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
Axiom (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.*
### 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
Axiom

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 (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!
1. Derivative-free optimization
2. Motivation and basic idea
3. A subspace decomposition framework
4. Global convergence
5. Global rate
6. Applications to derivative-free optimization
7. Very preliminary numerical results
8. Concluding remarks
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
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*
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
Two main classes of rigorous methods in DFO

- **Directional methods**, like direct search
- **Model-based methods**, like trust-region methods

Difficulty of large-scale problems

- Large-scale problems?

Traditional NLP:

- 10,000
- 100,000
- 1,000,000

Derivative-free:

- 100
- 1,000

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 \(O(n)\).

Motivation and basic idea 8/30
Difficulty of large-scale problems

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

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 \(O(n)\).

Motivation and basic idea
Difficulty of large-scale problems

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

- Divide a difficult problem into a sequence of easy problems, and solve each of them.
- Divide a large problem into a sequence of small problems, and solve each of them.
Basic idea:

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

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)

Divide et impera.

— Julius Caesar (1 BCE)
Subspace and decomposition techniques in optimization

- **Subspace techniques**
Subspace techniques


Decomposition techniques

Subspace and decomposition techniques in optimization

- **Subspace techniques**

- **Decomposition techniques**

- **Coordinate-search . . .**
Suppose that the current iterate is $x_k$. 

Decomposition:

Select spaces $S(k)$, $S(2)_k$, ..., $S(m)_k$ such that:

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

minimize $f(x_k + d)$ with respect to $d$ on $S(i)_k$, and obtain $d(i)_k$.
A subspace decomposition framework

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

Decomposition:

- select spaces $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that

$$\mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)};$$
Suppose that the current iterate is $x_k$.

**Decomposition:**

- Select spaces $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that
  \[ \mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)}; \]

- Minimize $f(x_k + d)$ with respect to $d$ on $S_k^{(i)}$, and obtain $d_k^{(i)}$ ($i = 1, 2, \ldots, m_k$).
Suppose that the current iterate is $x_k$.

Decomposition:

- select spaces $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that
  \[
  \mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)};
  \]

- minimize $f(x_k + d)$ with respect to $d$ on $S_k^{(i)}$, 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 $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that
    \[ \mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)}; \]
  - minimize $f(x_k + d)$ with respect to $d$ on $S_k^{(i)}$, and obtain $d_k^{(i)}$ ($i = 1, 2, \ldots, m_k$).

- How to obtain a single step $d_k$?
  - Set
    \[ d_k = \sum_{i=0}^{m_k} d_k^{(i)}? \]
Suppose that the current iterate is $x_k$.

Decomposition:
- select spaces $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that
  $$\mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)};$$
- minimize $f(x_k + d)$ with respect to $d$ on $S_k^{(i)}$, and obtain $d_k^{(i)}$ ($i = 1, 2, \ldots, m_k$).

Composition:
Suppose that the current iterate is $x_k$.

**Decomposition:**

- select spaces $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that
  \[ \mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)}; \]

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

**Composition:**

- set
  \[ S_k = \text{span} \left\{ d_k^{(1)}, d_k^{(2)}, \ldots, d_k^{(m_k)} \right\}; \]
Suppose that the current iterate is $x_k$.

**Decomposition:**

- select spaces $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(m_k)}$ such that
  \[ \mathbb{R}^n = \sum_{i=0}^{m_k} S_k^{(i)}; \]
  - minimize $f(x_k + d)$ with respect to $d$ on $S_k^{(i)}$, and obtain $d_k^{(i)}$ ($i = 1, 2, \ldots, m_k$).

**Composition:**

- set
  \[ S_k = \text{span} \left\{ d_k^{(1)}, d_k^{(2)}, \ldots, d_k^{(m_k)} \right\}; \]
  - minimize $f(x_k + d)$ with respect to $d$ on $S_k$, and obtain $d_k$. 

