

# A Derivative-Free Approach to Least-squares Minimization

Hongchao Zhang  
hozhang@math.lsu.edu

Department of Mathematics  
Center for Computation and Technology  
Louisiana State University

Joint work with Andrew R. Conn and Katya Scheinberg

August 2009, Northern Illinois University

# Motivation

- **Least-Squares problems** are a particular but **very important** class of problems. For example, **data fitting applications** and **solving systems of equations**.
- There are **many applications** where the analytic function is **not available** or computed by a **black box simulation package** and the **simulation is expensive**.
- We have a **well developed theory and algorithms** and our preferred approach uses a **polynomial basis and trust regions**
- Often there is **noise in the function values**.

Evidently the best methods **must exploit the problems structure**.

# Motivation

- **Least-Squares problems** are a particular but **very important** class of problems. For example, **data fitting applications** and **solving systems of equations**.
- There are **many applications** where the analytic function is **not available** or computed by a **black box simulation package** and the **simulation is expensive**.
- We have a **well developed theory and algorithms** and our preferred approach uses a **polynomial basis and trust regions**
- Often there is **noise in the function values**.

Evidently the best methods **must exploit the problems structure**.

# Motivation

- **Least-Squares problems** are a particular but **very important** class of problems. For example, **data fitting applications** and **solving systems of equations**.
- There are **many applications** where the analytic function is **not available** or computed by a **black box simulation package** and **the simulation is expensive**.
- We have a **well developed theory and algorithms** and our preferred approach uses a **polynomial basis and trust regions**
- Often there is **noise in the function values**.

Evidently the best methods **must exploit the problems structure**.

# Motivation

- **Least-Squares problems** are a particular but **very important** class of problems. For example, **data fitting applications** and **solving systems of equations**.
- There are **many applications** where the analytic function is **not available** or computed by a **black box simulation package** and **the simulation is expensive**.
- We have a **well developed theory and algorithms** and our preferred approach uses a **polynomial basis and trust regions**
  - Often there is **noise in the function values**.

Evidently the best methods **must exploit the problems structure**.

# Motivation

- **Least-Squares problems** are a particular but **very important** class of problems. For example, **data fitting applications** and **solving systems of equations**.
- There are **many applications** where the analytic function is **not available** or computed by a **black box simulation package** and **the simulation is expensive**.
- We have a **well developed theory and algorithms** and our preferred approach uses a **polynomial basis and trust regions**
- Often there is **noise in the function values**.

Evidently **the best methods must exploit the problems structure**.

# Motivation

- **Least-Squares problems** are a particular but **very important** class of problems. For example, **data fitting applications** and **solving systems of equations**.
- There are **many applications** where the analytic function is **not available** or computed by a **black box simulation package** and **the simulation is expensive**.
- We have a **well developed theory and algorithms** and our preferred approach uses a **polynomial basis and trust regions**
- Often there is **noise in the function values**.

Evidently **the best methods must exploit the problems structure**.

# The Problem

Optimize

$$\Phi(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^m f_i^2(\mathbf{x}) = \frac{1}{2} \|F(\mathbf{x})\|^2$$

where

- $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))^\top$
- $f_i(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ , continuously differentiable,  $n < 300$
- Derivatives not available
- Function evaluation expensive and often inaccurate

# Several Approaches

- Direct methods
  - ◇ Pattern search:  
geometric simplices, discrete grids
  - ◇ Random process:  
simulated annealing, genetic algorithms
  - ◇ Conjugate grids or conjugate directions
- Finite difference approximations:  
forward difference, central difference
- Trust region methods :  
Interpolation/regression models, typically linear or quadratic polynomial models

- Well-known results
- DFSL algorithm
- Global convergence
- Local convergence
- Numerical results

# Some Well-Known Results

- $\Lambda$ -poised set (Conn, Scheinberg, etc. 2005)  
(Essentially characterizes good geometry.)  
 $\mathbf{Y} = \{\mathbf{y}^0, \mathbf{y}^1, \dots, \mathbf{y}^p\}$  is  $\Lambda$ -poised in  $\mathcal{B}(1)$  for a polynomial space  $\mathcal{P}$  if and only if there exists a  $\lambda(\mathbf{x}) \in \mathbb{R}^{p+1}$ , such that

$$\sum_{i=0}^p \lambda_i(\mathbf{x}) \mathbf{b}(\mathbf{y}^i) = \mathbf{b}(\mathbf{x}) \quad \text{with } \|\lambda(\mathbf{x})\| \leq \Lambda,$$

where,  $\mathbf{b}(\mathbf{x}) = \{b_0(\mathbf{x}), b_1(\mathbf{x}), \dots, b_p(\mathbf{x})\}$  is a basis in the polynomial space  $\mathcal{P}$ .

- (Can realize it in practise.)  
A  $\Lambda$ -poised set  $\mathbf{Y}$  can be generated in a finite number, at most  $p$ , improvement steps.

- Fully Linear Model

(Essentially needed for an equivalent of a **Cauchy point**.)

If  $f \in C^1(\mathcal{B}(\Delta))$ ,  $\mathbf{Y} \subset \mathcal{B}(\Delta)$  is  $\Lambda$ -poised (with  $|\mathbf{Y}| \geq n + 1$ ), then the polynomial  $p(\cdot)$  with minimum possible degree satisfying

$$[\textit{Interpolating}] \quad p(\mathbf{y}^i) = f(\mathbf{y}^i), \quad i = 1, \dots, |\mathbf{Y}|,$$

is at least a fully linear model of  $f$  on  $\mathcal{B}(\Delta)$ , meaning

$$\begin{aligned} \|\nabla f(\mathbf{s}) - \nabla p(\mathbf{s})\| &\leq \kappa_g \Delta, \\ |f(\mathbf{s}) - p(\mathbf{s})| &\leq \kappa_f \Delta^2, \end{aligned}$$

for any  $\mathbf{s} \in \mathcal{B}(\Delta)$ .

