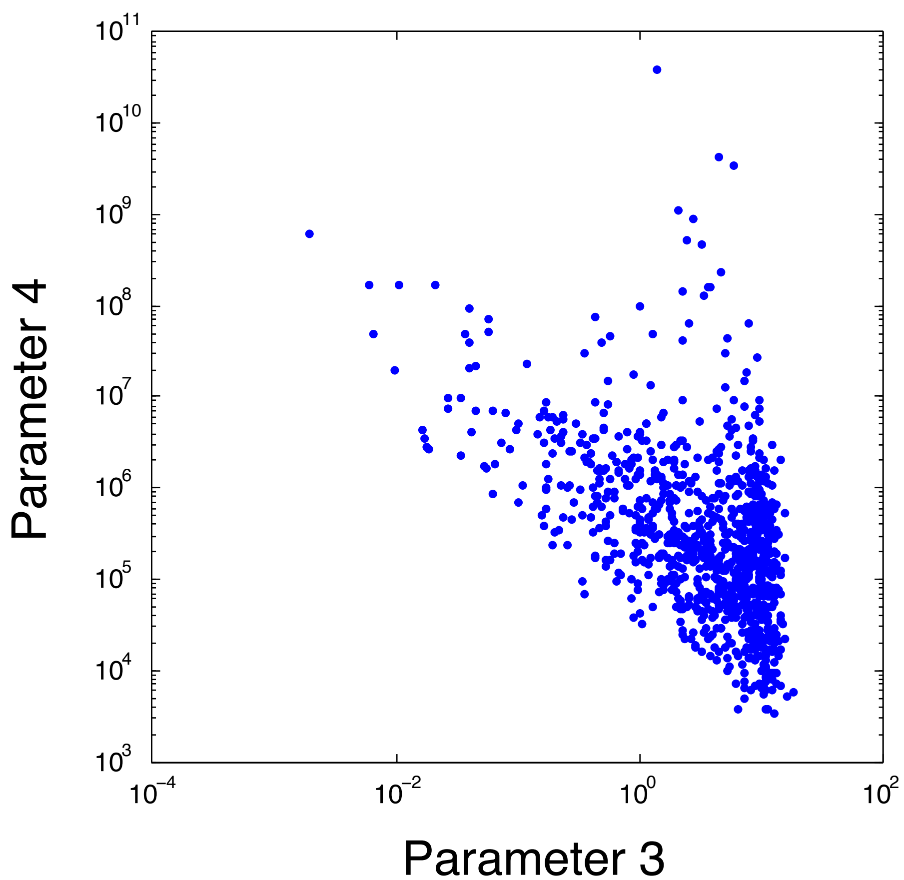
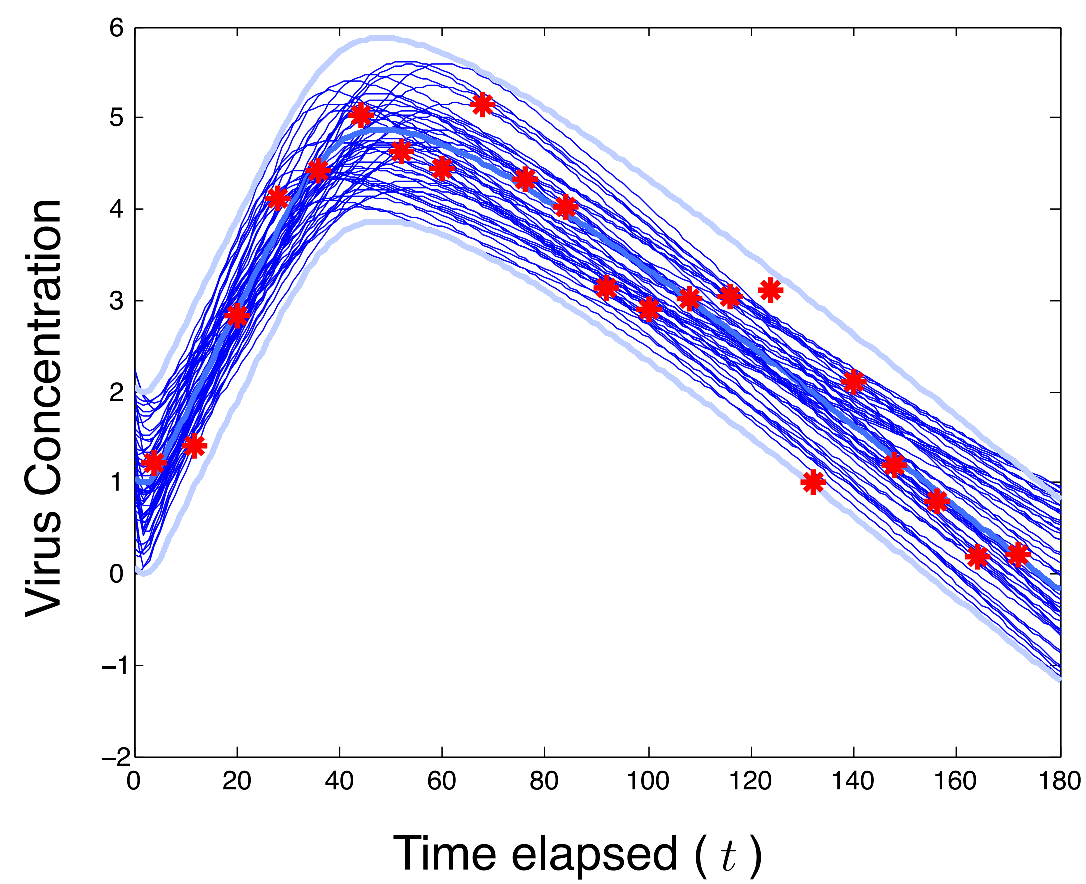
A Practical Algorithm
for
"Practical" Parameter Identifiability Analysis





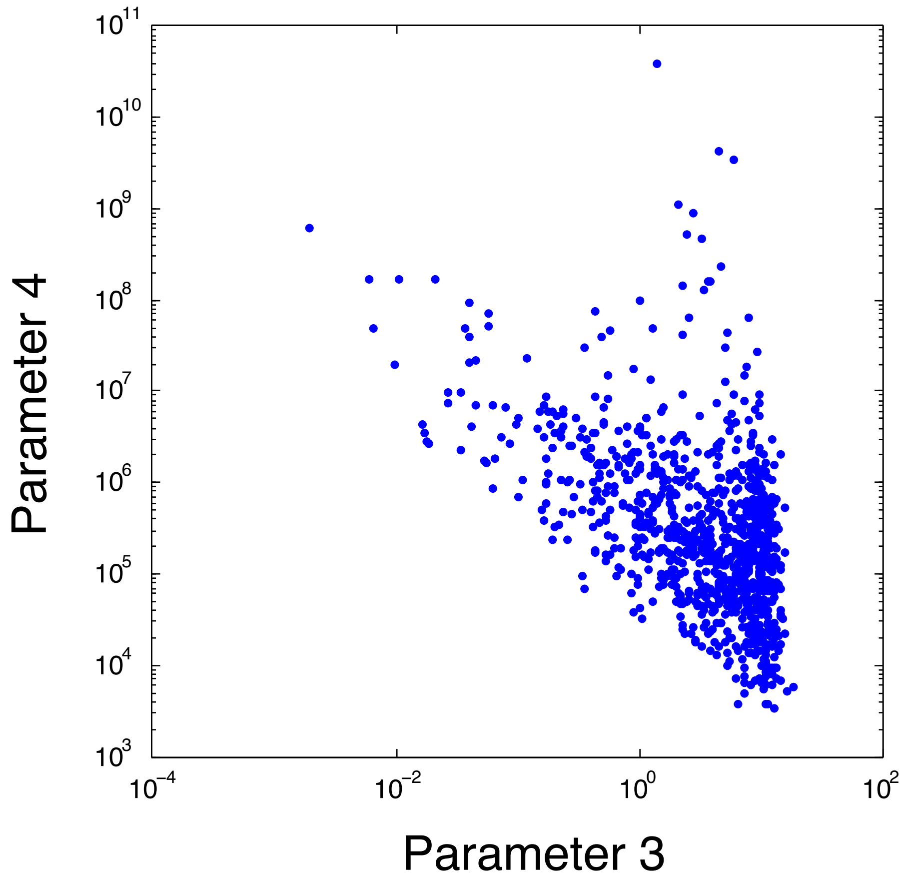
Yasunori Aoki
University of Waterloo

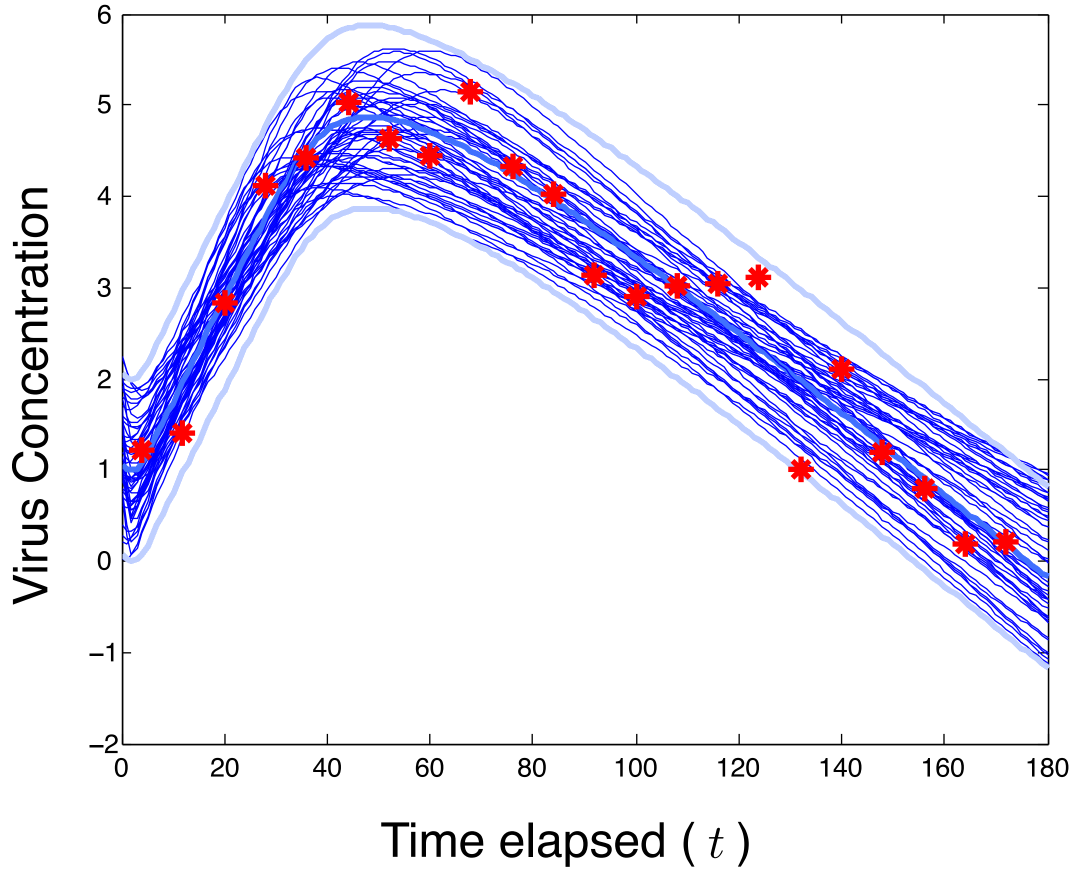
Ben Holder
Ryerson University

Hans De Sterck
University of Waterloo

Ken Hayami
National Institute of Informatics

A Practical Algorithm
for
"Practical" Parameter Identifiability Analysis





Yasunori Aoki
University of Waterloo

Ben Holder
Ryerson University

Hans De Sterck
University of Waterloo

Ken Hayami
National Institute of Informatics

Parameter Identifiability Analysis

Parameter Identifiability Analysis

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Structural Parameter Identifiability Analysis

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Structural Parameter Identifiability Analysis

Assuming there is no variability of data, can we uniquely estimate model parameters?

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Structural Parameter Identifiability Analysis

Assuming there is no variability of data, can we uniquely estimate model parameters?

Practical Parameter Identifiability Analysis

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Structural Parameter Identifiability Analysis

Assuming there is no variability of data, can we uniquely estimate model parameters?

Practical Parameter Identifiability Analysis

Given a certain level of variability of data, how precisely can we estimate model parameters?

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix		

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix	not-Robust ✗	

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix	not-Robust ✗	Fast ✓

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix	not-Robust ✗	Fast ✓
Monte Carlo / Bootstrap method		

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix	not-Robust ✗	Fast ✓
Monte Carlo / Bootstrap method	Robust ✓	

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix	not-Robust ✗	Fast ✓
Monte Carlo / Bootstrap method	Robust ✓	Impractically Slow... ✗

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

"Practical" Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Fisher Information Matrix	not-Robust ✗	Fast ✓
Monte Carlo / Bootstrap method	Robust ✓	Impractically Slow… ✗

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Example 0 : Influenza Kinetics model (Baccam et al.)

Example 0 : Influenza Kinetics model (Baccam et al.)

Example 0 : Influenza Kinetics model (Baccam et al.)

$$\frac{du_1}{dt} = -x_1 u_1 u_4$$
$$\frac{du_2}{dt} = x_1 u_1 u_4 - \frac{1}{x_2} u_2$$
$$\frac{du_3}{dt} = \frac{1}{x_2} u_2 - \frac{1}{x_3} u_3$$
$$\frac{du_4}{dt} = \frac{x_4}{x_5} u_3 - x_6 u_4$$

$$u_1(t=0) = x_5$$
$$u_2(t=0) = 0$$
$$u_3(t=0) = 0$$
$$u_4(t=0) = x_7$$

Example 0 : Influenza Kinetics model (Baccam et al.)

$$\frac{du_1}{dt} = -x_1 u_1 u_4$$

$$\frac{du_2}{dt} = x_1 u_1 u_4 - \frac{1}{x_2} u_2$$

$$\frac{du_3}{dt} = \frac{1}{x_2} u_2 - \frac{1}{x_3} u_3$$

$$\frac{du_4}{dt} = \frac{x_4}{x_5} u_3 - x_6 u_4$$

$$u_1(t=0) = x_5$$

$$u_2(t=0) = 0$$

$$u_3(t=0) = 0$$

$$u_4(t=0) = x_7$$

Example 0 : Influenza Kinetics model (Baccam et al.)

$$\frac{du_1}{dt} = -x_1 u_1 u_4$$

$$\frac{du_2}{dt} = x_1 u_1 u_4 - \frac{1}{x_2} u_2$$

$$\frac{du_3}{dt} = \frac{1}{x_2} u_2 - \frac{1}{x_3} u_3$$

$$\frac{du_4}{dt} = \frac{x_4}{x_5} u_3 - x_6 u_4$$

$$u_1(t=0) = x_5$$

$$u_2(t=0) = 0$$

$$u_3(t=0) = 0$$

$$u_4(t=0) = x_7$$

Example 0 : Influenza Kinetics model (Baccam et al.)

$$\frac{du_1}{dt} = -x_1 u_1 u_4$$

$$\frac{du_2}{dt} = x_1 u_1 u_4 - \frac{1}{x_2} u_2$$

$$\frac{du_3}{dt} = \frac{1}{x_2} u_2 - \frac{1}{x_3} u_3$$

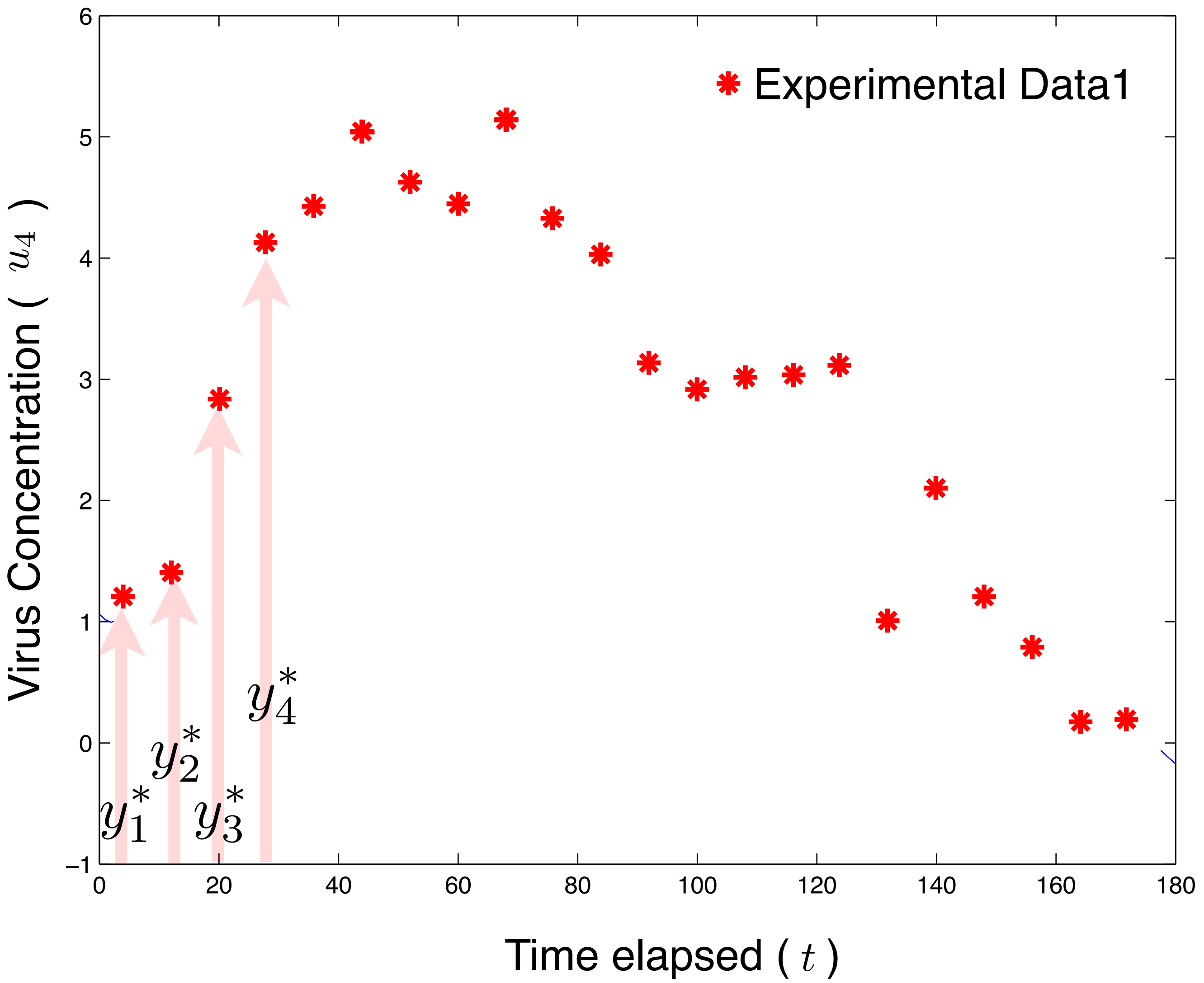
$$\frac{du_4}{dt} = \frac{x_4}{x_5} u_3 - x_6 u_4$$

