ISSN 1346-5597

National Institute of Informatics

NII Technical Report

Cluster Newton Method for Sampling Multiple Solutions of an Underdetermined Inverse Problem: Parameter Identification for Pharmacokinetics

Yasunori Aoki, Ken Hayami, Hans De Sterck
and Akihiko Konagaya

NII-2011-002E
Aug. 2011

URL: http://www.nii.ac.jp/ Tokyo 101-8430, JAPAN +81-3-4212-2000 FAX: +81-3-3556-1916

Cluster Newton Method for Sampling Multiple Solutions of an Underdetermined Inverse Problem: Parameter Identification for Pharmacokinetics

Yasunori Aoki[1], Ken Hayami[2], Hans De Sterck[3] and Akihiko Konagaya[4]

[1,3]University of Waterloo
200 University Ave. West, Waterloo, Ontario, Canada, N2L 3G1
[2] National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan, 101-8430
[4] Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama, Japan, 226-8503

E-mail: [1] yaoki@uwaterloo.ca, [2] hayami@nii.ac.jp, [3] hdesterck@uwaterloo.ca, [4] kona@dis.titech.ac.jp

Abstract. In daily life, if we lack information for making a decision, we often consider multiple possibilities. However, if we are given an underdetermined inverse problem, we often add mathematically convenient constraints and consider only one of many possible solutions even though it may be beneficial for the application to consider multiple solutions. We propose an algorithm for simultaneously finding multiple solutions of an underdetermined inverse problem that are suitably distributed, guided by *a priori* information on which part of the solution manifold is of interest. Through numerical experiments, we show that our algorithm is a fast, accurate and robust solution method, especially applicable to ODE coefficient identification problems. We give an example of applying this algorithm to a parameter identification problem in pharmacokinetics.

1. Introduction

Since the information we can obtain clinically from a live patient going through treatment is often much less than the complexity of the internal activity in a patient's body, underdetermined inverse problems appear often in the field of mathematical medicine.

In fact, our interest in the underdetermined inverse problem was initiated by the parameter identification problem of a pharmacokinetics model for the anti-cancer drug CPT-11 (also known as Irinotecan). Konagaya has proposed a framework called "virtual patient population convergence" [9], whose essential idea is to estimate the parameters of the whole body pharmacokinetics model based on the clinically observed data from patients. The essential difference of this framework from other parameter identification approaches is that instead of finding a single set of parameters that is suitable for the pharmacokinetics model to reproduce the clinically observed data, its aim is to find multiple sets of such parameters. The reason for finding these multiple sets of parameters is to take into account multiple relevant possibilities of the drug kinetics in the patient's body.

For underdetermined inverse problems, it is customary to add extra constraints to make the solution unique (e.g., the solution closest to some initial point). If only one of many solutions is considered, it is often hard to conclude if the characteristics of that solution are representative of the other solutions or if they are a consequence of the choice of the extra constraints. Hence we wish to sample many solutions from the solution manifold of the underdetermined inverse problem. However, for a problem as complicated as a pharmacokinetics model aiming to model the whole body drug kinetics, even to find one set of model parameters that fits a clinical observation can be time consuming. Thus, trying to find multiple sets of model parameters one set by one set can take computational time that is unreaslistic for practical use.

Motivated by this problem, we have constructed an algorithm to simultaneously find multiple solutions of an underdetermined inverse problem, in a new way that is significantly more robust and efficient than solving many separate inverse problems with different initial iterates. Our iterative scheme starts with a set (cluster) of initial points and by computing the forward problem at each point and fitting a hyperplane to the solution values in the sense of least squares, we obtain a linear approximation of the function that corresponds to the forward problem. This linear approximation aims to approximate the function in the broad domain covered by the cluster of initial points. Then by using this linear approximation, we estimate the solution of the inverse problem and move the cluster of points accordingly. By repeating this iteratively, the cluster of points becomes stationary and the points are close to being solutions of the inverse problem. In a next step, if the desired accuracy has not been met, we use Broyden's method to improve the accuracy by moving each point individually using different approximated Jacobians and achieve the desired accuracy. Throughout this paper, we shall refer to this method we have constructed as the Cluster Newton method

(CN method).

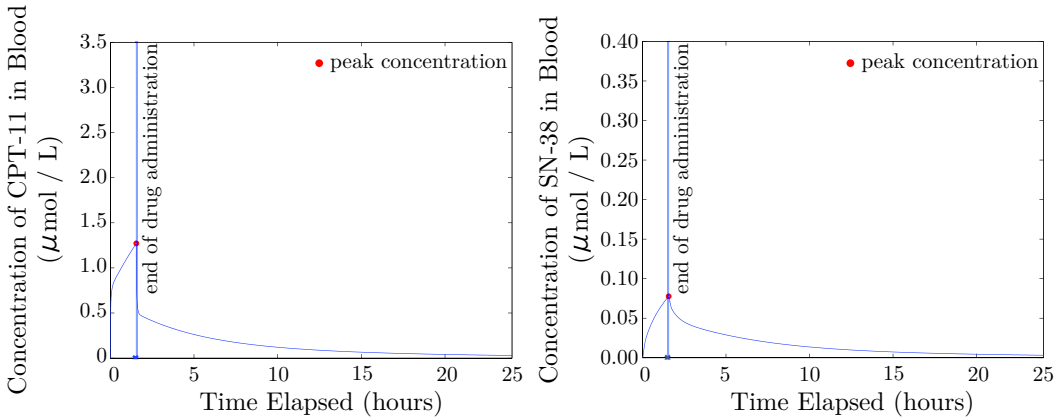
Through numerical experiments, we found that the Cluster Newton method requires far less function evaluations than applying the Levenberg-Marquardt method multiple times for various initial iterates. The Cluster Newton method is similar to Newton's method in the sense that it iteratively improves the approximation by approximating the forward problem by a linear function and solves the inverse of the linear approximation. Aside from moving a cluster of points instead of a single point, the Cluster Newton method differs from the traditional Newton's method in the sense that instead of approximating the Jacobian locally, we estimate it more globally in the domain covered by the cluster of points. The global approximation of the Jacobian acts as a regularization and we observed that the Cluster Newton algorithm is robust against local optima caused by small-scale "roughness" of the function (e.g., caused by the inaccuracy of solving the forward problem), compared to a method like Levenberg-Marquardt. Such roughness appears in the coefficient identification problem of a system of ODEs when the ODE is solved numerically.

1.1. Motivation for finding multiple solutions of the underdetermined inverse problem

Our motivation to find multiple solutions instead of a single solution of the underdetermined inverse problem came from the parameter identification problem of Arikuma et al.'s pharmacokinetics model for the anti-cancer drug CPT-11 [1]. Figure 1 shows the concentrations of CPT-11 and SN-38 (a metabolite of CPT-11) in blood simulated by the pharmacokinetics model using a set of model parameters found by the Levenberg-Marquardt method based on clinically measured data. The Levenberg-Marquardt method iteratively finds a solution of the underdetermined inverse problem near the initial iterate. We have chosen the initial iterate as the typical values of the model parameters listed in Arikuma et al. [1]. From Figure 1, we observe that the peak concentration happens at time $t = 1.5$ (we denote this time as T_{max}) for both CPT-11 and SN-38. Also, observe that the peak concentration is around 1.3 $\mu mol/L$ for CPT-11 and 0.08 $\mu mol/L$ for SN-38 (We denote peak concentrations by C_{max}.)

In order to investigate further whether these obtained values are specific to the choice of the initial iterate or similar for most of the solutions of this underdetermined inverse problem, we obtain multiple solutions (multiple sets of model parameters) using the Levenberg-Marquardt method with multiple different initial iterates. Figure 2 shows the concentrations of CPT-11 and SN-38 in blood simulated by the pharmacokinetics model using 1,000 sets of model parameters found by the Levenberg-Marquardt method with 1,000 different initial iterates.

We observe from Figure 2 that only the observation that $T_{max} = 1.5$ for CPT-11 seems independent of the choice of the initial iterate and may be a common feature among the solutions in the solution manifold of this underdetermined inverse problem. Other values (e.g., C_{max} for both CPT-11 and SN-38 and T_{max} for SN-38) are heavily dependent on the choice of initial iterate. That is to say, these values cannot be



(a) Concentration of the anti-cancer drug CPT-11.

(b) Concentration of the metabolite SN-38.

Figure 1. Drug and Metabolite concentration simulation based on a single set of model parameters found by the Levenberg-Marquardt Method.

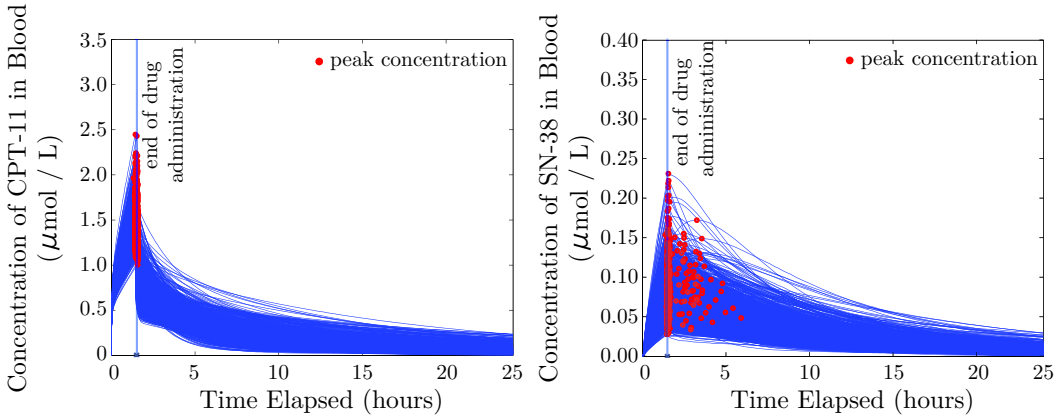
determined due to the underdetermined nature of the inverse problem. Although these values cannot be determined precisely, information on the range of C_{max} and T_{max} as obtained from the multiple solutions shown in Figure 2 is still of significant value in the context of the application problem.

Although it is only realistic in the clinical experiment setting and not necessarily reliable (e.g., 16 blood samples are required from a patient in a day), the values of C_{max} and T_{max} can be measured. Slatter et al. have obtained for 8 patients that C_{max} for CPT-11 is on average 2.26 $\mu mol/L$ (with standard deviation of 0.21), C_{max} of SN-38 is on average 0.04 $\mu mol/L$ (with standard deviation of 0.017), T_{max} of CPT-11 is 1.5 hours (with zero standard deviation) and T_{max} of SN-38 is on average 2.3 hours (with standard deviation of 1.0). All of the measured values except T_{max} of CPT-11 are different from what can be predicted from the single solution of Figure 1. However, Table 1 shows that these measured values for a small set of patients are within the range of the values of C_{max} and T_{max} obtained by solving for multiple solutions of the underdetermined inverse problem as shown in Figure 2. While the numbers in Table 1 indicate that the pharmacokinetics model is not perfect yet, this example does show that obtaining multiple solutions of the underdetermined inverse problem is useful for determining the general characteristics of the solutions in the solution manifold of the underdetermined inverse problem.

The MATLAB implementation of the Levenberg-Marquardt method we used took 3.3 minutes to compute a single set of model parameters used to plot Figure 1 using one core of an Intel Xeon X7350 3GHz processor. It took about 7 hours with a server machine with two quad-core Intel Xeon X7350 3GHz processors to find the 1,000 model parameters used to produce Figure 2. We wish to develop an algorithm so that we can find such sets of parameters with significantly less computational cost.

Table 1. Summary of C_{max} and T_{max} predicted from Figures 1 and 2, and clinically measured values.

	Predicted value from Figure 1	Range from Figure 2	Measured value in [11] (ave ± sd)
C_{max} of CPT-11 ($\mu mol/L$)	1.3	1.0∼2.5	2.26 ±0.21
C_{max} of SN-38 ($\mu mol/L$)	0.08	0.02∼0.23	0.04 ± 0.017
T_{max} of CPT-11 (hours)	1.5	1.5	1.5 ± 0
T_{max} of SN-38 (hours)	1.5	1.5∼6	2.3 ± 1.0



(a) Concentration of the anti-cancer drug CPT-11.

(b) Concentration of the metabolite SN-38.

Figure 2. 1,000 model parameter sets found by multiple application of the Levenberg-Marquardt Method.

1.2. Statement of the Problem

In this paper, we consider the following underdetermined inverse problem, (see Appendix A for an explanation of the matrix and vector notation used in this paper):
Find $\boldsymbol{x}$ such that

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}^* \,, \tag{1}$$

where $\boldsymbol{y}^*$ is a given constant vector in $\mathbb{R}^n$, $\boldsymbol{f}$ is a vector function from $\mathcal{X} \subset \mathbb{R}^m$ to $\mathbb{R}^n$ with $m > n$, and the solution of (1) is not unique. We assume this inverse problem has the following properties:

- The evaluation of $\boldsymbol{f}$ (solving the forward problem) is computationally expensive. Thus, we would like to minimize the number of function evaluations.
- The Jacobian of $\boldsymbol{f}$ is not explicitly known.

We now denote a subset $\mathcal{X}^*_\epsilon$ of $\mathcal{X}$ to be the set containing all the values of $\mathcal{X}$ which approximately satisfy (1) with maximum norm relative residual less than ϵ, i.e.,

$$\mathcal{X}^*_\epsilon := \{\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^m \,:\, \max_{i=1,\ldots,n} |\left(f_i(\boldsymbol{x}) - y^*_i\right)/y^*_i| < \epsilon\} \,. \tag{2}$$

We note that in most cases $\mathcal{X}^*_\epsilon$ is an infinite set and often is an unbounded set. We are only interested in a part of this set $\mathcal{X}^*_\epsilon$, namely the part that is relevant in the context

of the problem, and corresponds to a range of reasonable physiological parameters. We assume that we know the following regarding the relevant values of $\boldsymbol{x} \in \mathcal{X}$:

- a typical value of $\boldsymbol{x}$ is known (we denote this value as $\hat{\boldsymbol{x}}$)
- the typical relative range $\boldsymbol{v}$ (in the sense defined in (3) below) of the value $\boldsymbol{x}$ is known.

So we roughly know that the solutions we are interested in are somewhere near a "box" $\mathcal{X}^0$ defined as follows:

$$\mathcal{X}^0 := \{\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^m \,:\, \max_{i=1,\ldots,n} |\,(x_i - \hat{x}_i)\,/(\hat{x}_i\, v_i)| < 1\}\,. \tag{3}$$

We seek solutions that are close to this initial box $\mathcal{X}^0$. We now state the problem of interest mathematically: find a set of l column vectors $\{\boldsymbol{x}_{\cdot j}\}_{j=1}^{l}$ so that

$$\boldsymbol{x}_{\cdot j} \in \mathcal{X}^*_\epsilon \qquad \text{and} \qquad ||\boldsymbol{x}_{\cdot j} - \boldsymbol{x}^{(0)}_{\cdot j}||_2 \approx \min_{\boldsymbol{x} \in \mathcal{X}^*_\epsilon} ||\boldsymbol{x} - \boldsymbol{x}^{(0)}_{\cdot j}||_2 \tag{4}$$

for $j = 1, 2, ..., l$, where $\boldsymbol{x}^{(0)}_{\cdot j}$ is a randomly chosen point in the box $\mathcal{X}^0$.