# Derivative free least-squares (DFLS) algorithm

- Interpolation model  $\phi(\mathbf{y}, \mathbf{s})$  of  $\Phi(\cdot)$  at base point  $\mathbf{y} \in \mathcal{B}(\mathbf{z}, \Delta)$

- (a) Given  $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$ ,  $\Lambda$ -poised with  $|\mathbf{Y}| \geq n + 1$ , for all  $i = 1, \dots, m$ ,

$$\text{For each } f_i \text{ define } m_i(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y}} \ell_{\mathbf{y}}(\mathbf{x}) f_i(\mathbf{y}),$$

where  $\ell_{\mathbf{y}}(\mathbf{x})$  is a Lagrange polynomial of  $\mathbf{Y}$  at  $\mathbf{y}$ .

- (b) All the models  $m_i(\mathbf{x})$  are at least fully linear on  $\mathcal{B}(\mathbf{z}, \Delta)$ , so

$$\begin{aligned} \|\nabla f_i(\mathbf{s}) - \nabla m_i(\mathbf{s})\| &\leq \kappa_g \Delta, \\ \text{and } |f_i(\mathbf{s}) - m_i(\mathbf{s})| &\leq \kappa_f \Delta^2, \end{aligned}$$

for all  $i = 1, \dots, m$  and for all  $\mathbf{s} \in \mathcal{B}(\mathbf{z}, \Delta)$

# Derivative free least-squares (DFLS) algorithm

- Interpolation model  $\phi(\mathbf{y}, \mathbf{s})$  of  $\Phi(\cdot)$  at base point  $\mathbf{y} \in \mathcal{B}(\mathbf{z}, \Delta)$

- (a) Given  $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$ ,  $\Lambda$ -poised with  $|\mathbf{Y}| \geq n + 1$ , for all  $i = 1, \dots, m$ ,

$$\text{For each } f_i \text{ define } m_i(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y}} \ell_{\mathbf{y}}(\mathbf{x}) f_i(\mathbf{y}),$$

where  $\ell_{\mathbf{y}}(\mathbf{x})$  is a Lagrange polynomial of  $\mathbf{Y}$  at  $\mathbf{y}$ .

- (b) All the models  $m_i(\mathbf{x})$  are at least fully linear on  $\mathcal{B}(\mathbf{z}, \Delta)$ , so

$$\begin{aligned} \|\nabla f_i(\mathbf{s}) - \nabla m_i(\mathbf{s})\| &\leq \kappa_g \Delta, \\ \text{and } |f_i(\mathbf{s}) - m_i(\mathbf{s})| &\leq \kappa_f \Delta^2, \end{aligned}$$

for all  $i = 1, \dots, m$  and for all  $\mathbf{s} \in \mathcal{B}(\mathbf{z}, \Delta)$

# Derivative free least-squares (DFLS) algorithm

- Interpolation model  $\phi(\mathbf{y}, \mathbf{s})$  of  $\Phi(\cdot)$  at base point  $\mathbf{y} \in \mathcal{B}(\mathbf{z}, \Delta)$

- (a) Given  $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$ ,  $\Lambda$ -poised with  $|\mathbf{Y}| \geq n + 1$ , for all  $i = 1, \dots, m$ ,

$$\text{For each } f_i \text{ define } m_i(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y}} \ell_{\mathbf{y}}(\mathbf{x}) f_i(\mathbf{y}),$$

where  $\ell_{\mathbf{y}}(\mathbf{x})$  is a Lagrange polynomial of  $\mathbf{Y}$  at  $\mathbf{y}$ .

- (b) All the models  $m_i(\mathbf{x})$  are at least fully linear on  $\mathcal{B}(\mathbf{z}, \Delta)$ , so

$$\begin{aligned} \|\nabla f_i(\mathbf{s}) - \nabla m_i(\mathbf{s})\| &\leq \kappa_g \Delta, \\ \text{and } |f_i(\mathbf{s}) - m_i(\mathbf{s})| &\leq \kappa_f \Delta^2, \end{aligned}$$

for all  $i = 1, \dots, m$  and for all  $\mathbf{s} \in \mathcal{B}(\mathbf{z}, \Delta)$

# The DFSL algorithm background (continued)

(c) Define the interpolation model

$$\phi(\mathbf{y}, \mathbf{s}) = c_\phi(\mathbf{y}) + \mathbf{g}_\phi(\mathbf{y})^\top \mathbf{s} + \frac{1}{2} \mathbf{s}^\top H_\phi(\mathbf{y}) \mathbf{s},$$

where

$$c_\phi(\mathbf{y}) = \frac{1}{2} \mathbf{m}(\mathbf{y})^\top \mathbf{m}(\mathbf{y}), \quad \mathbf{g}_\phi(\mathbf{y}) = J(\mathbf{y})^\top \mathbf{m}(\mathbf{y})$$

and

$$H_\phi(\mathbf{y}) = \begin{cases} J(\mathbf{y})^\top J(\mathbf{y}) & \text{if } \|\mathbf{g}_\phi(\mathbf{y})\| \geq \gamma_1, \\ J(\mathbf{y})^\top J(\mathbf{y}) + \gamma_3 \|\mathbf{m}(\mathbf{y})\| I & \text{else if } c_\phi(\mathbf{y}) < \gamma_2 \|\mathbf{g}_\phi(\mathbf{y})\|, \\ J(\mathbf{y})^\top J(\mathbf{y}) + \sum_{i=1}^m m_i(\mathbf{y}) \nabla^2 m_i(\mathbf{y}) & \text{otherwise,} \end{cases}$$

with

$$\mathbf{m}(\mathbf{y}) = (m_1(\mathbf{y}), m_2(\mathbf{y}), \dots, m_m(\mathbf{y}))^\top \text{ and } J(\mathbf{y}) = \nabla \mathbf{m}(\mathbf{y}).$$

# Comments on the model definition

Exact linear model plus intelligent Hessian model

## Hessian

First definition is ideal for **small residual** problems.

