model in the 2 dimensional subspace spanned by the current gradient and the last search direction. The classical conjugate gradient methods are only the special cases where the objective function is quadratic and line searches are exact. Based on our analyses on the case where line searches are not exact, we construct new conjugate direction type algorithms.

Stoer and Yuan, 1992

- Conjugate gradient method:

$$x_{k+1} = x_k + \alpha_k d_k, \quad d_k = d_{k-1} - \beta_k \nabla f(x_k).$$

- Stoer–Yuan, 1992: From conjugate gradient to a subspace algorithm:

$$\mathcal{S}_k = \operatorname{span}\{\nabla f(x_k), d_{k-1}\}.$$

- Yuan, Y.-x. and Stoer, J., 1995, A Subspace Study on Conjugate Gradient Algorithms. Z. angew. Math. Mech., 75: 69–77.
- See also Conn, Toint, Sartenaer, and Gould, 1996.
- Recently studied in DRSOM (Zhang, Ge, Jiang, and Ye, 2022).

Some details on the implementation of NEWUOAs

1. At iteration k, obtain a 'sufficiently good model' $\hat{f}_k$ around x_k.
2. Define the subspace

$$\mathcal{S}_k = \text{span}\{\nabla \hat{f}_k(x_k),\ d_{k-1},\ \text{other directions}\}.$$

3. Call a solver (e.g., NEWUOA) to minimize f on $x_k + \mathcal{S}_k$ 'up to an extent'.

'Other directions':

- $\nabla \hat{f}_\ell(x_\ell)$ and/or $d_{\ell-1}$ with $\ell < k$ (Yuan, 2007, 2009, 2014). (cautious!)
- Directions reflecting second-order information (recall CG).

Z. Zhang, Scalable derivative-free optimization algorithms with low-dimensional subspace techniques, arXiv:2501.04536, 2025

An implementable framework of subspace algorithms

Algorithm 3.1 Derivative-free OptimIST

Input: $f : \mathbb{R}^n \to \mathbb{R}$, $x_0 \in \mathbb{R}^n$, $\delta_0 \in (0, \infty)$, $\eta \in (0, \infty)$.

For $k = 0, 1, 2, \ldots,$ iterate the following.

1. Generate an approximate gradient $\hat{g}_k$ for f at x_k.
2. Choose a subspace $\mathcal{S}_k \subset \mathbb{R}^n$ with $\hat{g}_k \in \mathcal{S}_k$.
3. Calculate $x_{k+1} \approx \operatorname{argmin}\{f(x) : x \in x_k + \mathcal{S}_k\}$ so that $f_{k+1} \le f_k$ and

$$f_{k+1} \;\le\; \mathbf{max}\left\{f_k - \eta\delta_k^2,\;\; f(x_k - \delta_k \hat{g}_k/\|\hat{g}_k\|)\right\}. \tag{3.1}$$

4. If $\|\hat{g}_k\| \ge \eta\delta_k$ and $f_{k+1} \le f_k - \eta\delta_k^2$, then $\delta_{k+1} = 2\delta_k$; otherwise, $\delta_{k+1} = \delta_k/2$.

The calculation of x_{k+1}:

- Let $x_k^{\mathsf{s}} \approx \arg\min\{f(x) : x \in x_k + \mathcal{S}_k\}$ and $x_k^{\mathsf{g}} = x_k - \delta_k \hat{g}_k/\|\hat{g}_k\|$. Then

$$x_{k+1} = \begin{cases} x_k^{\mathsf{s}} & \text{if } f(x_k^{\mathsf{s}}) \le f_k - \eta\delta_k^2, \\ \arg\min\{f(x) : x = x_k, x_k^{\mathsf{s}}, \text{ or } x_k^{\mathsf{g}}\} & \text{otherwise.} \end{cases}$$

- x_k^{s} is obtained by a DFO solver on $\mathcal{S}_k$; no requirement on its quality.
- x_k^{g} provides a safeguard when x_k^{s} fails to achieve a sufficient decrease.