$$u_1(t=0) = x_5$$

$$u_2(t=0) = 0$$

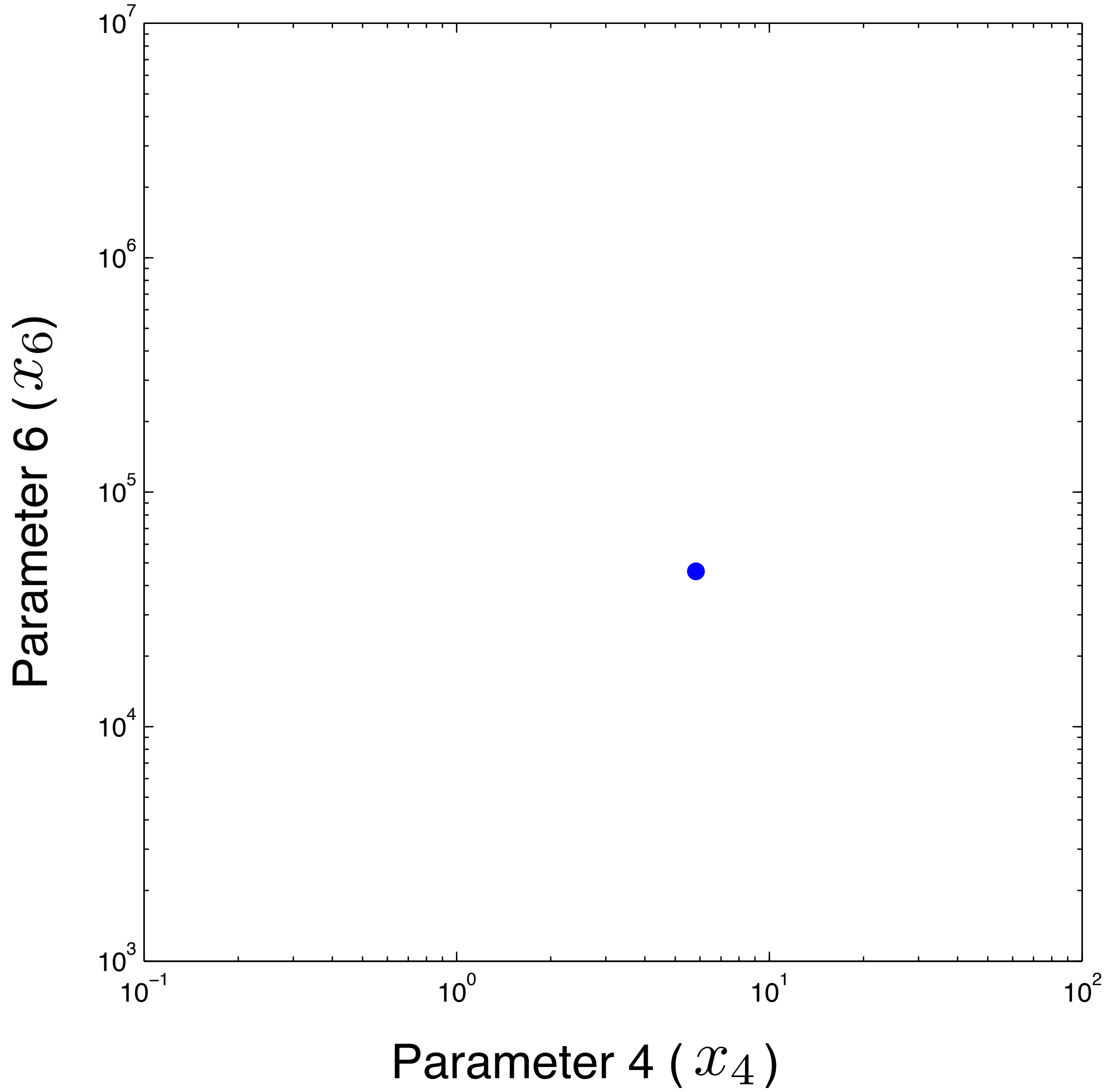
$$u_3(t=0) = 0$$

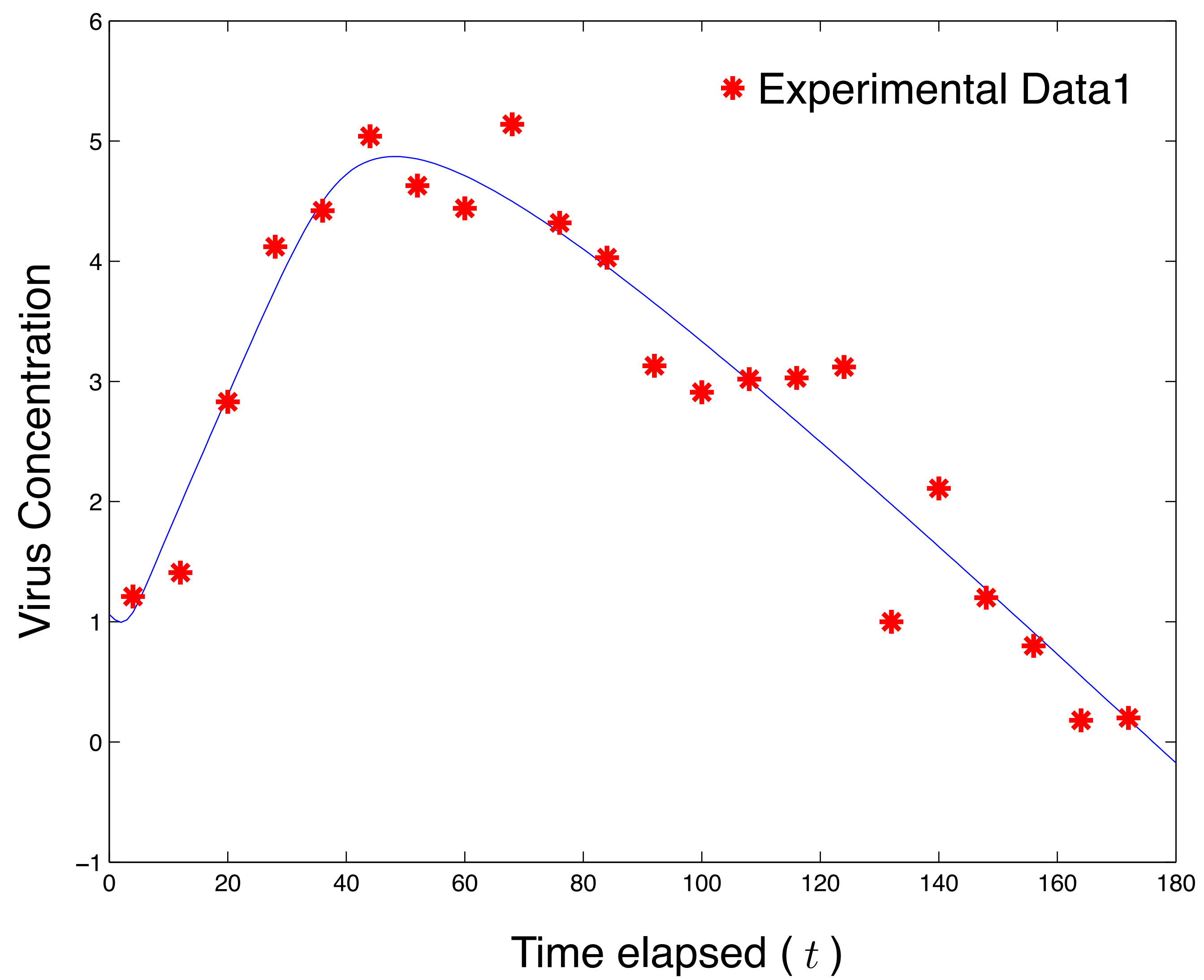
$$u_4(t=0) = x_7$$



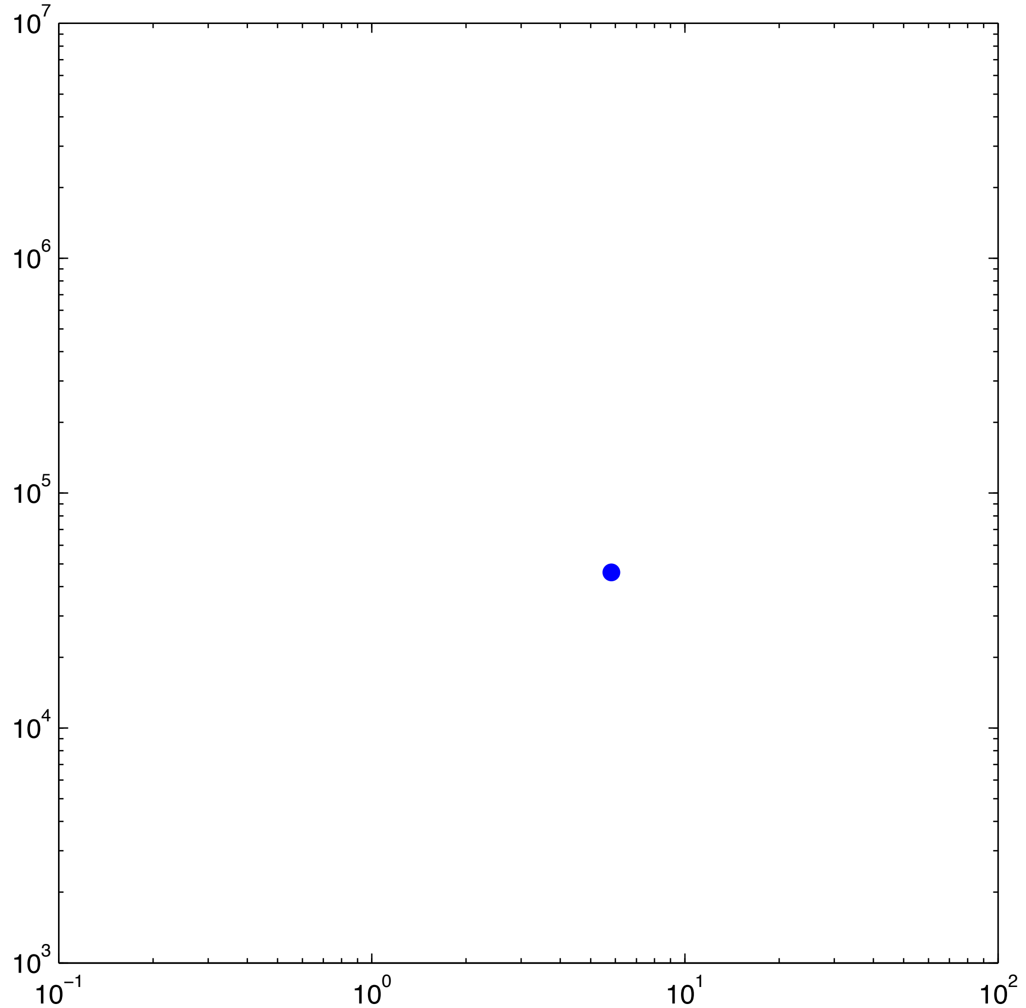
Monte Carlo / Bootstrap method

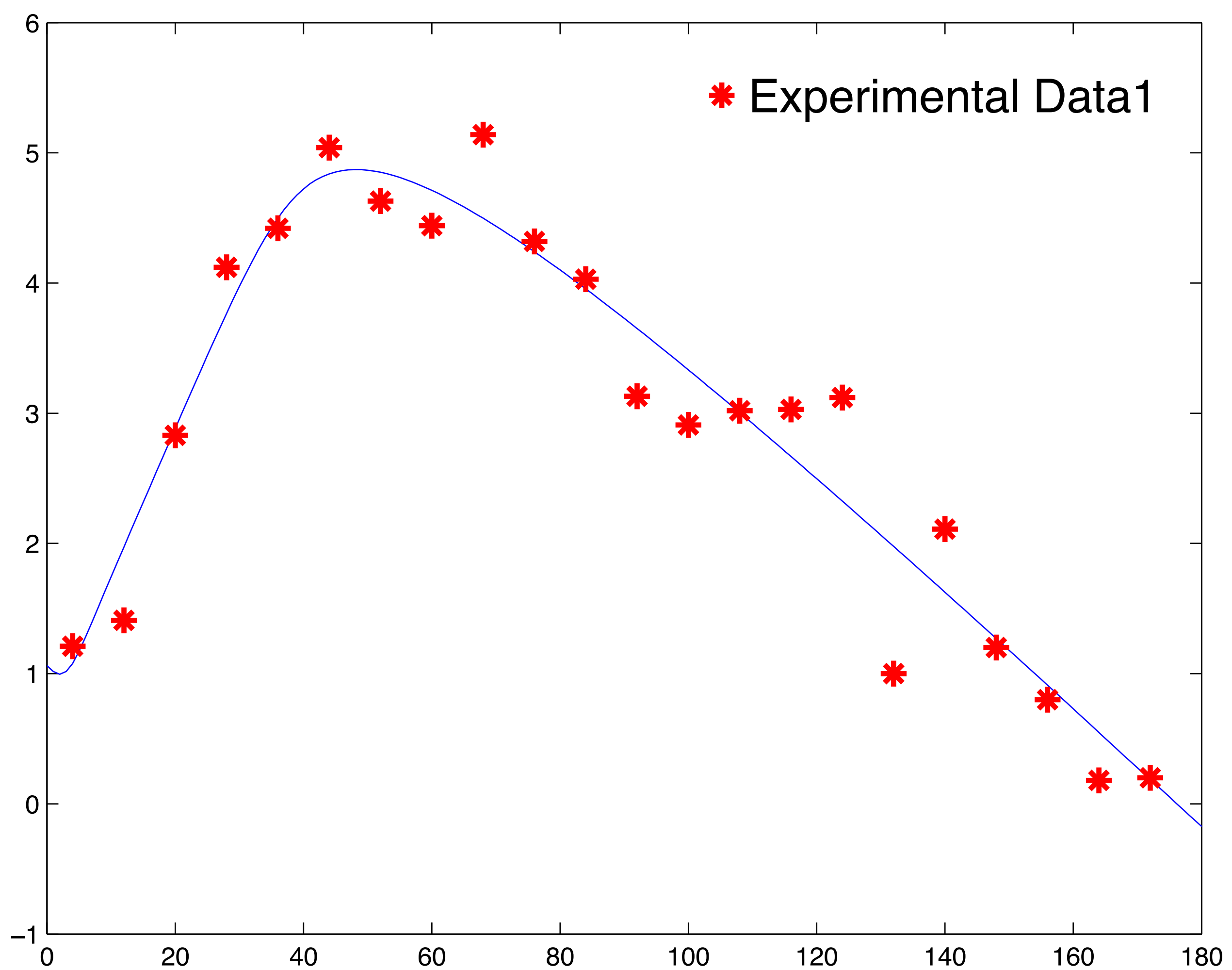
Monte Carlo / Bootstrap method





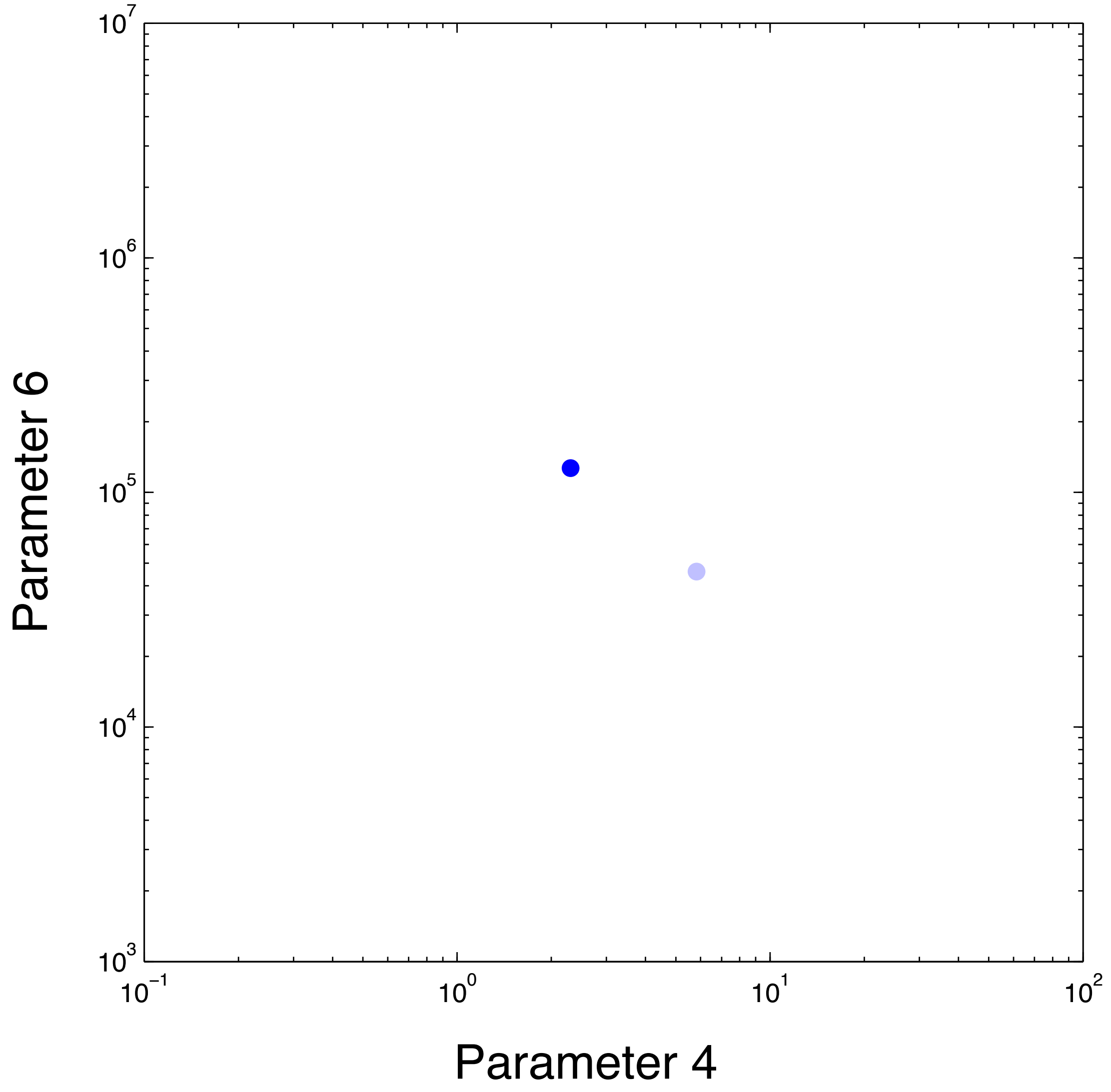
Monte Carlo / Bootstrap method

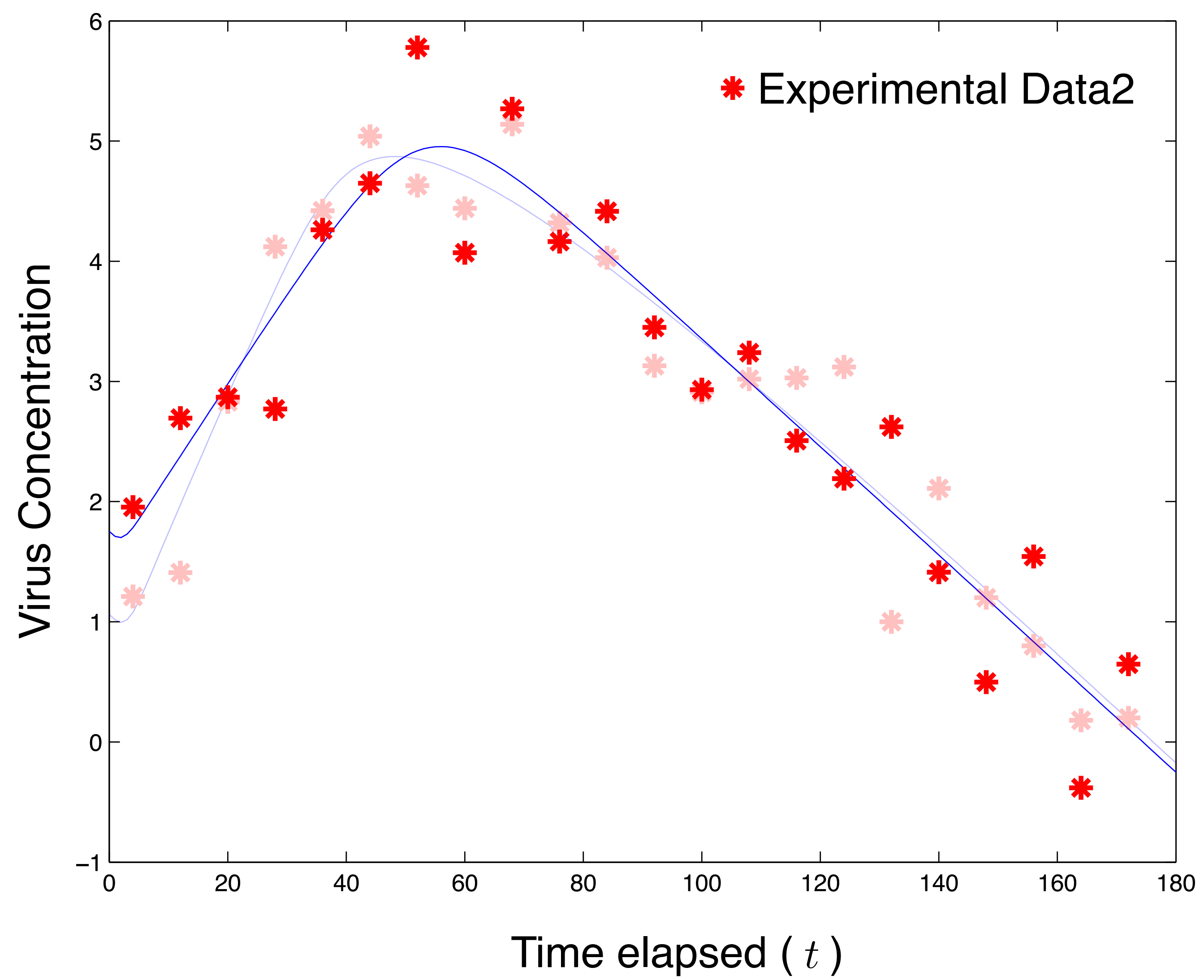




Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp1}}||_2$

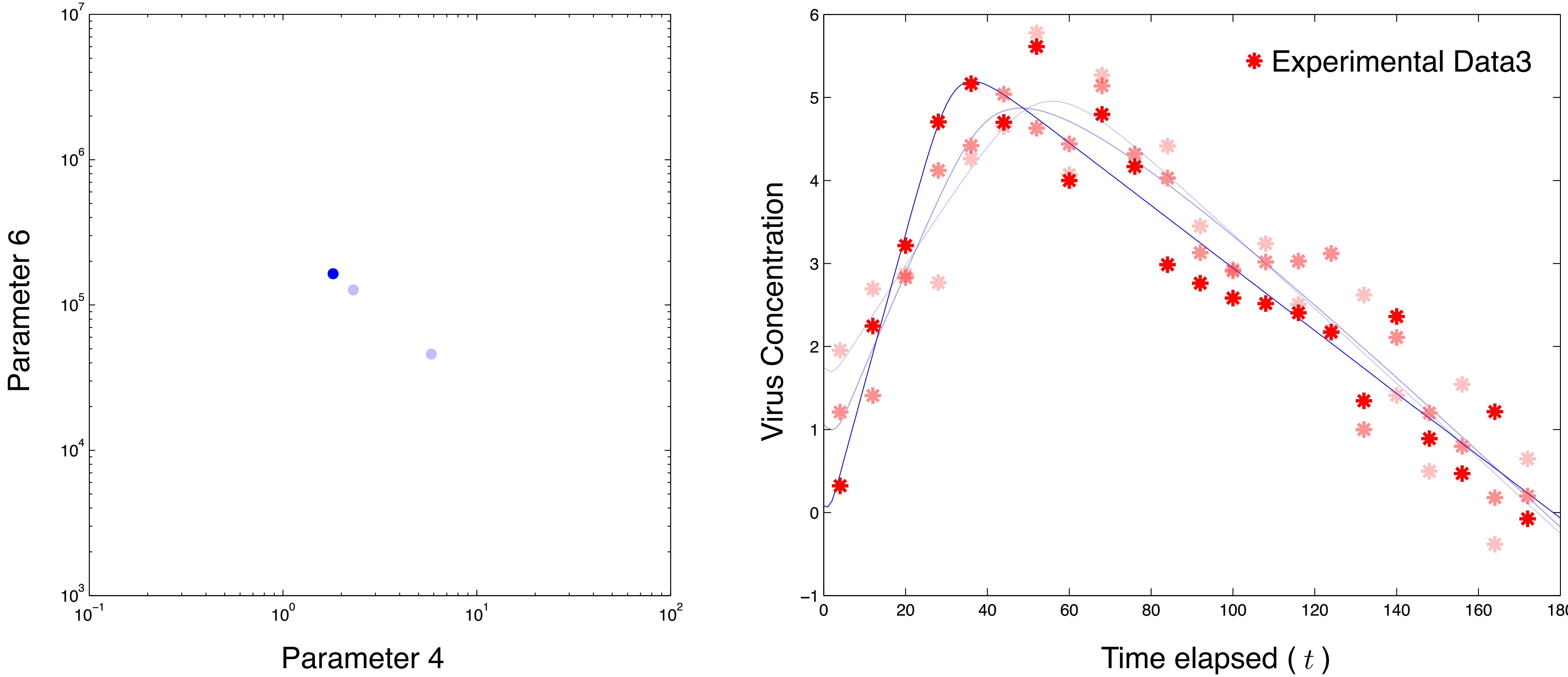
Monte Carlo / Bootstrap method





Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp2}}||_2$

Monte Carlo / Bootstrap method



Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp3}}||_2$

Monte Carlo / Bootstrap method

We do not usually have enough sets of experimental data to conduct statistical analyses on the sets of parameters found from them...

Monte Carlo / Bootstrap method

We do not usually have enough sets of experimental data to conduct statistical analyses on the sets of parameters found from them...

↓

Artificially create data.

Monte Carlo / Bootstrap method

We do not usually have enough sets of experimental data to conduct statistical analyses on the sets of parameters found from them...

↓

Artificially create data.

Explicitly specify variability level and distribution

↙

Monte Carlo method

Monte Carlo / Bootstrap method

We do not usually have enough sets of experimental data to conduct statistical analyses on the sets of parameters found from them...

Artificially create data.