Second definition is a **regularisation** (convexification).

Third definition is a full **second-order** model.

## But

Note that if  $\|g_\phi(\mathbf{y})\|$ , is sufficiently large, **first order changes dominate** even if the **residual is not relatively small**.

On the other hand, if the first-order change is **relatively small** and the **residuals are small** the second definition is a **compromise** over using the full second derivatives.

# Comments on the model definition

Exact linear model plus intelligent Hessian model

## Hessian

First definition is ideal for **small residual** problems.

Second definition is a **regularisation** (convexification).

Third definition is a full **second-order** model.

## But

Note that if  $\|\mathbf{g}_\phi(\mathbf{y})\|$ , is sufficiently large, **first order changes dominate** even if the **residual is not relatively small**.

On the other hand, if the first-order change is **relatively small** and the **residuals are small** the second definition is a **compromise** over using the full second derivatives.

# Description of the algorithm

## Step 0 (Initialization)

Given  $\mathbf{x}_0$ ,  $0 < \rho_0 \leq \bar{\Delta}_0 \leq \Delta_0 \leq \Delta_{max}$ ,  $\mathbf{Y}_0$  with  $\mathbf{x}_0 \in \mathbf{Y}_0 \subset \mathcal{B}(\mathbf{x}_0, \bar{\Delta}_0)$  and  $|\mathbf{Y}_0| = N_p \geq n + 1$ ,  $\epsilon_\beta \in (0, 1)$  and  $\beta > 0$ . Set  $k = 0$ .

## Step 1 (Critical step)

Choose a base point  $\mathbf{y}_k \in \mathbf{Y}_k$  and calculate

$$\mathbf{g}_{\phi_k} = J(\mathbf{y}_k)^\top \mathbf{m}(\mathbf{y}_k) + H_\phi(\mathbf{y}_k)(\mathbf{x}_k - \mathbf{y}_k). \quad (1)$$

If  $\|\mathbf{g}_{\phi_k}\| \leq \epsilon_\beta$ , let  $\bar{\Delta}_k^{(0)} = \bar{\Delta}_k$ ; possibly modifying  $\mathbf{Y}_k$  to make sure  $\mathbf{Y}_k$  is  $\Lambda$ -poised in  $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k)$ , with

$\bar{\Delta}_k = \min\{\bar{\Delta}_k^{(0)}, \beta\|\mathbf{g}_{\phi_k}\|\}$ , where  $\mathbf{g}_{\phi_k}$  is recalculated by (1).

Unless  $\mathbf{Y}_k$  is stationary, possible in a finite number of steps.

Would be nice to choose best point as base point but in practise too expensive. [ $O(n^3)$  versus  $O(n^2)$ ]

# Description of the algorithm

## Step 0 (Initialization)

Given  $\mathbf{x}_0$ ,  $0 < \rho_0 \leq \bar{\Delta}_0 \leq \Delta_0 \leq \Delta_{max}$ ,  $\mathbf{Y}_0$  with  $\mathbf{x}_0 \in \mathbf{Y}_0 \subset \mathcal{B}(\mathbf{x}_0, \bar{\Delta}_0)$  and  $|\mathbf{Y}_0| = N_p \geq n + 1$ ,  $\epsilon_\beta \in (0, 1)$  and  $\beta > 0$ . Set  $k = 0$ .

## Step 1 (Critical step)

Choose a base point  $\mathbf{y}_k \in \mathbf{Y}_k$  and calculate

$$\mathbf{g}_{\phi_k} = J(\mathbf{y}_k)^\top \mathbf{m}(\mathbf{y}_k) + H_\phi(\mathbf{y}_k)(\mathbf{x}_k - \mathbf{y}_k). \quad (1)$$

If  $\|\mathbf{g}_{\phi_k}\| \leq \epsilon_\beta$ , let  $\bar{\Delta}_k^{(0)} = \bar{\Delta}_k$ ; possibly modifying  $\mathbf{Y}_k$  to make sure  $\mathbf{Y}_k$  is  $\Lambda$ -poised in  $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k)$ , with

$\bar{\Delta}_k = \min\{\bar{\Delta}_k^{(0)}, \beta\|\mathbf{g}_{\phi_k}\|\}$ , where  $\mathbf{g}_{\phi_k}$  is recalculated by (1).

Unless  $\mathbf{Y}_k$  is stationary, possible in a finite number of steps.

Would be nice to choose best point as base point but in practise too expensive.  $[O(n^3)$  versus  $O(n^2)]$

# Description of the algorithm (continued)

## Step 2 (Step calculation)

Solve the trust region subproblem :

$$\begin{array}{ll} \min & \phi_k(\mathbf{d}) \\ \text{s.t.} & \|\mathbf{d}\| \leq \Delta_k, \end{array}$$

to obtain the step  $\mathbf{d}_k$ , where  $\phi_k(\mathbf{d}) = \phi(\mathbf{y}_k, (\mathbf{x}_k - \mathbf{y}_k) + \mathbf{d})$ .