2. Simple model problem (Example 1)

Before we attempt to solve the parameter identification problem of the pharmacokinetics model, we will explain our algorithm by a simple example that is easy to visualize.

2.1. Model problem description

Our model problem is as follows: find a set of l points in $\mathbb{R}^2$ near a box $\mathcal{X}^0$, such that

$$f(\boldsymbol{x}) = y^*\,, \tag{5}$$

where

$$f(\boldsymbol{x}) = (x_1^2 + x_2^2) + \sin(10000x_1) \cdot \sin(10000x_2)/100\,, \tag{6}$$

$$y^* \quad = 100\,, \tag{7}$$

$$\mathcal{X}^0 \quad = \{\boldsymbol{x} \in \mathbb{R}^2 \,:\, \max_{i=1,2} |\,(x_i - 2.5)\,/2.5| < 1\}\,. \tag{8}$$

The function is basically a paraboloid with a small-amplitude wildly oscillatory perturbation. As depicted in Figure 3, the solution manifold of this inverse problem $\mathcal{X}^*$ is approximately a circle of radius 10 centred at the origin in the x_1-x_2 plane. Thus, we aim to find the points on this curve $\mathcal{X}^*$ near the box $\mathcal{X}^0$. The perturbation mimics 'roughness' that can be found in many realistic high-dimensional applications. For example, as we will illustrate in the following section, when the forward problem involves numerical solution of a system of ODEs, a similar kind of 'roughness' can be observed for the function. The initial box $\mathcal{X}^0$ may signify some *a priori* knowledge about where physically relevant solutions are expected. We choose $l = 100$ in the numerical examples below.

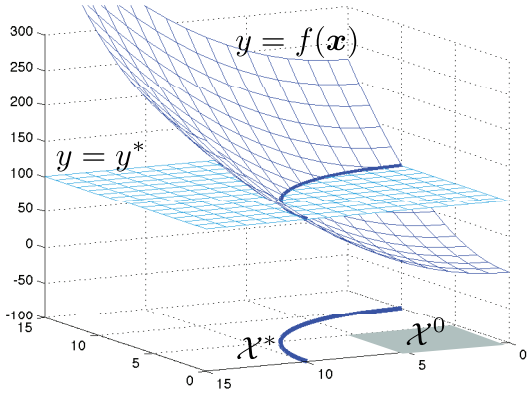
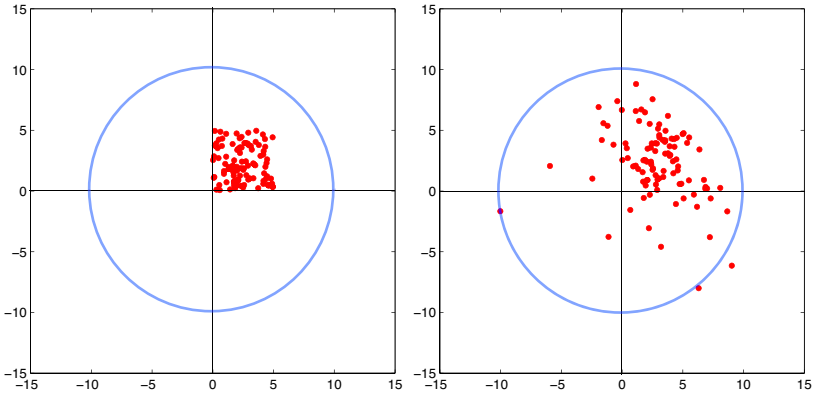


Figure 3. Example 1: the function f and the solution manifold $\mathcal{X}^*$ (in the first quadrant).



(a) Initial points in $\mathcal{X}^0$.

(b) Final Points obtained by the Levenberg-Marquardt method.

Figure 4. Example 1: an attempt with the Levenberg-Marquardt method.

2.2. Levenberg-Marquardt Method

We first discuss how the well-known Levenberg-Marquardt method (see e.g. [2]) performs when applied to this problem. We create l random points in $\mathcal{X}^0$ and then apply the Levenberg-Marquardt method using each random point as an initial point. We have used the Levenberg-Marquardt implementation in the MATLAB optimization toolbox (version 2010b) for our numerical experiment. We observe that the algorithm terminates with the error "Algorithm appears to be converging to a point that is not a root" for all initial points we tried. As can be seen in Figure 4, we fail to find points close to the solution manifold $\mathcal{X}^*$.

2.3. Cluster Newton (CN) Method

We now introduce our algorithm, which finds multiple points on the level curve and at the same time avoids the local optima created by the wildly oscillatory perturbation term of the function $f(\boldsymbol{x})$. Our algorithm described in detail below first approximates the function $f(x_1, x_2)$ linearly by a least squares fitting of a plane to $(\boldsymbol{x}, f(\boldsymbol{x}))$ for multiple points $\boldsymbol{x}$ in $\mathbb{R}^2$. Least squares fitting of a plane, instead of local gradient approximation in each point, acts as a regularization and helps to avoid convergence to the local optima created by the oscillatory perturbation. Then, using this linear approximation, we can obtain the next cluster of points. We can iteratively apply this scheme until the cluster of points becomes stationary. This constitutes the first stage of our algorithm. In a second stage, if these iterations do not achieve the desired accuracy, we use Broyden's method to improve the accuracy. The convergence of Broyden's method is fast and reliable when we use the cluster of points found by the first stage as initial points. Also, by using the slope of the linear approximation obtained from the first stage of the algorithm as the initial guess of the gradient, we avoid the need for approximating the gradient locally.

2.3.1. Algorithm: Cluster Newton method We present the algorithm we have developed to simultaneously find multiple solutions of underdetermined inverse problems, applied to the problem from Section 2.1. We first give a simplified version of the newly proposed Cluster Newton method for this simple model problem and give a complete description of the full algorithm in a later section. To understand the first stage of the algorithm visually, we suggest the reader to refer to Figure 5 while reading the following description of the algorithm.

Stage 1

1: Set up the initial points and the target values.

1-1: Randomly choose initial points $\{\boldsymbol{x}_{\cdot j}^{(0)}\}_{j=1}^{l}$ in the box $\mathcal{X}^0$. We put the initial points in a $2 \times l$ matrix $X^{(0)}$, where each column corresponds to a point $\boldsymbol{x}_{\cdot j}^{(0)}$ in $\mathbb{R}^2$.

1-2: In order to maintain well-posedness of the least squares problem in step 2-2 below, generate randomly perturbed target values $\{y_j^*\}_{j=1}^{l}$ near y^*. We choose each value y_j^* so that

$$\left| \frac{y_j^* - y^*}{y^*} \right| < \eta \,. \tag{9}$$

For our numerical experiments we choose $\eta = 0.1$. That is to say, the target relative residual for Stage 1 is $\pm 10\%$.

Let

$$\boldsymbol{y}^* = (y_1^*, y_2^*, ..., y_l^*). \tag{10}$$

2: For $k = 0, 1, 2, ..., K_1$

2-1: Evaluate f for each point in $X^{(k)}$:

$$\boldsymbol{y}^{(k)} = f(X^{(k)})\,, \tag{11}$$

that is

$$y_j^{(k)} = f(\boldsymbol{x}_{\cdot j}^{(k)}) \qquad \text{for } j = 1, 2, ..., l \tag{12}$$

$$\boldsymbol{y}^{(k)} = (y_1^{(k)}, y_2^{(k)}, ..., y_l^{(k)}) \tag{13}$$

$$X^{(k)} = [\boldsymbol{x}_{\cdot 1}^{(k)}, \boldsymbol{x}_{\cdot 2}^{(k)}, ..., \boldsymbol{x}_{\cdot l}^{(k)}] \tag{14}$$

2-2: Construct a linear approximation of f by fitting a plane

$$\boldsymbol{a}^{(k)\mathrm{T}}\boldsymbol{x} + y_o^{(k)}\,, \tag{15}$$

to the points

$$\{(\boldsymbol{x}_{\cdot j}^{(k)}, y_j^{(k)})\}_{j=1}^{l}, \tag{16}$$

that is to say find a vector $\boldsymbol{a}^{(k)}$ and a scalar value $y_o^{(k)}$ such that,

$$f(\boldsymbol{x}) \approx \boldsymbol{a}^{(k)\mathrm{T}}\boldsymbol{x} + y_o^{(k)}\,. \tag{17}$$

The slope vector $\boldsymbol{a}^{(k)}$ and the shift constant $y_o^{(k)}$ can be found as a least squares solution of an overdetermined system of linear equations:

$$\min_{\boldsymbol{a}^{(k)} \in \mathbb{R}^2\,,\, y_o^{(k)} \in \mathbb{R}} ||\boldsymbol{y}^{(k)} - ((\boldsymbol{a}^{(k)\mathrm{T}} X^{(k)})^{\mathrm{T}} + \boldsymbol{y_o}^{(k)})||_2\,, \tag{18}$$

where

$$\boldsymbol{y_o}^{(k)} = (y_o^{(k)}, y_o^{(k)}, ..., y_0^{(k)}) \in \mathbb{R}^{1\times l}\,. \tag{19}$$

2-3: Find an update vector $\boldsymbol{s}_{\cdot j}^{(k)}$ for each point $\boldsymbol{x}_{\cdot j}^{(k)}$ using a linear approximation, i.e., find $\boldsymbol{s}_{\cdot j}^{(k)}$ s.t.

$$y_j^* = \boldsymbol{a}^{(k)\mathrm{T}}(\boldsymbol{x}_{\cdot j}^{(k)} + \boldsymbol{s}_{\cdot j}^{(k)}) + y_o{}^{(k)} \qquad \text{for } j = 1, 2, ..., l\,. \tag{20}$$

The vectors $\boldsymbol{x}_{\cdot j}^{(k)} + \boldsymbol{s}_{\cdot j}^{(k)}$ satisfying (20) can be viewed as the projection of the intersection of planes $y = y_j^*$ and $y = \boldsymbol{a}^{(k)\mathrm{T}}\boldsymbol{x} + y_o{}^{(k)}$ on to the x_1-x_2 plane, which is a line. Thus, we cannot uniquely determine $\boldsymbol{s}_{\cdot j}^{(k)}$. Hence, we choose the shortest vector $\boldsymbol{s}_{\cdot j}^{(k)}$. The vectors $\{\boldsymbol{s}_{\cdot j}^{(k)}\}_{j=1}^{l}$ written as a matrix $S^{(k)}$ are the minimum norm solution of the underdetermined system of linear equations:

$$\min_{S^{(k)} \in \mathbb{R}^{2\times l}} ||S^{(k)}||_F \qquad \text{s.t.} \qquad \boldsymbol{y}^* = (\boldsymbol{a}^{(k)\mathrm{T}}(X^{(k)} + S^{(k)}))^{\mathrm{T}} + \boldsymbol{y_o}^{(k)}\,. \tag{21}$$

2-4: Find new points approximating the solution manifold $\mathcal{X}^*$ by updating $X^{(k)}$, i.e.,

$$X^{(k+1)} = X^{(k)} + S^{(k)}\,. \tag{22}$$

(20) and (22) give $y_j^* = \boldsymbol{a}^{(k)\mathrm{T}}\boldsymbol{x}_{\cdot j}^{(k+1)} + y_o{}^{(k)}$. Hence, if $\{y_j^*\}_{j=1}^{l}$ are chosen to be the same for all j, then vectors $\{\boldsymbol{x}_{\cdot j}^{(k+1)}\}_{j=1}^{l}$ all lie on one line and the least squares problem (18) becomes rank-deficient. However, as the values $\{y_j^*\}_{j=1}^{l}$ were chosen randomly around y^*, we maintain (18) to be full-rank so that we can obtain a linear approximation at the next iteration in Equation (18) without complications.

End for.

Stage 2: Broyden's method

3: Set up the initial gradient approximation using the slope vector obtained in the last iteration of Stage 1 as follows:

$$\boldsymbol{g}_{(j)}^{(K_1+1)} = \boldsymbol{a}^{(K_1)} \qquad \text{for } j = 1, 2, ..., l\,. \tag{23}$$

4: For $k = K_1 + 1, ..., K_2$

4-1: Solve the forward problem for each point in $X^{(k)}$, i.e.,

$$\boldsymbol{y}^{(k)} = f(X^{(k)})\,. \tag{24}$$

4-2: If $k \neq K_1+1$ then update the gradient for each point using Broyden's method as follows (cf. [7]):

$$\boldsymbol{g}_{(j)}^{(k)} = \boldsymbol{g}_{(j)}^{(k-1)} + (y_j^{(k)} - y^*)\frac{\boldsymbol{s}_{\cdot j}^{(k-1)}}{||\boldsymbol{s}_{\cdot j}^{(k-1)}||^2} \qquad \text{for } j = 1, 2, ..., l. \tag{25}$$

4-3: Find the update vectors $\boldsymbol{s}_{\cdot j}^{(k)}$ for all points in $X^{(k)}$ using the approximate gradient, i.e., find $\boldsymbol{s}_{\cdot j}^{(k)}$ given by the minimum norm solution of an underdetermined system of linear equations:

$$\min_{\boldsymbol{s}_{\cdot j}^{(k)} \in \mathbb{R}^2} ||\boldsymbol{s}_{\cdot j}^{(k)}||_2 \qquad \text{s.t.} \qquad y^* - y_j^{(k)} = \boldsymbol{g}_{(j)}^{(k)\mathrm{T}} \boldsymbol{s}_{\cdot j}^{(k)} \tag{26}$$

for $j = 1, 2, ..., l$.

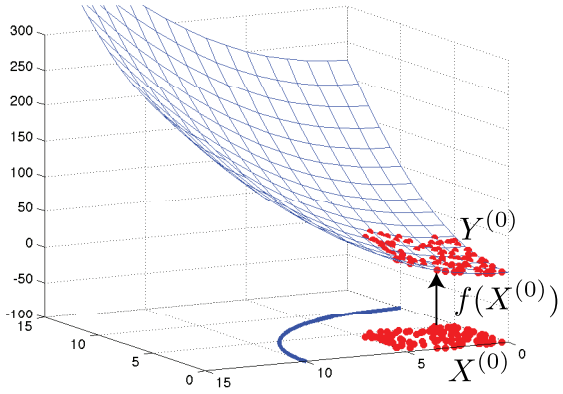
4-4: Update the points approximating the solution manifold $\mathcal{X}^*$:

$$X^{(k+1)} = X^{(k)} + S^{(k)}\,. \tag{27}$$

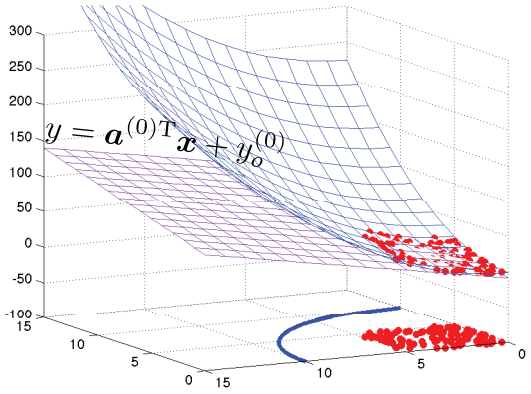
End for.