Localization

Trust-region:
\[ \min f(x_k + d) \]
subject to \( d \in S(i_k) \)
\[ \|d\| \leq \Delta_k \]

Levenberg-Marquardt:
\[ \min d \in S(i_k) \]
\[ f(x_k + d) + \frac{1}{2} \sigma_k^2 \|d\|^2 \]

How to update \( \Delta_k \) or \( \sigma_k \)?
Localization

- Trust-region:

\[
\begin{align*}
\text{min } & f(x_k + d) \\
\text{s.t. } & d \in S_k^{(i)} \\
& \|d\| \leq \Delta_k
\end{align*}
\]
Localization

- **Trust-region:**

\[
\begin{align*}
\min & \quad f(x_k + d) \\
\text{s.t.} & \quad d \in S_k^{(i)} \\
& \quad \|d\| \leq \Delta_k
\end{align*}
\]

- **Levenberg-Marquardt:**

\[
\min_{d \in S_k^{(i)}} f(x_k + d) + \frac{1}{2}\sigma_k \|d\|^2
\]
Trust-region:

\[
\begin{align*}
\min & \quad f(x_k + d) \\
\text{s.t.} & \quad d \in S_k^{(i)} \\
& \quad \|d\| \leq \Delta_k
\end{align*}
\]

Levenberg-Marquardt:

\[
\min_{d \in S_k^{(i)}} f(x_k + d) + \frac{1}{2}\sigma_k \|d\|^2
\]

How to update \(\Delta_k\) or \(\sigma_k\)?
Localization

- Trust-region:

\[
\begin{align*}
\min & \ f(x_k + d) \\
\text{s.t.} & \ d \in S_k^{(i)} \\
& \|d\| \leq \Delta_k
\end{align*}
\]

- Levenberg-Marquardt:

\[
\min_{d \in S_k^{(i)}} f(x_k + d) + \frac{1}{2} \sigma_k ||d||^2
\]

- How to update \(\Delta_k\) or \(\sigma_k\)?

\[
\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\sum_{i=1}^{m_k} \left[ f(x_k) - f(x_k + d_k^{(i)}) \right]}.
\]
### Algorithm (Trust-region framework)

1. Select a constant $\eta \in [0, 1)$, pick a starting point $x_0 \in \mathbb{R}^n$, choose $\Delta_0 > 0$, and set $k = 0$.
2. Choose subspaces $S(i)_k$ of $\mathbb{R}^n$ so that $\sum_{i=1}^{m_k} S(i)_k = \mathbb{R}^n$.
3. For $i = 1, 2, \ldots, m_k$, solve $\min f(x_k + d)$ s.t. $d \in S(i)_k \|d\| \leq \Delta_k$, to get $d(i)_k$. 

A subspace decomposition 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 $k = 0$. 
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 $S_k^{(i)}$ ($i = 1, 2, \ldots, m_k$) of $\mathbb{R}^n$ so that

$$
\sum_{i=1}^{m_k} S_k^{(i)} = \mathbb{R}^n.
$$
**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 $S_k^{(i)}$ ($i = 1, 2, \ldots, m_k$) of $\mathbb{R}^n$ so that

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

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

$$
\min f(x_k + d) \\
\text{s.t. } d \in S_k^{(i)} \\
\|d\| \leq \Delta_k,
$$

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

Step 4. Obtain $d_k$ by solving

\[
\min f(x_k + d)
\]

s.t. 
\[
d = \sum_{i=1}^{m_k} t^{(i)} d_k^{(i)}
\]

\[
0 \leq t^{(i)} \leq 1, \quad i = 1, 2, \ldots, m_k.
\]
Algorithm (Trust-region framework cont.)

**Step 4.** Obtain $d_k$ by solving

$$
\begin{align*}
\min & \quad f(x_k + d) \\
\text{s.t.} & \quad d = \sum_{i=1}^{m_k} t^{(i)} d^{(i)}_k \\
& \quad 0 \leq t^{(i)} \leq 1, \quad i = 1, 2, \cdots, m_k.
\end{align*}
$$

**Step 5.** Let

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

and set $\Delta_{k+1}$ so that

$$
\Delta_{k+1} \geq \Delta_k \quad \text{whenever} \quad \rho_k > \eta.
$$
Algorithm (Trust-region framework cont.)

**Step 4.** Obtain $d_k$ by solving

$$
\min f(x_k + d) \\
\text{s.t. } d = \sum_{i=1}^{m_k} t^{(i)} d^{(i)}_k \\
0 \leq t^{(i)} \leq 1, \quad i = 1, 2, \ldots, m_k.
$$

**Step 5.** Let

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

and set $\Delta_{k+1}$ so that

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

**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)

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

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

3. For $i = 1, 2, \ldots, m_k$, solve $\min_{d \in S(i)_k} f(x_k + d) + \frac{1}{2} \sigma_k \|d\|^2$ to get $d(i)_k$. 

A subspace decomposition framework 16/30
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 $\sigma_0$, 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 $\sigma_0$, and set $k = 0$.

