A Subspace Decomposition Framework for Nonlinear Optimization: Global Convergence and Global Rate

Zaikun Zhang
University of Coimbra
(Joint work with S. Gratton and L. N. Vicente)

July 25, Toulouse

http//www.mat.uc.pt/~zhang

Opening

Axiom

师徒如父子；一日为师，终身为父。

(Teacher and student are like father and son; if he/she is your teacher for one day, then he/she is your father for one life.)

Axiom

师徒如父子；一日为师，终身为父。

(Teacher and student are like father and son; if he/she is your teacher for one day, then he/she is your father for one life.)

Question

Provided that you call your professor 师父(Academic Father), how should you call a person whose professor is also the professor of your professor?

Axiom

师徒如父子；一日为师，终身为父。

(Teacher and student are like father and son; if he/she is your teacher for one day, then he/she is your father for one life.)

Question

Provided that you call your professor 师父(Academic Father), how should you call a person whose professor is also the professor of your professor?

Answer

师伯 师叔

Opening

Axiom

师徒如父子；一日为师，终身为父。

(Teacher and student are like father and son; if he/she is your teacher for one day, then he/she is your father for one life.)

Question

Provided that you call your professor 师父(Academic Father), how should you call a person whose professor is also the professor of your professor?

Answer

师伯 师叔 (Academic Uncle, “older” or “younger”).

Axiom

师徒如父子；一日为师，终身为父。

(Teacher and student are like father and son; if he/she is your teacher for one day, then he/she is your father for one life.)

Question

Provided that you call your professor 师父(Academic Father), how should you call a person whose professor is also the professor of your professor?

Answer

师伯 师叔 (Academic Uncle, "older" or "younger").

生日快乐，Toint 师伯！

Happy birthday, Academic (Older) Uncle Toint!

Joyeux anniversaire, Oncle Toint !

Outline

In this talk, to make things simple:

- we consider unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x);$$

In this talk, to make things simple:

- we consider unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x);$$

- we suppose that

In this talk, to make things simple:

- we consider unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x);$$

- we suppose that
  - f is smooth, but the derivatives are unavailable.

Derivative-free optimization

- Important & difficult

Derivative-free optimization

- Important & difficult

> *We consider optimization without derivatives one of the most important, open, and challenging areas in computational science and engineering, and one with enormous practical potential.*

— A. R. Conn, K. Scheinberg, L. N. Vicente
Introduction to Derivative-Free Optimization

Derivative-free optimization

- Important & difficult

> *We consider optimization without derivatives one of the most important, open, and challenging areas in computational science and engineering, and one with enormous practical potential.*

— A. R. Conn, K. Scheinberg, L. N. Vicente
Introduction to Derivative-Free Optimization

> *Why work on derivative-free optimization? Because the problems are important and cool.*

— J. Dennis
July 24th, 2013, Toulouse

Existing methods

- Two main classes of rigorous methods in DFO

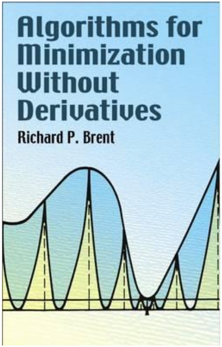
Existing methods

- Two main classes of rigorous methods in DFO
  - Directional methods, like direct search

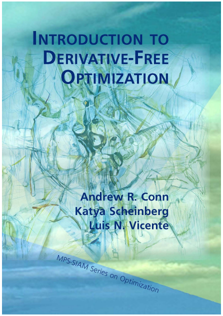
Existing methods

- Two main classes of rigorous methods in DFO
  - Directional methods, like direct search
  - Model-based methods, like trust-region methods

Books



R. P. Brent, Algorithms for Minimization Without Derivatives, Prentice-Hall, Englewood Cliffs, NJ, 1973



A. R. Conn, K. Scheinberg, and L. N. Vicente, Introduction to Derivative-Free Optimization, MOS-SIAM Series on Optimization, SIAM, Philadelphia, 2009

Difficulty of large-scale problems

- Large-scale problems?

Difficulty of large-scale problems

- Large-scale problems?
    - Traditional NLP: $10,000$? $100,000$? $1,000,000$?

Difficulty of large-scale problems

- Large-scale problems?
  - Traditional NLP: $10,000$? $100,000$? $1,000,000$?
  - Derivative-free: 100? 1000?