2.3.2. A few notes on the algorithm The fundamental differences between Stage 1 and Stage 2 is that in Stage 1 we use one linear approximation for all the points; however, in Stage 2 each point has its own gradient approximation. Also note that both Stage 1 and Stage 2 only require one function evaluation per point per iteration. For simplicity in presenting our algorithm, we have fixed the number of iterations for each of the stages (K_1 and K_2). However, one can easily modify the implementation so that the iteration stops once a desired accuracy has been achieved.

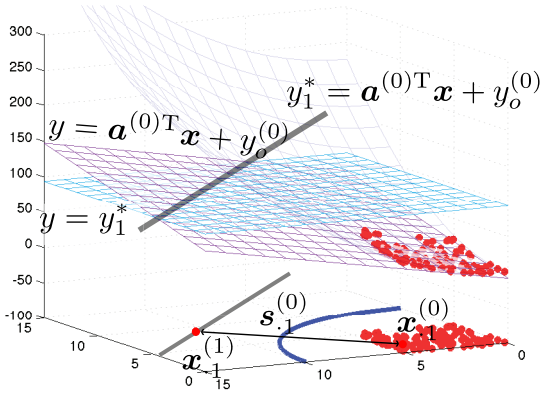
2.3.3. Pictorial description of the solutions found by the Cluster Newton method We choose $K_1 = 5$ and $K_2 = 24$ in the numerical results shown here. As depicted in Figure 6(i), the multiple points found by our algorithm trace part of the solution curve accurately. As can be seen in Figure 6(a), we start with an initial cluster of points uniformly distributed in the domain bounded by $x_1 = 0$, $x_1 = 5$ and $x_2 = 0$, $x_2 = 5$. Figures 6(a)-6(f) show that Stage 1 of the Cluster Newton method moves the points roughly close to the solution manifold $\mathcal{X}^*$. Once the algorithm moves onto Stage 2 when $k = 6$, we see that the points quickly line up with the solution curve as they were already near the solution owing to Stage 1 of our scheme. Recall from Figure 4 that the Levenberg-Marquardt method was not able to find solutions.



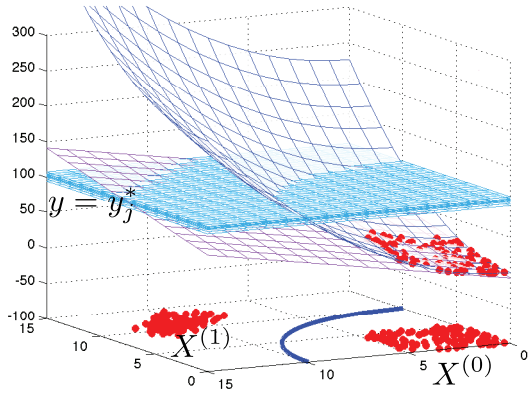
(a) Line 2-1: Evaluate f at each point in $X^{(0)}$ to generate $Y^{(0)}$.



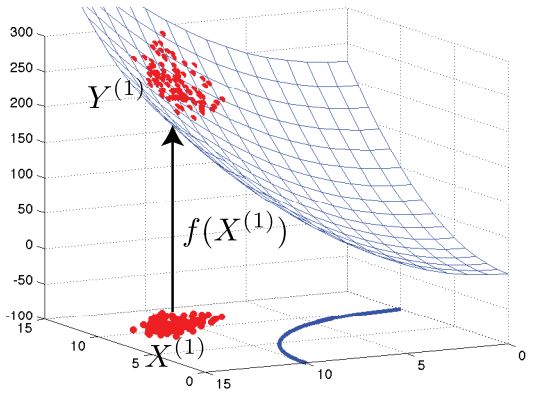
(b) Line 2-2: Construct a linear approximation $y = \boldsymbol{a}^{(0)\mathrm{T}}\boldsymbol{x} + y_o^{(0)}$.



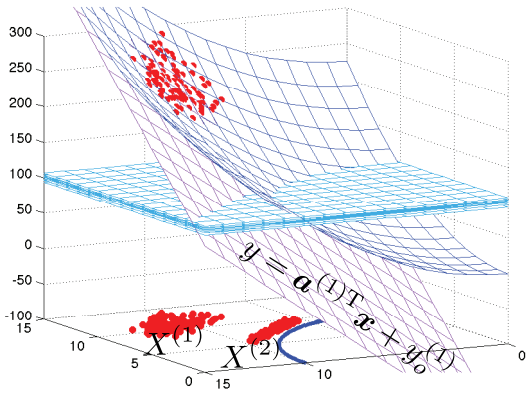
(c) Line 2-3: Find the update vector $\boldsymbol{s}_{\cdot j}$.



(d) Line 2-4: Find new cluster of points $X^{(1)}$ that approximate $\mathcal{X}^*$.

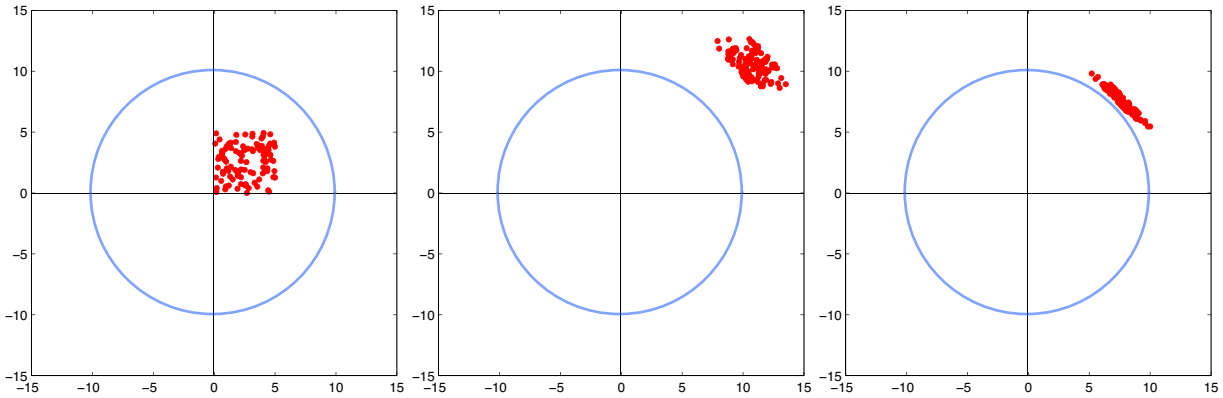


(e) Line 2-1: Evaluate f at each point in $X^{(1)}$ to generate $Y^{(1)}$.

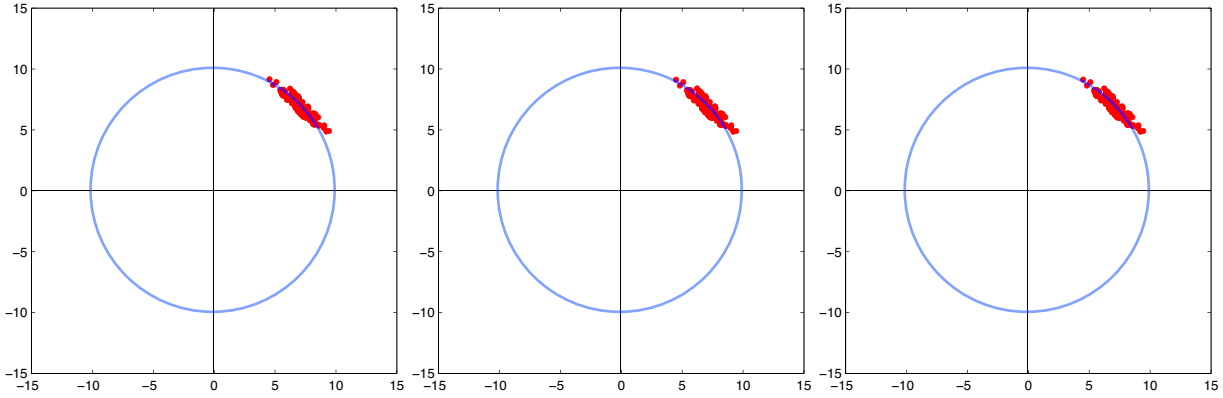


(f) Line 2-2,...,2-4: Obtain $X^{(2)}$

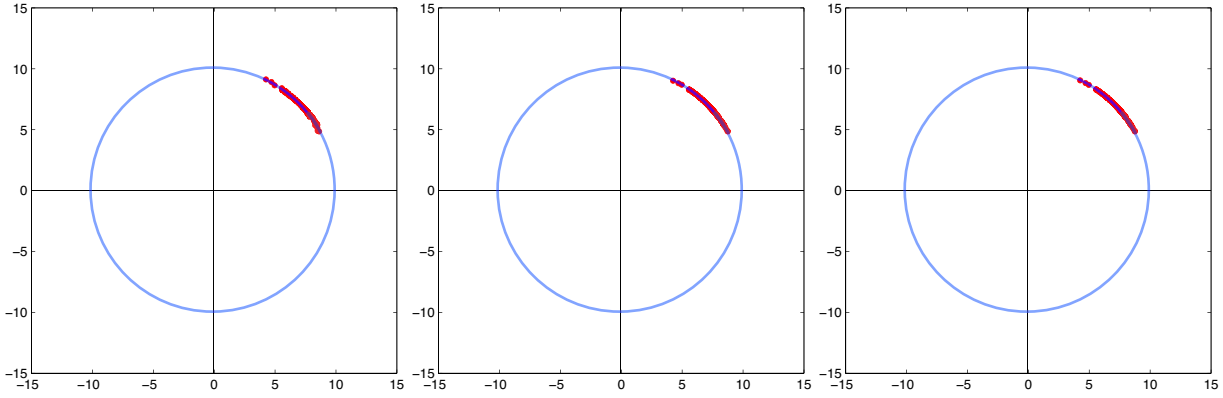
Figure 5. Example 1: Pictorial description of Stage 1 of the Cluster Newton method.



(a) $X^{(0)}$ (b) $X^{(1)}$ (c) $X^{(2)}$

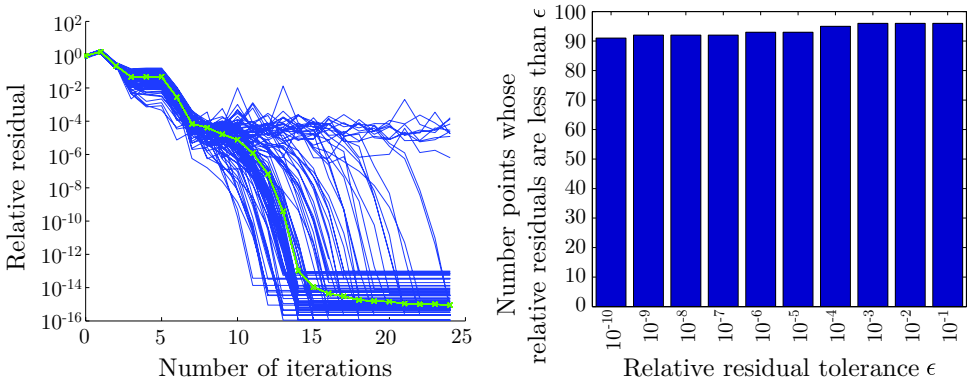


(d) $X^{(3)}$ (e) $X^{(4)}$ (f) $X^{(5)}$



(g) $X^{(6)}$ (h) $X^{(7)}$ (i) $X^{(8)}$

Figure 6. Example 1: Plots of the points in $X^{(k)}$. $X^{(0)}$ to $X^{(5)}$ correspond to Stage 1, and $X^{(6)}$ to $X^{(8)}$ to Stage 2.



(a) Speed of relative residual reduction. (b) Accuracy of the final set $X^{(24)}$ after 2500 total function evaluations.

Figure 7. Example 1: Speed and Accuracy of the Cluster Newton method.

2.3.4. Speed and accuracy of the Cluster Newton method In Figure 7(a), the relative residuals are plotted against the number of iterations. For Example 1, we define the relative residual to be the following:

$$r_j^{(k)} = \frac{|y_j^{(k)} - y^*|}{y^*} = \frac{|y_j^{(k)} - 100|}{100}. \tag{28}$$

As can be seen in Figure 7(a), the residual reduces rapidly.

In Figure 7(b), the number of points in the final set $X^{(24)}$ whose relative residuals are less than a relative residual tolerance ϵ is plotted. The graph shows that more than 90% of the points in the final set have relative residuals less than 10^{-10}.

Thus we have shown that the Cluster Newton method can find multiple accurate solutions of an underdetermined inverse problem. The method is highly efficient because of the collective operator fitting in Stage 1, and it also acts as a regularization against the small 'roughness' of the function. Hence this method is robust.

3. Pharmacokinetics ODE Coefficient Identification Problem (Example 2)

We now introduce the original problem that led us to construct the Cluster Newton algorithm for simultaneously finding multiple solutions of an underdetermined inverse problem. This inverse problem can be categorized as a coefficient identification problem of a system of ordinary differential equations. We use the system of ODEs modelling the metabolic and transportation processes of the anti-cancer drug CPT-11 and its metabolites developed by Arikuma et al. [1]. By solving this system of ODEs with fixed parameters, we simulate the amount of CPT-11 and its metabolites excreted in urine and bile. The amount of these chemical compounds in urine and bile can be measured clinically for individual patients (cf. Slatter et al. [11] and de Jong et al. [4]). The set of model parameters (we denote them as $x_1, x_2, ..., x_{60}$) represents a biological state of a patient (e.g., amounts of enzymes, blood flow rates) that cannot be measured directly.

Table 2. The amount of drug and its metabolite in excreta in units [nmol/kg] calculated from published data of Slatter et al. [11].

		Patient 1
CPT-11 in Urine	y_1^*	859.0
SN-38 in Urine	y_2^*	35.5
SN-38G in Urine	y_3^*	473.9
NPC in Urine	y_4^*	3.55
APC in Urine	y_5^*	305.0
CPT-11 in Bile + Faeces	y_6^*	975.4
SN-38 in Bile + Faeces	y_7^*	127.1
SN-38G in Bile + Faeces	y_8^*	105.4
NPC in Bile + Faeces	y_9^*	24.5
APC in Bile + Faeces	y_{10}^*	219.4
total dosage	$\sum_{i=1}^{10} y_i^*$	3946

By solving this inverse problem, we aim to find multiple possible biological states of a patient that are consistent with the clinical observations.

3.1. Inverse Problem: Parameter identification of a Pharmacokinetics Model

Using Arikuma et al.'s PBPK model (see Appendix B), we have a function that maps the parameters of the PBPK model to the quantities of CPT-11 and its metabolites excreted in urine and bile (faeces). We shall refer to a set of quantities of excreted CPT-11 and its metabolites as *an excretion profile*. In order to create the mathematical model, we wish to identify these model parameters using clinically measured excretion profiles. As there are 60 model parameters and 10 clinically measurable excretion profile quantities, we have an underdetermined inverse problem.

We note that the model parameters appear as coefficients of the system of ODEs. Also note that solving the forward problem involves solving a system of ODEs numerically and is computationally expensive.