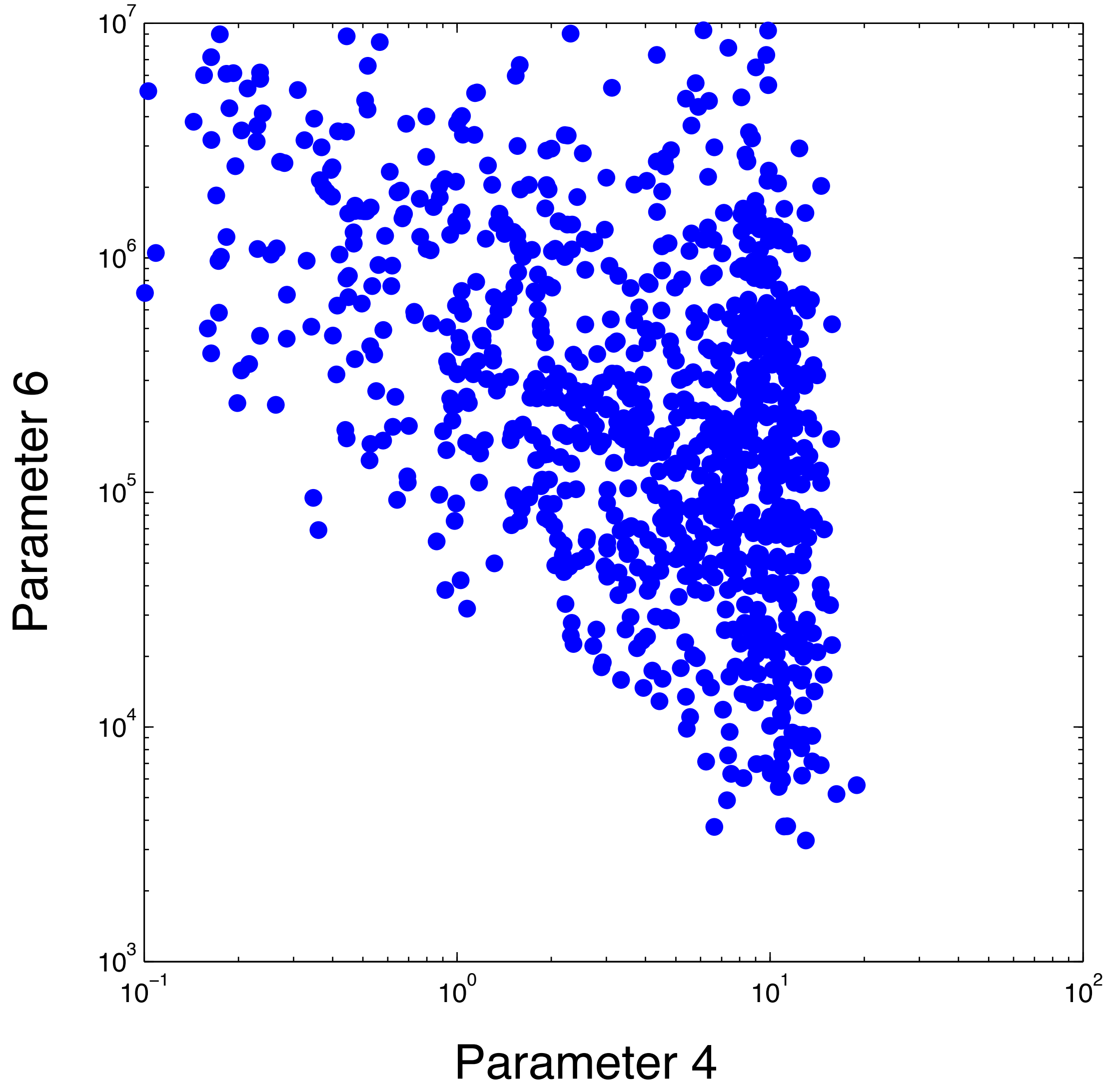
Explicitly specify variability level and distribution

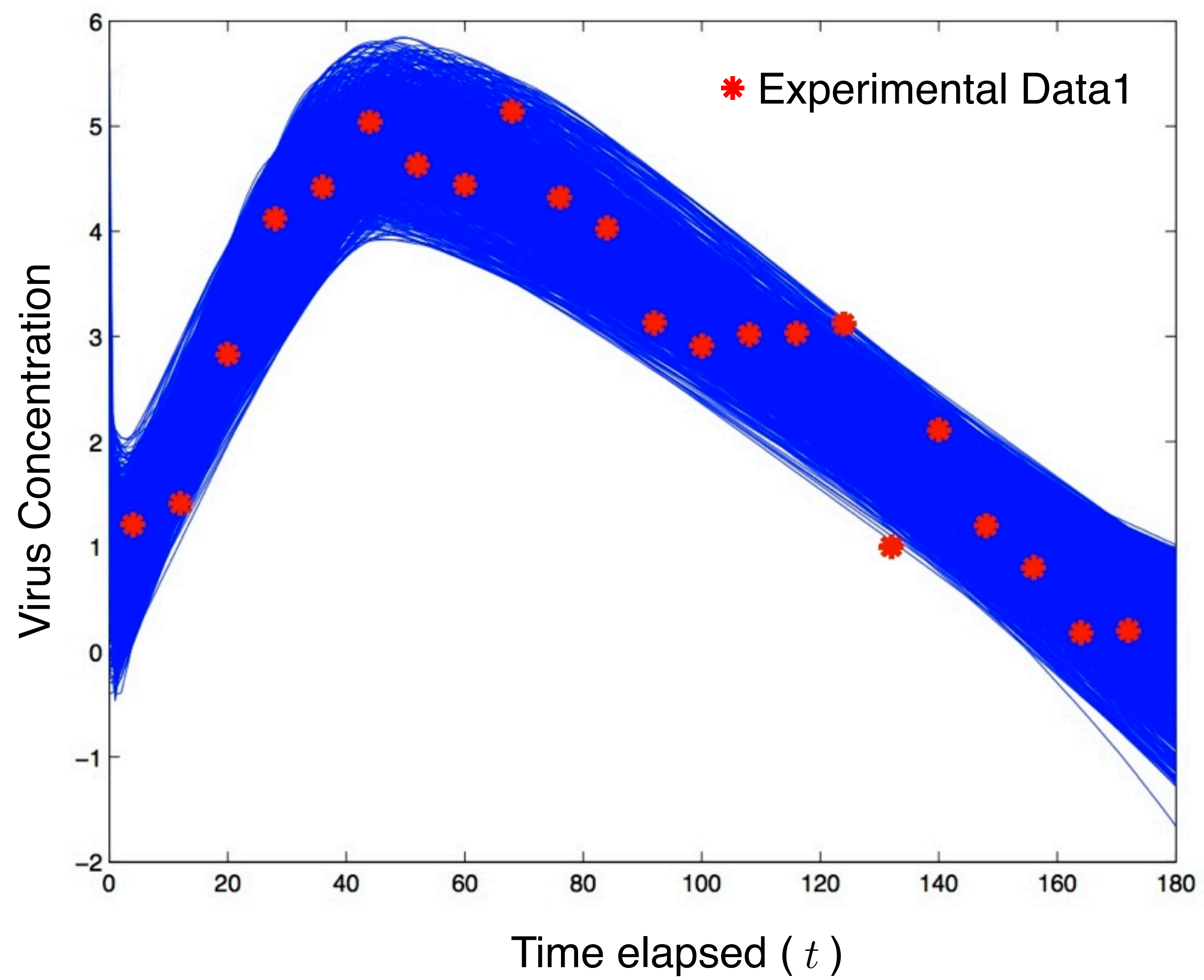
Resample residuals

Monte Carlo method

Bootstrap method

Monte Carlo / Bootstrap method





Monte Carlo / Bootstrap method

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\mathrm{exp1}}||_2$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\mathrm{exp1}}||_2$

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\mathrm{exp2}}||_2$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp1}}||_2$

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp2}}||_2$

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp3}}||_2$

$$\vdots$$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp1}}||_2$

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp2}}||_2$

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp3}}||_2$

$\vdots$

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*_{\text{exp1000}}||_2$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*||_2$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*||_2$

Gauss Newton method

$$\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{x}_{\mathrm{old}} + (J^{\mathrm{T}}J)^{-1}(J^{\mathrm{T}}(\boldsymbol{f}(\boldsymbol{x}_{\mathrm{old}}) - \boldsymbol{y}^*))$$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*||_2$

Levenberg-Marquardt method

$$\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{x}_{\mathrm{old}} + (J^{\mathrm{T}}J + \lambda I)^{-1}(J^{\mathrm{T}}(\boldsymbol{f}(\boldsymbol{x}_{\mathrm{old}}) - \boldsymbol{y}^*))$$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*||_2$

Levenberg-Marquardt method

$$\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{x}_{\mathrm{old}} + (J^{\mathrm{T}} J + \lambda I)^{-1} (J^{\mathrm{T}} (\boldsymbol{f}(\boldsymbol{x}_{\mathrm{old}}) - \boldsymbol{y}^*))$$

Monte Carlo / Bootstrap method

Find $\boldsymbol{x}$ that minimizes $||\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}^*||_2$

Levenberg-Marquardt method

$$\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{x}_{\mathrm{old}} + (J^{\mathrm{T}} J + \lambda I)^{-1} (J^{\mathrm{T}} (\boldsymbol{f}(\boldsymbol{x}_{\mathrm{old}}) - \boldsymbol{y}^*))$$

Approximating the Jacobian matrix

is computationally **expensive**...

Conventional Algorithm

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

$$\begin{matrix} \frac{\partial f_1}{\partial x_1} \\ \frac{\partial f_2}{\partial x_1} \\ \vdots \\ \frac{\partial f_n}{\partial x_1} \end{matrix}$$

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} \\ \frac{\partial f_2}{\partial x_1} \\ \vdots \\ \frac{\partial f_n}{\partial x_1} \end{bmatrix} \approx$$

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\frac{\partial f_1}{\partial x_1}} \\ \boldsymbol{\frac{\partial f_2}{\partial x_1}} \\ \vdots \\ \boldsymbol{\frac{\partial f_n}{\partial x_1}} \end{bmatrix} \approx \quad \boldsymbol{f}(x_1, x_2, \cdots x_m)$$

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} \\ \frac{\partial f_2}{\partial x_1} \\ \vdots \\ \frac{\partial f_n}{\partial x_1} \end{bmatrix} \approx \boldsymbol{f}(x_1, x_2, \cdots x_m) \; - \boldsymbol{f}(x_1 - \epsilon, x_2, \cdots x_m)$$

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} \\ \frac{\partial f_2}{\partial x_1} \\ \vdots \\ \frac{\partial f_n}{\partial x_1} \end{bmatrix} \approx \frac{\boldsymbol{f}(x_1, x_2, \cdots x_m) - \boldsymbol{f}(x_1 - \epsilon, x_2, \cdots x_m)}{\epsilon}$$

Conventional Algorithm

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \frac{\partial f_2}{\partial x_i} \\ \vdots \\ \frac{\partial f_n}{\partial x_i} \end{bmatrix} \approx \frac{\boldsymbol{f}(x_1, \cdots, x_i, \cdots, x_m) - \boldsymbol{f}(x_1, \cdots, x_i - \epsilon, \cdots, x_m)}{\epsilon}$$

$$i = 1, 2, \ldots, m$$

Conventional Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

Conventional Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

(Number of parameters + 1)

Conventional Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

(Number of parameters + 1) x Number of iterations

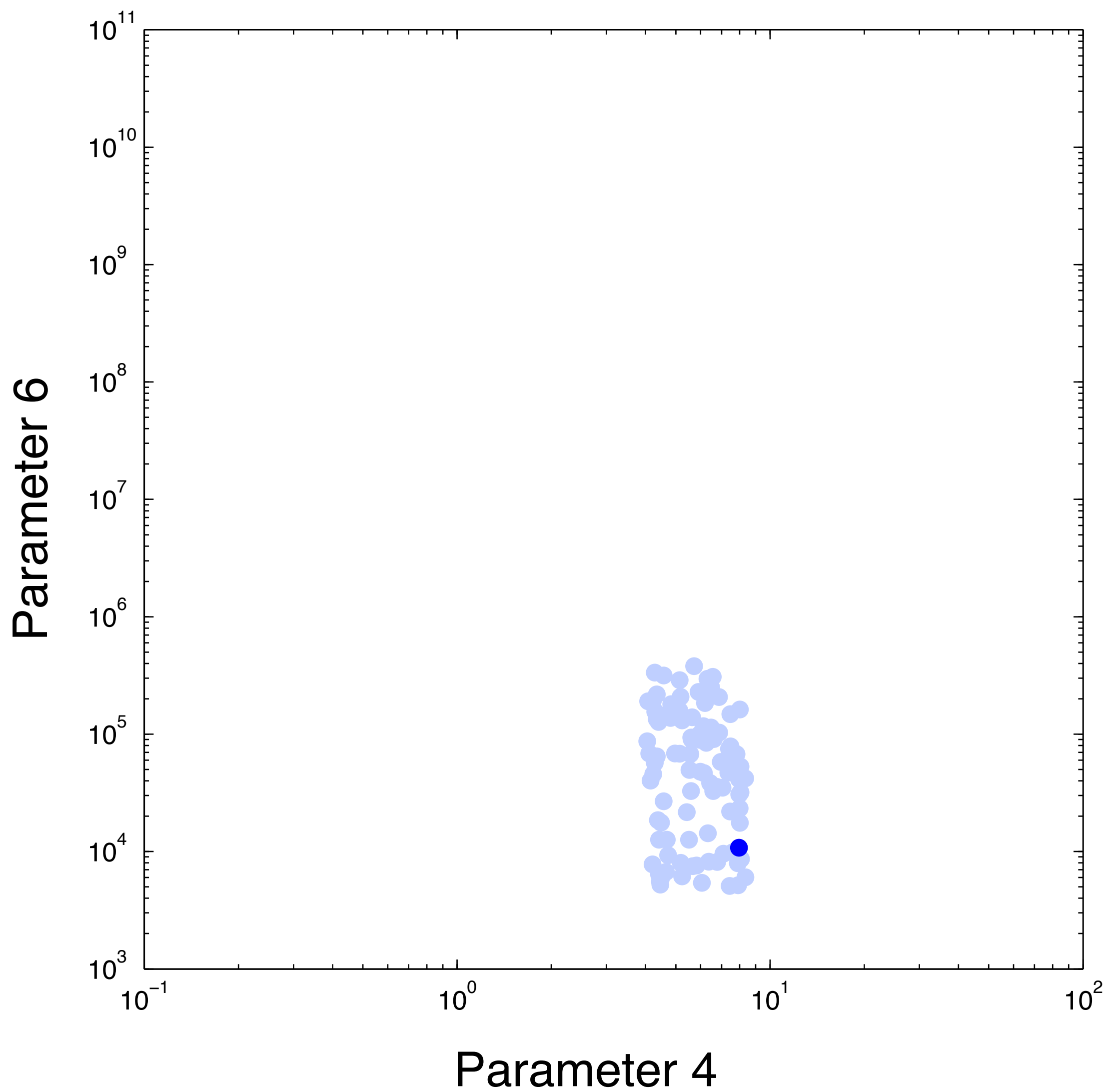
Conventional Algorithm

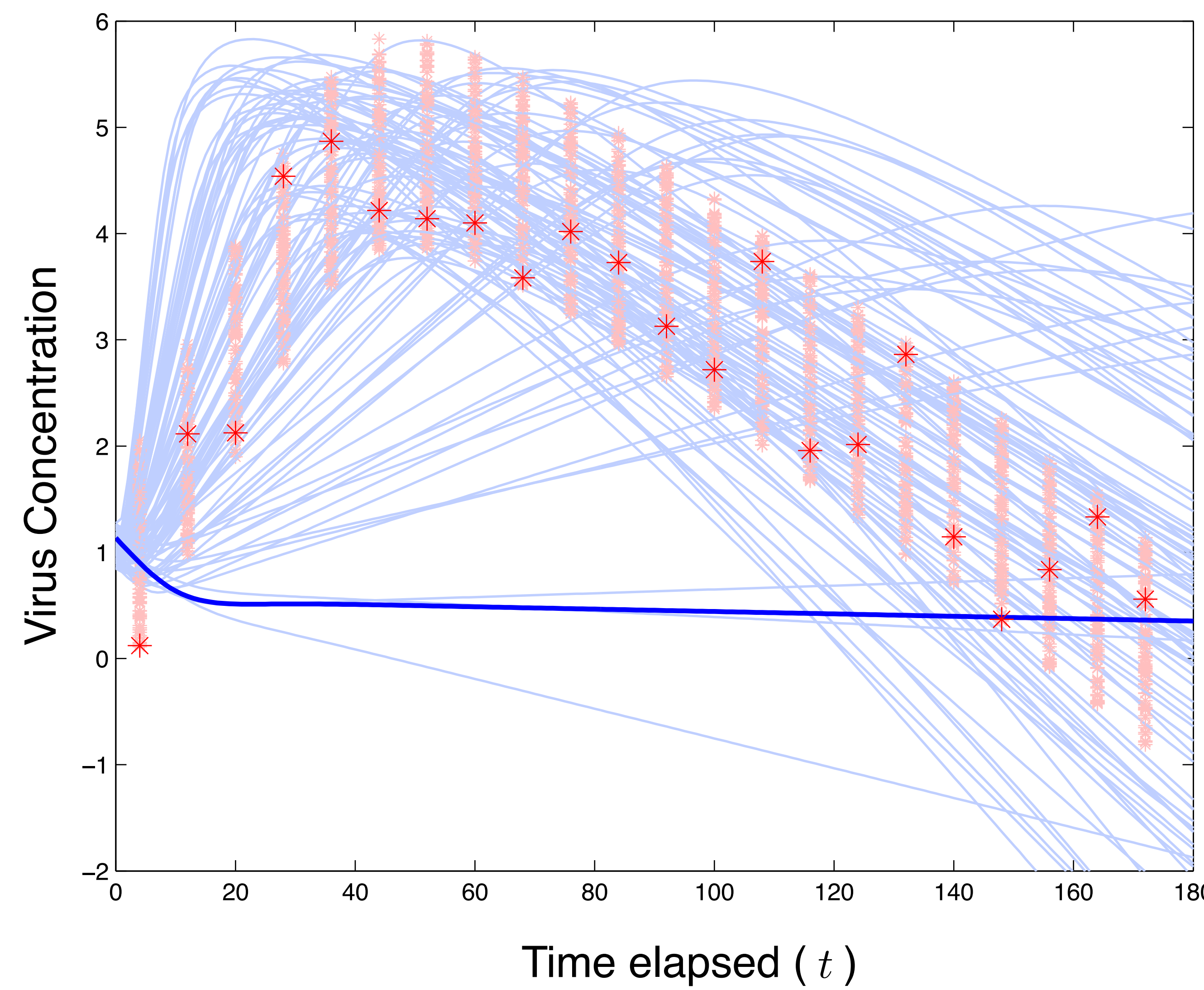
Number of simulations needed to obtain 1,000 sets of parameters.