Difficulty of large-scale problems

- Large-scale problems?
  - Traditional NLP: $10,000$? $100,000$? $1,000,000$?
  - Derivative-free: 100? 1000?
- Large-scale derivative-free problems are very difficult:

Difficulty of large-scale problems

- Large-scale problems?
  - Traditional NLP: $10,000$? $100,000$? $1,000,000$?
  - Derivative-free: 100? 1000?
- Large-scale derivative-free problems are very difficult:
  - quadratic-model-based methods:
    - in principle, the degree of freedom of a full quadratic model is $(n+1)(n+2)/2$
    - in practice, we hope the algorithms finish the job with number of function evaluations of $O(n)$

Difficulty of large-scale problems

- Large-scale problems?
  - Traditional NLP: $10,000$? $100,000$? $1,000,000$?
  - Derivative-free: 100? 1000?
- Large-scale derivative-free problems are very difficult:
  - quadratic-model-based methods:
    - in principle, the degree of freedom of a full quadratic model is $(n+1)(n+2)/2$
    - in practice, we hope the algorithms finish the job with number of function evaluations of $O(n)$
  - difficult to exploit problem structure

- Basic idea:

Basic idea

- Basic idea:
  - divide a difficult problem into a sequence of easy problems, and solve each of them;

Basic idea

- Basic idea:
  - divide a difficult problem into a sequence of easy problems, and solve each of them;

  more specifically,
  - divide a large problem into a sequence of small problems, and solve each of them.

An old idea, very old

- Not a new idea, of course.

An old idea, very old

- Not a new idea, of course.

分而治之 Divide and conquer

An old idea, very old

- Not a new idea, of course.

分而治之

Divide and conquer

故用兵之法，十则围之，五则攻之，倍则分之
凡治众如治寡，分数是也

— Sun Tzu, The Art of War
(6 BCE)

An old idea, very old

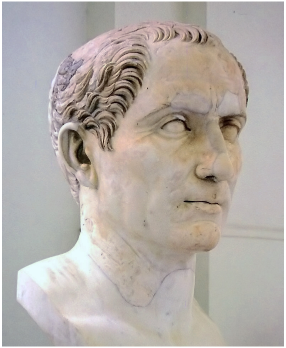
- Not a new idea, of course.

分而治之

故用兵之法，十则围之，五则攻之，倍则分之
凡治众如治寡，分数是也

— Sun Tzu, The Art of War
(6 BCE)

Divide and conquer



Divide et impera.

— Julius Caesar
(1 BCE)

Subspace and decomposition techniques in optimization

- Subspace techniques
  - Gould, Nick, A. Sartenaer, and Ph L. Toint. On iterated-subspace minimization methods for nonlinear optimization. Rutherford Appleton Laboratory, 1994.
  - Yuan, Ya-xiang. Subspace techniques for nonlinear optimization. Some topics in industrial and applied mathematics 8 (2007): 206-218.

Subspace and decomposition techniques in optimization

- Subspace techniques
  - Gould, Nick, A. Sartenaer, and Ph L. Toint. On iterated-subspace minimization methods for nonlinear optimization. Rutherford Appleton Laboratory, 1994.
  - Yuan, Ya-xiang. Subspace techniques for nonlinear optimization. Some topics in industrial and applied mathematics 8 (2007): 206-218.
- Decomposition techniques
  - Boyd, Stephen, Lin Xiao, Almir Mutapcic, and Jacob Mattingley. Notes on decomposition methods. Notes for EE364B, Stanford University (2007).

Subspace and decomposition techniques in optimization

- Subspace techniques
  - Gould, Nick, A. Sartenaer, and Ph L. Toint. On iterated-subspace minimization methods for nonlinear optimization. Rutherford Appleton Laboratory, 1994.
  - Yuan, Ya-xiang. Subspace techniques for nonlinear optimization. Some topics in industrial and applied mathematics 8 (2007): 206-218.
- Decomposition techniques
  - Boyd, Stephen, Lin Xiao, Almir Mutapcic, and Jacob Mattingley. Notes on decomposition methods. Notes for EE364B, Stanford University (2007).
- Coordinate-search . . .

A subspace decomposition framework

- Suppose that the current iterate is x_k.