3.1.1. Clinically measured excretion profile by Slatter et al. [11] Although it would be ideal to use raw clinical data, due to a limitation of resources, we used the clinical data published by Slatter et al. [11]. Based on their data, we calculated the excretion profile of two patients as shown in Table 2. The published data by de Jong et al. [4] lacks the information of the patient's weight so that we could not calculate the excretion profile for this data. Without loss of generality, for our numerical experiments, we use the clinically measured data of patient 1 as the target for the output of the pharmacokinetics model: the goal is to determine multiple sets of parameter values that are consistent with the excretion profile of Patient 1.

3.1.2. Typical value and variability of the model parameters $x_1, x_2, ..., x_{60}$ We use the typical values of the model parameters derived through literature search and educated

estimates by Arikuma et al. [1] and denote them as $\hat{x}_1, ..., \hat{x}_{60}$. These values are re-tabulated in Tables B1-B5 in Appendix B.

We choose the variability to be $\pm 50\%$ for the kinetic parameters (Tables B1-B3), $\pm 30\%$ for the physiological parameters (Table B4) and $\pm 5\%$ for the drug administration parameters (Table B5), i.e.,

$$v_i = \begin{cases} 0.5 & \text{for } i = 1, 2, ..., 50 \\ 0.3 & \text{for } i = 51, 52, ..., 58 \\ 0.05 & \text{for } i = 59, 60. \end{cases} \tag{29}$$

The variability of the kinetic parameters was chosen guided by the fact that the inter-subject variability of these values is usually less than $\pm 50\%$ as shown in [3, 5, 6, 12]. The variability of the physiological parameters was motivated by [13]. The variability of the drug administration parameters is chosen to be small since it is only influenced by the experimental precision of the drug administration procedure.

3.1.3. Statement of the inverse problem We now state the PBPK model parameter identification problem as follows: find a set of l points in $\mathcal{X} \subset \mathbb{R}^{60}$ near a box $\mathcal{X}^0$, such that

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}^* \,, \tag{30}$$

where

$$\boldsymbol{f} \;:\; \mathcal{X} \subset \mathbb{R}^{60} \to \mathbb{R}^{10} \qquad \text{a function that maps the model parameters to the excretion profile, as defined in Appendix B}\,, \tag{31}$$

$$\boldsymbol{y}^* \;:\; \text{clinically measured data from patient 1 as in Table 2}\,, \tag{32}$$

$$\mathcal{X} \;= \{\boldsymbol{x} \in \mathbb{R}^{60} : x_i > 0 \text{ and } \sum_{i=55}^{58} x_i < 1000\}\,, \tag{33}$$

$$\mathcal{X}^0 = \{\boldsymbol{x} \in \mathbb{R}^{60} \;:\; \max_{i=1,2,...,60} |\,(x_i - \hat{x}_i)\,/(\hat{x}_i v_i)| < 1\}\,. \tag{34}$$

We note that evaluating function value $\boldsymbol{f}(\boldsymbol{x})$ involves numerically solving the system of ODEs, hence the function cannot be evaluated exactly. The computational cost of computing the forward problem with various discretization accuracy is tabulated in Table B6.

3.2. Levenberg-Marquardt Method

We first create random points in $\mathcal{X}^0$ and then apply the Levenberg-Marquardt method using each point as an initial point. We do this computation in parallel as each run of the Levenberg-Marquardt method is parallel to each other. Due to the limitation of the Matlab Parallel Computing Toolbox we utilize at most 8 cores. A visual representation of 1,000 randomly chosen points in $\mathcal{X}^0$ is given in Figure 8. A red $\times$ indicates the centre of a set of points (average of normalized parameters). We have used the implementation

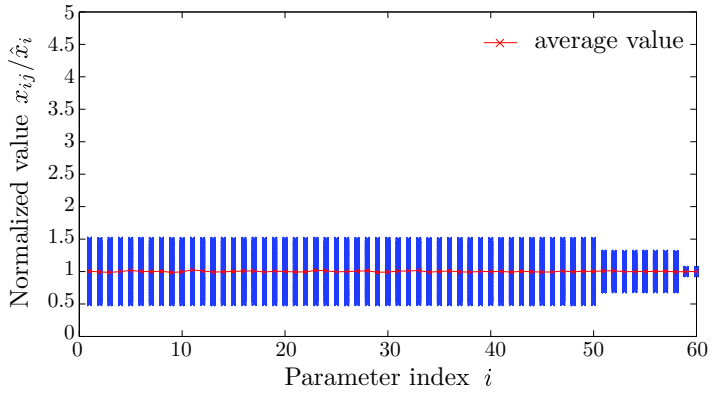


Figure 8. Initial set of points in $\mathcal{X}^0$.

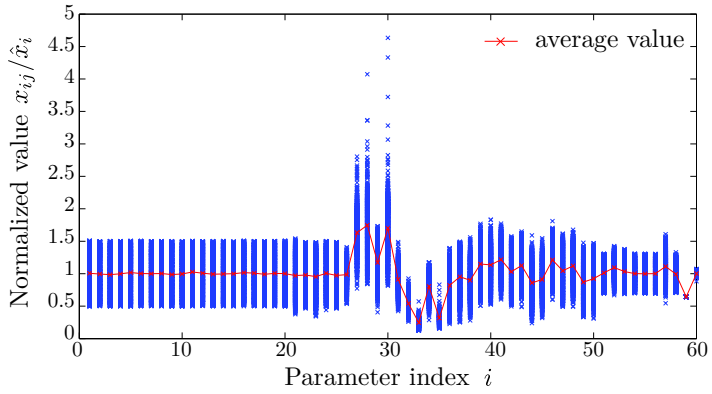


Figure 9. Final set of points found by the Levenberg-Marquardt method.

in the MATLAB optimization toolbox (version 2010b) to find the root of the following function in our numerical experiment:

$$\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}) = \begin{cases} (diag(\boldsymbol{y}^*))^{-1}\boldsymbol{f}(diag(\hat{\boldsymbol{x}})\tilde{\boldsymbol{x}}) - 1 & \text{if } (diag(\hat{\boldsymbol{x}})\tilde{\boldsymbol{x}}) \in \mathcal{X} \\ 10^5\,[1,1,1,1,1,1,1,1,1,1]^{\mathrm{T}} & \text{otherwise} \end{cases}, \tag{35}$$

where $\tilde{\boldsymbol{x}}$ is a normalized model parameter. Note that we used the normalization by $diag(\boldsymbol{y}^*)$ because it improved convergence. The way to enforce solutions in the domain of the function $\boldsymbol{f}$ in (35) turns out to work satisfactorily.

3.2.1. Visual representation of the solution found by the Levenberg-Marquardt method Figure 9 visually represents the final set of points found by the Levenberg-Marquardt method after 469,439 function evaluations at the error tolerance of 10^{-9} (i.e., we set the error tolerance to be RelTol= δ_{ODE}, AbsTol= δ_{ODE} for MATLAB ODE solver ODE15s, when solving the system of ODEs with $\delta_{ODE} = 10^{-9}$). Note that this small tolerance is required for the Levenerg-Marquardt method to converge, while the Cluster Newton method converges with larger tolerance, see below.

3.2.2. Speed and accuracy of the Levenberg-Marquardt method In Figure 10(a), the relative residual was plotted against the number of iterations. For Example 2, we define the relative residual to be the following:

$$r_j^{(k)}(\boldsymbol{x}) = \max_{i=1,2,\ldots,10} \left| \frac{y_{ij}^{(k)} - y_i^*}{y_i^*} \right|, \tag{36}$$

where $\boldsymbol{y}_{\cdot j}^{(k)} = \boldsymbol{f}(\boldsymbol{x}_{\cdot j}^{(k)})$ with $\delta_{ODE} = 10^{-11}$. As can be seen in Figure 10(a), it takes on average seven iterations to find solutions accurate up to the accuracy of the function evaluation (δ_{ODE}). We note that as the Jacobian of the function is not explicitly given, the MATLAB implementation of the Levenverg- Marquardt method estimates the Jacobian by finite differences. Hence, at each iteration, the function is evaluated at least 61 times. In each function evaluation, we solve the system of ODEs to high accuracy. Thus, this method can be computationally very expensive (e.g., to obtain the solution presented in Figure 10(b), about 8 hours of computation is used on a server machine with two quad-core Intel Xeon X7350 3GHz processors).

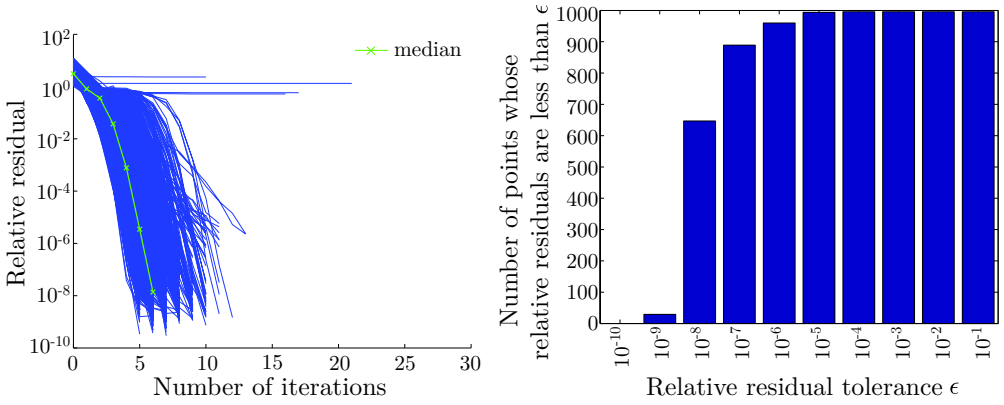
In Figure 10(b), the number of points in the final set obtained by the Levenberg-Marquardt method (after 469,439 function evaluations) whose relative residual is less than the relative residual tolerance ϵ is plotted. As can be seen in Figure 10(b), about 95% of the points achieve a relative residual less than 10^{-6} and about 65% of the points achieve a relative residual less than 10^{-8}.

As stated in Section 1.1, we confirm that we can obtain 1,000 different model parameter sets of interest through multiple application of the Levenberg-Marquardt method. However, it requires accurate function evaluation tolerances δ_{ODE} and also a large number of function evaluations per iteration. Thus, multiple applications of the Levenberg-Marquardt method to obtain multiple model parameters is computationally very expensive.

3.2.3. Influence of the accuracy of the function evaluation to the convergence of the Levenberg-Marquardt method In Figure 11, we have plotted the median relative residual against the number of iterations with different accuracy of the function evaluation. As can be seen from Figure 11, the function needs to be evaluated accurately in order for the Levenberg-Marquardt method to find the solutions with small relative residual. Numerical experiments show that an accuracy of function evaluations of at least 10^{-9} is required for the MATLAB implementation of the Levenberg-Marquard method to stably find the solution.

3.3. Cluster Newton Method

We now show that our Cluster Newton method is a much more computationally efficient way to find multiple solutions of the underdetermined inverse problem than multiple applications of the Levenberg-Marquardt method. The computational efficiency of the Cluster Newton method is due to the following characteristics: robustness against



(a) Speed of relative residual reduction.

(b) Accuracy of the final set of points after 469,439 function evaluations at the error tolerance of 10^{-9}.

Figure 10. Speed and accuracy of the Levenberg-Marquardt method applied to Example 2.

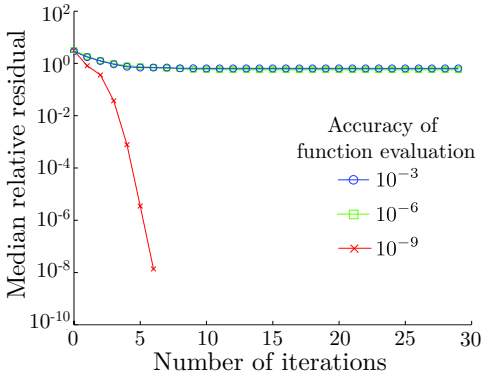


Figure 11. Influence of the accuracy of the function evaluation (δ_{ODE}) on the Levenberg-Marquardt method.

'roughness' (numerical error) in the function evaluation, and less number of required function evaluations per iteration. These characteristics follow from the collective way in which the points are updated and linear approximations are computed in Stage 1 of the Cluster Newton method.

3.3.1. Algorithm: Cluster Newton method There are three major differences between Example 1 and 2 and we need to modify the algorithm accordingly. These major differences are as follows:

- The function $\boldsymbol{f}$ is a vector function, i.e., $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{10}$.
- The domain $\mathcal{X}$ of function $\boldsymbol{f}$ is not the entire $\mathbb{R}^{60}$.
- The magnitude of the parameters to be identified differs significantly, i.e., the typical values of the model parameters varies in order from 10^{-1} to 10^{3}.

Extension of the algorithm presented in Section 2.3.1 to a vector function is straightforward. Also, we can simply modify lines 2-4 and 4-4 of this algorithm to adjust the step-size so that the points remain in the domain $\mathcal{X}$. Lastly, by scaling the matrix S when the minimum norm solutions of the underdetermined systems of linear equations are solved in lines 2-3 and 4-3, we can address the issues originating from the differences in magnitude of the parameters. The resulting algorithm is as follows:
Stage 1

1: Set up the initial points and the target values.

1-1: Randomly choose initial points $\{\boldsymbol{x}_{\cdot j}\}_{j=1}^{l}$ in the box $\mathcal{X}^0$. The initial points are stored in a $60 \times l$ matrix $X^{(0)}$ where each column corresponds to a point $\boldsymbol{x}_{\cdot j}$ in $\mathbb{R}^{60}$.

1-2: Generate randomly perturbed target values $\{\boldsymbol{y}^*_{\cdot j}\}_{j=1}^{l}$ (to maintain well-posedness of step 2-2) near $\boldsymbol{y}^*$. We choose each value $\boldsymbol{y}^*_{\cdot j}$ so that

$$\max_{i=1,2,...,10} \left| \frac{y^*_{ij} - y^*_i}{y^*_i} \right| < \eta \,, \tag{37}$$

with $\eta = 0.1$ meaning that the target accuracy for Stage 1 is $\pm 10\%$. We put these vectors in a matrix Y^* where column j corresponds to $\boldsymbol{y}^*_{\cdot j}$.

2: For $k = 0, 1, 2, ..., K_1$

2-1: Solve the forward problem for each point in (column vector of) $X^{(k)}$, i.e.,

$$Y^{(k)} = \boldsymbol{f}(X^{(k)}) \,, \tag{38}$$

where each column of $Y^{(k)}$ corresponds to the solution of the function $\boldsymbol{f}$ at each column of $X^{(k)}$.

2-2: Construct a linear approximation of $\boldsymbol{f}$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x}) \approx A^{(k)}\boldsymbol{x} + \boldsymbol{y}_o^{(k)} \,, \tag{39}$$

by fitting a hyperplane to $Y^{(k)}$. The slope matrix $A^{(k)}$ and the shift constant $\boldsymbol{y}_o^{(k)}$ can be found as a least squares solution of an over-determined system of linear equations:

$$\min_{A^{(k)} \in \mathbb{R}^{10\times 60}\,,\, \boldsymbol{y}_o^{(k)} \in \mathbb{R}^{10}} ||Y^{(k)} - (A^{(k)}X^{(k)} + Y_o^{(k)})||_F \,, \tag{40}$$

where $Y_o^{(k)}$ is a $10 \times l$ matrix whose columns are all $\boldsymbol{y}_o^{(k)}$.