The trust region radius has a dual purpose:

- (a) restricts stepsize to region where model assumed good
- (b) determines the sampling space.

# Description of the algorithm (continued)

## Step 2 (Step calculation)

Solve the trust region subproblem :

$$\begin{array}{ll} \min & \phi_k(\mathbf{d}) \\ \text{s.t.} & \|\mathbf{d}\| \leq \Delta_k, \end{array}$$

to obtain the step  $\mathbf{d}_k$ , where  $\phi_k(\mathbf{d}) = \phi(\mathbf{y}_k, (\mathbf{x}_k - \mathbf{y}_k) + \mathbf{d})$ .

The trust region radius has a dual purpose:

- (a) restricts stepsize to region where model assumed good
- (b) determines the sampling space.

## Description of the algorithm (continued)

Steps 3, 5 and 6 are meant to ensure that the trust region doesn't shrink too fast without checking that the model is good. Since unnecessarily small step sizes at best make the method inefficient.

Step 3 (Safety step) only applies when the step is too small (less than half of the trust region radius lower bound  $\rho_k$ ), but we are not stationary.

We don't even calculate 'truth' since we assume the point is bad, as it may not only give a worse  $\phi$  (expensive to calculate) but it may also destroy the geometry.

Have to be **poised to reduce** T-R radius.

## Description of the algorithm (continued)

Steps 3, 5 and 6 are meant to ensure that the trust region doesn't shrink too fast without checking that the model is good. Since unnecessarily small step sizes at best make the method inefficient.

Step 3 (Safety step) only applies when the step is too small (less than half of the trust region radius lower bound  $\rho_k$ ), but we are not stationary.

We don't even calculate 'truth' since we assume the point is bad, as it may not only give a worse  $\phi$  (expensive to calculate) but it may also destroy the geometry.

Have to be **poised to reduce** T-R radius.

## Description of the algorithm (continued)

Step 3 (**Safety step**) Used only when  $\|\mathbf{d}_k\| < \frac{1}{2}\rho_k$  and  $\|\mathbf{g}_{\phi_k}\| > \epsilon_\beta$

Let  $\mathbf{x}_{k+1} = \mathbf{x}_k$  and set  $r_k = -1$

3.1 Let  $i = 0$ ,  $\Delta_k^{(0)} = \Delta_k$ .

3.2 Choose  $\bar{\Delta}_k^{(i)} \in [\rho_k, \Delta_k^{(i)}]$ .

3.3 If  $\bar{\Delta}_k^{(i)} > \rho_k$ , then

If  $\mathbf{Y}_k$  is  $\Lambda$ -poised in  $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k^{(i)})$ , then

$\Delta_k^{(i+1)} = \max\{\Delta_k^{(i)}/10, \rho_k\}$ .  $i = i + 1$ , go to 3.2.

Else  $\Delta_{k+1} = \bar{\Delta}_k^{(i)}$ ,  $\rho_{k+1} = \rho_k$ .

Else (i.e.  $\bar{\Delta}_k^{(i)} = \rho_k$ )

If  $\mathbf{Y}_k$  is  $\Lambda$ -poised  $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k^{(i)})$ , then

$\Delta_{k+1} = \rho_k/2$ ,  $\rho_{k+1} = \rho_k/10$ .

Else  $\Delta_{k+1} = \bar{\Delta}_k^{(i)}$ ,  $\rho_{k+1} = \rho_k$ .

Endif

Choose  $\bar{\Delta}_{k+1} \in [\rho_k, \Delta_{k+1}]$ .  $\mathbf{Y}_k \rightarrow \mathbf{Y}_{k+1}$  with  $\mathbf{Y}_{k+1}$

$\Lambda$ -poised in  $\mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$ .  $k = k + 1$ , go to Step 1.

## Description of the algorithm (continued)

Step 4 (**Acceptance of trial step**) when  $\|\mathbf{d}_k\| \geq 1/2\rho_k$  or  $\|\mathbf{g}_{\phi_k}\| \leq \epsilon\beta$   
Compute  $\Phi(\mathbf{x}_k + \mathbf{d}_k)$  and

$$r_k = \frac{\Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_k + \mathbf{d}_k)}{\phi_k(\mathbf{0}) - \phi_k(\mathbf{d}_k)}.$$

If  $r_k > 0$ ,  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$ ; otherwise,  $\mathbf{x}_{k+1} = \mathbf{x}_k$ .

Note that Steps 3 and 4 imply  $\rho_k$  is a lower bound on the sampling region unless we are close to stationarity

Step 5 (**Trust region radius update**)

$$\tilde{\Delta}_{k+1} = \begin{cases} \frac{1}{2} \|\mathbf{d}_k\| & \text{if } r_k < 0.1, \\ \max\{\frac{1}{2}\Delta_k, \|\mathbf{d}_k\|\} & \text{if } 0.1 \leq r_k < 0.7, \\ \max\{\Delta_k, 2\|\mathbf{d}_k\|\} & \text{if } r_k \geq 0.7, \end{cases}$$

$$\Delta_{k+1} = \min\{\max\{\tilde{\Delta}_{k+1}, \rho_k\}, \Delta_{max}\}.$$

If  $r_k \geq 0.1$ ,  $\mathbf{Y}_{k+1} = \mathbf{x}_{k+1} \cup \{\mathbf{Y}_k \setminus \mathbf{y}_t\}$ ; set  
 $\bar{\Delta}_{k+1} = \Delta_{k+1}, \rho_{k+1} = \rho_k, k = k + 1$ , go to Step 1.