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that

$$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that

$$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$

  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k^{(i)}$, and obtain $d_k^{(i)}$ $(i = 1, 2, \ldots, m_k)$.

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that
$$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$
  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k^{(i)}$, and obtain $d_k^{(i)}$ $(i = 1, 2, \ldots, m_k)$.
- How to obtain a single step d_k ?

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that
  $$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$
  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k^{(i)}$, and obtain $d_k^{(i)}$ $(i = 1, 2, \ldots, m_k)$.
- How to obtain a single step d_k ?
  - Set
  $$d_k = \sum_{i=0}^{m_k} d_k^{(i)} \ ?$$

A subspace decomposition framework

- Suppose that the current iterate is x_k.

- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that

$$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$

  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k^{(i)}$, and obtain $d_k^{(i)}$ $(i = 1, 2, \ldots, m_k)$.

- Composition:

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that
$$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$
  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k^{(i)}$, and obtain $d_k^{(i)}$ $(i = 1, 2, \ldots, m_k)$.
- Composition:
  - set
$$\mathcal{S}_k = \operatorname{span}\left\{d_k^{(1)}, d_k^{(2)}, \ldots, d_k^{(m_k)}\right\};$$

A subspace decomposition framework

- Suppose that the current iterate is x_k.
- Decomposition:
  - select spaces $\mathcal{S}_k^{(1)}, \mathcal{S}_k^{(2)}, \ldots, \mathcal{S}_k^{(m_k)}$ such that
$$\mathbb{R}^n = \sum_{i=0}^{m_k} \mathcal{S}_k^{(i)};$$
  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k^{(i)}$, and obtain $d_k^{(i)}$ $(i = 1, 2, \ldots, m_k)$.
- Composition:
  - set
$$\mathcal{S}_k = \operatorname{span}\left\{ d_k^{(1)}, d_k^{(2)}, \ldots, d_k^{(m_k)} \right\};$$
  - minimize $f(x_k + d)$ with respect to d on $\mathcal{S}_k$, and obtain d_k.

Localization

Localization

- Trust-region:

$$\begin{aligned} \min \ & f(x_k + d) \\ \text{s.t. } & d \in \mathcal{S}_k^{(i)} \\ & \|d\| \le \Delta_k \end{aligned}$$

Localization

- Trust-region:

$$\begin{aligned} \min\ & f(x_k + d) \\ \text{s.t. } & d \in \mathcal{S}_k^{(i)} \\ & \|d\| \le \Delta_k \end{aligned}$$

- Levenberg-Marquardt:

$$\min_{d \in \mathcal{S}_k^{(i)}} f(x_k + d) + \frac{1}{2}\sigma_k \|d\|^2$$

Localization

- Trust-region:

$$\begin{aligned} \min \ & f(x_k + d) \\ \text{s.t. } & d \in \mathcal{S}_k^{(i)} \\ & \|d\| \le \Delta_k \end{aligned}$$

- Levenberg-Marquardt:

$$\min_{d \in \mathcal{S}_k^{(i)}} f(x_k + d) + \frac{1}{2}\sigma_k \|d\|^2$$

- How to update Δ_k or σ_k?

Localization

- Trust-region:

$$\begin{aligned} \min\ & f(x_k + d) \\ \text{s.t. } & d \in \mathcal{S}_k^{(i)} \\ & \|d\| \le \Delta_k \end{aligned}$$

- Levenberg-Marquardt:

$$\min_{d \in \mathcal{S}_k^{(i)}} f(x_k + d) + \frac{1}{2}\sigma_k \|d\|^2$$

- How to update Δ_k or σ_k?

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\sum_{i=1}^{m_k} \left[f(x_k) - f(x_k + d_k^{(i)}) \right]}$$

Trust-region framework

Algorithm (Trust-region framework)

Trust-region framework

Algorithm (Trust-region framework)

Step 1. Select a constant $\eta \in [0, 1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose $\Delta_0 > 0$, and set $k = 0$.

Trust-region framework

Algorithm (Trust-region framework)

Step 1. Select a constant $\eta \in [0, 1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose $\Delta_0 > 0$, and set $k = 0$.