(Number of parameters + 1) x Number of iterations x 1,000

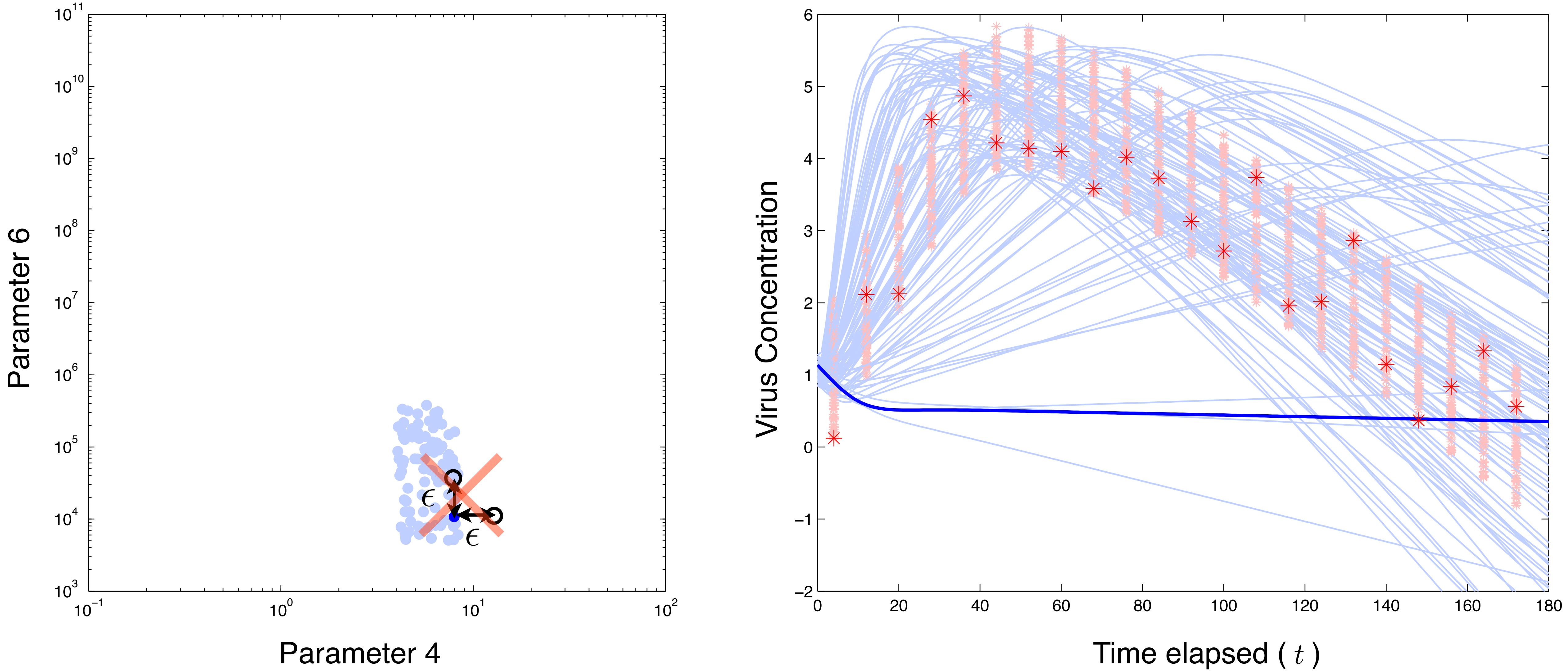
New Algorithm

New Algorithm

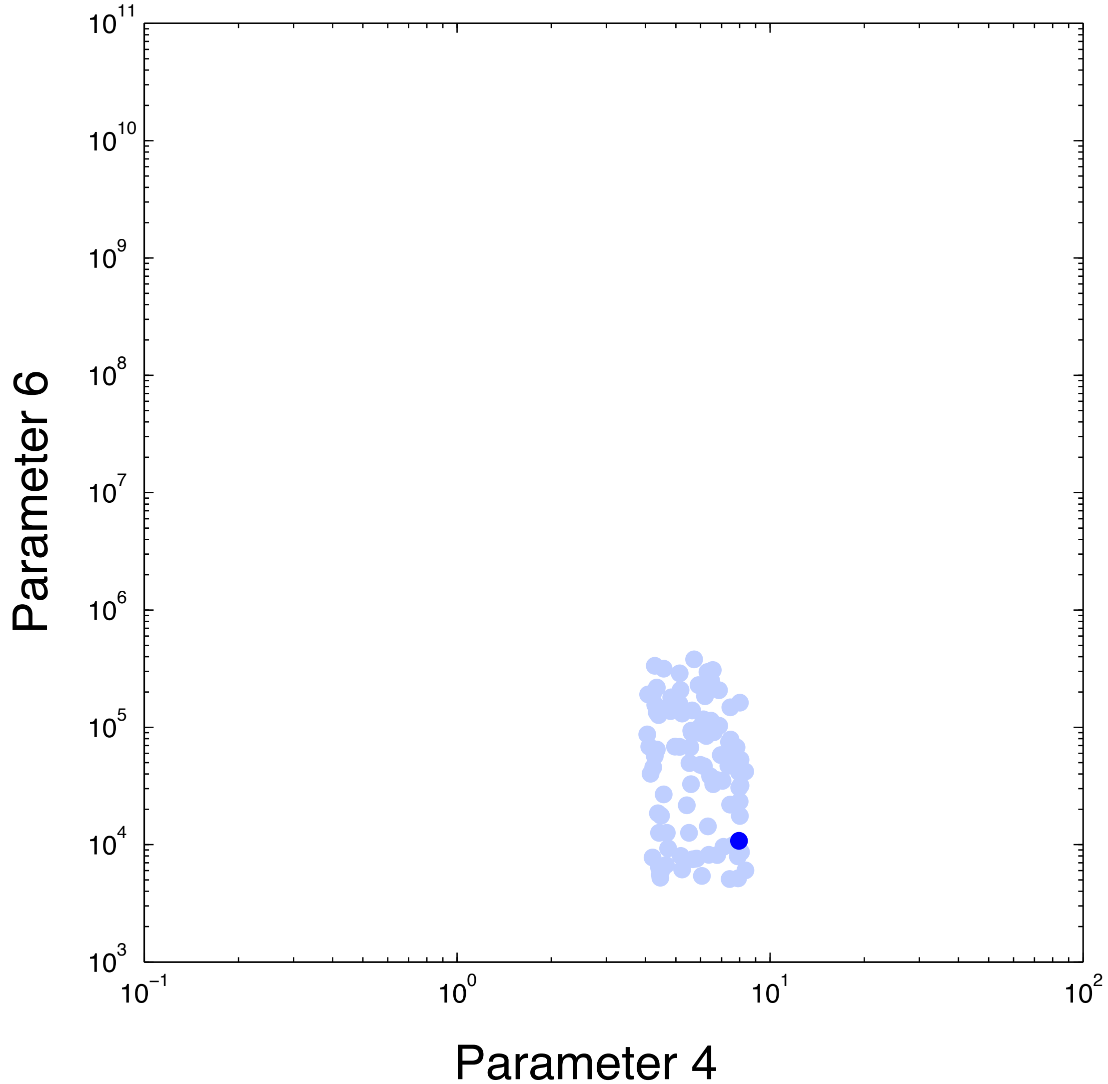


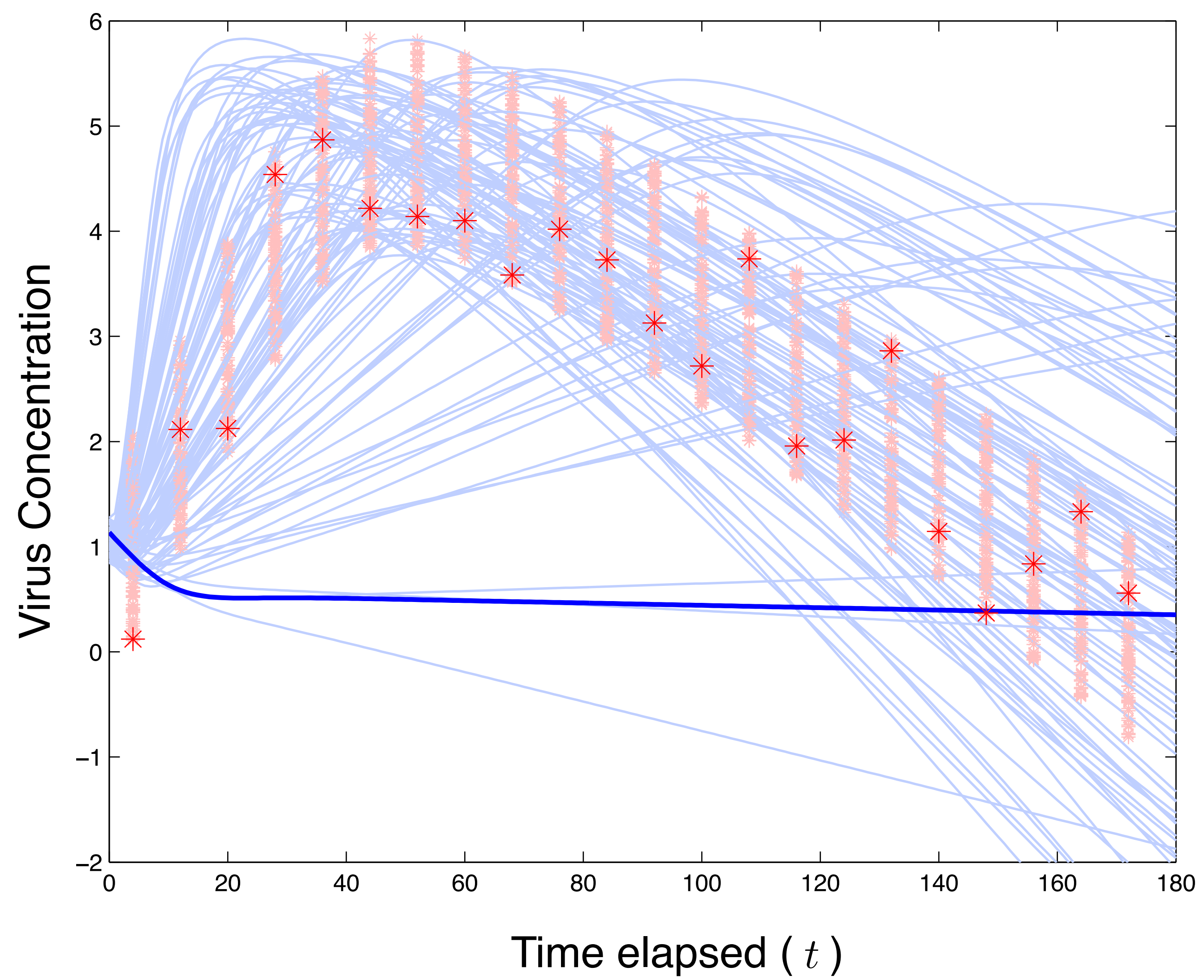


New Algorithm



New Algorithm





New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

Jacobian is like... Slope

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

Jacobian is like...

$$\text{Slope} \approx \frac{\text{Rise}}{\text{Run}}$$

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

Jacobian is like...

$$\text{Slope} \approx \frac{\text{Rise}}{\text{Run}}$$

Rise: $\Delta \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_{\text{neighbour}})$

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

Jacobian is like... $\text{Slope} \approx \dfrac{\text{Rise}}{\text{Run}}$

Rise: $\Delta \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_{\text{neighbour}})$

Run: $\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{\text{neighbour}}$

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

Jacobian is like...

$$J_{\text{approx}} = \frac{\Delta \boldsymbol{y}}{\Delta \boldsymbol{x}}$$

Rise: $\Delta \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_{\text{neighbour}})$

Run: $\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{\text{neighbour}}$

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

$$\Delta \boldsymbol{y} \quad = J_{\text{approx}} \Delta \boldsymbol{x}$$

$$\Delta \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_{\text{neighbour}})$$

$$\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{\text{neighbour}}$$

New Algorithm

Approximating the Jacobian matrix using the **neighbours**...

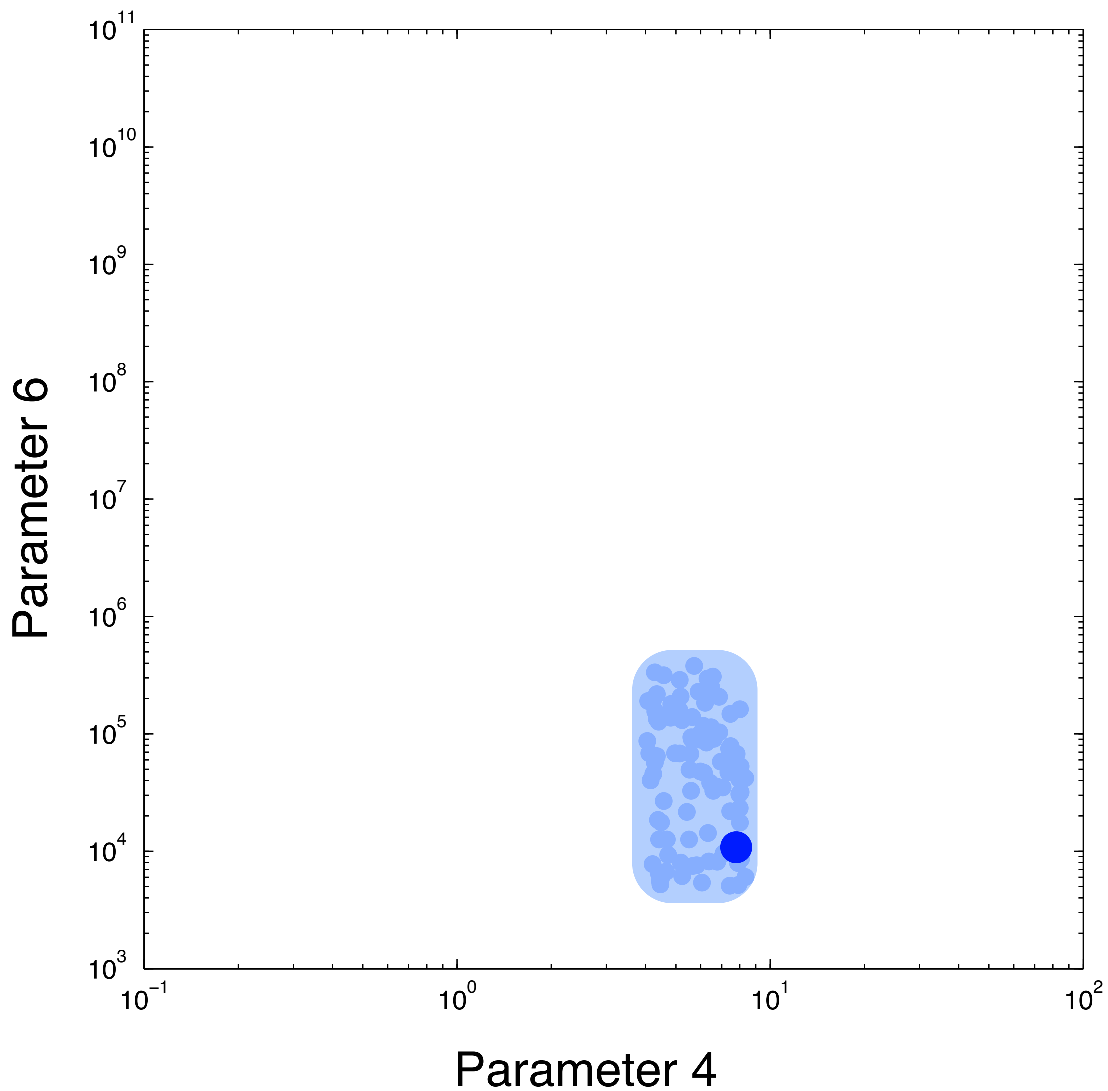
$$\Delta \boldsymbol{y}^{(i)} = J_{\mathrm{approx}} \Delta \boldsymbol{x}^{(i)}$$

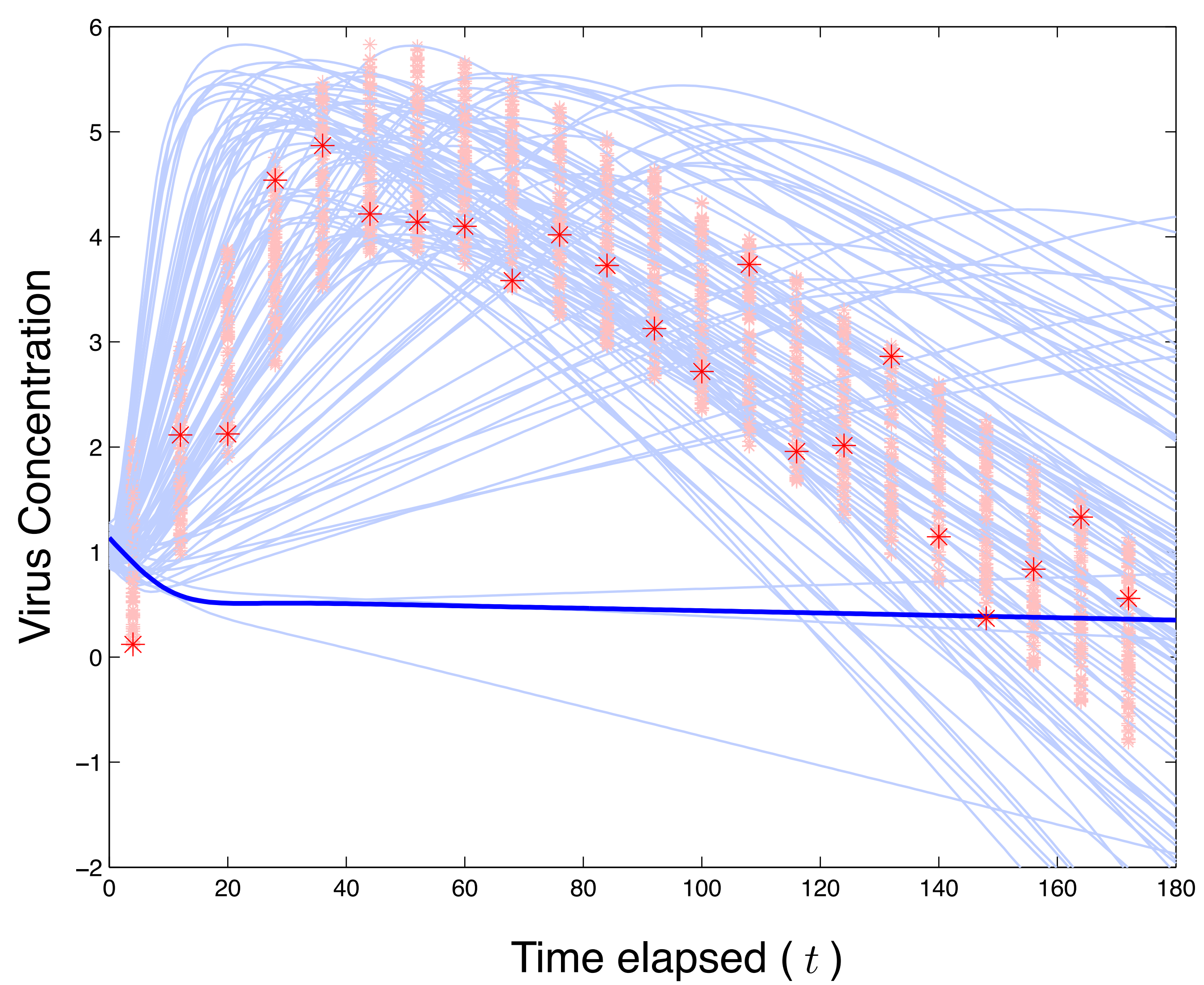
$$\text{for } i = 1, 2, \ldots, n_{\mathrm{neighbour}}$$

$$\Delta \boldsymbol{y}^{(i)} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}^{(i)}_{\mathrm{neighbour}})$$

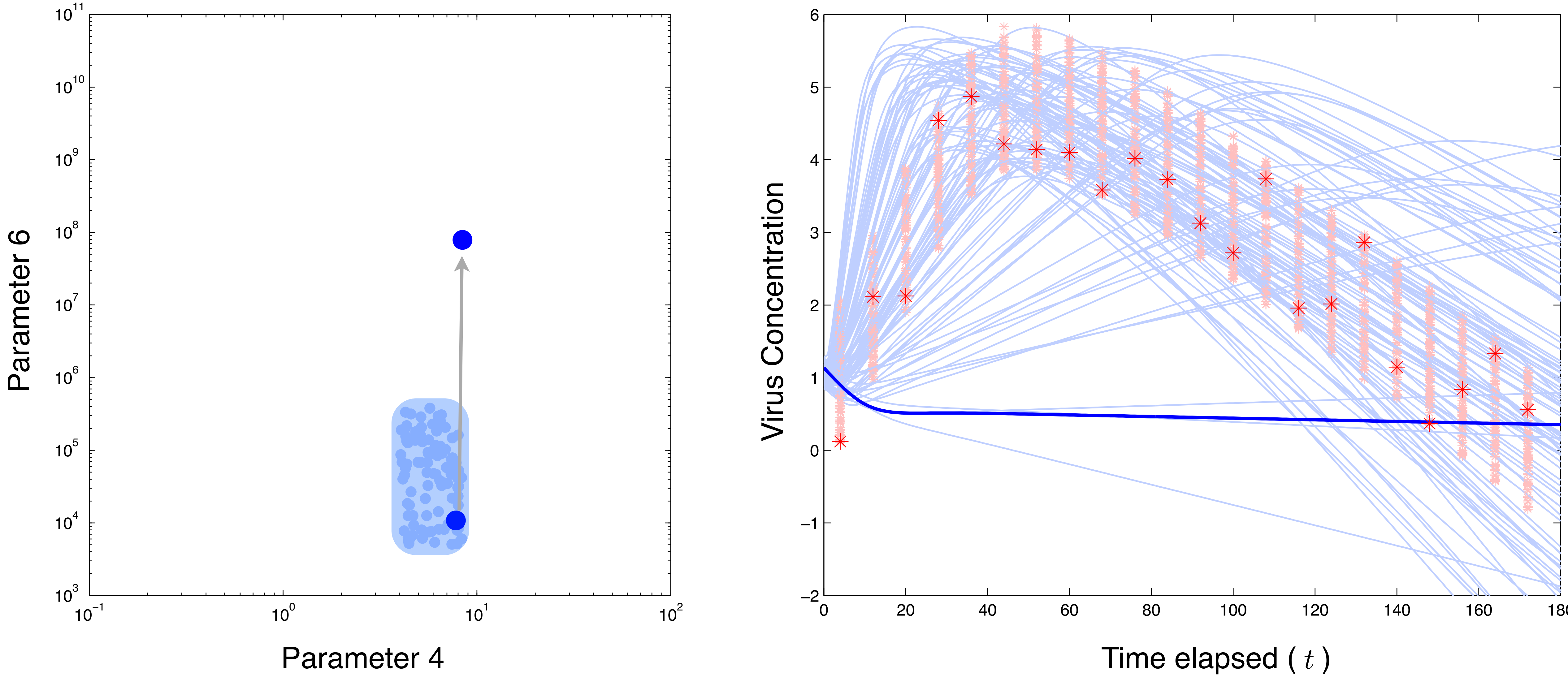
$$\Delta \boldsymbol{x}^{(i)} = \boldsymbol{x} - \boldsymbol{x}^{(i)}_{\mathrm{neighbour}}$$