Iteration complexity

Theorem

Suppose that f is L-smooth and there exists a constant $\zeta \geq 0$ such that

$$\|\hat{g}_k - \nabla f(x_k)\| \leq \zeta \delta_k \qquad \text{for each} \qquad k \geq 0.$$

Then the number of iterations needed to find an x_ϵ with $\|\nabla f(x_\epsilon)\| \leq \epsilon$ is

$$K_\epsilon = \mathcal{O}((L+\zeta)^2 \epsilon^{-2}).$$

Function evaluation complexity

Theorem

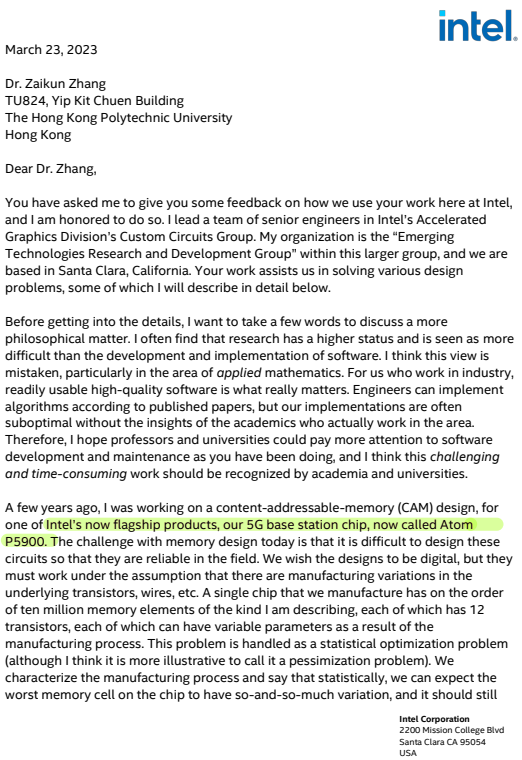
If $\hat{g}_k$ is obtained by forward finite differences with step size $h_k = \mathcal{O}(\delta_k/\sqrt{n})$, then the number of function evaluations to find an x_ϵ as above is

$$K_\epsilon^f \;=\; \mathcal{O}(n\epsilon^{-2}).$$

Comparison with other methods:

- Standard full-space trust-region DFO methods: $\mathcal{O}(n^2\epsilon^{-2})$.
- The subspace method RSDFO (Cartis and Roberts, 2023): $\mathcal{O}(n^2\epsilon^{-2})$.
- Finite-difference gradient methods: $\mathcal{O}(n\epsilon^{-2})$; but our method can be much more efficient in practice due to the subspace exploration.

Applications at Intel: Atom P5900



intel

March 23, 2023

Dr. Zaikun Zhang
TU824, Yip Kit Chuen Building
The Hong Kong Polytechnic University
Hong Kong

Dear Dr. Zhang,

You have asked me to give you some feedback on how we use your work here at Intel, and I am honored to do so. I lead a team of senior engineers in Intel's Accelerated Graphics Division's Custom Circuits Group. My organization is the "Emerging Technologies Research and Development Group" within this larger group, and we are based in Santa Clara, California. Your work assists us in solving various design problems, some of which I will describe in detail below.

Before getting into the details, I want to take a few words to discuss a more philosophical matter. I often find that research has a higher status and is seen as more difficult than the development and implementation of software. I think this view is mistaken, particularly in the area of *applied* mathematics. For us who work in industry, readily usable high-quality software is what really matters. Engineers can implement algorithms according to published papers, but our implementations are often suboptimal without the insights of the academics who actually work in the area. Therefore, I hope professors and universities could pay more attention to software development and maintenance as you have been doing, and I think this *challenging and time-consuming* work should be recognized by academia and universities.