## Description of the algorithm (continued)

Step 4 (**Acceptance of trial step**) when  $\|\mathbf{d}_k\| \geq 1/2\rho_k$  or  $\|\mathbf{g}_{\phi_k}\| \leq \epsilon_\beta$   
Compute  $\Phi(\mathbf{x}_k + \mathbf{d}_k)$  and

$$r_k = \frac{\Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_k + \mathbf{d}_k)}{\phi_k(\mathbf{0}) - \phi_k(\mathbf{d}_k)}.$$

If  $r_k > 0$ ,  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$ ; otherwise,  $\mathbf{x}_{k+1} = \mathbf{x}_k$ .

Note that Steps 3 and 4 imply  $\rho_k$  is a lower bound on the sampling region unless we are close to stationarity

Step 5 (**Trust region radius update**)

$$\tilde{\Delta}_{k+1} = \begin{cases} \frac{1}{2} \|\mathbf{d}_k\| & \text{if } r_k < 0.1, \\ \max\{\frac{1}{2}\Delta_k, \|\mathbf{d}_k\|\} & \text{if } 0.1 \leq r_k < 0.7, \\ \max\{\Delta_k, 2\|\mathbf{d}_k\|\} & \text{if } r_k \geq 0.7, \end{cases}$$

$$\Delta_{k+1} = \min\{\max\{\tilde{\Delta}_{k+1}, \rho_k\}, \Delta_{max}\}.$$

If  $r_k \geq 0.1$ ,  $\mathbf{Y}_{k+1} = \mathbf{x}_{k+1} \cup \{\mathbf{Y}_k \setminus \mathbf{y}_t\}$ ; set  
 $\bar{\Delta}_{k+1} = \Delta_{k+1}$ ,  $\rho_{k+1} = \rho_k$ ,  $k = k + 1$ , go to Step 1.

## Description of the algorithm (continued)

### Step 6 (Model improvement)

When  $r_k < 0.1$

If  $\mathbf{Y}_k$  is  $\Lambda$ -poised in  $\mathcal{B}(\mathbf{x}_k, \Delta_k)$  and  $\Delta_k = \rho_k$ , then

Let  $\Delta_{k+1} = \rho_k/2$ ,  $\rho_{k+1} = \rho_k/10$ .

Choose  $\bar{\Delta}_{k+1} \in [\rho_{k+1}, \Delta_{k+1}]$ ,  $\mathbf{Y}_k \rightarrow \mathbf{Y}_{k+1} \in \mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$

Else

Let  $\rho_{k+1} = \rho_k$ . Choose  $\bar{\Delta}_{k+1} \in [\rho_{k+1}, \Delta_{k+1}]$ ,

$\mathbf{Y}_k \rightarrow \mathbf{Y}_{k+1}$  s. t.  $\mathbf{Y}_{k+1}$  is  $\Lambda$ -poised in  $\mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$ .

Endif

Set  $k = k + 1$ , go to Step 1.

Step 6 says we have to improve the model and/or ensure poisedness if agreement is poor.

- (a) Set  $|\mathbf{Y}_k| = NP = 2n + 1$ ,  $\mathbf{x}_k \in \mathbf{Y}_k$  for all  $k$ .
- (b) Least Frobenius norm updating of models (Powell, 2004)

$$\begin{array}{ll} \min & \|\nabla^2 \ell_t^+\|_F \\ \text{s.t.} & \ell_t(\mathbf{x}^+) = 1, \ell_t(\mathbf{y}) = 0, \mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_t. \end{array}$$

For  $i = 1, \dots, m$ ,

$$m_i^+(\mathbf{x}) = m_i(\mathbf{x}) + \{f_i(\mathbf{x}^+) - m_i(\mathbf{x}^+)\} \ell_t^+(\mathbf{x}).$$

## Theorem:

Suppose

1.  $\nabla F(\mathbf{x})$  is bounded, Lipschitz Continuous on  $L_{enl}(\mathbf{x}_0)$  with

$$L_{enl}(\mathbf{x}_0) = \bigcup_{\mathbf{x} \in L(\mathbf{x}_0)} \mathcal{B}(\mathbf{x}, \Delta_{max}), \quad L(\mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq f(\mathbf{x}_0)\}.$$

2.  $\{\mathbf{x}_k\}$  generated by DFSL algorithm,

then

$$\liminf_{k \rightarrow \infty} \|\nabla \Phi(\mathbf{x}_k)\| = \liminf_{k \rightarrow \infty} \|\nabla F(\mathbf{x}_k)^\top F(\mathbf{x}_k)\| = 0.$$

## Assumptions:

1. **Zero residual problem**, i.e. there exists solution set  $\mathbf{X}^* \in \mathbb{R}^n$

$$\Phi(\mathbf{x}^*) = 0, \quad \text{for any } \mathbf{x}^* \in \mathbf{X}^*.$$

2. **Local error bound**, i.e. there exist positive constants  $\alpha$  and  $\rho$  such that

$$\|F(\mathbf{x})\| \geq \alpha \|\mathbf{x} - \bar{\mathbf{x}}\|, \quad \text{whenever } \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \rho,$$

where  $\bar{\mathbf{x}} \in \mathbf{X}^*$  and  $\|\mathbf{x} - \bar{\mathbf{x}}\| = \text{dist}(\mathbf{x}, \mathbf{X}^*)$ .

(This is a generalization of assuming a non-singular Jacobian at the solution.)