New Algorithm

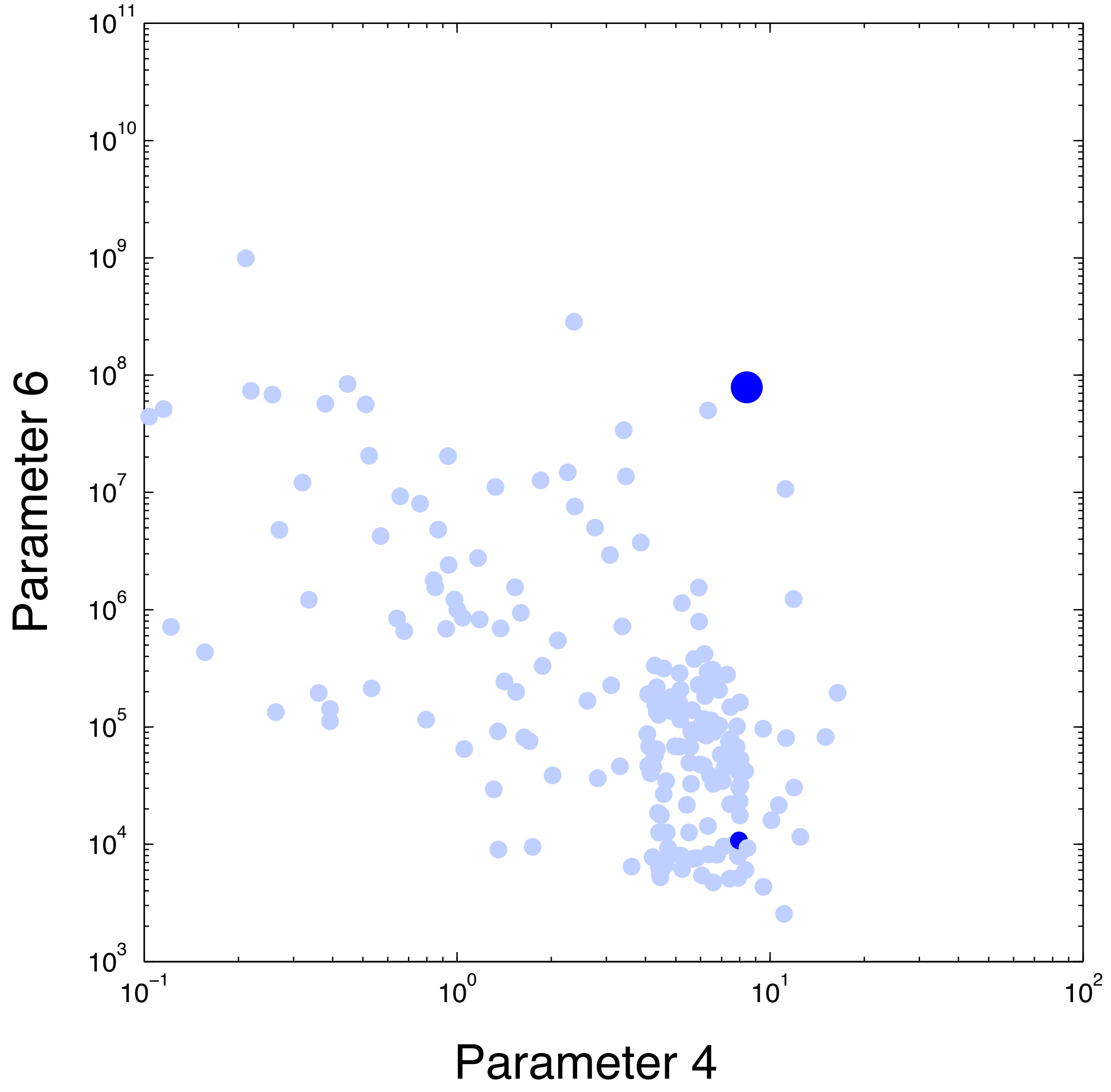


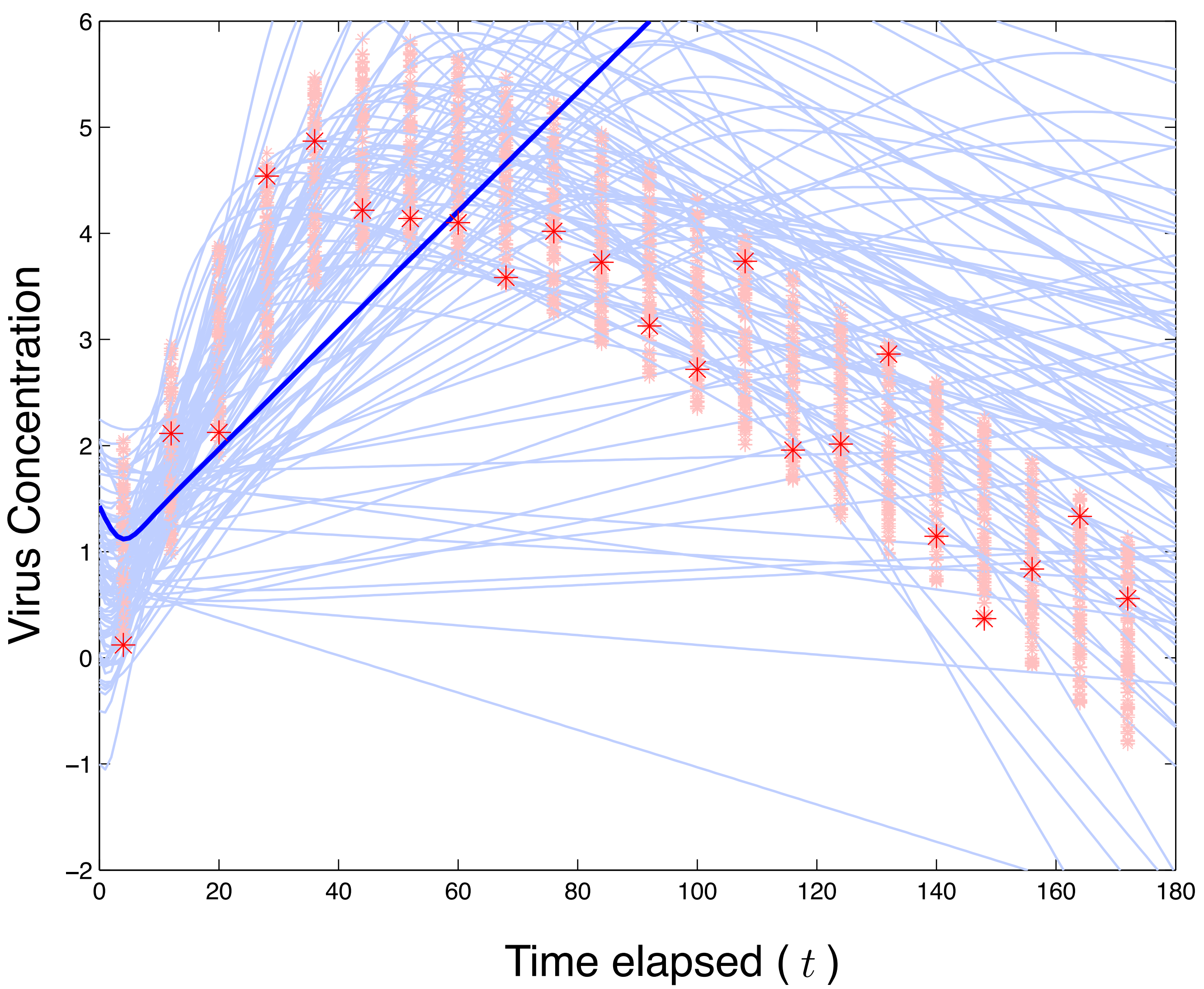


New Algorithm

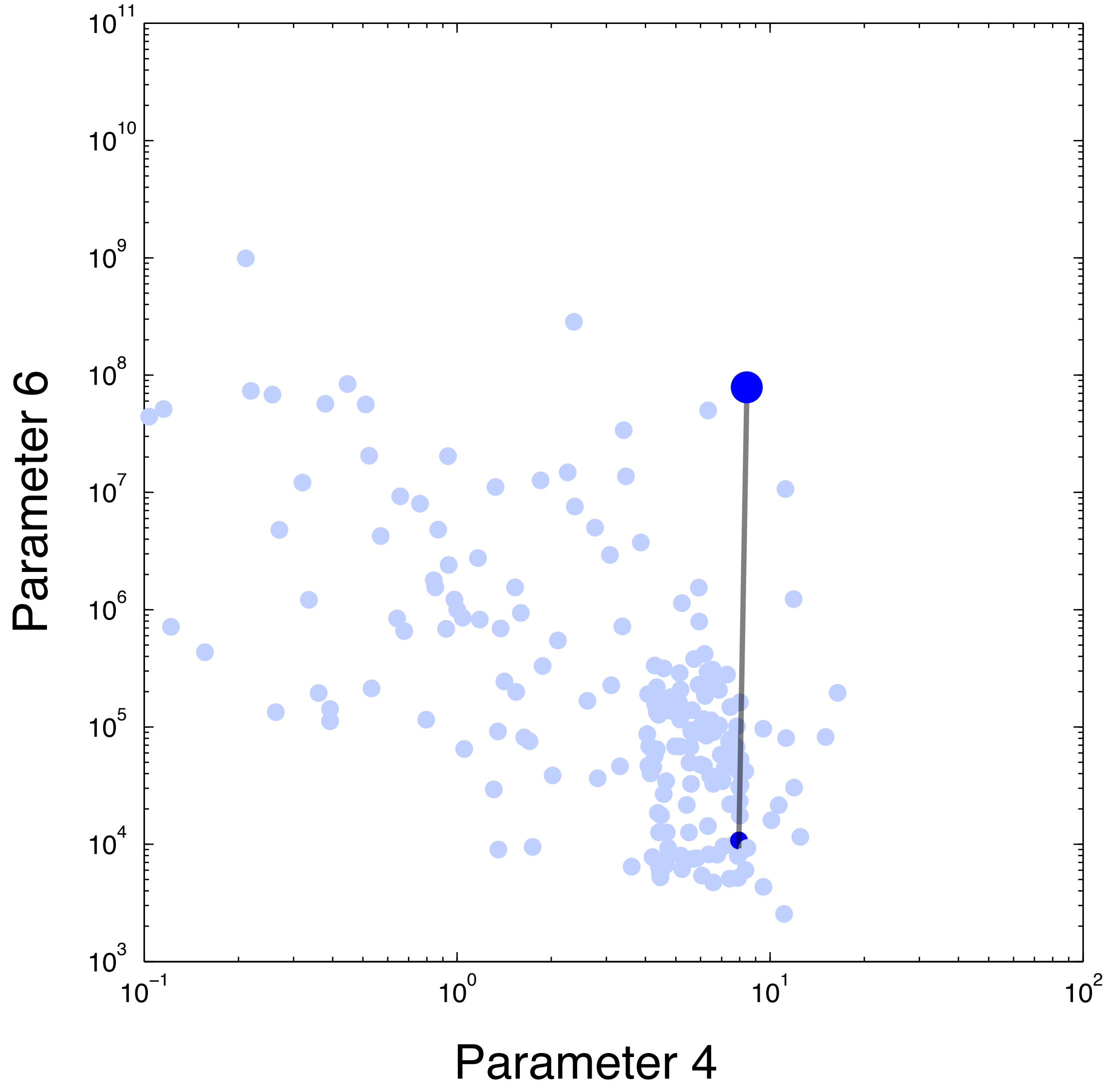


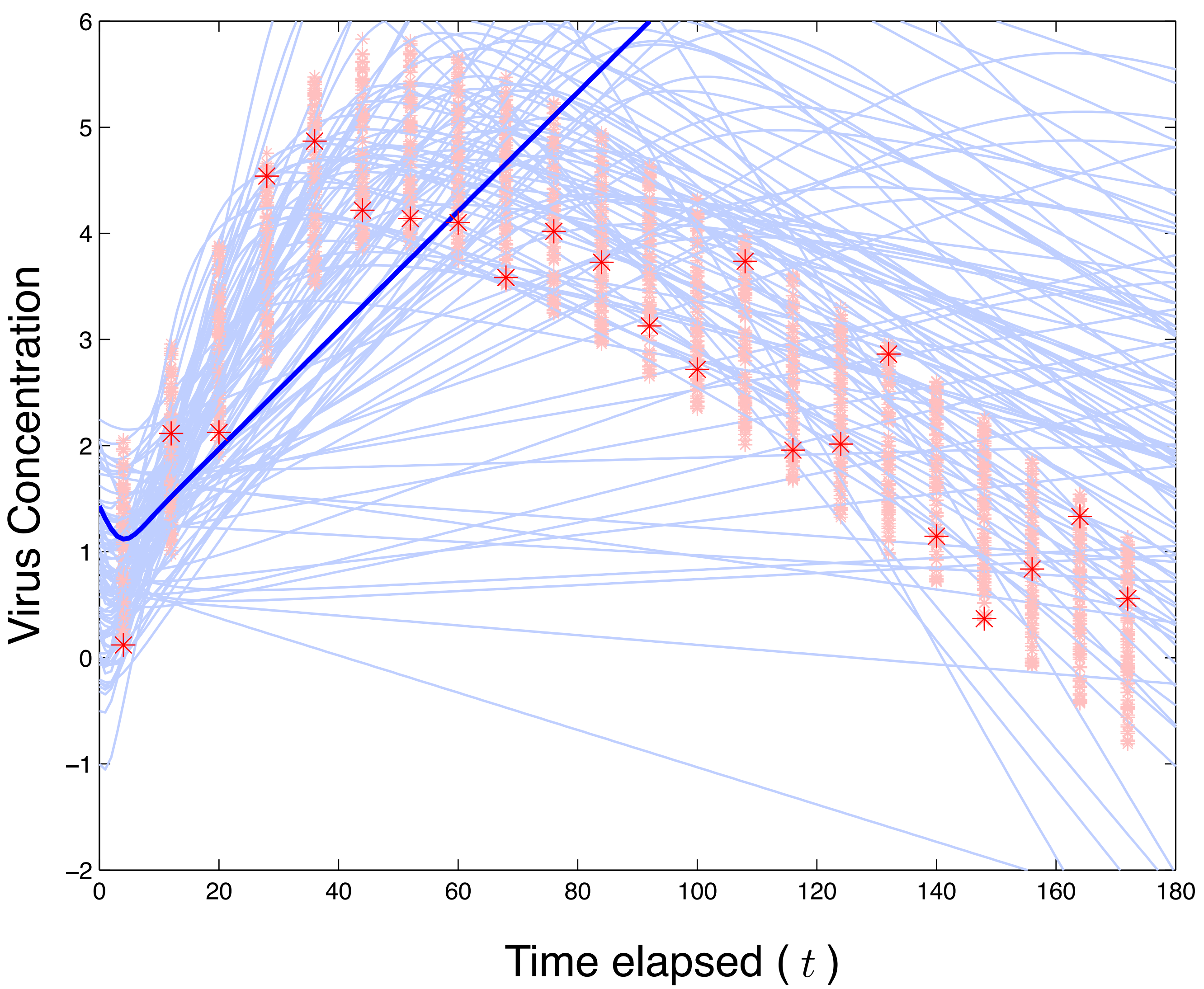
New Algorithm



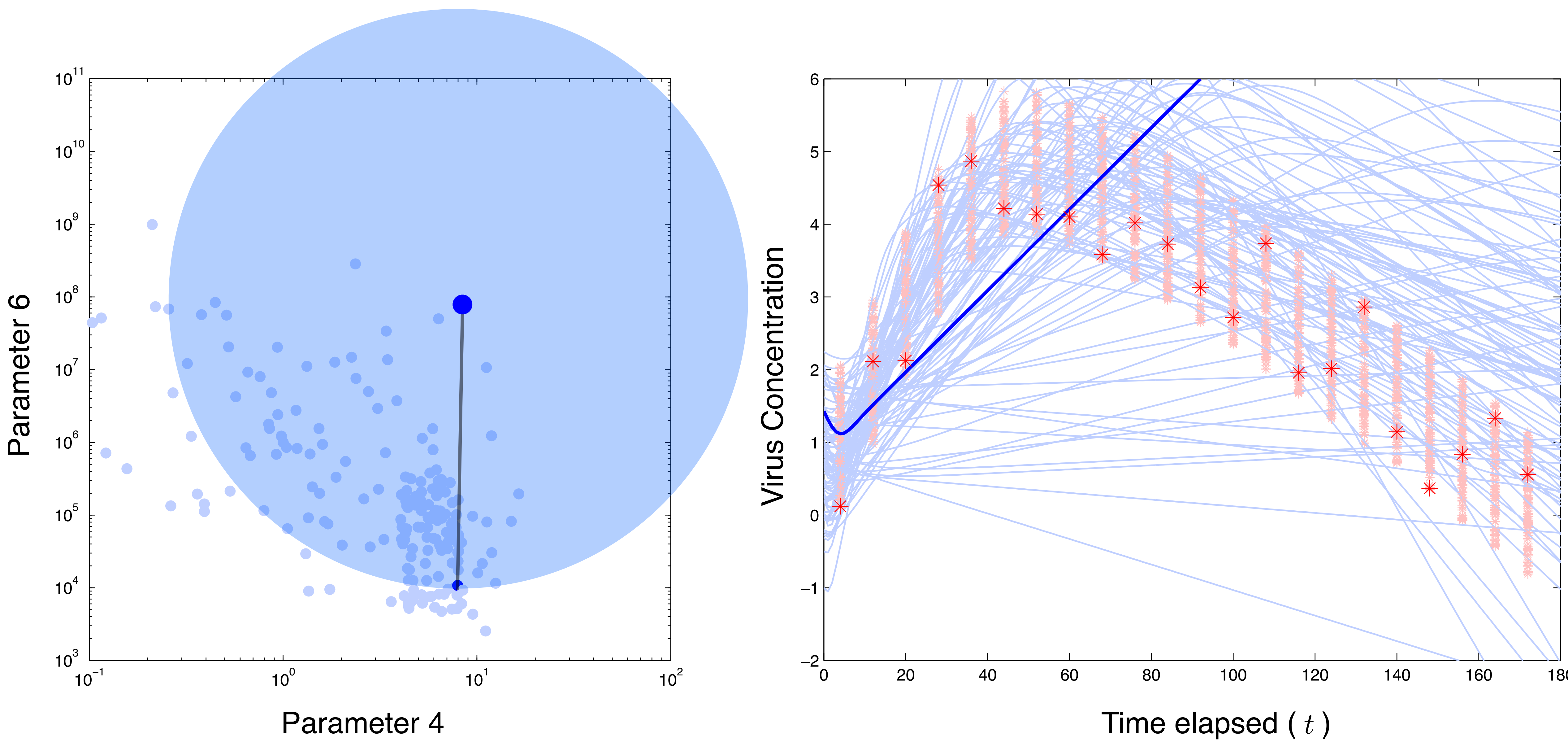


New Algorithm

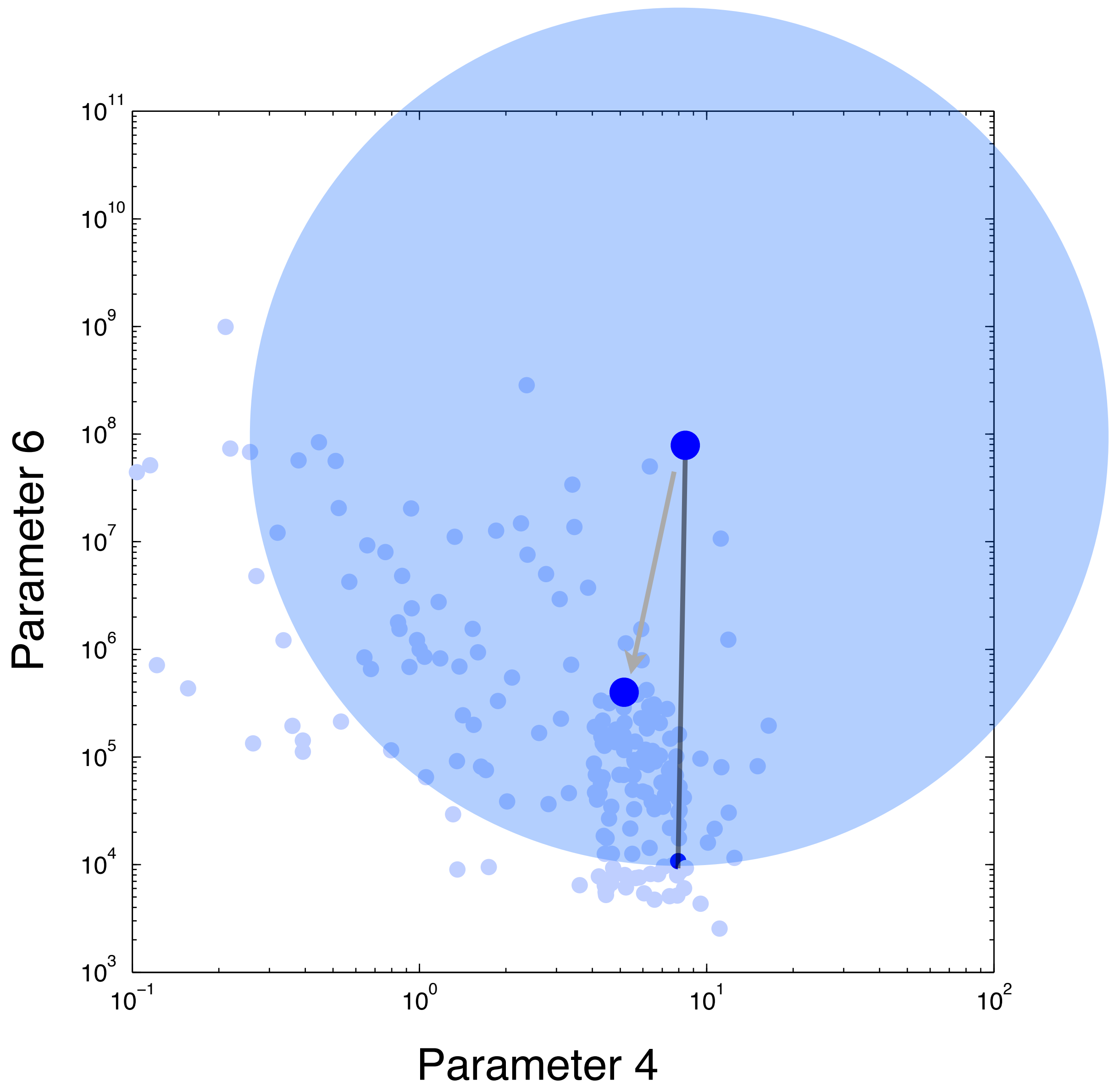


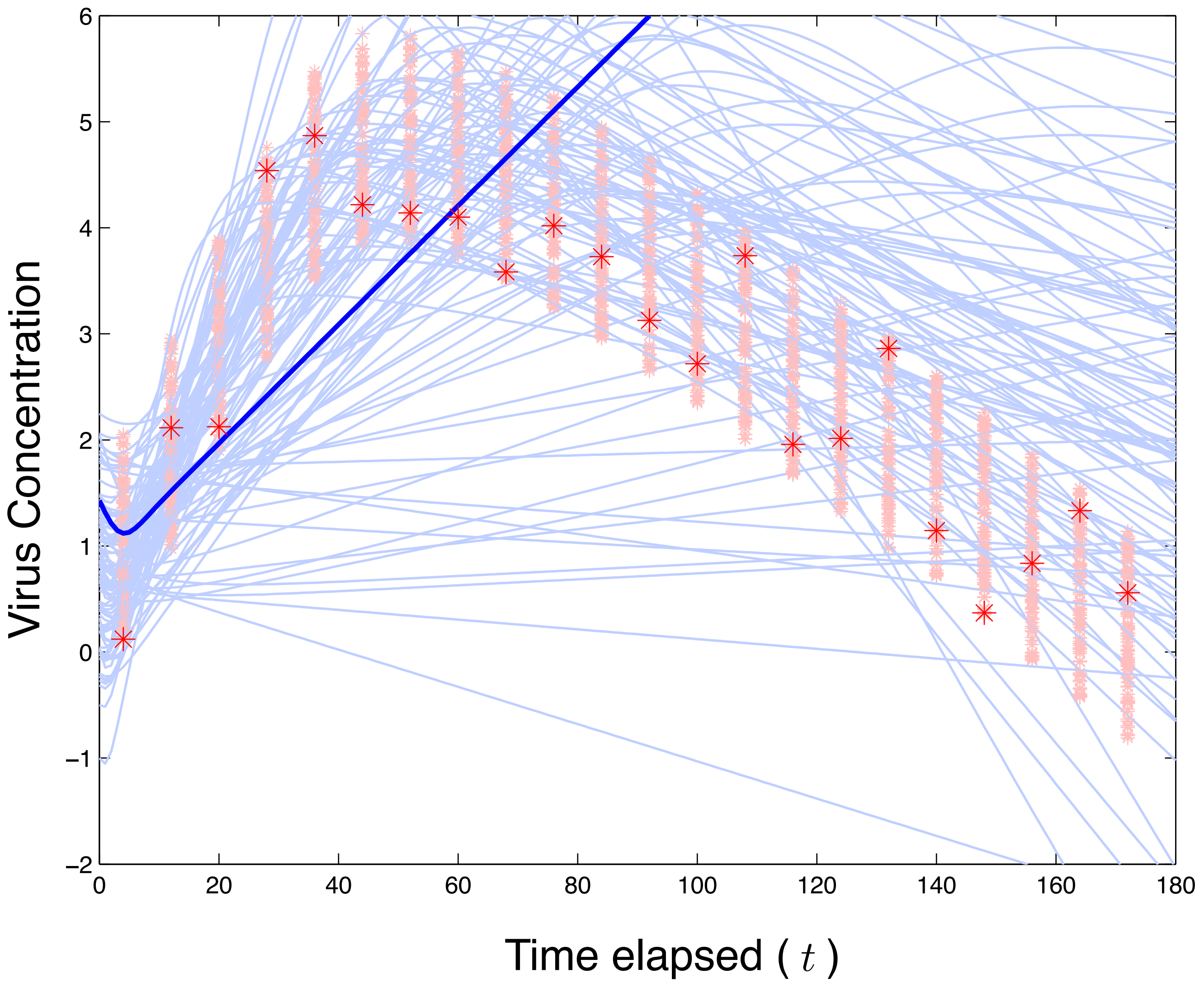


New Algorithm

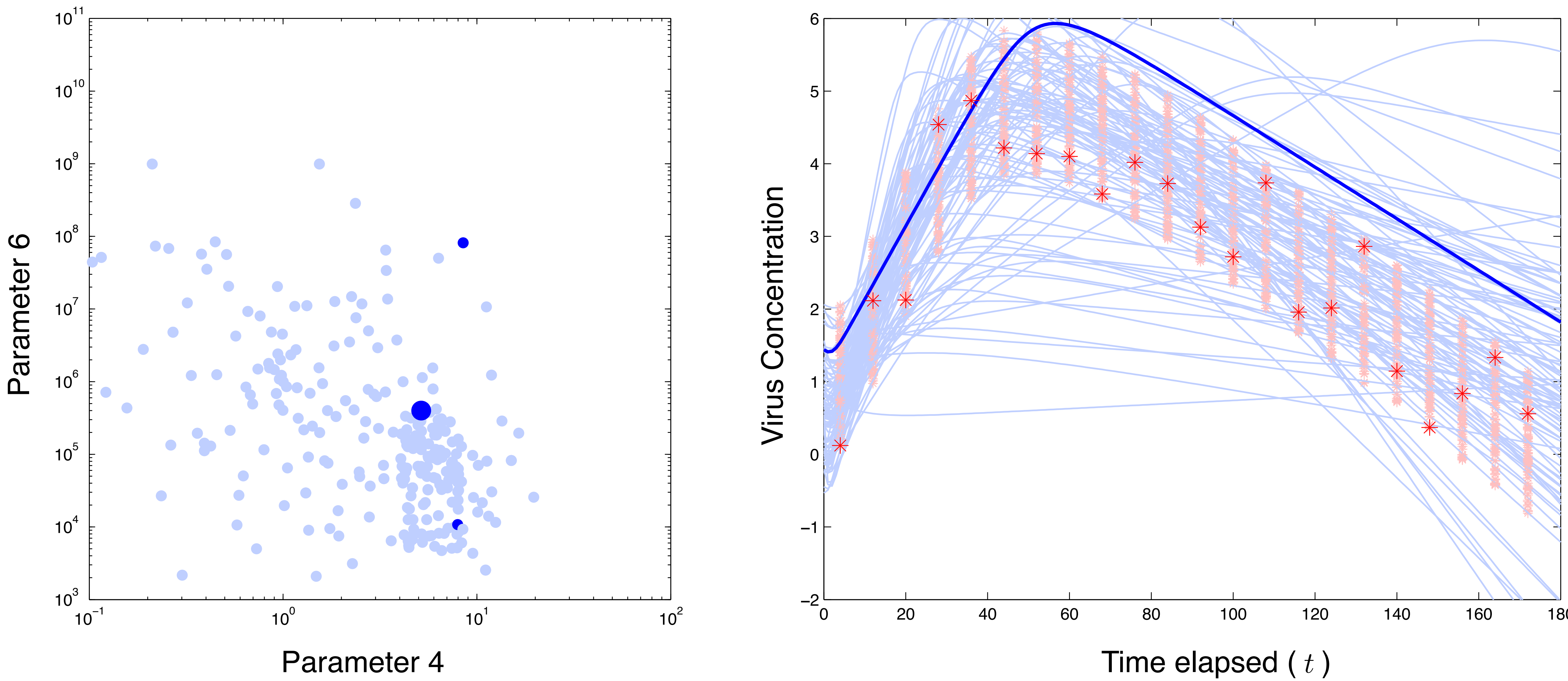


New Algorithm

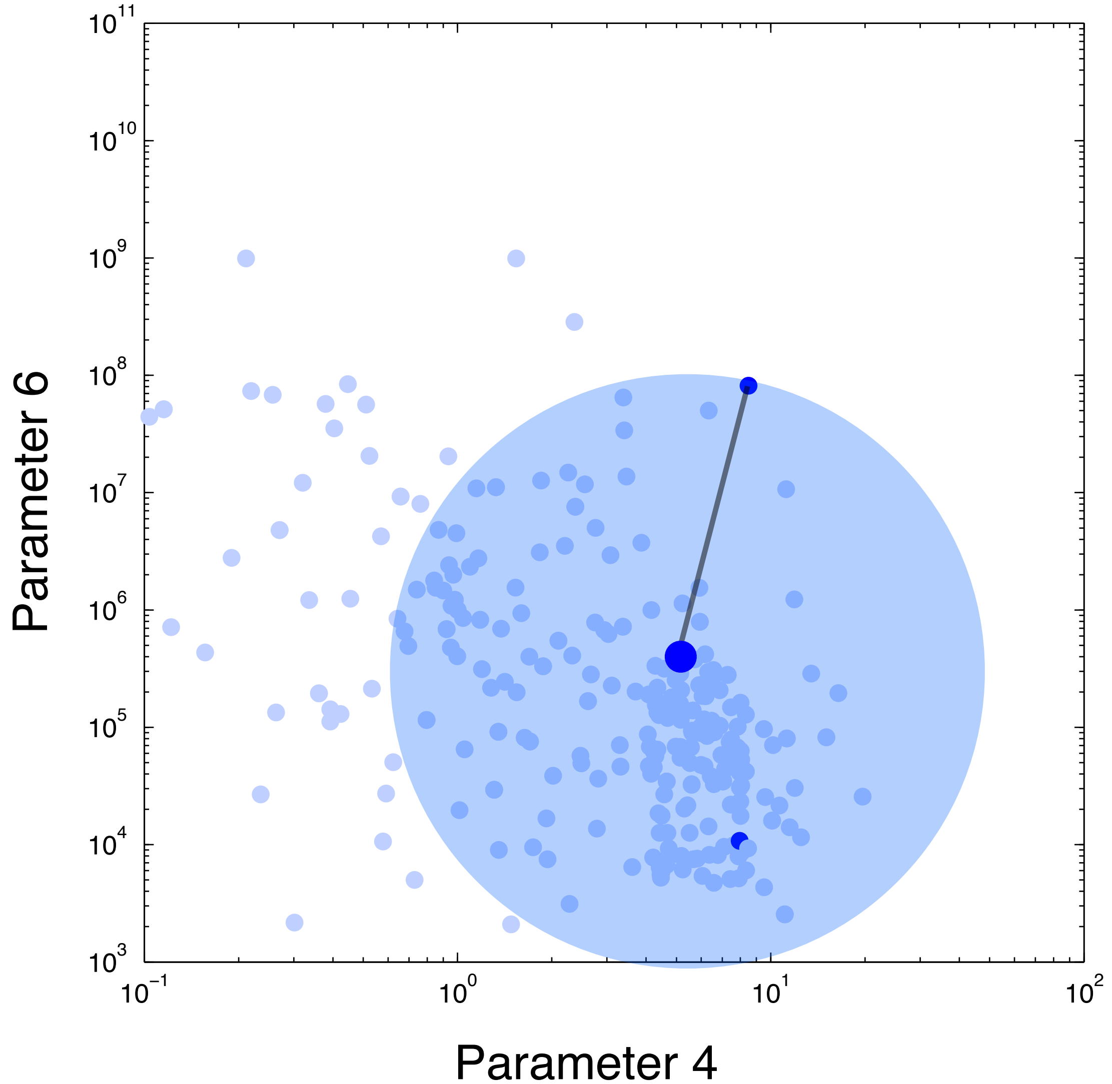


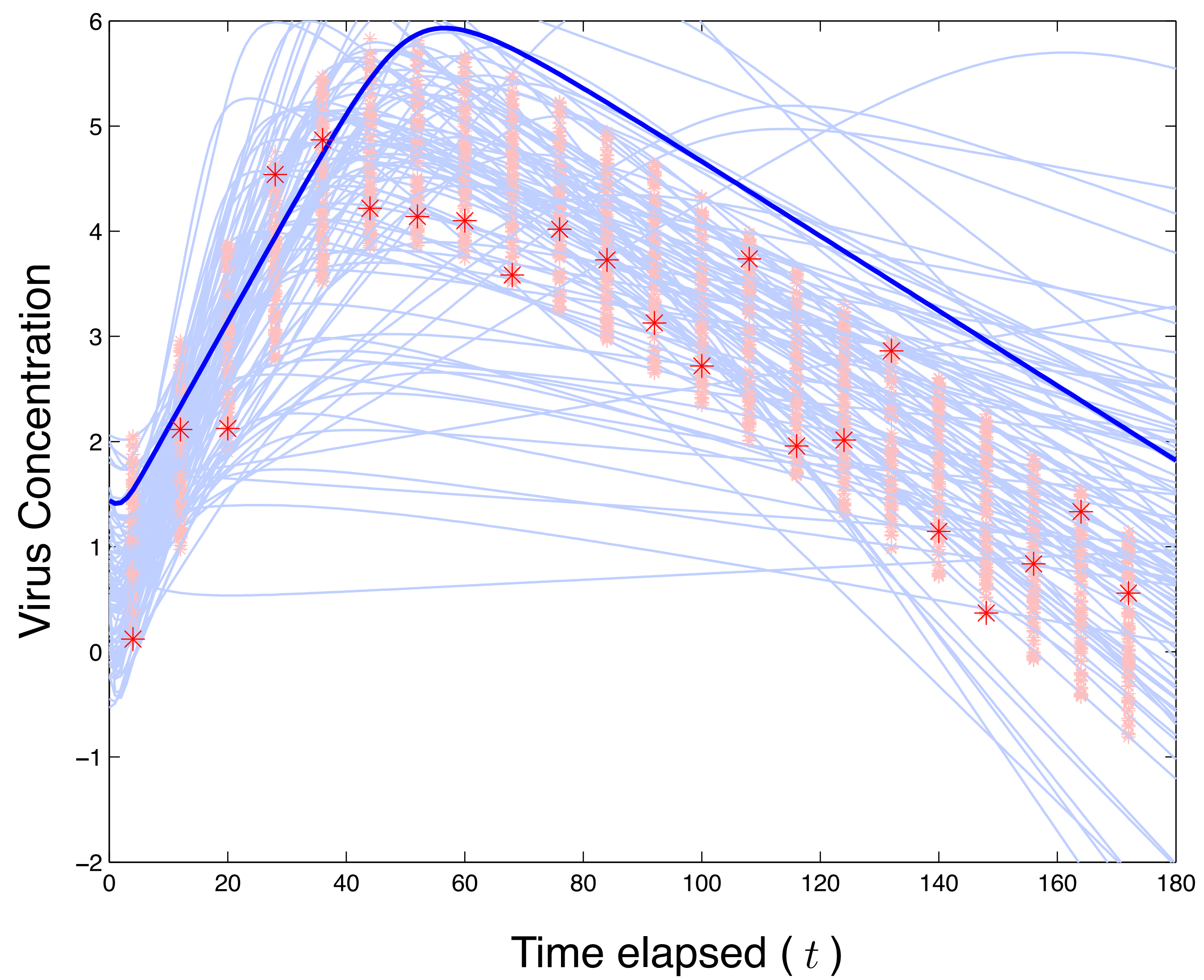


New Algorithm

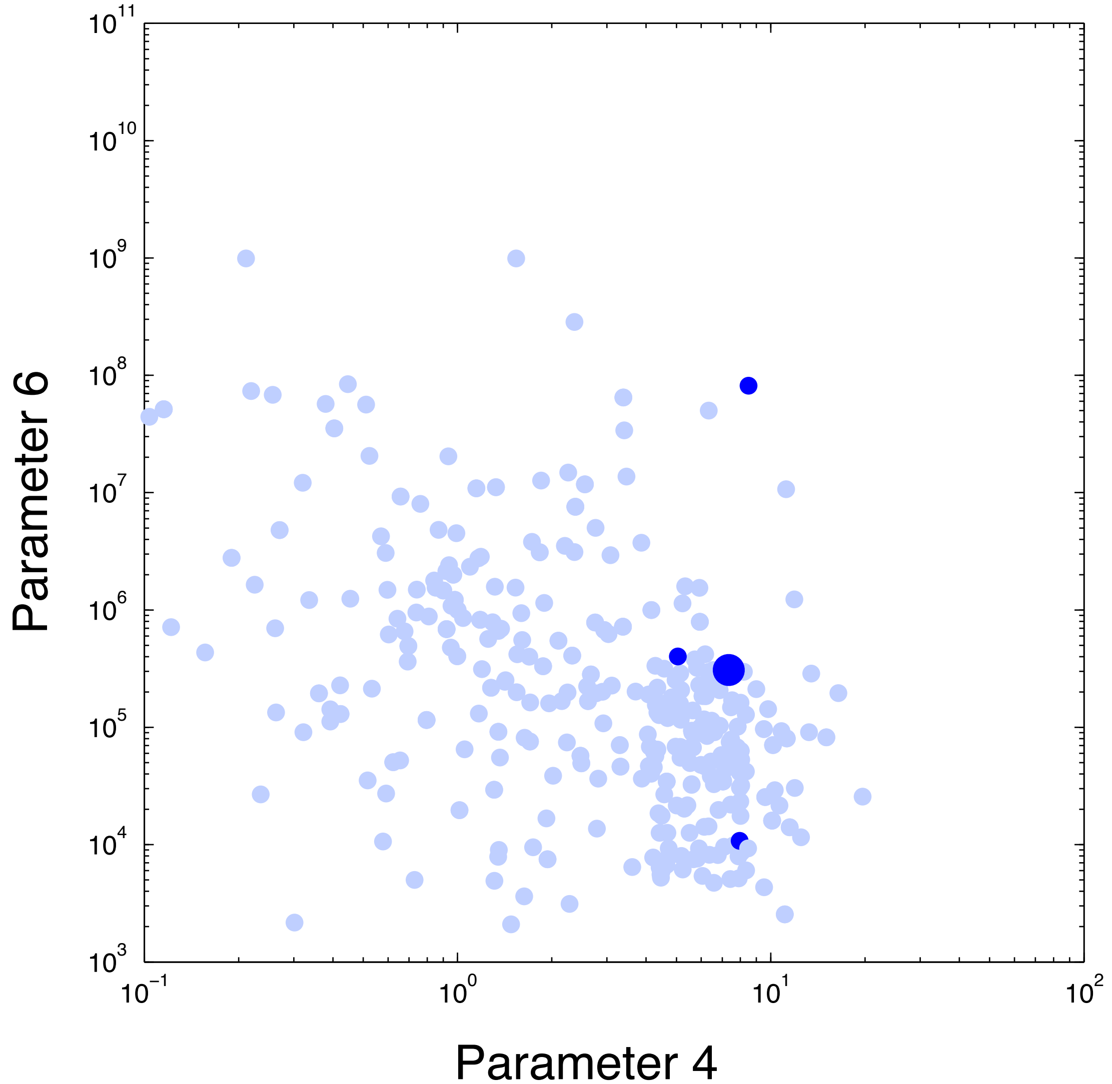


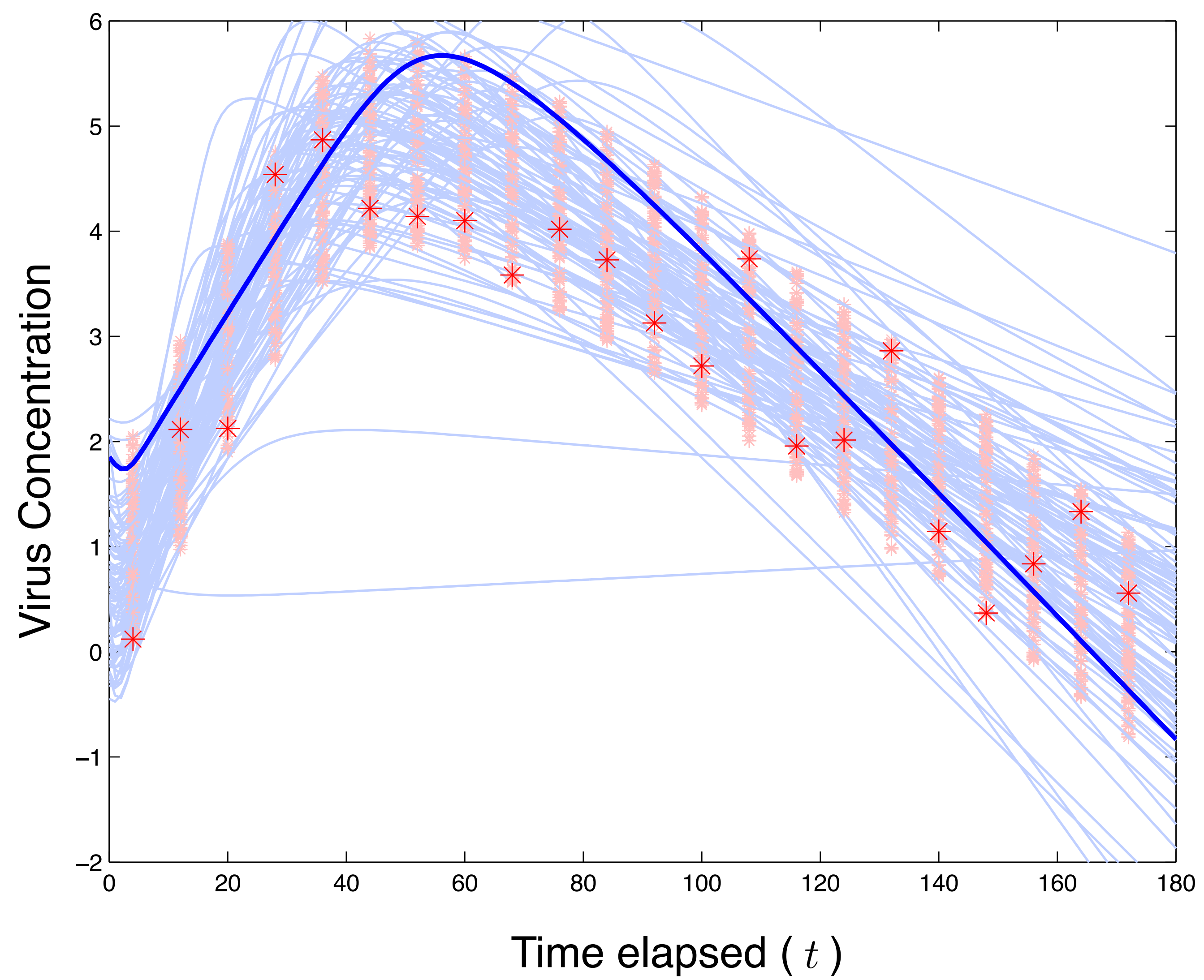
New Algorithm



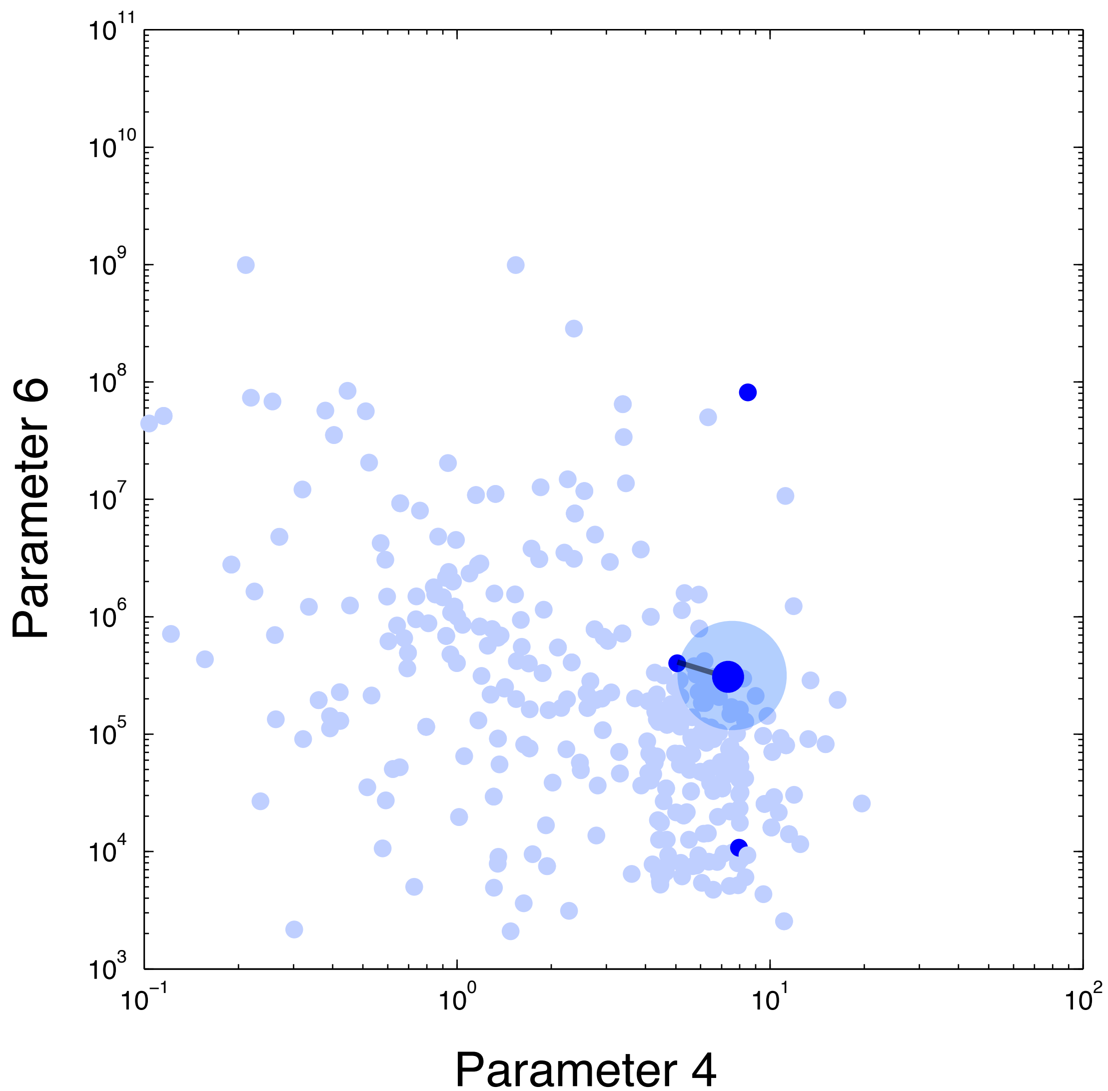


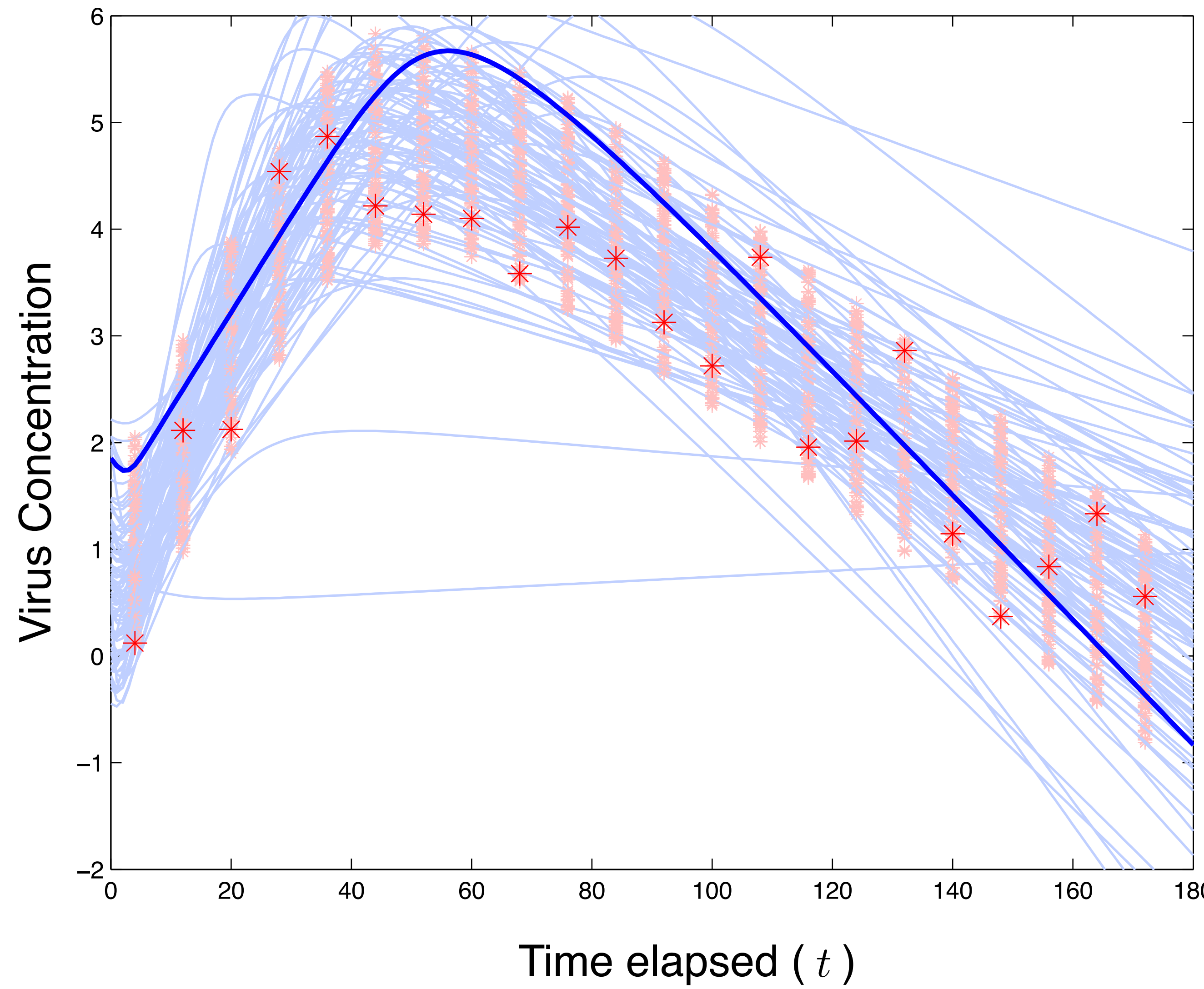
New Algorithm



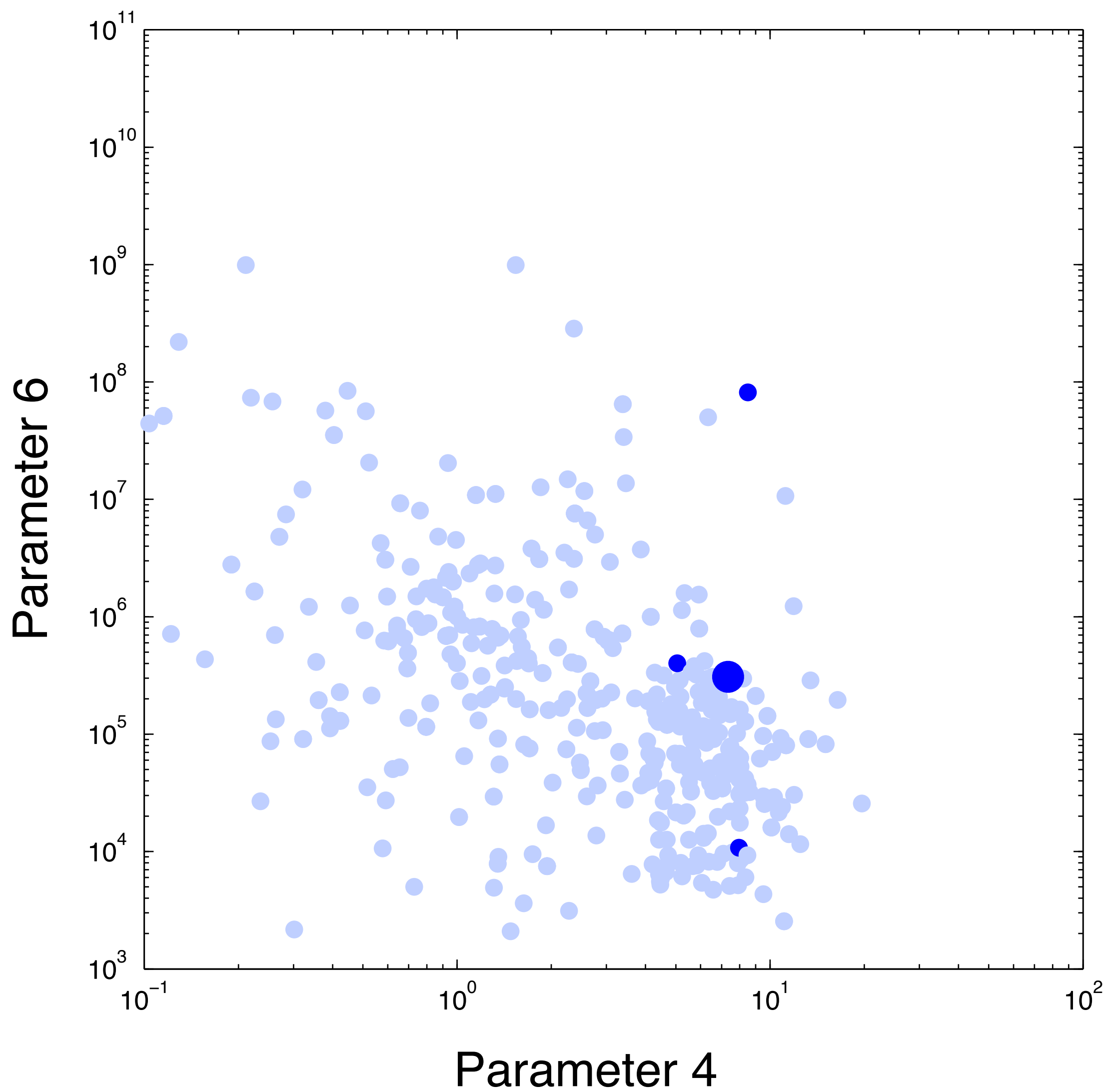


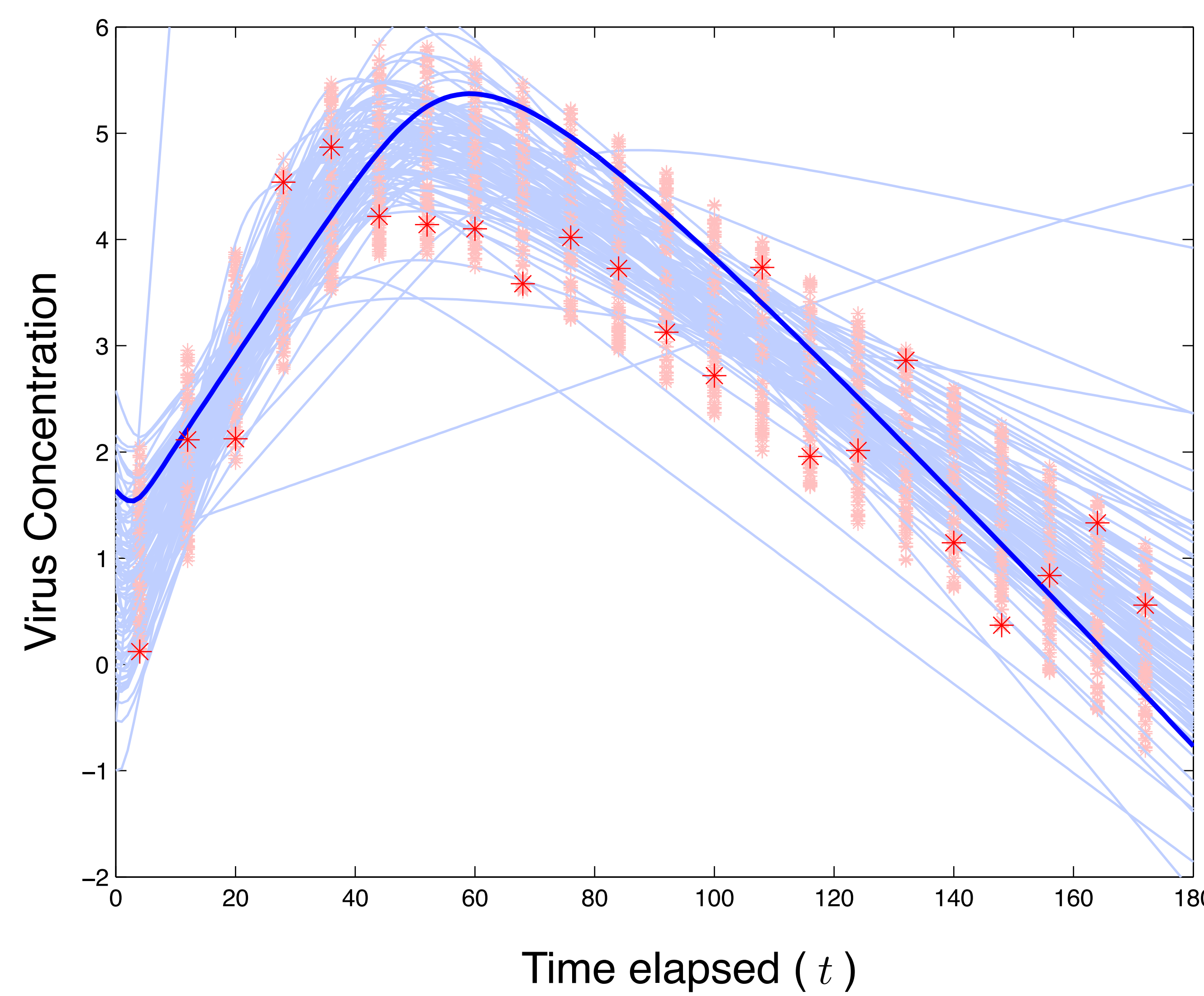
New Algorithm





New Algorithm





New Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

New Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