2-3: Find the update vectors $\boldsymbol{s}_{\cdot j}$ for all cloumns of $X^{(k)}$ using the linear approximation, i.e., find $\boldsymbol{s}_{\cdot j}$ s.t.,

$$\boldsymbol{y}^*_{\cdot j} = A^{(k)}(\boldsymbol{x}^{(k)}_{\cdot j} + \boldsymbol{s}^{(k)}_{\cdot j}) + \boldsymbol{y}_o{}^{(k)} \qquad \text{for } j = 1, 2, ..., l \,. \tag{41}$$

As can be seen from the fact that matrix A is a rectangular matrix with more columns than rows, this is an underdetermined system of linear equations. Hence we cannot uniquely determine $\boldsymbol{s}^{(k)}_{\cdot j}$ that satisfies Equation (41). Among all the solutions of Equation (41), we choose the vector $\boldsymbol{s}^{(k)}_{\cdot j}$ with the shortest scaled length as follows. The vectors $\{\boldsymbol{s}^{(k)}_{\cdot j}\}_{j=1}^{l}$ written as a matrix $S^{(k)}$ are the minimum norm solution of an underdetermined system of linear equations:

$$\min_{S^{(k)} \in \mathbb{R}^{60\times l}} ||(diag(\hat{\boldsymbol{x}}))^{-1} S^{(k)}||_F \tag{42}$$

$$\text{s.t.} \qquad Y^* = (A^{(k)}(X^{(k)} + S^{(k)}) + Y_o^{(k)})\,. \tag{43}$$

Here, $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_{60})$. We note that we have scaled Equation (42) as the order of magnitudes of the values in vector $\boldsymbol{s}_{\cdot j}^{(k)}$ are different.

2-4: Find new points approximating the solution manifold $\mathcal{X}^*$ by updating $X^{(k)}$. If necessary, we shrink the length of the vector $\boldsymbol{s}_{\cdot j}^{(k)}$ until the point $(\boldsymbol{x}_{\cdot j}^{(k)} + \boldsymbol{s}_{\cdot j}^{(k)})$ is in the domain of the function $\boldsymbol{f}$, i.e.,

For $j = 1, 2, ..., l$

While $(\boldsymbol{x}_{\cdot j}^{(k)} + \boldsymbol{s}_{\cdot j}^{(k)}) \notin \mathcal{X}$

$$\boldsymbol{s}_{\cdot j}^{(k)} = \frac{1}{2}\boldsymbol{s}_{\cdot j}^{(k)} \tag{44}$$

End while

End for

$$X^{(k+1)} = X^{(k)} + S^{(k)}\,. \tag{45}$$

End for.

Stage 2: Broyden's method

3: Set up the initial Jacobian approximation:

$$J_{(j)}^{(K_1+1)} = A^{(K_1)} \qquad \text{for } j = 1, 2, ..., l\,. \tag{46}$$

4: For $k = K_1 + 1, ..., K_2$

4-1: Solve the forward problem for each column of $X^{(k)}$, i.e.,

$$Y^{(k)} = \boldsymbol{f}(X^{(k)})\,. \tag{47}$$

4-2: If $k \neq K_1 + 1$ then update the Jacobian for each point using Broyden's method as follows:

$$J_{(j)}^{(k)} = J_{(j)}^{(k-1)} + (\boldsymbol{y}_{\cdot j}^{(k)} - \boldsymbol{y}^*)\frac{(\boldsymbol{s}_{\cdot j}^{(k-1)})^{\mathrm{T}}}{||\boldsymbol{s}_{\cdot j}^{(k-1)}||^2} \qquad \text{for } j = 1, 2, ..., 1000. \tag{48}$$

4-3: Find the search direction vectors $\boldsymbol{s}_{\cdot j}^{(k)}$ for each point in $X^{(k)}$ using the approximated Jacobian, i.e., find $\boldsymbol{s}_{\cdot j}^{(k)}$ given as the minimum norm solution of an underdetermined linear system:

$$\min_{\boldsymbol{s}_{\cdot j}^{(k)} \in \mathbb{R}^{60}} ||(diag(\hat{\boldsymbol{x}}))^{-1}\boldsymbol{s}_{\cdot j}^{(k)}||_2 \tag{49}$$

$$\text{s.t.} \qquad \boldsymbol{y}^* - \boldsymbol{y}_{\cdot j}^{(k)} = J_{(j)}^{(k)}\boldsymbol{s}_{\cdot j}^{(k)} \tag{50}$$

for $j = 1, 2, ..., l$.

4-4: Find new points approximating the solution manifold $\mathcal{X}^*$ by updating $X^{(k)}$, i.e.,

For $j = 1, 2, ..., l$

While $(\boldsymbol{x}_{\cdot j}^{(k)} + \boldsymbol{s}_{\cdot j}^{(k)}) \notin \mathcal{X}$

$$\boldsymbol{s}_{\cdot j}^{(k)} = \frac{1}{2}\boldsymbol{s}_{\cdot j}^{(k)} \tag{51}$$

End while

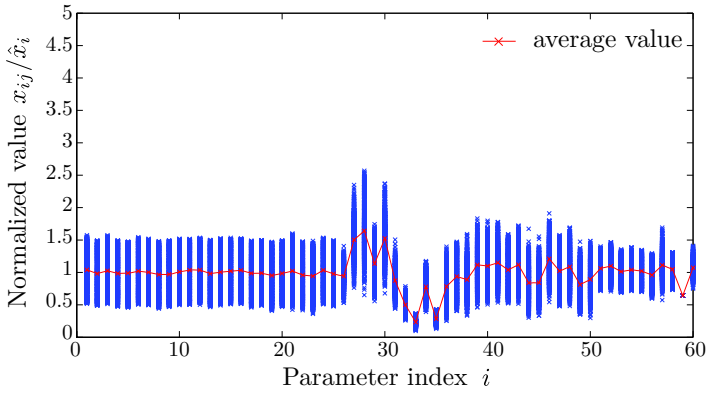


Figure 12. Example 2: Set of points $X^{(30)}$ found by the Cluster Newton method.

End for

$$X^{(k+1)} = X^{(k)} + S^{(k)}\,. \tag{52}$$

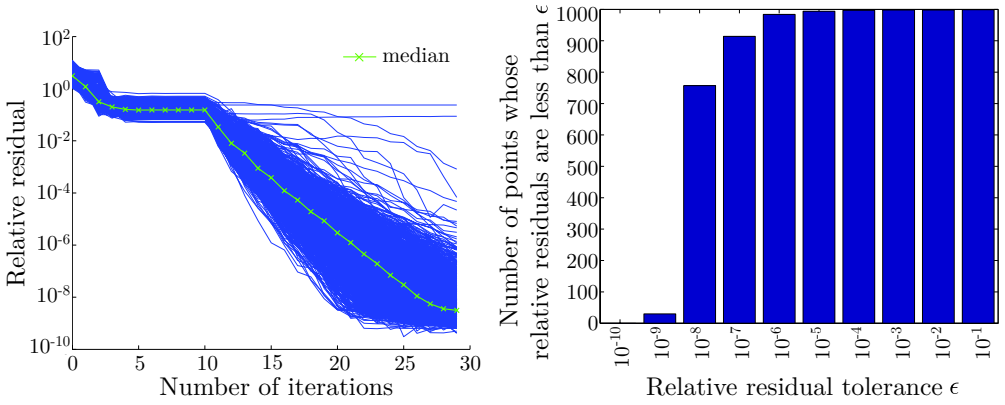
End for.

Note that the function evaluations at each point in lines 2-1 and 4-1 are independent of each other. Hence, these lines can be implemented embarrassingly in parallel. Since most of the computational cost is spent on computing the function in these two lines, the computation time required by the Cluster Newton method is almost inversely proportional to the number of CPU cores that we can utilize.

3.3.2. Visual representation of the solution found by the Cluster Newton method In the numerical result presented here, we choose $l = 1{,}000, K_1 = 10$ and $K_2 = 29$. Figure 12 visually represents the final set of points found by the Cluster Newton method after 30,000 function evaluations with the error tolerance of 10^{-9} (i.e., we set the error tolerance to be RelTol= δ_{ODE}, AbsTol= δ_{ODE} with $\delta_{ODE} = 10^{-9}$ for MATLAB ODE solver ODE15s, when solving the system of ODEs).

3.3.3. Speed and accuracy of the Cluster Newton method In Figure 13(a), the relative residual calculated as in Equation (36) is plotted against the number of iterations. As can be seen in Figure 13(a), 30 iterations are sufficient to find the solution accurate up to the accuracy of the function evaluation. We note that the Cluster Newton method only requires one function evaluation per point per iteration, thus, to get the final solution, only 30,000 function evaluations are necessary (recall that the Levenberg-Marquardt method required 469,439 function evaluations).

In Figure 13(b), the number of points in the final set obtained by the Cluster Newton method whose relative residuals are less than the relative residual tolerance ϵ is plotted. As can be seen in Figure 13(b), almost all the points achieve relative residual less than 10^{-6} and about 75% of the points achieve relative residual less than 10^{-8}.



(a) Speed of relative residual reduction.

(b) Accuracy of the final set of points after 30,000 function evaluations at the error tolerance of 10^{-9}.

Figure 13. Example 2: Speed and accuracy of the Cluster Newton method.

As the Cluster Newton method only requires one function evaluation per point per iteration, we can obtain the solution presented in Figure 13(b) in about 30 minutes on a server machine with two quad-core of Intel Xeon X7350 3GHz processors, which is a factor of 16 faster than the Levenberg-Marquardt method.

3.3.4. Influence of the accuracy of the function evaluations on the convergence of the Cluster Newton method In Figure 14, we have plotted the median of the relative residuals against the number of iterations with different accuracy of function evaluation. As can be seen from Figure 14, the Cluster Newton method finds the solutions with accuracy close to the accuracy of the function evaluation for all the function evaluation accuracies. Thus, we observe that the Cluster Newton method is robust against small errors in the function evaluation caused by the discretization of the system of ODEs. This characteristic is especially advantageous if the desired accuracy for the solution is not very high so that we can reduce the accuracy of the numerical solution of the ODEs to save computational cost. In this way the Cluster Newton method can for example obtain 1,000 solutions with relative accuracy of 10^{-3} within 4.7 minutes which is 100 times faster than the Levenberg-Marquardt method.

3.4. Comparison between the Levenberg-Marquardt method and the Cluster Newton method

We now compare the Levenberg-Marquardt method and the Cluster Newton method. We first show that the Cluster Newton method finds solutions similar to what the Levenberg-Marquardt method finds by solving the inverse problem with multiple initial points. Then we compare the time it takes to compute each set of solutions.

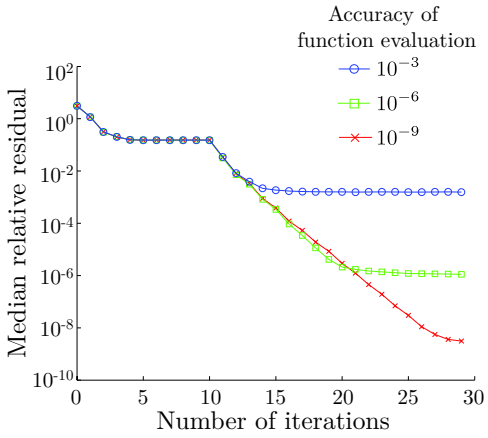


Figure 14. Influence of the accuracy of the function evaluation to the Cluster Newton method.

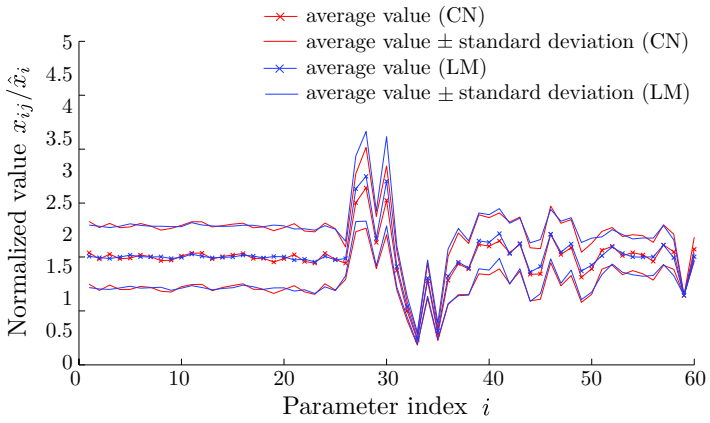


Figure 15. Average and standard deviation of the set of points found by the Levenberg-Marquardt method and the Cluster Newton method.

3.4.1. Solutions found by the Levenberg-Marquardt method and the Cluster Newton method In Figure 15, we have plotted the average and the standard deviation of the points found by the Levenberg-Marquardt method and the Cluster Newton method. The average values correspond to the location of the centre of mass of the set of these points. The standard deviation can be interpreted as the size of the point set. As can be seen in Figure 15, the difference between the set of points found by the Levenberg-Marquardt method and the Cluster Newton method is marginal.

3.4.2. Computational cost of the Levenberg-Marquardt method and the Cluster Newton method for finding multiple solutions of an underdetermined problem In Figure 16, we have plotted the relative residual versus the computational time. We have conducted this numerical experiment on a server machine with two quad-core Intel Xeon X7350 3GHz processors with fully-parallelized code using the MATLAB parallel computing toolbox for both methods.

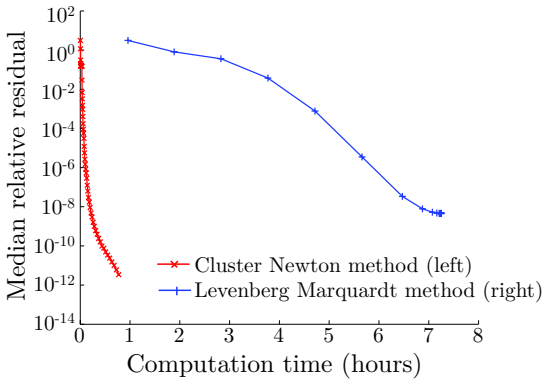


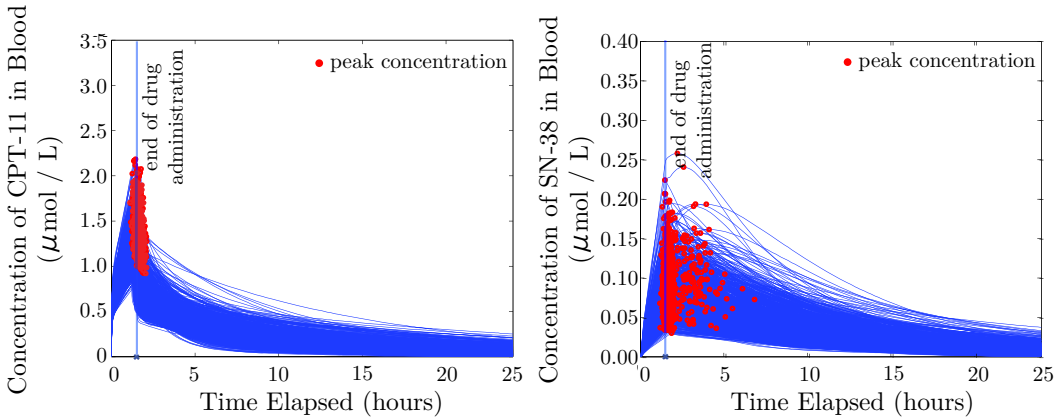
Figure 16. Comparison of computational cost between the Levenberg-Marquardt method and the Cluster Newton method.