# Local Convergence Assumptions (continued)

3. Choose the base point  $\mathbf{y}_k \in \mathbf{Y}_k$  to be  $\mathbf{x}_k$ , i.e.

$$\mathbf{y}_k = \mathbf{x}_k, \text{ for all large } k.$$

4.  $\{\mathbf{x}_k\}$  generated by the DFLS algorithm converges to the solution set  $\mathbf{X}^*$ , i.e.

$$\lim_{k \rightarrow \infty} \|\mathbf{x}_k - \bar{\mathbf{x}}_k\| = 0.$$

# Local Convergence

**Lemma:** Under the above conditions and assumptions, for any  $\gamma_2 > 0$ , if  $k$  is sufficiently large, then

$$c_\phi(\mathbf{x}_k) = \frac{1}{2} \mathbf{m}(\mathbf{x}_k)^\top \mathbf{m}(\mathbf{x}_k) \leq \gamma_2 \|\mathbf{g}_\phi(\mathbf{x}_k)\|,$$

and

$$\begin{aligned} \phi_k(\mathbf{d}) = \phi(\mathbf{x}_k, \mathbf{d}) &= \frac{1}{2} \|\mathbf{m}(\mathbf{x}_k) + J(\mathbf{x}_k) \mathbf{d}\|^2 + \lambda_k \|\mathbf{d}\|^2 \\ &= \frac{1}{2} \|\mathbf{m}(\mathbf{x}_k) + \nabla \mathbf{m}(\mathbf{x}_k) \mathbf{d}\|^2 + \lambda_k \|\mathbf{d}\|^2, \end{aligned}$$

where  $\lambda_k = \gamma_3/2 \|\mathbf{m}(\mathbf{x}_k)\|$ .

$$\text{( Recall: } \phi(\mathbf{y}, \mathbf{s}) = c_\phi(\mathbf{y}) + \mathbf{g}_\phi(\mathbf{y})^\top \mathbf{s} + \frac{1}{2} \mathbf{s}^\top H_\phi(\mathbf{y}) \mathbf{s}. \text{ )}$$

**Theorem:** Under the above conditions and assumptions, there exists a  $\mathbf{x}^* \in \mathbf{X}^*$ , such that

$$\lim_{k \rightarrow \infty} \|\mathbf{x}_k - \mathbf{x}^*\| = 0,$$

and

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq c \|\mathbf{x}_k - \mathbf{x}^*\|^2,$$

for constant  $c > 0$ .

## Comparison Methods

### LMDIF:

A derivative free version (1980) of the Levenberg-Marquardt algorithm for minimizing the sum of squares developed by Garbow, Hillstom and Moré. The Jacobian is calculated by a forward-difference approximation.

### NEWUOA:

Software (2006) for unconstrained optimization without derivatives developed by Powell.

All written and compiled in Fortran and run on the same ThinkPad T40.

# Numerical results (continued)

- Long-term Behavior Comparisons

- ♣ Smooth Problems

- Maximum number of function evaluations  $10^5$

- (a) 25 Zero Residue least square problems

$$\Phi(\mathbf{x}_k) \leq \max\{10^{-12}, 10^{-20}\Phi(\mathbf{x}_0)\}$$

- (b) 22 Nonzero Residue least square problems

$$\text{F-ReErr} \leq 10^{-12} \quad \text{with} \quad \text{F-ReErr} = \frac{\Phi_{end} - \Phi^*}{\Phi^*}$$

## ♣ Stochastic Noisy Problems

Maximum number of function evaluations  $10^5$ ; 10 runs for each problem

- (c) 25 Zero Residue problems with noise level  $10^{-2}$

$$F(\mathbf{x}) = F_{true}(\mathbf{x}) + 10^{-2}E.$$

- (b) 22 Nonzero Residue problems with relative noise level  $10^{-2}$

$$F(\mathbf{x}) = F_{true}(\mathbf{x}) + 10^{-2}E \circ |F(\mathbf{x})|.$$

where  $E \in \mathbb{R}^m$  is a random vector with normal distribution in  $[-0.5, 0.5]$ .

# Numerical results (continued)

Problem Name	$n$	$m$	LMDF NF/F-Error	NEWUOA NF/F-Error	DFLS NF/F-Error
HS26	3	2	890/6.22e-9	3706	378
HS27	3	2	281/2.40e-11	405	338
HS28	3	2	107	247	48
HS39	4	3	81	585	53
HS40	4	4	41	344	30
HS42	4	3	139/4.86e-12	655	48
HS46	5	3	1762/3.75e-8	7060	1076
HS47	5	4	2186/4.40e-11	2151	263
HS49	5	3	$10^5$ /4.18e-7	2546	2395
HS56	7	5	266/1.55e-9	1851	168
HS60	3	2	123/2.29e-10	415	65
HS77	5	3	638/8.35e-5	1439	145
HS78	5	4	79	727	48
HS81	5	4	232/6.06e-7	2262	107
HS111	10	4	4434/8.77e-6	$10^5$ /2.32e-8	2193/7.85e-12
CONN	20	20	667	639	113
CHROSEN	20	38	169	635	96
CHROSEN	80	158	649	3094	346
TRIGSSQS	10	10	67	490	81
TRIGSSQS	10	20	45	236	49
TRIGSSQS	20	20	170	2267	145
BUCKLEY21	6	13	113	2076	171
HEART	6	6	*	$10^5$ /1.21e-2	863
MORE25	20	22	400	10359	198
RORE20	30	31	295	$10^5$ /1.23e-8	431/8.00e-12

Table: Numerical Comparisons (zero residuals)