A few years ago, I was working on a content-addressable-memory (CAM) design, for one of Intel's now flagship products, our 5G base station chip, now called Atom P5900. The challenge with memory design today is that it is difficult to design these circuits so that they are reliable in the field. We wish the designs to be digital, but they must work under the assumption that there are manufacturing variations in the underlying transistors, wires, etc. A single chip that we manufacture has on the order of ten million memory elements of the kind I am describing, each of which has 12 transistors, each of which can have variable parameters as a result of the manufacturing process. This problem is handled as a statistical optimization problem (although I think it is more illustrative to call it a pessimization problem). We characterize the manufacturing process and say that statistically, we can expect the worst memory cell on the chip to have so-and-so-much variation, and it should still

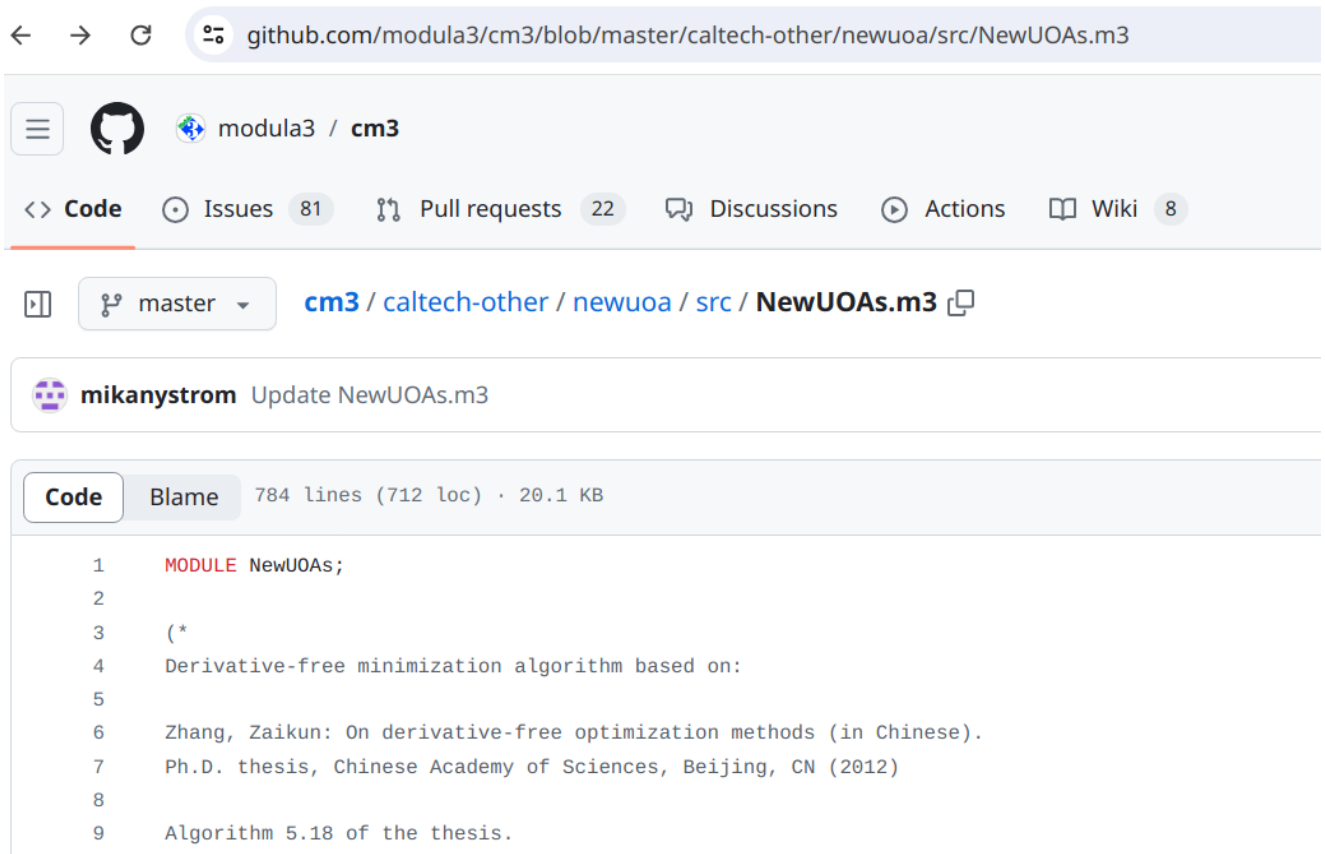
Intel Corporation
2200 Mission College Blvd
Santa Clara CA 95054
USA