Step 2. Choose subspaces $\mathcal{S}_k^{(i)}$ $(i = 1, 2, \cdots, m_k)$ of $\mathbb{R}^n$ so that

$$\sum_{i=1}^{m_k} \mathcal{S}_k^{(i)} = \mathbb{R}^n.$$

Trust-region framework

Algorithm (Trust-region framework)

Step 1. Select a constant $\eta \in [0,1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose $\Delta_0 > 0$, and set $k = 0$.

Step 2. Choose subspaces $\mathcal{S}_k^{(i)}$ $(i = 1, 2, \cdots, m_k)$ of $\mathbb{R}^n$ so that

$$\sum_{i=1}^{m_k} \mathcal{S}_k^{(i)} = \mathbb{R}^n.$$

Step 3. For $i = 1, 2, \ldots, m_k$, solve

$$\begin{aligned} \min\ & f(x_k + d) \\ \text{s.t. } & d \in \mathcal{S}_k^{(i)} \\ & \|d\| \le \Delta_k, \end{aligned}$$

to get $d_k^{(i)}$.

Algorithm (Trust-region framework cont.)

Step 4. Obtain d_k by solving

$$\begin{aligned} &\min\ f(x_k + d) \\ &\text{s.t. } d = \sum_{i=1}^{m_k} t^{(i)} d_k^{(i)} \\ &\qquad 0 \le t^{(i)} \le 1, \quad i = 1, 2, \cdots, m_k. \end{aligned}$$

Algorithm (Trust-region framework cont.)

Step 4. Obtain d_k by solving

$$\begin{aligned} &\min\ f(x_k + d) \\ &\text{s.t. } d = \sum_{i=1}^{m_k} t^{(i)} d_k^{(i)} \\ &\qquad 0 \le t^{(i)} \le 1, \quad i = 1, 2, \cdots, m_k. \end{aligned}$$

Step 5. Let

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\sum_{i=1}^{m_k} \left[f(x_k) - f(x_k + d_k^{(i)}) \right]},$$

and set Δ_{k+1} so that

$$\Delta_{k+1} \ge \Delta_k \ \text{ whenever } \ \rho_k > \eta.$$

Trust-region framework

Algorithm (Trust-region framework cont.)

Step 4. Obtain d_k by solving

$$\begin{aligned} &\min\ f(x_k + d) \\ &\text{s.t. } d = \sum_{i=1}^{m_k} t^{(i)} d_k^{(i)} \\ &\qquad 0 \le t^{(i)} \le 1, \quad i = 1, 2, \cdots, m_k. \end{aligned}$$

Step 5. Let

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\sum_{i=1}^{m_k} \left[f(x_k) - f(x_k + d_k^{(i)}) \right]},$$

and set Δ_{k+1} so that

$$\Delta_{k+1} \ge \Delta_k \ \text{ whenever } \ \rho_k > \eta.$$

Step 6. Let $x_{k+1} = x_k + d_k$, increment k by 1, and go to **Step 2**.

Levenberg-Marquardt framework

Algorithm (Levenberg-Marquardt framework)

Levenberg-Marquardt framework

Algorithm (Levenberg-Marquardt framework)

Step 1. Select a constant $\eta \in [0, 1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose a positive number σ_0, and set $k = 0$.

Levenberg-Marquardt framework

Algorithm (Levenberg-Marquardt framework)

Step 1. Select a constant $\eta \in [0, 1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose a positive number σ_0, and set $k = 0$.

Step 2. Choose nonzero subspaces $\mathcal{S}_k^{(i)}$ $(i = 1, 2, \cdots, m_k)$ of $\mathbb{R}^n$ so that

$$\sum_{i=1}^{m_k} \mathcal{S}_k^{(i)} = \mathbb{R}^n.$$

Algorithm (Levenberg-Marquardt framework)

Step 1. Select a constant $\eta \in [0, 1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose a positive number σ_0, and set $k = 0$.

Step 2. Choose nonzero subspaces $\mathcal{S}_k^{(i)}$ $(i = 1, 2, \cdots, m_k)$ of $\mathbb{R}^n$ so that

$$\sum_{i=1}^{m_k} \mathcal{S}_k^{(i)} = \mathbb{R}^n.$$

Step 3. For $i = 1, 2, \ldots, m_k$, solve

$$\min_{d \in \mathcal{S}_k^{(i)}} f(x_k + d) + \frac{1}{2}\sigma_k \|d\|^2$$

to get $d_k^{(i)}$.