# Numerical results (continued)

Problem Name	$n$	$m$	LMDF NF/AsReErr	NEWUOA NF/AsReErr	DFLS NF/AsReErr
HS26	3	2	90/1.97e-7	189	155
HS27	3	2	208/1.06e-3	248	108
HS28	3	2	1950	107	143
HS39	4	3	104/5.30e-8	245	100
HS40	4	4	54/8.91e-7	125	76
HS42	4	3	165/3.91e-6	84	59
HS46	5	3	1788/1.54e-5	600/1.99e-12	586
HS47	5	4	*	92	92
HS49	5	3	10 <sup>b</sup> /6.86e-5	832/3.51e-12	516
HS56	7	5	169/3.21e-5	234	160
HS60	3	2	139/3.93e-5	217	81
HS77	5	3	959/2.44e-3	655	162
HS78	5	4	226/7.76e-7	158	105
HS81	5	4	194/1.69e-2	949	214
HS111	10	4	90451/1.16e-6	10562	673
DENNIS	4	20	2034/2.38e-11	254	130
SPHRPTS	20	45	*	2263	1406
TRIGSSQS	10	10	230/7.10e-7	241	227
TRIGSSQS	10	20	215/4.70e-6	217	241
TRIGSSQS	20	20	378/6.33e-5	580	576
PENALTY1	20	21	219/5.87e-5	9855	433
PENALTY2	20	40	442/1.89e-10	1982	336

Table: Numerical Comparisons (nonzero residuals)

# Numerical results (continued)

Problem Name	$n$	$m$	LMDIF NF/F-Error	NEUWOA NF/F-Error	DFLS NF/F-Error
HS26	3	2	52.2/4.22e-2	91.6/4.09e-3	87.4/1.80e-5
HS27	3	2	F	106.1/1.85e-2	94.3/2.48e-5
HS28	3	2	63.3/1.22e-1	93.9/9.32e-5	85.1/1.29e-5
HS39	4	3	83.6/2.29e-1	122.6/0.22	99.4/1.28e-6
HS40	4	4	43.1/3.11e-2	102.9/9.88e-5	95.4/2.32e-5
HS42	4	3	45.2/2.78e-1	115.4/3.65e-3	93.8/3.55e-6
HS46	5	3	80.8/1.77e-1	161.9/1.36e-2	133.8/5.65e-5
HS47	5	4	89.8/7.77e-1	171.9/6.15e-3	136.3/6.90e-5
HS49	5	3	F	F	192/6.05e-4
HS56	7	5	F	174.6/6.41e-4	168/2.64e-4
HS60	3	2	63.34.12e-3	81.1/1.04e-2	85.7/1.91e-5
HS77	5	3	95.1/5.53e-2	158/7.95e-2	149.4/1.95e-3
HS78	5	4	82.1/3.83e-3	117.4/1.46e-2	116.7/2.14e-5
HS81	5	4	83.4/3.61e-5	148.9/7.25e-2	115.4/1.34e-4
HS111	10	4	136.6/1.92e-2	248.7/8.01e-2	235.3/1.32e-4
CONN	20	20	F	625.9/2.95e-3	415.5/7.96e-5
CHROSEN	20	38	F	479.8/4.36e-3	385.4/1.14e-4
CHROSEN	80	158	F	2292.7/4.64e-1	1866.4/4.03e-4
TRIGSSQS	10	10	167.5/4.06e-4	347.3/3.22e-3	205/1.02e-4
TRIGSSQS	10	20	155.5/1.20e-4	254.1/3.14e-4	158/3.65e-5
TRIGSSQS	20	20	449.1/5.00e-3	816.5/2.1e-1	375.7/1.61e-4
BUCKLEY21	6	13	60.1/7.9e-1	159.8/1.5e-1	168.7/3.94e-5
HEART	6	6	F	F	196.6/4.2e-1
MORE25	20	22	F	F	450.2/2.28e-4
RORE20	30	31	F	961/2.3e-1	588.1/1.18e-4

Table: Numerical Comparisons (zero residuals, noise level 1.e-2)

# Numerical results (continued)

Problem Name	$n$	$m$	LMDIF NF/F-Error	NEUJQA NF/F-Error	DFLS NF/F-Error
HS26	3	2	40.9/6.63e-1	84/1.81e-2	94.3/1.79e-3
HS27	3	2	F	98.2/6.29e-1	113.3/4.14e-4
HS28	3	2	38.8/5.21e-1	83/1.41e-2	92.1/2.20e-2
HS39	4	3	F	124.5/2.09e-2	116.7/3.22e-3
HS40	4	4	59.3/9.38e-3	107.5/1.11e-2	105.4/8.60e-4
HS42	4	3	F	94.4/1.48e-3	105.1/2.31e-3
HS46	5	3	49.2/1.92e-1	131.4/1.13e-2	132.5/3.30e-3
HS47	5	4	*	92/1.33e-1	92/5.20e-2
HS49	5	3	F	F	156.3/5.25e-1
HS56	7	5	F	185.2/1.20e-2	174.2/8.71e-3
HS60	3	2	71.9/3.74e-3	124.2/6.02e-2	102/1.57e-4
HS77	5	3	84.2/6.47e-1	F	157.9/4.81e-2
HS78	5	4	63.9/2.26e-2	141.6/1.51e-3	121.5/1.69e-3
HS81	5	4	84.9/1.05e-1	F	154/2.94e-3
HS111	10	4	F	238.6/2.76e-1	248.3/2.09e-2
DENNIS	4	20	F	164.8/8.59e-3	133.9/3.61e-4
SPHRPTS	20	45	F	562.5/8.18e-3	541.4/6.59e-3
TRIGSSQS	10	10	74.7/2.03e-1	241.1/5.38e-3	248.9/4.29e-3
TRIGSSQS	10	20	59.3/2.93e-1	249.4/3.05e-3	260.2/1.89e-3
TRIGSSQS	20	20	114.8/3.43e-1	468.5/7.28e-3	473.5/8.78e-3
PENALTY1	20	21	135.7/2.60e-1	590.7/1.26e-1	475.8/5.67e-3
PENALTY2	20	40	F	447.4/3.26e-3	417.3/3.50e-3