1

New Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

1 x Number of iterations

New Algorithm

Number of simulations needed to obtain 1,000 sets of parameters.

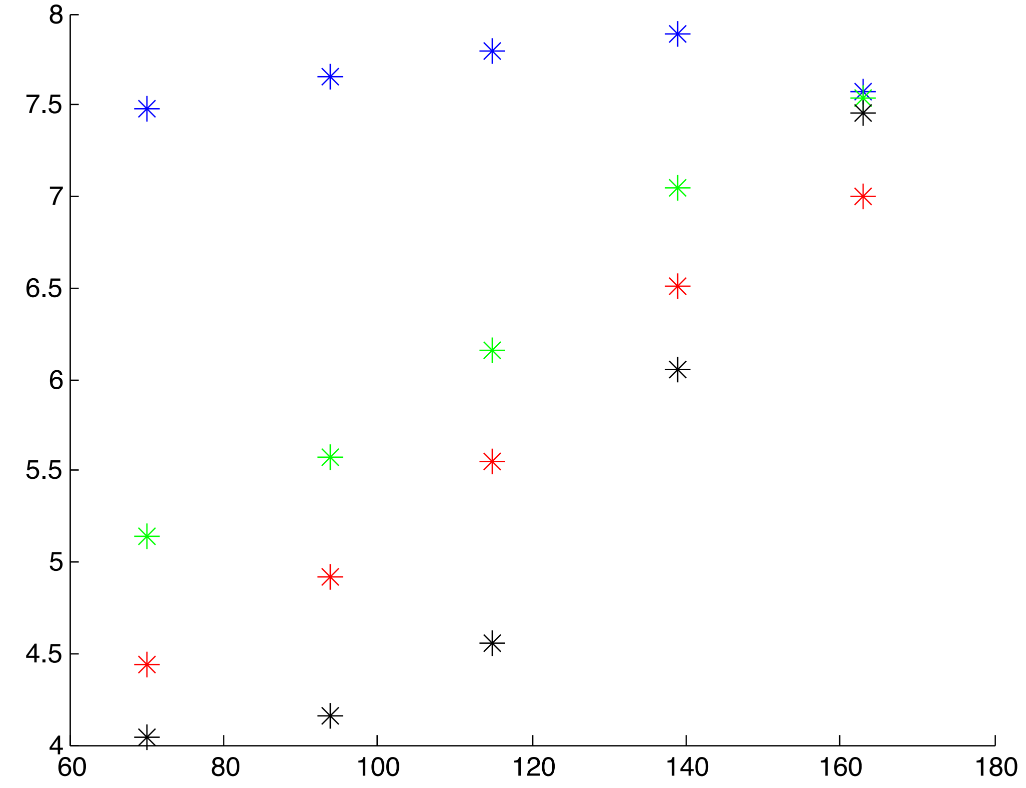
1 x Number of iterations x 1,000

Example 1 HIV kinetics model of Miao et al.

Example 1 HIV kinetics model of Miao et al.

$$\frac{du_1}{dt} = (x_1 - x_5 u_2 - x_6 u_3 - x_7 u_4) u_1$$
$$\frac{du_2}{dt} = (x_2 + x_5 u_1 - x_8 u_3) u_2 + \frac{1}{4} x_7 u_4 u_1$$
$$\frac{du_3}{dt} = (x_3 + x_6 u_1 - x_9 u_2) u_3 + \frac{1}{4} x_7 u_4 u_1$$
$$\frac{du_4}{dt} = (x_4 + \frac{1}{2} x_7 u_1) u_4 + (x_8 + x_9) u_3 u_2$$

$$u_1(t=0) = x_{10}$$
$$u_2(t=0) = x_{11}$$
$$u_3(t=0) = x_{12}$$
$$u_4(t=0) = x_{13}$$



Hongyu Miao, Carrie Dykes, Lisa M. Demeter, James Cavenaugh, Sung Yong Park, Alan S. Perelson, and Hulin Wu. Modeling and Estimation of Kinetic Parameters and Replicative Fitness of HIV-1 from Flow-Cytometry-Based Growth Competition Experiments. Bulletin of Mathematical Biology, 70:1749– 1771, 2008.