In order for the Cluster Newton method to take advantage of the robustness against the numerical error of the function evaluation, we have implemented the Cluster Newton method so that the error tolerance of the function evaluation is initially set to 10^{-3} and then increases as the number of iterations increases. The Levenberg-Marquardt method requires 61 times more function evaluations per iteration in order to estimate the Jacobian by finite differences, and the function evaluation tolerance has to be less than 10^{-9} in order for the method to converge. Thus the computational time required by the Levenberg-Marquardt method is significantly more than for the Cluster Newton method when finding multiple solutions of the underdetermined inverse problem.

This difference in computational time becomes prominent if the desired accuracy of the solution is not very high. For example, if one only requires to find solutions whose relative residual is around 10^{-3}, then the Cluster Newton method takes only about 5 minutes to find 1,000 sets of solutions. However, the MATLAB implementation of the Levenberg-Marquardt method requires over 7 hours in order to find a similar set of solutions.

4. Conclusion

We have introduced a new idea of sampling multiple points from the solution manifold of an underdetermined inverse problem for problems for which multiple solutions are of interest. We have also proposed a new computationally efficient, easy to parallelize, robust algorithm for simultaneously finding these multiple solutions of an underdetermined inverse problem. Our algorithm was applied to a coefficient identification problem of a system of non-linear ODEs modelling the drug kinetics of an anti-cancer drug, and we demonstrated that it efficiently traces the part of the solution manifold of our interest. Multiple solutions are of interest in this application because they give representative samples of the possible biological states of a patient which according to the model, reproduces the observed data. This information about the patient can potentially be used, for example, to assess or design treatment plans.



(a) Concentration of the anti-cancer drug CPT-11

(b) Concentration of the metabolite SN-38

Figure 17. 1,000 model parameter sets found by the Cluster Newton method.

By our algorithm, 1,000 sets of model parameters can be estimated with relative accuracy 10^{-3} from clinically observed data in half an hour using a four-years-old laptop computer (MacBook Pro with 2.33 GHz Intel Core2Duo processor with 4 GB of RAM; the Cluster Newton method is implemented in MATLAB 2010b with the Parallel Computing Toolbox). Figure 17 shows the predicted concentration of CPT-11 and SN-38 in blood calculated using the model parameters found by the Cluster Newton method. Detailed comparison verifies that this solution and the solution obtained through multiple applications of the Levenberg-Marquardt method (cf. Figure 2), after 8 hours of computation using a server machine, are very similar.

We recognize that there are many ways to further accelerate our algorithm. For example, we can choose the number of Stage 1 iterations based on the residual, or can use more sophisticated step-size selection algorithms.

Similarly to what has been done for the inverse problem of the drug kinetics model, we expect that applying our algorithm to other underdetermined inverse problems will efficiently provide useful information, and will also lead to new insights about the applicability of our algorithm.

5. Acknowledgements

We would like to thank Professor Kunio Tanabe of Waseda University for his extensive advice and useful comments on our research. We would like to thank Mr. Keiichi Morikuni of the Graduate University of Advanced Studies for useful discussions. This work was supported by the Grand-in-Aid for Scientific Research (C) of the Ministry of Education, Sports, Science and Technology, Japan, and by the Natural Science and Engineering Research Council of Canada. The first author would like to thank the National Institute of Informatics and the University of Waterloo as the research opportunity which led to this paper was given to him by their MOU internship program.

Appendix A. Matrix Notation

In general, we use a capital letter for a matrix, a bold lowercase letter for a vector and a lower case letter for a scalar quantity. Also, we introduce the following matrix related notations:

$$\begin{aligned}
&\boldsymbol{y} && : \text{column vector}, && \text{(A.1)}\\
&y_i && : i\text{th component of column vector } \textit{boldsymboly}, && \text{(A.2)}\\
&x_{ij} && : \text{element of matrix } X \text{ in column } i \text{ and row } j, && \text{(A.3)}\\
&\boldsymbol{x}_{\cdot j} && : j^{th} \text{ column of matrix } X \text{ as a column vector}, && \text{(A.4)}\\
&diag(\boldsymbol{y}) && : \text{ a diagonal matrix whose } i\text{th column, } i\text{th row entry is } y_i, && \text{(A.5)}\\
&||X||_F && : \text{Frobenius norm of a matrix } X. && \text{(A.6)}
\end{aligned}$$

Appendix B. Forward problem of Example 2

Appendix B.1. Forward Problem: Physiologically Based Pharmacokinetics Model

We here briefly describe the physiologically based pharmacokinetics (PBPK) model of the intravenous (i.v.) drip infusion of CPT-11 (also known as Irinotecan). We use the PBPK model developed by Arikuma et al. [1] to model the concentration $u_1(t), ..., u_{25}(t)$ of CPT-11 and its metabolites (SN-38, SN-38G, NPC and APC) in each compartment of the body (Blood, Adipose, Gastrointestinal tract (denote as GI), Liver and everything-else (denoted as NET)). Each chemical compound in the compartment is connected by pathways, $l_1, ..., l_{55}$, representing the inflow and outflow of chemical compounds. By noting that the change in concentration is due to the flow of the chemical compounds, we can construct a system of first order ordinary differential equations (ODEs) of the concentrations denoted by $u_i(t)$ as a function of time t.

As the information presented in Arikuma et al. [1] is not sufficient to construct the ODE model, for the sake of reproducibility of our results, we here present enough details to construct a system of ODEs that models the drug kinetics of CPT-11. We shall refer the reader to [1] for biological justification and verification of this PBPK model.

Appendix B.1.1. Mathematical model of the pathways Arikuma et al. have modelled the pathways (labeled as $l_1, ..., l_{55}$ in Figure B1) of the PBPK model as listed below. There are four kinds of pathways: i.v. drip pathways, blood flow pathways, metabolic pathways, and excretion pathways. Each quantitatively describes the flow rate of the drug, with units of $[nmol/min]$. There are 60 parameters associated with these pathways and these parameters are labelled as $x_1, ..., x_{60}$ and listed in Tables B1-B5 with typical values. The goal of this inverse problem will be to determine better estimates of these parameters than the typical values listed in the tables, based on clinically observed data.

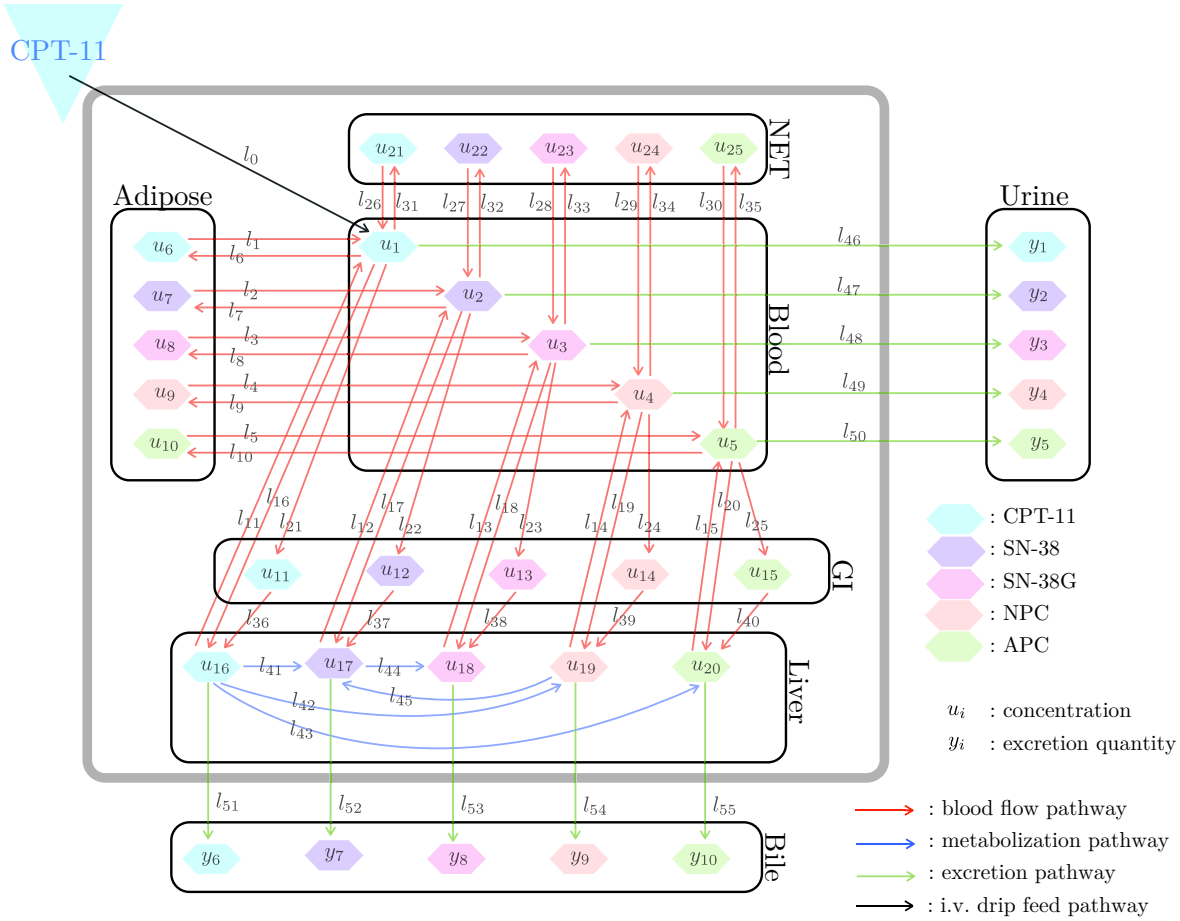


Figure B1. Schematic Diagram of the PBPK model.

Table B1. Kinetic parameters related to blood flow transportation of the drugs, where PBR is the protein binding ratio.

Organ		Tissue-Blood Distribution Ratio Adipose	GI	Liver	NET	PBR
Compound	Label	$i = 1$	$i = 6$	$i = 11$	$i = 16$	$i = 21$
CPT-11	$\hat{x}_i$	10.00	1.00	1.00	3.00	0.37
SN-38	$\hat{x}_{i+1}$	2.00	1.00	1.00	0.70	0.05
SN-38G	$\hat{x}_{i+2}$	2.80	1.00	1.00	0.08	1.00
NPC	$\hat{x}_{i+3}$	6.00	1.00	1.00	2.00	0.37
APC	$\hat{x}_{i+4}$	1.50	1.00	1.00	0.06	0.37

Table B2. Kinetic parameters related to the clearances.

Unit		Urinary Clearance $[mL/min/kg]$	Biliary Clearance $[mL/min/kg]$
Compound	Label	$i = 26$	$i = 31$
CPT-11	$\hat{x}_i$	6.15	10.6
SN-38	$\hat{x}_{i+1}$	9.91	103
SN-38G	$\hat{x}_{i+2}$	1.44	2.03
NPC	$\hat{x}_{i+3}$	1.49	14.5
APC	$\hat{x}_{i+4}$	1.47	5.45

Table B3. Kinetic parameters related to the Michaelis-Menten kinetics equation, where the units of K_m, V_{max} and α are $[nmol/mL]$, $[nmol/min/mg_{protein}]$ and $[mg_{protein}/g_{tissue}]$, respectively.

				K_m	V_{max}	α
Enzyme	Substrate	Product	Label	$i = 36$	$i = 41$	$i = 46$
Carboxylesterase	CPT-11	SN-38	$\hat{x}_i$	2.30	0.00211	128
Carboxylesterase	NPC	SN-38	$\hat{x}_{i+1}$	2.30	0.00211	128
CYP3A4	CPT-11	APC	$\hat{x}_{i+2}$	18.4	0.0260	73.3
CYP3A4	CPT-11	NPC	$\hat{x}_{i+3}$	48.2	0.0741	11.7
UGT1A1	SN-38	SN-38G	$\hat{x}_{i+4}$	3.80	0.0508	750

Table B4. Physiological parameters obtained from Willmann et al. [13].

Unit	Blood flow rate $[mL/min/kg]$		Volume $[mL/kg]$	
Organ	Label	Typical Value	Label	Typical Value
Blood	-	-	$\hat{x}_{55}$	51.0
Adipose	$\hat{x}_{51}$	4.45		$1000 - (x_{55} + x_{56} + x_{57} + x_{58})$
GI	$\hat{x}_{52}$	13.4	$\hat{x}_{56}$	32.1
Liver	$\hat{x}_{53}$	5.79	$\hat{x}_{57}$	32.3
NET	$\hat{x}_{54}$	37.4	$\hat{x}_{58}$	681

Table B5. Drug administration parameters.

Parameters	Unit	Label	Value
Dosing amount	$[nmol/kg]$	$\hat{x}_{59}$	4860
Drip feed duration	$[min]$	$\hat{x}_{60}$	90

The i.v. drip feed into blood is expressed by the following step function:

$$l_0 = \begin{cases} x_{59}/x_{60} & \text{for } 0 < t < x_{60} \\ 0 & \text{otherwise} \end{cases} . \tag{B.1}$$

The drug transportation by the blood flow pathways are modelled using the fact that the amount of drug flowing out of the compartment is proportional to the concentration of the drug in the compartment, with the parameters x_i used to model the constants of

proportionality. This can be written as follows:

$$l_i = \begin{cases} x_{51}/x_i \cdot u_{i+5}(t) & \text{for } i = 1, ..., 5\,, \\ x_{51} \cdot u_{i-5}(t) & \text{for } i = 6, ..., 10\,, \\ (x_{52} + x_{53})/x_i \cdot u_{i+5}(t) & \text{for } i = 11, ..., 15\,, \\ x_{53} \cdot u_{i-15}(t) & \text{for } i = 16, ..., 20\,, \\ x_{52} \cdot u_{i-20}(t) & \text{for } i = 21, ..., 25\,, \\ x_{54}/x_{i-10} \cdot u_{i-5}(t) & \text{for } i = 26, ..., 30\,, \\ x_{54} \cdot u_{i-30}(t) & \text{for } i = 31, ..., 35\,, \\ x_{52}/x_{i-30} \cdot u_{i-25}(t) & \text{for } i = 36, ..., 40\,, \end{cases} \tag{B.2}$$

where $u_1(t), ..., u_{25}(t)$ are the concentrations of CPT-11 and its metabolites in the compartments, as labelled in Figure B1. The drug transformation by the metabolic pathways are modelled using the Michaelis-Menten kinetics equation as follows:

$$l_i = \begin{cases} (x_{41} \cdot x_{46} \cdot x_{57})/(\frac{x_{36} \cdot x_{11}}{x_{21} \cdot u_{16}(t)} + 1) & \text{for } i = 41\,, \\ (x_{44} \cdot x_{49} \cdot x_{57})/(\frac{x_{39} \cdot x_{11}}{x_{21} \cdot u_{16}(t)} + 1) & \text{for } i = 42\,, \\ (x_{43} \cdot x_{48} \cdot x_{57})/(\frac{x_{38} \cdot x_{11}}{x_{21} \cdot u_{16}(t)} + 1) & \text{for } i = 43\,, \\ (x_{45} \cdot x_{50} \cdot x_{57})/(\frac{x_{40} \cdot x_{12}}{x_{22} \cdot u_{17}(t)} + 1) & \text{for } i = 44\,, \\ (x_{42} \cdot x_{47} \cdot x_{57})/(\frac{x_{37} \cdot x_{14}}{x_{24} \cdot u_{19}(t)} + 1) & \text{for } i = 45\,. \end{cases} \tag{B.3}$$

The drug elimination by the excretion pathway is modelled using the fact that the amount of drug flowing out of the compartment is proportional to the concentration of the drug in the compartment. This can be written as

$$l_i = \begin{cases} x_{i-20} \cdot x_{i-25} \cdot u_{i-45}(t) & \text{for } i = 46, ..., 50\,, \\ (x_{i-20} \cdot x_{i-30})/x_{i-40} \cdot u_{i-35}(t) & \text{for } i = 51, ..., 55\,. \end{cases} \tag{B.4}$$

These pathways are assembled together to form the system of ODEs, as we discuss next.