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

$$
\sum_{i=1}^{m_k} S_k^{(i)} = \mathbb{R}^n.
$$
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 $\sigma_0$, and set $k = 0$.

**Step 2.** Choose nonzero subspaces $S_k^{(i)}$ ($i = 1, 2, \ldots, m_k$) of $\mathbb{R}^n$ so that

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

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

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

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

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 $D_k = (d_k^{(1)} \ d_k^{(2)} \ \cdots \ d_k^{(m_k)}).$
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 \( \sigma_{k+1} \) so that

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

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

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

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 $\sum_{i=1}^{m_k} P(i)k$ are bounded away from zero, where $P(i)k$ is the orthogonal projection matrix from $\mathbb{R}^n$ onto $S(i)k$. 

Global convergence
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 $S_k^{(i)}$. 
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 \( \{m_k\} \).
We have thus the worst case complexity: $O(\varepsilon^{-2} m)$
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 $O(n^{-\frac{1}{4}})$
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.*
Properties of the framework

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

↓

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
### Table: Numerical results of VARDIM

<table>
<thead>
<tr>
<th>$n$</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>$#f$</td>
<td>8343</td>
<td>8926</td>
<td>12689</td>
<td>17741</td>
</tr>
<tr>
<td></td>
<td>3592</td>
<td>6222</td>
<td>7507</td>
<td>16653</td>
</tr>
<tr>
<td>$f_{\text{final}}$</td>
<td>1.61E-11</td>
<td>4.08E-11</td>
<td>4.93E-11</td>
<td>1.76E-10</td>
</tr>
<tr>
<td></td>
<td>9.74E-11</td>
<td>6.85E-10</td>
<td>5.74E-11</td>
<td>7.89E-13</td>
</tr>
<tr>
<td></td>
<td>NEWUOA</td>
<td>SSD</td>
<td>NEWUOA</td>
<td>SSD</td>
</tr>
</tbody>
</table>

$$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 + \left[ \sum_{i=1}^{n} i(x_i - 1) \right]^2 + \left[ \sum_{i=1}^{n} i(x_i - 1) \right]^4$$
### Table: Numerical results of PENALTY1

<table>
<thead>
<tr>
<th>$n$</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>$# f$</td>
<td>9532</td>
<td>10947</td>
<td>14427</td>
<td>13577</td>
</tr>
<tr>
<td></td>
<td>2089</td>
<td>2784</td>
<td>2348</td>
<td>2812</td>
</tr>
<tr>
<td>$f_{\text{final}}$</td>
<td>2.03E-04</td>
<td>2.48E-04</td>
<td>2.93E-04</td>
<td>3.39E-04</td>
</tr>
<tr>
<td></td>
<td>2.04E-04</td>
<td>2.50E-04</td>
<td>2.95E-04</td>
<td>3.41E-04</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>( n )</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td># ( f )</td>
<td>968</td>
<td>576</td>
<td>2052</td>
<td>2363</td>
</tr>
<tr>
<td></td>
<td>27889</td>
<td>53103</td>
<td>90304</td>
<td>206608</td>
</tr>
<tr>
<td>( f_{\text{final}} )</td>
<td>235</td>
<td>326</td>
<td>342</td>
<td>395</td>
</tr>
<tr>
<td></td>
<td>3.08</td>
<td>3.08</td>
<td>3.08</td>
<td>3.08</td>
</tr>
<tr>
<td>( \text{SSD} )</td>
<td>134</td>
<td>284</td>
<td>233</td>
<td>229</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>n</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1123</td>
<td>1445</td>
<td>1717</td>
<td>1859</td>
<td>NEWUOA</td>
<td>SSD</td>
</tr>
<tr>
<td></td>
<td>96040</td>
<td>103296</td>
<td>127726</td>
<td>142272</td>
<td>SSD</td>
<td>SSD</td>
</tr>
<tr>
<td>#f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_{final}</td>
<td>8.94E-12</td>
<td>1.07E-11</td>
<td>1.13E-11</td>
<td>3.14E-11</td>
<td>NEWUOA</td>
<td>SSD</td>
</tr>
<tr>
<td></td>
<td>2.95E-10</td>
<td>5.49E-10</td>
<td>7.26E-10</td>
<td>8.09E-10</td>
<td>SSD</td>
<td></td>
</tr>
</tbody>
</table>

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

- A subspace decomposition framework (two versions) with global convergence and convergence rate

- Possible to develop parallel and multilevel methods without using derivatives
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...
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