Example 1 HIV kinetics model of Miao et al.

$$\frac{du_1}{dt} = (x_1 - x_5 u_2 - x_6 u_3 - x_7 u_4) u_1$$

$$\frac{du_2}{dt} = (x_2 + x_5 u_1 - x_8 u_3) u_2 + \frac{1}{4} x_7 u_4 u_1$$

$$\frac{du_3}{dt} = (x_3 + x_6 u_1 - x_9 u_2) u_3 + \frac{1}{4} x_7 u_4 u_1$$

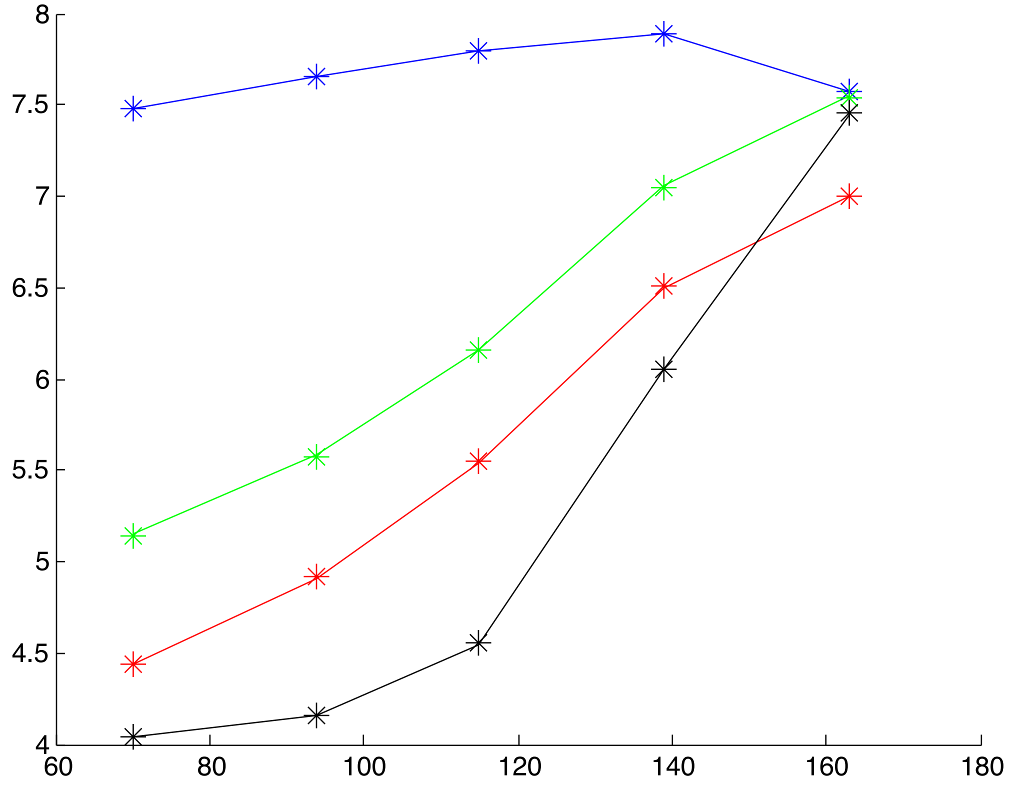
$$\frac{du_4}{dt} = (x_4 + \frac{1}{2} x_7 u_1) u_4 + (x_8 + x_9) u_3 u_2$$

$$u_1(t=0) = x_{10}$$

$$u_2(t=0) = x_{11}$$

$$u_3(t=0) = x_{12}$$

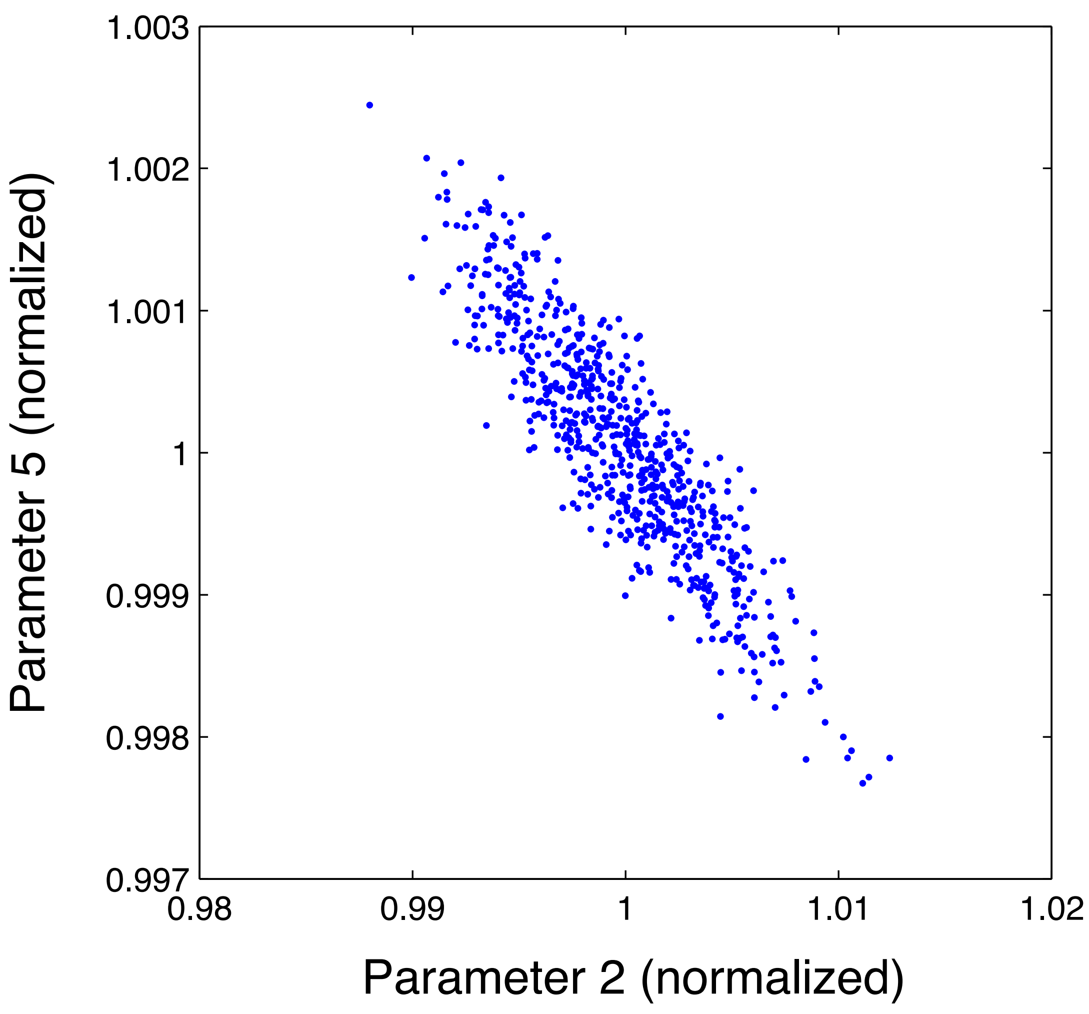
$$u_4(t=0) = x_{13}$$



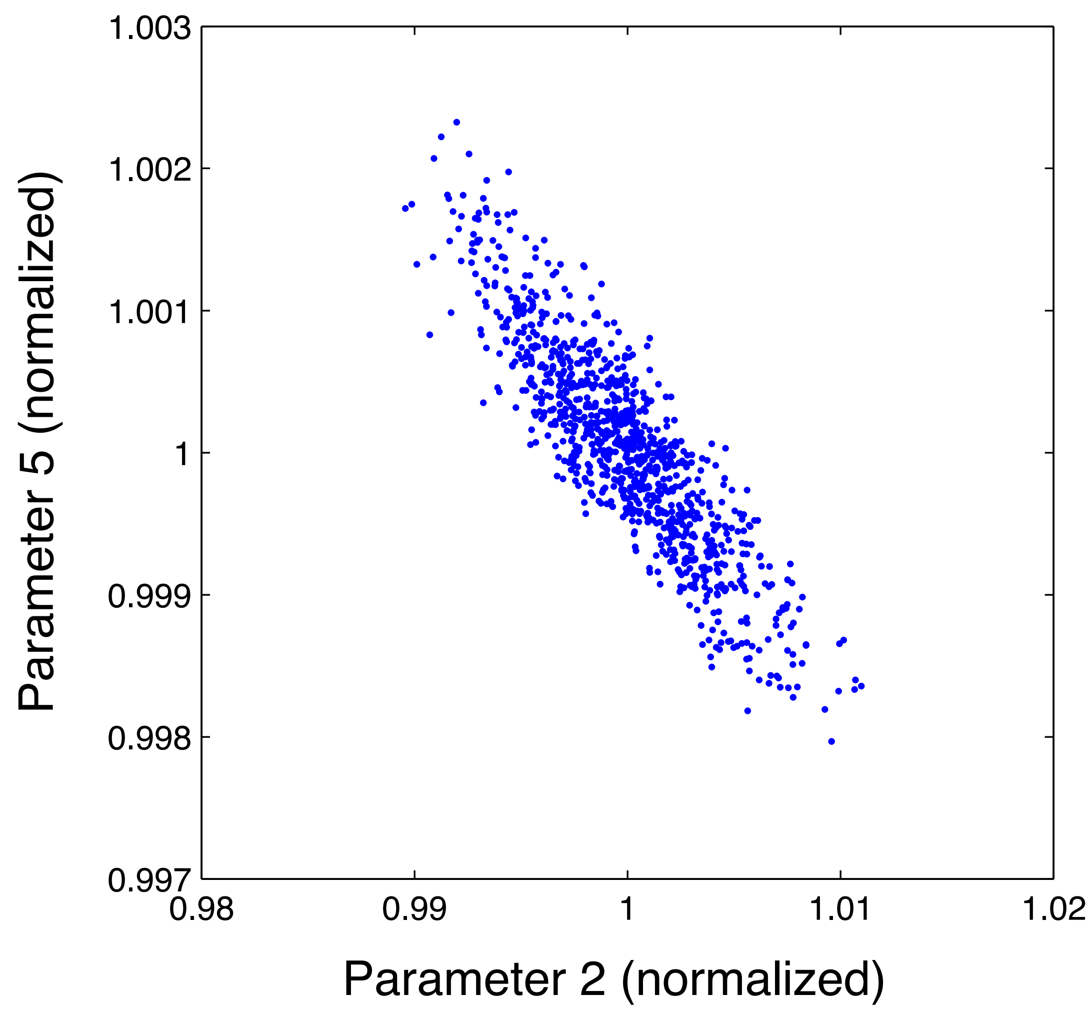
Hongyu Miao, Carrie Dykes, Lisa M. Demeter, James Cavenaugh, Sung Yong Park, Alan S. Perelson, and Hulin Wu. Modeling and Estimation of Kinetic Parameters and Replicative Fitness of HIV-1 from Flow-Cytometry-Based Growth Competition Experiments. Bulletin of Mathematical Biology, 70:1749– 1771, 2008.