Table: Numerical Comparisons (nonzero residuals, relative noise  $1.e-2$ )

# Numerical results (continued)

- Short-term Behavior Comparisons

Maximum number of function evaluations  $50(n_p + 1)$   
(Equivalent to 50 simplex gradients)

(a) Test problem set  $\mathcal{P}$  consists of 53 problems  
([http:// www.mcs.anl.gov/~more/dfo](http://www.mcs.anl.gov/~more/dfo) )

(b) Stopping condition

$$\Phi(\mathbf{x}_0) - \Phi(\mathbf{x}_k) \geq (1 - \tau)(\Phi(\mathbf{x}_0) - \Phi_L), \quad 0 \leq \tau < 1$$

(c) Performance

$$d_s(\alpha) = \frac{|\{p \in \mathcal{P} : t_{p,s} \leq \alpha(n_p + 1)\}|}{|\mathcal{P}|}, \quad \alpha \in [1, 50]$$

- (d) Test problem set  $\mathcal{P}_{\mathcal{N}}$  consists of 53 problems with added non-random noise  
([http:// www.mcs.anl.gov/~more/dfo](http://www.mcs.anl.gov/~more/dfo) )

$$\begin{aligned}\Phi_{\mathcal{N}}(\mathbf{x}) &= (1 + \epsilon_{\mathcal{N}}\pi(\mathbf{x}))\Phi(\mathbf{x}) \\ &= \frac{1}{2}(1 + \epsilon_{\mathcal{N}}\pi(\mathbf{x})) \sum_{i=1}^m f_i^2(\mathbf{x}),\end{aligned}$$

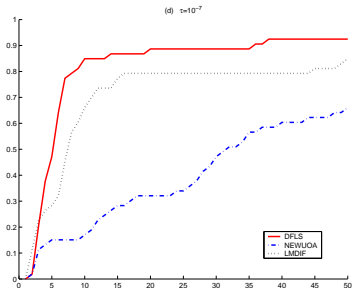
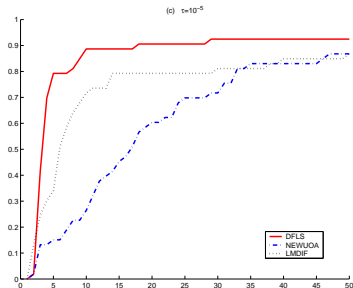
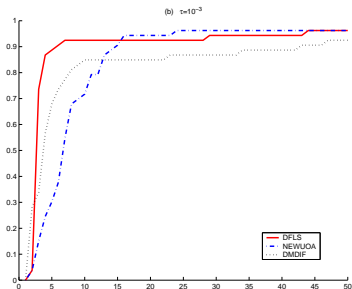
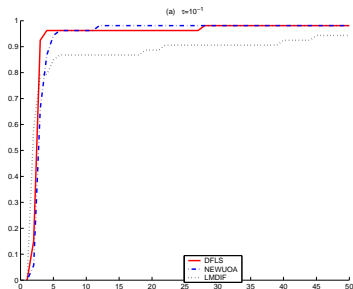
where

$$\pi(\mathbf{x}) = T_3(\pi_0(\mathbf{x})), \quad T_3(\alpha) = \alpha(4\alpha^2 - 3), \quad \epsilon_{\mathcal{N}} = 1.e - 3,$$

and

$$\pi_0(\mathbf{x}) = 0.9 \sin(100\|\mathbf{x}\|_1) \cos(100\|\mathbf{x}\|_\infty) + 0.1 \cos(\|\mathbf{x}\|_2).$$

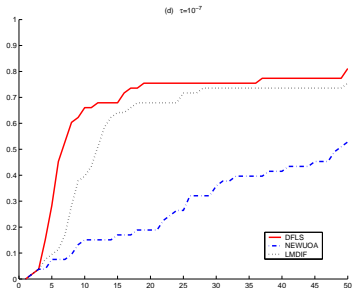
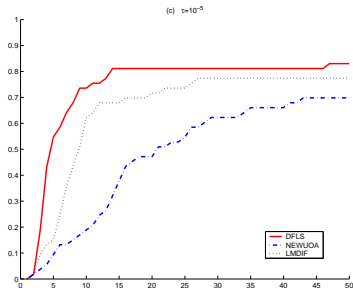
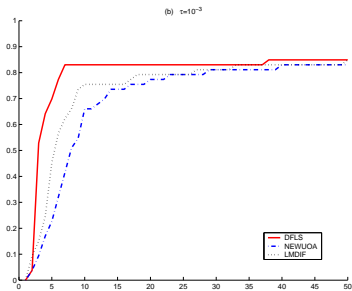
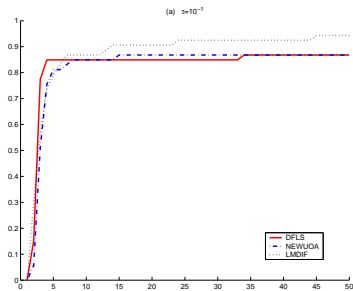
# Numerical results (continued)



Data profiles of function  $d_5(\alpha)$  for smooth problems:

(a)  $\tau = 10^{-1}$ , (b)  $\tau = 10^{-3}$ , (c)  $\tau = 10^{-5}$ , (d)  $\tau = 10^{-7}$

# Numerical results (continued)



Data profiles  $d_5(\alpha)$  for non-stochastic noisy problems:

(a)  $\tau = 10^{-1}$ , (b)  $\tau = 10^{-3}$ , (c)  $\tau = 10^{-5}$ , (d)  $\tau = 10^{-7}$