Appendix B.1.2. Mathematical model of the concentrations Let $u_1(t), ..., u_{25}(t)$ be the concentrations of CPT-11 and its metabolites in the compartments, as labelled in Figure B1, with the unit of $[\mu mol/L]$. As the change of the concentration $\frac{\mathrm{d}u_i}{\mathrm{d}t}$ is due to the flow in/out of the drug via pathways, we can construct the following system of ODEs to model the concentrations $u_i(t)$:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} u_{5(j-1)+k} = & \left(\sum_{i \in \mathcal{N}_{5(j-1)+k}} l_i(x_1, ..., x_{60}, u_1, ..., u_{25}; t) \right) / V_j \\ & - \left(\sum_{i \in \mathcal{M}_{5(j-1)+k}} l_i(x_{1,2,...,60}, u_{1,2,...,25}; t) \right) / V_j\,, \end{aligned} \tag{B.5}$$

where

$$\begin{aligned} & j = 1, ..., 5 : \text{compartments} \\ & \qquad \text{(Blood, Adipose, GI, Liver, NET)}, \end{aligned}$$

$k = 1, ..., 5$: drug and its metabolites
(CPT-11, SN-38, SN-38G, NPC, APC),

and

t : time in minutes,
$\mathcal{N}_i$: index set of the inflow pathways of u_i as listed in Equations (B.11),
$\mathcal{M}_i$: index set of the outflow pathways of u_i as listed in Equations (B.12),
l_i : flow rate of the drug in each pathway as listed in Equations (B.1)-(B.4),
x_i : model parameter,
V_j : volume of the compartment j per weight of the patient $[mL/kg]$.

The compartment volume V_j can be written as

$$V_1 = x_{55}\,, \tag{B.6}$$
$$V_2 = 1000 - (x_{55} + x_{56} + x_{57} + x_{58})\,, \tag{B.7}$$
$$V_3 = x_{56}\,, \tag{B.8}$$
$$V_4 = x_{57}\,, \tag{B.9}$$
$$V_5 = x_{58}\,. \tag{B.10}$$

Equation (B.7) is derived based on the formulation used in Willmann et al. [13] and assuming that the volume mass ratio of the human body is $1000mL/kg$. The following index sets indicate the inflow pathways of u_i as in Figure B1:

$$\mathcal{N}_i = \begin{cases} \{0,\ 1,\ 11,\ 26\} & \text{for } i = 1\,, \\ \{i,\ i+10,\ i+25\} & \text{for } i = 2, ..., 5\,, \\ \{i\} & \text{for } i = 6, ..., 10\,, \\ \{i+10\} & \text{for } i = 11, ..., 15\,, \\ \{16,\ 36\} & \text{for } i = 16\,, \\ \{17,\ 37,\ 41,\ 45\} & \text{for } i = 17\,, \\ \{18,\ 38,\ 44\} & \text{for } i = 18\,, \\ \{19,\ 39,\ 42\} & \text{for } i = 19\,, \\ \{20,\ 40,\ 43\} & \text{for } i = 20\,, \\ \{i+10\} & \text{for } i = 21, ..., 25\,. \end{cases} \tag{B.11}$$

The following index sets indicate the outflow pathways of u_i as in Figure B1:

$$\mathcal{M}_i = \begin{cases} \{i+5,\ i+15,\ i+20,\ i+30,\ i+45\} & \text{for } i = 1, ..., 5\,, \\ \{i-5\} & \text{for } i = 6, ..., 10\,, \\ \{i+25\} & \text{for } i = 11, ..., 15\,, \\ \{11,\ 41,\ 42,\ 43,\ 51\} & \text{for } i = 16\,, \\ \{12,\ 44,\ 52\} & \text{for } i = 17\,, \\ \{13,\ 53\} & \text{for } i = 18\,, \\ \{14,\ 45,\ 54\} & \text{for } i = 19\,, \\ \{15,\ 55\} & \text{for } i = 20\,, \\ \{i+5\} & \text{for } i = 21, ..., 25\,. \end{cases} \tag{B.12}$$

Table B6. Computational costs of numerically solving the forward problem for Example 2. The computational time was measured on a server machine with Intel Xeron X7350 3GHz processor.

Absolute/Relative tolerance	Computational time (sec)
10^{-2}	0.097
10^{-3}	0.12
10^{-4}	0.19
10^{-5}	0.25
10^{-6}	0.37
10^{-7}	0.42
10^{-8}	0.56
10^{-9}	0.76
10^{-10}	1.13

For example, the ODE model for the concentration of SN-38G ($k = 3$) in Liver ($j = 4$) can be written as follows:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}u_{18} &= \left(\sum_{i\in\{18,38,44\}} l_i - \sum_{i\in\{13,53\}} l_i\right)/V_4 \\ &= \left(x_{53}\cdot u_3(t) + \frac{x_{52}}{x_8}\cdot u_{13}(t) + \frac{x_{45}\cdot x_{50}\cdot x_{57}}{\frac{x_{40}\cdot x_{12}}{x_{22}\cdot u_{17}(t)}+1}\right)/x_{57} \\ &\quad - \left(\frac{x_{52}+x_{53}}{x_{13}}\cdot u_{18}(t) + \frac{x_{33}\cdot x_{23}}{x_{13}}\cdot u_{18}(t)\right)/x_{57}\,. \end{aligned} \tag{B.13}$$

By solving this system of ODEs, we obtain a vector valued function $\boldsymbol{u}(x_1, ..., x_{60}; t)$ depending on the variable t and parameters $x_1, ..., x_{60}$. Using the typical parameters $\hat{\boldsymbol{x}}$, the concentration of SN-38G in the Liver can be simulated by the above system of ODEs as depicted in Figure B2(a).

Note:

- We solve this system of ODEs using the MATLAB 2010b stiff ODE solver ODE15s [10]. The computational costs of numerically solving this system of ODEs with various integration tolerances are tabulated in Table B6.
- It can be shown easily that this ODE model conserves the amount of drug. Assuming all the parameters x_i and V_2 are positive real numbers, we further can show that $u_i(t) \geq 0$ for all i which leads to the proof of the existence of $u_i(t)$ for all $0 < t < \infty$.
- We have numerically observed that if one or more of the parameters x_i or V_2 is negative, the solution of this system of ODEs blows up in finite time. Thus, we will only consider the physiologically relevant cases where all parameters x_i and the compartment volume V_2 are positive.
- Although the analytical existence of the solution of this system of ODEs is guaranteed when all x_i and V_2 are nonnegative, this system of ODEs cannot easily

be solved numerically. We first note that one of the terms on the right hand side of the ODEs corresponding to the i.v. drip feed is discontinuous at $t = x_{60}$ (cf. Equation (B.1)). This discontinuous term causes abrupt changes of the solution u_i at time $t = x_{60}$. Hence, this system of ODEs is considered a stiff system numerically. It requires a stiff ODE solver. Also, it is noteworthy that the ODE system is well-posed only if $u_i(t) \geq 0$ for all i and $t \geq 0$, while $\lim_{t\to\infty} u_i(t) = 0$. Thus we have to make sure that the ODE solver chooses small enough time steps especially when $u_i(t)$ is small, so that the numerical solution does not go below 0. This stiffness and positivity requirement causes the high computational cost of solving the forward problem.

Appendix B.1.3. Numerical simulation of the excretion profile using the PBPK model We now describe how to compute the excretion profile using the PBPK model. Let $y_1, ..., y_{10}$ be the total excretion amount of CPT-11 and its metabolites through urine and bile in units of $[nmol/kg]$. More specifically, the variables $y_1, ..., y_5$ are the total excretion amount of CPT-11, SN-38, SN-38G, NPC and APC in urine, respectively, and the variables $y_6, ..., y_{10}$ are the total excretion amount of CPT-11, SN-38, SN-38G, NPC and APC in bile, respectively. The set of total excretion amounts of CPT-11 and its four metabolites through urine and bile (a set of ten numbers) is an *excretion profile*. The total excretion amount of each chemical compound can be computed as follows:

$$y_i = \begin{cases} \int_0^\infty x_{i+25} \cdot x_{i+20} \cdot u_i(t)\,\mathrm{d}t & \text{for } i = 1, ..., 5 \\ \int_0^\infty (x_{i+25} \cdot x_{i+15})/x_{i+5} \cdot u_{i+10}(t)\,\mathrm{d}t & \text{for } i = 6, ..., 10 \end{cases}, \tag{B.14}$$

$$= \int_0^\infty l_{i+45}\,\mathrm{d}t \qquad \text{for } i = 1, ..., 10\,, \tag{B.15}$$

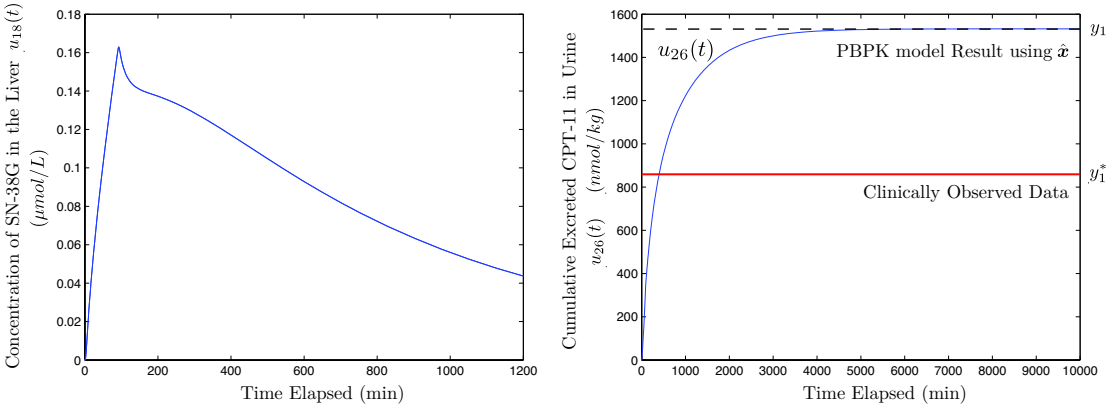
$$= \lim_{t\to\infty} \int_0^t l_{i+45}\,\mathrm{d}\tau \qquad \text{for } i = 1, ..., 10\,, \tag{B.16}$$

$$\approx \int_0^{\mathrm{T}} l_{i+45}\,\mathrm{d}t \qquad \text{for } i = 1, ..., 10\,, \tag{B.17}$$

where $u_i(t)$ is the concentration of CPT-11 and its metabolites obtained by solving the system of ODEs (B.5), and T is a sufficiently large constant. In our implementation, we have chosen $T = 10^5$ to ensure that $u_i(T) \approx 0$ for all $i = 1, ..., 5, 16, ..., 20$. It is a direct consequence of the conservation property and $u_i > 0$ that $\int_0^t l_{i+45}\,\mathrm{d}\tau$ is bounded and monotonically increasing. Thus, by the monotone convergence theorem, there exist y_i which satisfy Equations (B.14). Also by the conservation property we have $x_{59} = \sum_{i=1}^{10} y_i$.

Since the u_i are independent of the y_i, it is possible to compute the y_i at the same time as solving the PBPK model. That is to say, this numerical integration can be included in the system of ODEs by adding 10 unknown functions and re-writing Equation (B.17) in the following form:

$$\frac{\mathrm{d}}{\mathrm{d}t} u_{i+25} = l_{i+45} \qquad \text{for } i = 1, ..., 10\,, \tag{B.18}$$



(a) Concentration of SN-38G in the Liver (b) Excretion amount of CPT-11 in Urine

Figure B2. Example solutions from a PBPK model simulation using the typical value $\hat{\boldsymbol{x}}$

which leads to

$$y_i \approx u_{i+25}(T) \qquad \text{for } i = 1, ..., 10\,. \tag{B.19}$$

As an example, the graphs of $u_{26}(t)$ and y_1 are depicted in Figure B2(b) using the typical values of the parameters $\hat{\boldsymbol{x}}$.