- "I was working on a content-addressable-memory design, for one of Intel's now flagship products, our 5G base station chip, now called Atom P5900 ."
- "The impact of the performance of `NEWUOAs` as compared to prior methods was to massively simplify our verification software flow.
- "The optimization that takes a couple of hours with `NEWUOAs` would take weeks with `NEWUOA`."
- "I find myself reaching for `NEWUOAs` more often than any other numerical algorithm, just to access its parallel speedup.

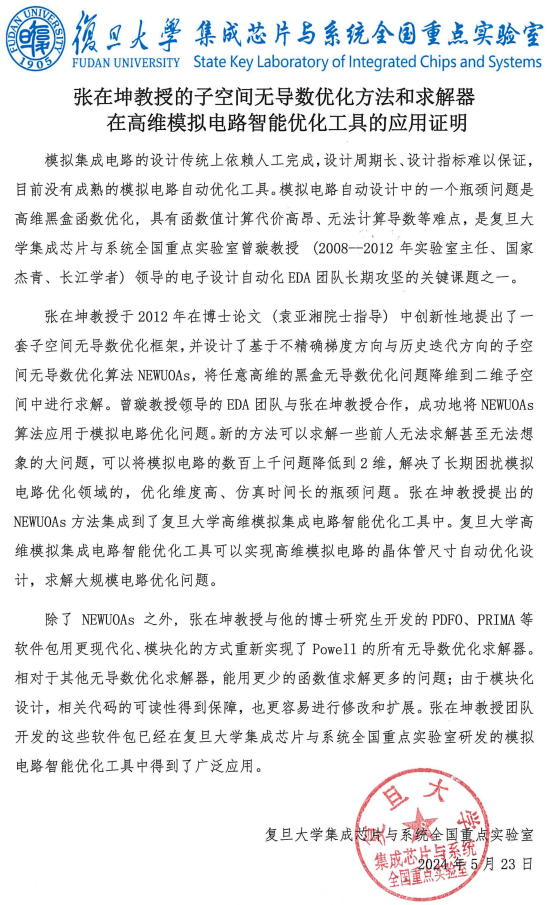
NEWUOAs included in CM3 by Dr. M. Nyström

NEWUOAs is included as a DFO solver in the open-source package cm3 at

https://github.com/modula3/cm3



Applications at State Key Laboratory of ASIC & System



复旦大學 集成芯片与系统全国重点实验室
FUDAN UNIVERSITY State Key Laboratory of Integrated Chips and Systems

张在坤教授的子空间无导数优化方法和求解器
在高维模拟电路智能优化工具的应用证明

模拟集成电路的设计传统上依赖人工完成，设计周期长、设计指标难以保证，目前没有成熟的模拟电路自动优化工具。模拟电路自动设计中的一个瓶颈问题是高维黑盒函数优化，具有函数值计算代价高昂、无法计算导数等难点，是复旦大学集成芯片与系统全国重点实验室曾璇教授（2008—2012 年实验室主任、国家杰青、长江学者）领导的电子设计自动化 EDA 团队长期攻坚的关键课题之一。

张在坤教授于 2012 年在博士论文（袁亚湘院士指导）中创新性地提出了一套子空间无导数优化框架，并设计了基于不精确梯度方向与历史迭代方向的子空间无导数优化算法 NEWUOAs，将任意高维的黑盒无导数优化问题降维到二维子空间中进行求解。曾璇教授领导的 EDA 团队与张在坤教授合作，成功地将 NEWUOAs 算法应用于模拟电路优化问题。新的方法可以求解一些前人无法求解甚至无法想象的大问题，可以将模拟电路的数百上千问题降低到 2 维，解决了长期困扰模拟电路优化领域的，优化维度高、仿真时间长的瓶颈问题。张在坤教授提出的 NEWUOAs 方法集成到了复旦大学高维模拟集成电路智能优化工具中。复旦大学高维模拟集成电路智能优化工具可以实现高维模拟电路的晶体管尺寸自动优化设计，求解大规模电路优化问题。