Algorithm (Levenberg-Marquardt framework cont.)

Step 4. Solve

$$\min_{t \in \mathbb{R}^{m_k}} f(x_k + D_k t) + \frac{1}{2}\sigma_k \|t\|^2,$$

to obtain t_k, and then set

$$d_k = D_k t_k,$$

where $D_k = (d_k^{(1)} \; d_k^{(2)} \; \cdots \; d_k^{(m_k)})$.

Algorithm (Levenberg-Marquardt framework cont.)

Step 4. Solve

$$\min_{t \in \mathbb{R}^{m_k}} f(x_k + D_k t) + \frac{1}{2}\sigma_k \|t\|^2,$$

to obtain t_k, and then set

$$d_k = D_k t_k,$$

where $D_k = (d_k^{(1)} \; d_k^{(2)} \; \cdots \; d_k^{(m_k)})$.

Step 5. Let

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\sum_{i=1}^{m_k} \left[f(x_k) - f(x_k + d_k^{(i)}) \right]},$$

and set σ_{k+1} so that

$$\sigma_{k+1} \leq \sigma_k \quad \text{whenever} \quad \rho_k > \eta.$$

Levenberg-Marquardt framework

Algorithm (Levenberg-Marquardt framework cont.)

Step 4. Solve

$$\min_{t \in \mathbb{R}^{m_k}} f(x_k + D_k t) + \frac{1}{2}\sigma_k \|t\|^2,$$

to obtain t_k, and then set

$$d_k = D_k t_k,$$

where $D_k = (d_k^{(1)} \ d_k^{(2)} \ \cdots \ d_k^{(m_k)})$.

Step 5. Let

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\sum_{i=1}^{m_k} \left[f(x_k) - f(x_k + d_k^{(i)}) \right]},$$

and set σ_{k+1} so that

$$\sigma_{k+1} \le \sigma_k \quad \text{whenever} \quad \rho_k > \eta.$$

Step 6. Let $x_{k+1} = x_k + d_k$, increment k by 1, and go to **Step 2**.

Assumptions

Assumption

Assumption

1. *The function f is bounded from below and twice continuously differentiable, and $\nabla^2 f$ is bounded on $\mathbb{R}^n$.*

Assumptions

Assumption

1. *The function f is bounded from below and twice continuously differentiable, and $\nabla^2 f$ is bounded on $\mathbb{R}^n$.*

2. *The sequence $\{m_k\}$ is bounded.*

Assumptions

Assumption

1. *The function f is bounded from below and twice continuously differentiable, and $\nabla^2 f$ is bounded on $\mathbb{R}^n$.*
2. *The sequence $\{m_k\}$ is bounded.*
3. *The smallest eigenvalues of $\sum_{i=1}^{m_k} P_k^{(i)}$ are bounded away from zero, where $P_k^{(i)}$ is the orthogonal projection matrix from $\mathbb{R}^n$ onto $\mathcal{S}_k^{(i)}$.*

Global convergence

Theorem

Suppose that the assumptions stated before hold, then the iterates $\{x_k\}$ *generated by either of the frameworks satisfy*

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

Theorem

Suppose that the assumptions stated before hold, and additionally

$$\Delta_{k+1} \geq \alpha \Delta_k$$

for some constant $\alpha \in (0, 1]$, *then the iterates* $\{x_k\}$ *generated by the trust-region framework satisfy*

$$\min_{0 \leq \ell \leq k} \|\nabla f(x_\ell)\| \leq C_1 \sqrt{\frac{m}{k}},$$

where m *is an upper bound of* $\{m_k\}$.

Theorem

Suppose that the assumptions stated before hold, and additionally

$$\sigma_{k+1} \leq \beta \sigma_k$$

for some constant $\beta \geq 1$, then the iterates $\{x_k\}$ generated by the Levenberg-Marquardt framework satisfy

$$\min_{0 \leq \ell \leq k} \|\nabla f(x_\ell)\| \leq C_2 \sqrt{\frac{m}{k}},$$

where m is an upper bound of $\{m_k\}$.