Recalling that $u_1, ..., u_{25}$ depend on parameters $x_1, ..., x_{60}$, now we have obtained a function that maps the parameters to the excretion profile. For simplicity we denote this vector function by $\boldsymbol{f}$, i.e.,

$$[y_1, y_2, ..., y_{10}]^{\mathrm{T}} = [u_{26}(x_1, ..., x_{60}; T), ..., u_{35}(x_1, ..., x_{60}; T)]^{\mathrm{T}}\,, \tag{B.20}$$

$$= [f_1(x_1, ..., x_{60}), ..., f_{10}(x_1, ..., x_{60})]^{\mathrm{T}}\,, \tag{B.21}$$

or

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})\,, \tag{B.22}$$

where

$$\boldsymbol{f}: \ \mathbb{R}^{60} \to \mathbb{R}^{10}\,, \tag{B.23}$$

$$\boldsymbol{x} = [x_1, ..., x_{60}]^{\mathrm{T}} \ : \text{ a vector that represents the parameters}\,, \tag{B.24}$$

$$\boldsymbol{y} = [y_1, ..., y_{10}]^{\mathrm{T}} \ : \text{ a vector that represents the excretion profile}\,. \tag{B.25}$$

Appendix C. Preconditioned Cluster Newton method

As our algorithm tries to approximate the function $\boldsymbol{f}$ by a hyper-plane in a large domain, how close $\boldsymbol{f}$ is to a linear function influences the accuracy of the solution obtained by our algorithm. By choosing an appropriate pre-conditioning function $\mathbf{g}$ so that $\boldsymbol{f} \circ \boldsymbol{g}^{-1}$ is close to linear, and applying the Cluster Newton method to the composite function $\boldsymbol{f} \circ \boldsymbol{g}^{-1}$, we can increase the speed and chance of finding the solution. It is often not trivial to find such a function. However, for our ODE coefficient identification problem,

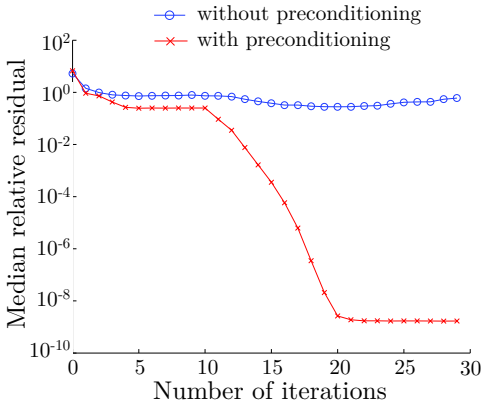


Figure C1. Median relative residual with or without preconditioning.

it is not very difficult to find one. For example, from Equation (B.13), we can observe that the x_i interact with each other mostly by multiplications and divisions. Thus, by re-defining the parameters x_i by $x_i = e^{\breve{x}_i}$ we can make them interact by additions and subtractions, i.e.,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}u_{18} = &\left(e^{\breve{x}_{53}-\breve{x}_{57}}\cdot u_3(t) + e^{\breve{x}_{52}-\breve{x}_8-\breve{x}_{57}}\cdot u_{13}(t) + \frac{e^{\breve{x}_{45}+\breve{x}_{50}}}{e^{\breve{x}_{40}+\breve{x}_{12}-\breve{x}_{22}}/u_{17}(t)+1}\right)\\ &-\left((e^{\breve{x}_{52}-\breve{x}_{13}-\breve{x}_{57}}+e^{\breve{x}_{53}-\breve{x}_{13}-\breve{x}_{57}})\cdot u_{18}(t) - e^{\breve{x}_{33}+\breve{x}_{23}-\breve{x}_{13}}\cdot u_{18}(t)\right).\end{aligned}$$

In addition to making the parameters of the function $\boldsymbol{f}$ interact almost linearly, this choice of re-definition of $\boldsymbol{x}$ has the benefit that the values of $\boldsymbol{x}$ remain positive. This helps us reduce the complication of points going outside of the domain where the function $\boldsymbol{f}$ is defined. This re-definition of $\boldsymbol{x}$ can be done easily for our algorithm by setting the pre-conditioning function as $\boldsymbol{g}(\boldsymbol{x}) = \ln(\boldsymbol{x})$ and directly applying the Cluster Newton method to $\boldsymbol{f}\circ\boldsymbol{g}^{-1}$.

The effect of this preconditioning function can be demonstrated by solving Example 2 with an initial set of points with wider than usual variability, and that brings the initial box close to the boundary of the domain $\mathcal{X}$, i.e.,

$$v_i = \begin{cases} 0.95 & \text{for } i = 1,2,...,50\\ 0.3 & \text{for } i = 51,52,...,58\\ 0.05 & \text{for } i = 59,60\end{cases}. \tag{C.1}$$

Figure C1 plots the median of the relative residual versus the number of iterations for the Cluster Newton method with or without preconditioning. As can be seen from Figure C1, without preconditioning, the Cluster Newton method fails to find the parameters of interest. However, with the preconditioning, the Cluster Newton method finds the set of points with small relative residual. For the numerical results presented in the main body of the paper preconditioning was not necessary and was thus not used.

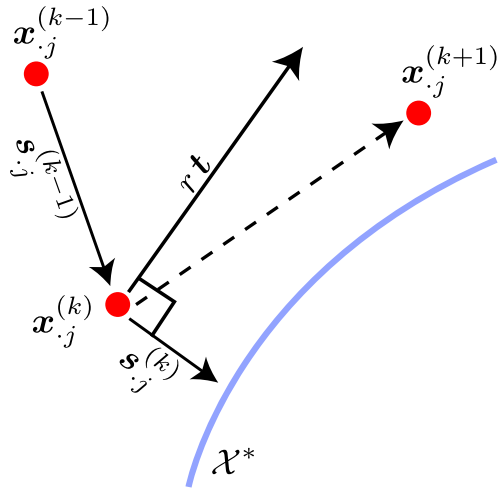


Figure D1. Main idea of the Global Cluster Newton method.

Appendix D. Global Cluster Newton method

Finding multiple solutions of the underdetermined inverse problem near the initial cluster is useful when there is *a priori* knowledge about which part of the solution manifold is of interest. However, it is a natural question to ask, if we can sample multiple solutions on the solution manifold not necessarily close to the initial cluster. We can do so using our Cluster Newton method by small modifications to line 4-4 of the algorithm. We demonstrate this idea using Example 1, that is to say, instead of just placing points on the contour curve near the initial cluster in box $\mathcal{X}^0$, we aim to place the points on the entire contour curve. In order to achieve the above goal, we simply slide the points in the direction tangent to the solution manifold once the point is sufficiently close to the solution manifold, as illustrated in Figure D1. This can be done by replacing line 4-4 in Stage 2 of the algorithm (Section 2.3.1) with the following lines.

4-4: If $k < K_3$
  For $j = 1, 2, ..., l$
    If $||\boldsymbol{s}_{\cdot j}^{(k)}||_2 < \xi$
      $t_1 = -\boldsymbol{s}_{2j}^{(k)} / ||\boldsymbol{s}_{\cdot j}^{(k)}||_2$
      $t_2 = \boldsymbol{s}_{1j}^{(k)} / ||\boldsymbol{s}_{\cdot j}^{(k)}||_2$
      $\boldsymbol{s}_{\cdot j}^{(k)} = \boldsymbol{s}_{\cdot j}^{(k)} + r\boldsymbol{t}$
    End if.
  End for.
End if.

$$X^{(k+1)} = X^{(k)} + S^{(k)}\,. \tag{D.1}$$

Hence, K_3 is the maximum number of iterations that the algorithm spreads the points in the direction tangential to the solution manifold, ξ is a parameter that selects points

sufficiently close to the solution manifold, and r is a randomly generated real number in a certain range symmetrically about 0 (see below) which is different every time it appears in the for-loop.

In Figure D2, we show the solutions found by the Global Cluster Newton method. For our numerical experiment, we have chosen $l = 100$, $K_1 = 5$, $K_2 = 100$, $K_3 = 80$, $\xi = 10^{-2}$ and r to be a uniformly randomly generated number between -5 and 5. We can observe that the points are placed on the entire contour curve.

An extension of this idea is possible for the multi-dimensional problem (e.g., Example 2). However, the tangential direction cannot be uniquely determined if the degree of freedom on the solution manifold is more than one. In such a case, we choose the direction of the tangential vector $\boldsymbol{t}$ randomly at each time it appears in the for-loop.

References

[1] Arikuma T, Yoshikawa S, Azuma R, Watanabe K, Matsumura K and Konagaya A. 2008. Drug interaction prediction using ontology-driven hypothetical assertion framework for pathway generation followed by numerical simulation. *BMC Bioinformatics* , **9**(suppl 6):S11.

[2] Björck Å, Numerical methods for least squares problems. Society for Industrial and Applied Mathematics: Philadelphia; 1996.

[3] Gagne JF, Montminy V, Belanger P, Journault K, Gaucher G and Guillemette C. 2002. Common human UGT1A polymorphisms and the altered metabolism of irinotecan active metabolite 7-ethyl-10-hydroxycamptothecin (SN-38). *Molecular Pharmacology* , **62**(3): 608-617.

[4] de Jong FA, Kitzen JJ, de Bruijn P, Verweij J and Loos WJ. 2006. Hepatic transport, metabolism and biliary excretion of irinotecan in a cancer patient with an external bile drain. *Cancer Biology and Therapy* **5**(9): 1105-10.

[5] Haaz M, Rivory L, Rich C, Vernillet L and Robert J. 1998. Metabolism of Irinotecan (CPT-11) by Human Hepatic Microsomes: Participation of Cytochrome P-450 3A and Drug Interactions. *Cancer Research* , **58**(3):468-472.

[6] Haaz M, Riche C, Rivory LP and Robert J. 1998. Biosynthesis of an aminopiperidino metabolite of irinotecan [7-ethyl-10-[4-(1-piperidino)-1-piperidino]carbonyloxycamptothecine] by human hepaticmicrosomes. *Drug Metabolism and Disposition* , **26**(8):769-774.

[7] Kelley C, Iterative Methods for Optimization: Society for Industrial and Applied Mathematics: Philadelphia; 1999.

[8] Kirkwood L, Nation R and Somogri A. 2003. Characterization of the human cytochrome P450 enzymes involved in the metabolism of dihydrocodeine. *British Journal of Clinical Pharmacology*, **44**(6): 549-555.

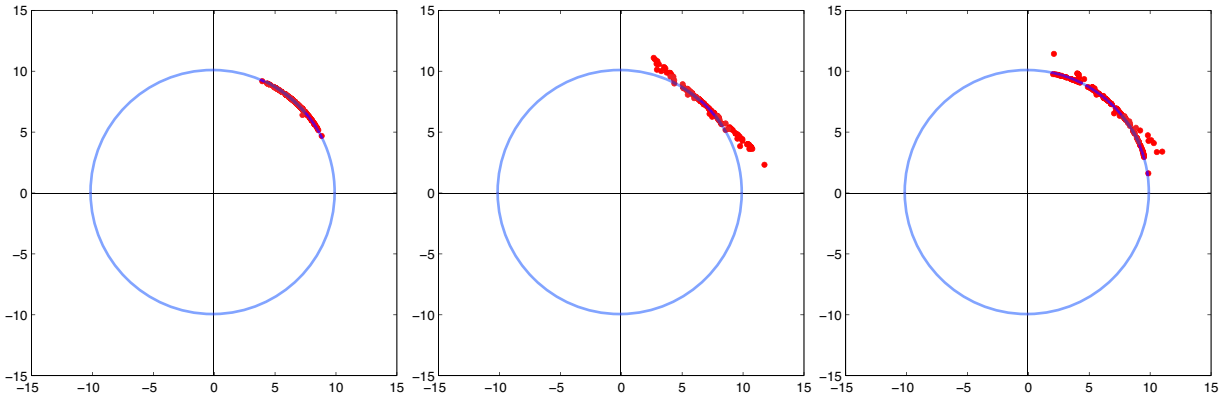
[9] Konagaya A, Bioinformatics (in Japanese). The Institute of Electronics, Information and Communication Engineering: Tokyo; 2009.

[10] Shampine LF and Reichelt MW. 1997. The matlab ODE suite. *SIAM Journal on Scientific Computing*, **18**(1):1-22.

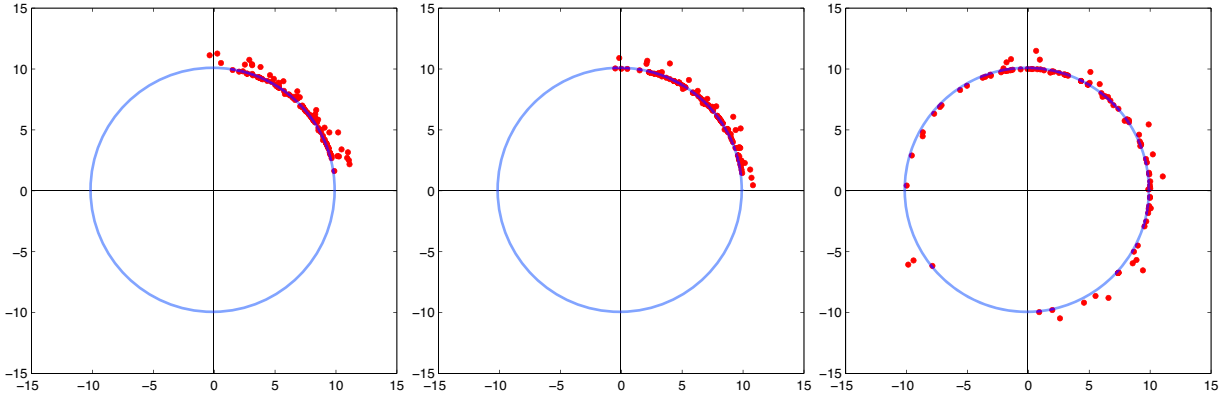
[11] Slatter JG, Schaaf LJ, Sams JP, Feenstra KL, Johnson MG, Bombardt PA et al. 2000. Pharmacokinetics, Metabolism, and Excretion of Irinotecan (CPT-11) Following I.V. Infusion of [14C]CPT-11 in Cancer Patients. *Drug Metabolism and Disposition* , **28**(4):423-433.

[12] Slatter J, Su P, Sams J, Schaaf L, Wienkers L. 1997. Bioactivation of the Anticancer agent CPT-11 to SN-38 by human hepatic microsomal carboxylesterases and the in vitro Assessment of potential drug interactions. *Drug Metabolism and Disposition* , **25**(10):1157-1164.

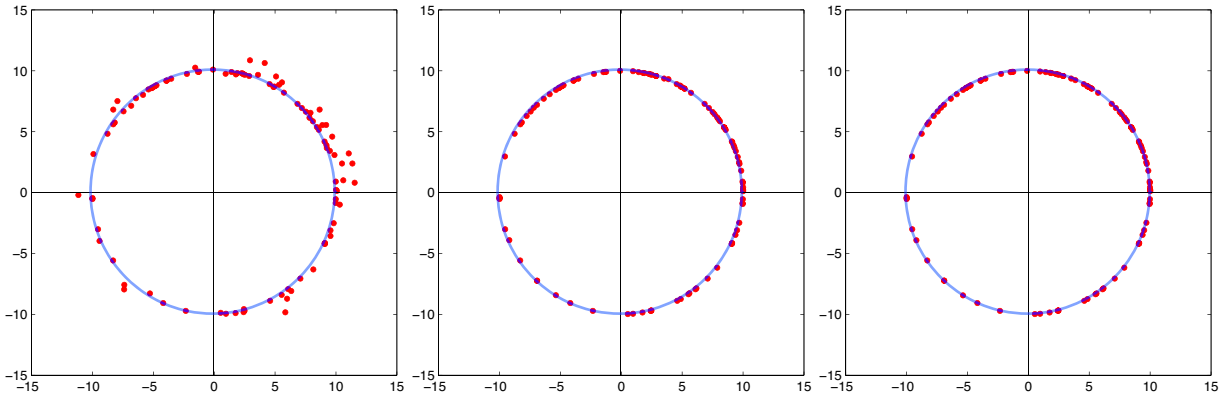
[13] Willmann S, Hohn K, Edginton A, Sevestre M, Solodenko J, Weiss W et al. 2007.



(a) $X^{(6)}$ (b) $X^{(7)}$ (c) $X^{(8)}$



(d) $X^{(9)}$ (e) $X^{(10)}$ (f) $X^{(40)}$



(g) $X^{(79)}$ (h) $X^{(80)}$ (i) $X^{(100)}$

Figure D2. Example 1: Plots of the points $X^{(k)}$ found by the Global Cluster Newton method. $X^{(0)}$ to $X^{(5)}$ are the same as in Figure 6 so the plots were omitted.

Development of a physiology-based whole-body population model for assessing the influence of individual variability on the pharmacokinetics of drugs. *Journal of Pharmacokinetics and Pharmacodynamics*, **34**(3):401-431.