除了 NEWUOAs 之外，张在坤教授与他的博士研究生开发的 PDFO、PRIMA 等软件包用更现代化、模块化的方式重新实现了 Powell 的所有无导数优化求解器。相对于其他无导数优化求解器，能用更少的函数值求解更多的问题；由于模块化设计，相关代码的可读性得到保障，也更容易进行修改和扩展。张在坤教授团队开发的这些软件包已经在复旦大学集成芯片与系统全国重点实验室研发的模拟电路智能优化工具中得到了广泛应用。

复旦大学集成芯片与系统全国重点实验室
2024 年 5 月 23 日

- "The team of Professor Xuan Zeng (曾璇) ... has successfully applied NEWUOAs to Analog Circuit Optimization ..."
- "The new method is capable of solving some large problems that were unsolvable or even unimaginable"
- " ... solved bottleneck problems with high dimensions and expensive simulations, which had been long-standing obstacles in Analog Circuit Optimization ..."

Outline

From unconstrained to constrained optimization

- Penalty methods
- Augmented Lagrangian
  - Powell, A method for nonlinear constraints in minimization problems, *Optimization* (edited by Fletcher), 1969
  - Hestenes, Multiplier and gradient methods, *J. Optim. Theory Appl.*, 1969
- Constraint dissolving
  - Xiao, Liu, and Toh, Dissolving constraints for Riemannian optimization, *Math. Oper. Res.*, 2023
  - Hu, Xiao, Liu, and Toh, A constraint dissolving approach for nonsmooth optimization over the Stiefel manifold, *IMA J. Numer. Anal.*, 2024
  - ...

OptimIST: Optimization by Iterated Subspace Techniques

$$\min_{x \in \mathcal{X}} \; f(x)$$

Algorithm (OptimIST $\sim$ Conn, Toint, Sartenaer, and Gould, 1996)

Step 1. Pick the starting point x_0. $k := 0$.

Step 2. Pick a subspace $\mathcal{S}_k$ of $\mathbb{R}^n$.

Step 3. Solve the subspace subproblem

$$\min_{d \in \mathcal{S}_k \cap (\mathcal{X} - x_k)} \; f(x_k + d)$$

exactly or approximately, obtaining d_k.

Step 4. $x_{k+1} := x_k + d_k$ and $k := k + 1$. Go to Step 2.

N.B.: If $\mathcal{X}$ is "hard", then we may replace it with $\mathcal{X}_k \approx \mathcal{X}$ in Step 3.

Beyond NEWUOAs: SPRIMA

- SPRIMA aims at $1,000 \sim 10,000$-dimensional (constrained) problems:

$$\text{SPRIMA} \;=\; \text{Subspace} + \text{PRIMA}.$$

- At each iteration,
  1. SPRIMA defines a low-dimensional subspace;
  2. SPRIMA solves a low-dimensional DFO subproblem by PRIMA.

Outline

Concluding remarks

- DFO is an exciting research area with wide applications in industry, engineering, and recently in machine learning & AI.
- SPRIMA $=$ Subspace $+$ PRIMA can solve $10,000$-dimensional problems using inaccurate function values.
- To have real impact, DFO methods must be implemented into reliable, efficient, and user-friendly software packages.

Thank you very much!

Open-source software packages by my team

SPRIMA
(dates back to 2012)

COBYQA
(highlight of SciPy 1.14.0)

PRIMA
(highlight of SciPy 1.16.0)

BDS

OptiProfiler
(testing platform)

Users: State Key Lab of ASIC and System (Fudan), Airbus, Intel, NVIDIA