Example 1 HIV kinetics model of Miao et al.

Conventional Algorithm

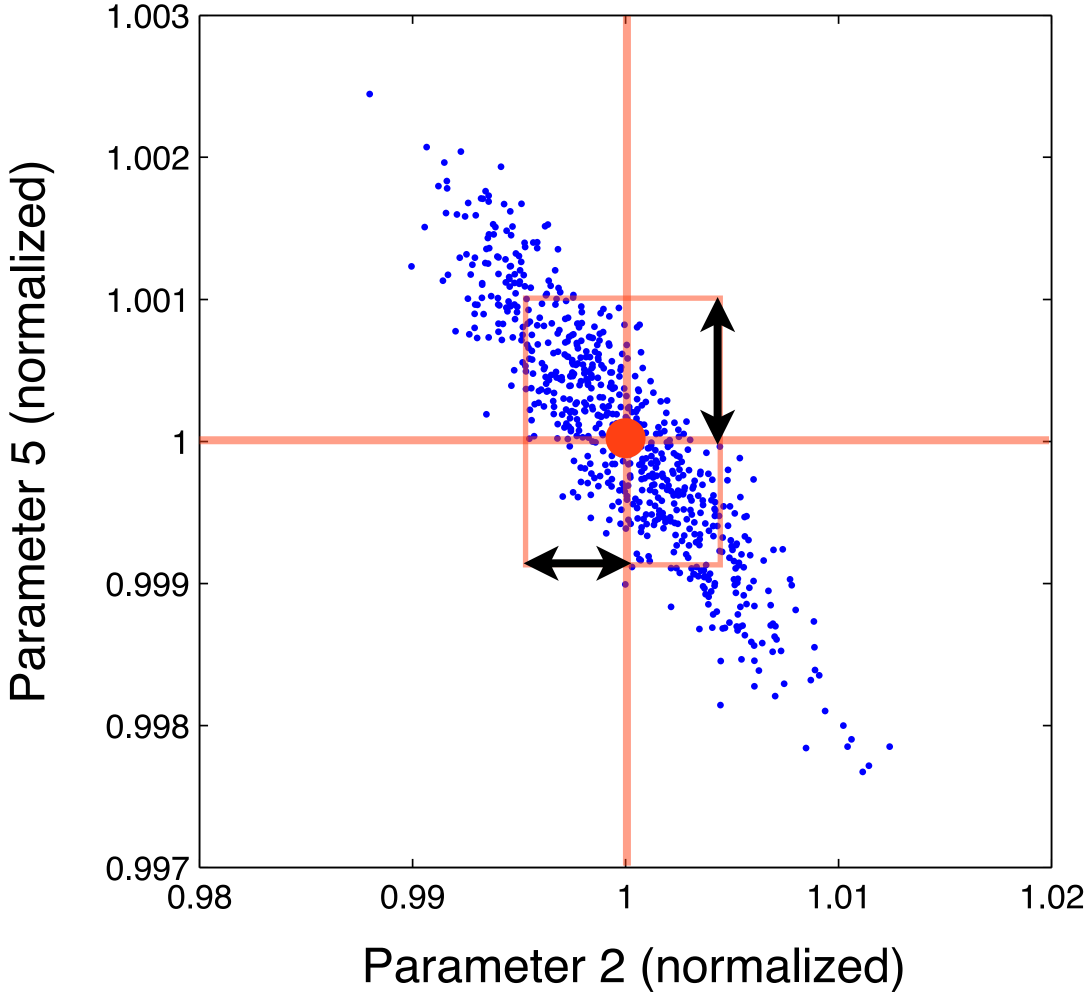


New Algorithm



Example 1 HIV kinetics model of Miao et al.

Average **R**elative Parameter **E**stimation Error

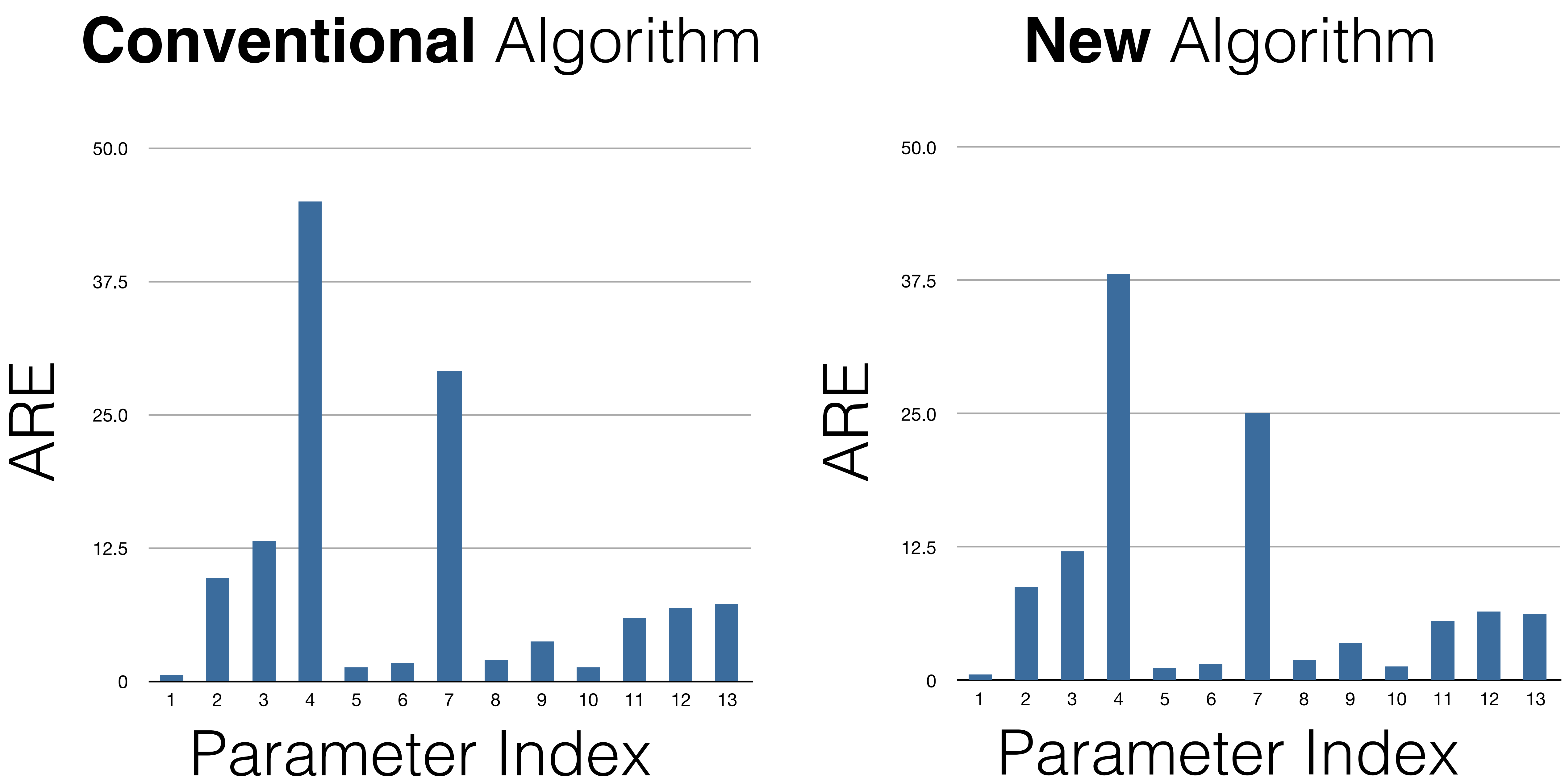


ARE of the i th parameter

$$\frac{1}{N}\sum_{j=1}^{N}\frac{x_{ij}-x_i^*}{x_i^*}\times 100\%$$

Hongyu Miao, Xiaohua Xia, Alan S. Perelson, and Hulin Wu.
On Identifiability of Nonlinear ODE Models and Applications in Viral Dynamics. SIAM REVIEW, 53(1):3–39, 2011.

Example 1 HIV kinetics model of Miao et al.



Example 1 HIV kinetics model of Miao et al.

Conventional Algorithm		**New** Algorithm
	Number of simulations	
	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 1 HIV kinetics model of Miao et al.

Conventional Algorithm		**New** Algorithm
131,508	Number of simulations	
	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 1 HIV kinetics model of Miao et al.

Conventional Algorithm		**New** Algorithm
131,508	Number of simulations	11,280
	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 1 HIV kinetics model of Miao et al.

Conventional Algorithm		**New** Algorithm
131,508	Number of simulations	11,280
2 hours	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 1 HIV kinetics model of Miao et al.

Conventional Algorithm		**New** Algorithm
131,508	Number of simulations	11,280
2 hours	CPU time	**10 min**

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 2 Benchmark Problem of Moles et al.

Example 2 Benchmark Problem of Moles et al.

$$
\begin{aligned}
\frac{du_1}{dt} &= x_1/(1 + (p/x_2)^{x_3} + (x_4/s)^{x_5}) - x_6 u_1 \\
\frac{du_2}{dt} &= x_7/(1 + (p/x_8)^{x_9} + (x_{10}/u_7)^{x_{11}}) - x_{12} u_2 \\
\frac{du_3}{dt} &= x_{13}/(1 + (p/x_{14})^{x_{15}} + (x_{16}/u_8)^{x_{17}}) - x_{18} u_3 \\
\frac{du_4}{dt} &= x_{19}/(x_{20} + u_1) u_1 - x_{21} u_4 \\
\frac{du_5}{dt} &= x_{22}/(x_{23} + u_2) u_2 - x_{24} u_5 \\
\frac{du_6}{dt} &= x_{25}/(x_{26} + u_3) u_3 - x_{27} u_6 \\
\frac{du_7}{dt} &= x_{28}/x_{29} u_4 (s - u_7)/(1 + s/x_{29} - u_7/x_{30}) \\
&\quad - x_{31}/x_{32} u_5 (u_7 - u_8)/(1 + u_7/x_{32} + u_8/x_{33}) \\
\frac{du_8}{dt} &= x_{31}/x_{32} u_5 (u_7 - u_8)/(1 + u_7/x_{32} + u_8/x_{33}) \\
&\quad - x_{34}/x_{35} u_6 (u_8 - p)/(1 + u_8/x_{35} + p/x_{36})
\end{aligned}
$$

Carmen G. Moles, Pedro Mendes, and Julio R. Banga. Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. Genome Research, 13:2467–2474, 2003.

Example 2 Benchmark Problem of Moles et al.

$$\frac{du_1}{dt} = x_1/(1 + (p/x_2)^{x_3} + (x_4/s)^{x_5}) - x_6 u_1$$

$$\frac{du_2}{dt} = x_7/(1 + (p/x_8)^{x_9} + (x_{10}/u_7)^{x_{11}}) - x_{12} u_2$$

$$\frac{du_3}{dt} = x_{13}/(1 + (p/x_{14})^{x_{15}} + (x_{16}/u_8)^{x_{17}}) - x_{18} u_3$$

$$\frac{du_4}{dt} = x_{19}/(x_{20} + u_1)u_1 - x_{21} u_4$$

$$\frac{du_5}{dt} = x_{22}/(x_{23} + u_2)u_2 - x_{24} u_5$$

$$\frac{du_6}{dt} = x_{25}/(x_{26} + u_3)u_3 - x_{27} u_6$$

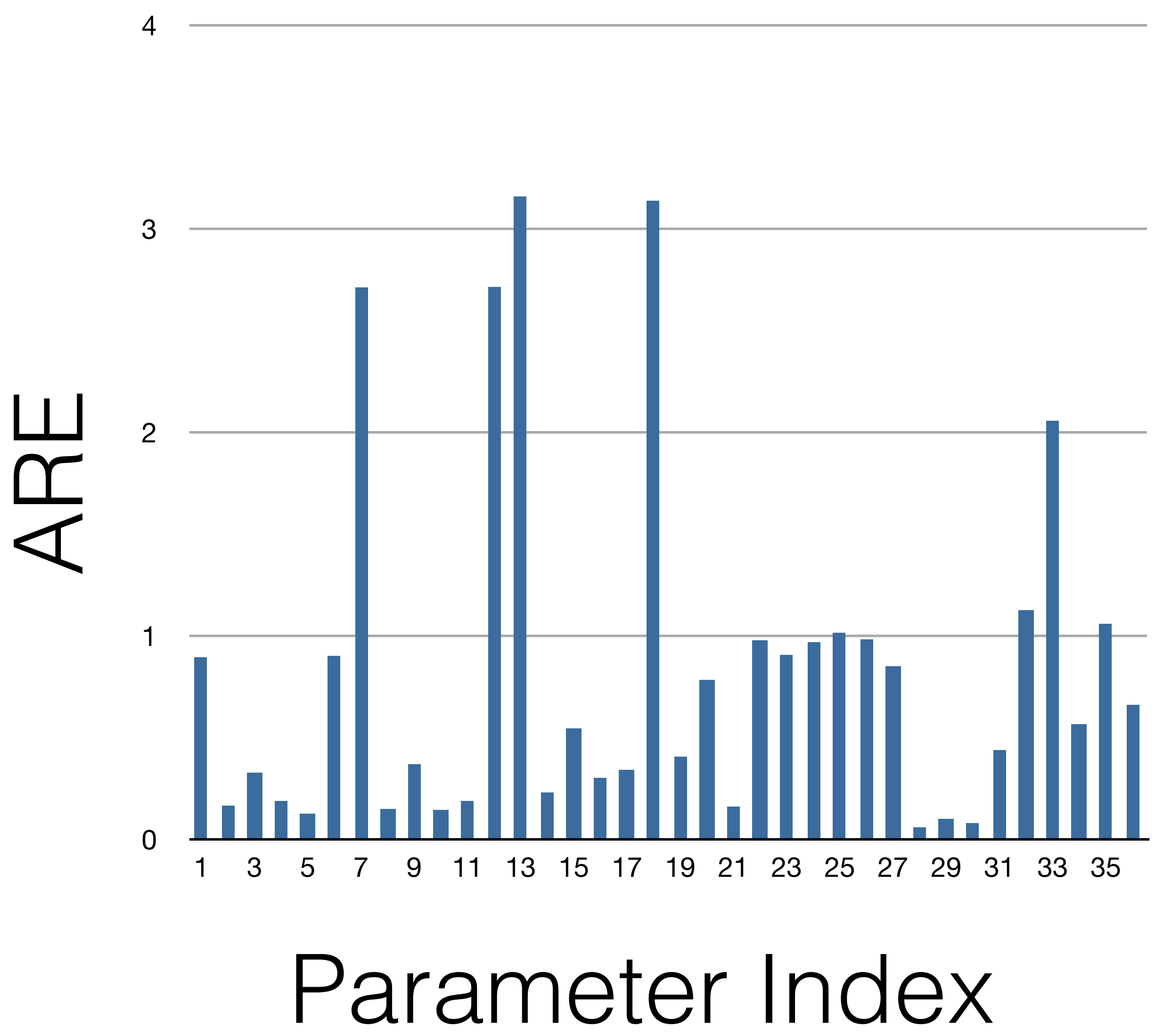
$$\frac{du_7}{dt} = x_{28}/x_{29} u_4 (s - u_7)/(1 + s/x_{29} - u_7/x_{30}) - x_{31}/x_{32} u_5 (u_7 - u_8)/(1 + u_7/x_{32} + u_8/x_{33})$$

$$\frac{du_8}{dt} = x_{31}/x_{32} u_5 (u_7 - u_8)/(1 + u_7/x_{32} + u_8/x_{33}) - x_{34}/x_{35} u_6 (u_8 - p)/(1 + u_8/x_{35} + p/x_{36})$$

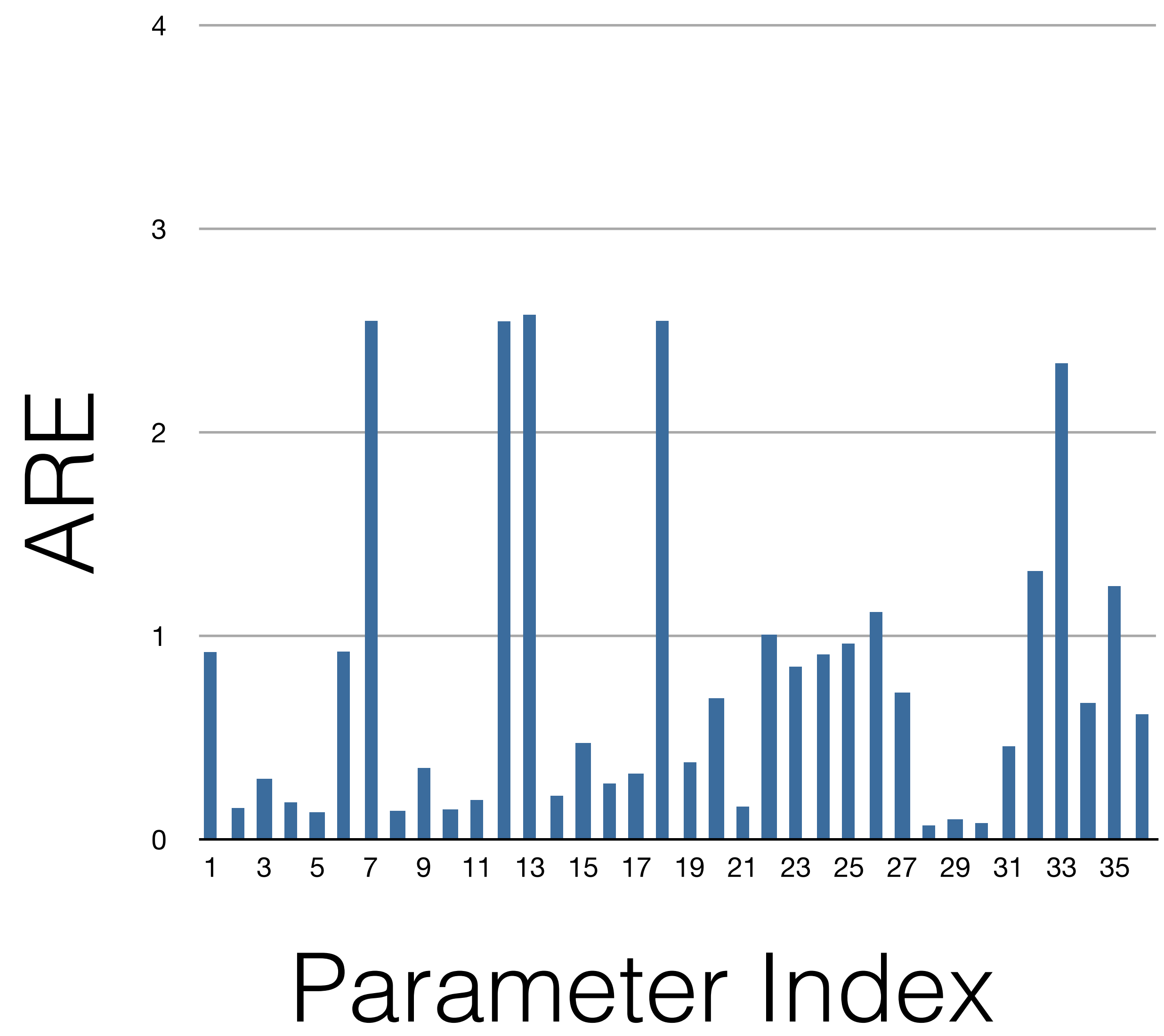
Carmen G. Moles, Pedro Mendes, and Julio R. Banga. Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. Genome Research, 13:2467–2474, 2003.

Example 2 Benchmark Problem of Moles et al.

Conventional Algorithm



New Algorithm



Example 2 Benchmark Problem of Moles et al.

Conventional Algorithm		**New** Algorithm
	Number of simulations	
	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 2 Benchmark Problem of Moles et al.

Conventional Algorithm		**New** Algorithm
174,586	Number of simulations	
	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 2 Benchmark Problem of Moles et al.

Conventional Algorithm		**New** Algorithm
174,586	Number of simulations	3,167
	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 2 Benchmark Problem of Moles et al.

Conventional Algorithm		**New** Algorithm
174,586	Number of simulations	3,167
3 days	CPU time	

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Example 2 Benchmark Problem of Moles et al.

Conventional Algorithm		**New** Algorithm
174,586	Number of simulations	3,167
3 days	CPU time	**1.5 hours**

CPU time measured on Intel Xeon 2.66GHz Processor on Mac Pro

Conclusion

“Practical” Parameter Identifiability Analysis

"**Practical**" Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method		

"Practical" Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method	Robust ✔	

“Practical” Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method	Robust ✓	Impractically Slow... ✗

“**Practical**” Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method	Robust ✔	Impractically Slow... ✘
Monte Carlo / Bootstrap method + **New Algorithm**		

“**Practical**” Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method	Robust ✓	Impractically Slow... ✗
Monte Carlo / Bootstrap method + **New Algorithm**	Robust ✓	

“**Practical**” Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method	Robust ✓	Impractically Slow... ✗
Monte Carlo / Bootstrap method + **New Algorithm**	Robust ✓	Practically acceptable computational cost ✓

Practical Parameter Identifiability Analysis

	Nonlinear Model	Computational Cost
Monte Carlo / Bootstrap method	Robust ✓	Impractically Slow… ✗
Monte Carlo / Bootstrap method + **New Algorithm**	Robust ✓	Practically acceptable computational cost ✓

Research funded and supported by

Natural Science and Engineering Research Council of Canada

Department of Applied Mathematics, University of Waterloo