Number of subspaces and subproblem solution precision

We have thus the worst case complexity: $O(\varepsilon^{-2}m)$

Number of subspaces and subproblem solution precision

We have thus the worst case complexity: $O(\varepsilon^{-2}m)$

Using this and the WCC $O(n^2\varepsilon^{-2})$ for subproblems,

- a reasonable choice for m is $O(\sqrt{n})$
- a reasonable subproblem solution accuracy is $O(n^{-\frac{1}{4}})$

Applications to derivative-free optimization

Properties of the framework

Applications to derivative-free optimization

Properties of the framework

- *It does not explicitly require derivatives.*

Properties of the framework

- *It does not explicitly require derivatives.*
- *It is naturally parallel.*

Applications to derivative-free optimization

Properties of the framework

- *It does not explicitly require derivatives.*
- *It is naturally parallel.*
- *It is naturally multilevel.*

Applications to derivative-free optimization

Properties of the framework

- *It does not explicitly require derivatives.*
- *It is naturally parallel.*
- *It is naturally multilevel.*

$$\Downarrow$$

Our goal

Parallel and multilevel algorithms without using derivatives and capable of solving relatively large problems.

Very preliminary numerical results

- Use the Levenberg-Marquardt framework
- Subproblem solver: `NEWUOA`
- Number of subspaces: $\sqrt{n/2}$
- Benchmark: `NEWUOA`
- Very preliminary: not parallel, not multilevel, not large-scale . . .
- Dimension of test problems: 25, 30, 35, 40
- Denote our code as `SSD`

VARDIM

Table : Numerical results of VARDIM

n	25	30	35	40	
$\#f$	8343	8926	12689	17741	NEWUOA
	3592	6222	7507	16653	SSD
f_{final}	1.61E-11	4.08E-11	4.93E-11	1.76E-10	NEWUOA
	9.74E-11	6.85E-10	5.74E-11	7.89E-13	SSD

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

Table : Numerical results of PENALTY1

n	25	30	35	40	
$\#f$	9532	10947	14427	13577	NEWUOA
	2089	2784	2348	2812	SSD
f_{final}	2.03E-04	2.48E-04	2.93E-04	3.39E-04	NEWUOA
	2.04E-04	2.50E-04	2.95E-04	3.41E-04	SSD

$$f(x) = 10^{-15}\sum_{i=1}^{n}(x_i - 1)^2 + \left(\frac{1}{4} - \sum_{i=1}^{n} x_i^2\right)^2$$

SBRYBND

Table : Numerical results of SBRYBND

n	25	30	35	40	
$\#f$	968	576	2052	2363	`NEWUOA`
	27889	53103	90304	206608	`SSD`
f_{final}	235	326	342	395	`NEWUOA`
	3.08	3.08	3.08	3.08	`SSD`
	134	284	233	229	

$$f(x) = \sum_{i=1}^{n} \left[(2 + 5p_i^2 x_i^2) p_i x_i + 1 - \sum_{j \in J_i} p_j x_j (1 + p_j x_j) \right]^2,$$

where $J_i = \{ j \mid j \neq i, \max\{1, i-5\} \leq j \leq \min\{n, j+1\}\}$, and $p_i = \exp\left(6 \frac{i-1}{n-1}\right)$.

CHROSEN

Table : Numerical results of CHROSEN

n	25	30	35	40	
$\#f$	1123	1445	1717	1859	NEWUOA
	96040	103296	127726	142272	SSD
f_{final}	8.94E-12	1.07E-11	1.13E-11	3.14E-11	NEWUOA
	2.95E-10	5.49E-10	7.26E-10	8.09E-10	SSD

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

Concluding remarks

- A subspace decomposition framework (two versions) with global convergence and convergence rate
- Possible to develop parallel and multilevel methods without using derivatives

Concluding remarks

- A subspace decomposition framework (two versions) with global convergence and convergence rate
- Possible to develop parallel and multilevel methods without using derivatives
- "Clever" way of choosing subspaces ...

Concluding remarks

- A subspace decomposition framework (two versions) with global convergence and convergence rate
- Possible to develop parallel and multilevel methods without using derivatives
- "Clever" way of choosing subspaces . . .
    - not try to cover the whole space, but . . .

Concluding remarks

- A subspace decomposition framework (two versions) with global convergence and convergence rate
- Possible to develop parallel and multilevel methods without using derivatives
- "Clever" way of choosing subspaces . . .
  - not try to cover the whole space, but . . .
  - choose subspaces randomly

Merci! 谢谢！

zhang@mat.uc.pt