Nonlinear Data Fitting for ODEs

John Bagterp Jørgensen

Department of Informatics and Mathematical Modeling
Technical University of Denmark

02610 Optimization and Data Fitting
Lecture 12
November 26, 2012

Learning Objectives

Apply numerical techniques for

data fitting / regression

in nonlinear models involving

ordinary differential equations (ODEs).

Outline

1. The Nonlinear ODE Model and Sensitivities
2. Least Squares Estimation
3. Statistical Inference

Parameter Estimation in ODEs

Given the data points $\{(t_i, y_i)\}_{i=1}^{m}$ compute the parameters estimates, p, such that

$$
\begin{aligned}
\min_{p \in \mathbb{R}^{n_p}} \quad & \phi = \frac{1}{2}\sum_{i=1}^{m} \|\hat{y}(t_i) - y_i\|_2^2 \\
s.t. \quad & \frac{dx}{dt}(t) = f(t, x(t), p) \qquad x(t_0) = x_0 \\
& \hat{y}(t) = g(x(t), p) \\
& p_l \leq p \leq p_u
\end{aligned}
$$

We denote the minimizer to this optimization problem as $\hat{p}$ and say that it is the nonlinear least squares estimate.

Parameter Estimation in ODEs

Note that we can express the output function $\hat{y}(t) = \hat{y}(t; p, x_0)$

$$\begin{aligned}\hat{y}(t) &= \hat{y}(t;p) = \hat{y}(t;p,x_0)\\ &= \left\{\hat{y}(t) = g(x(t),p) :\ \frac{dx}{dt}(t) = f(t,x(t),p),\ x(t_0) = x_0\right\}\end{aligned}$$

Then we can reformulate

$$\begin{aligned}\min_{p\in\mathbb{R}^{n_p}} \quad & \phi = \frac{1}{2}\sum_{i=1}^{m} \|\hat{y}(t_i) - y_i\|_2^2\\ s.t. \quad & \frac{dx}{dt}(t) = f(t,x(t),p) \qquad x(t_0) = x_0\\ & \hat{y}(t) = g(x(t),p)\\ & p_l \le p \le p_u\end{aligned}$$

as

$$\begin{aligned}\min_{p\in\mathbb{R}^{n_p}} \quad & \phi = \frac{1}{2}\sum_{i=1}^{m} \|\hat{y}(t_i;p,x_0) - y_i\|_2^2\\ s.t. \quad & p_l \le p \le p_u\end{aligned}$$

Parameter Estimation in ODEs

$$\begin{aligned} \min_{p \in \mathbb{R}^{n_p}} \quad & \phi = \frac{1}{2} \sum_{i=1}^{m} \| \hat{y}(t_i; p, x_0) - y_i \|_2^2 \\ s.t. \quad & p_l \leq p \leq p_u \end{aligned}$$

with

$$\begin{aligned} \hat{y}(t) &= \hat{y}(t; p) = \hat{y}(t; p, x_0) \\ &= \left\{ \hat{y}(t) = g(x(t), p) : \frac{dx}{dt}(t) = f(t, x(t), p),\ x(t_0) = x_0 \right\} \end{aligned}$$

is a nonlinear least squares problem.

The key questions are

- How to compute $\hat{y}(t) = \hat{y}(t; p, x_0)$
- How to compute $\frac{\partial \hat{y}}{\partial p}(t; p, x_0)$

The Nonlinear Data Fitting Model

Measurements:

$$\{(t_1, y_1), (t_2, y_2), \dots, (t_m, y_m)\} = \{(t_i, y_i)\}_{i=1}^{m}$$

Model:

$$\frac{dx}{dt}(t) = f(t, x(t); p) \qquad x(t_0) = x_0 \qquad f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^{n_p} \mapsto \mathbb{R}^n$$

$$\hat{y}(t) = g(x(t), p) \qquad g : \mathbb{R}^n \times \mathbb{R}^{n_p} \mapsto \mathbb{R}$$

$$y_i = \hat{y}(t_i) + e_i \qquad e_i \sim N(0, \sigma^2) \qquad i = 1, 2, \dots, m$$

Parameters:

$$p \in \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n_p} \end{bmatrix}$$

Sensitivity Equation

Ordinary Differential Equation (ODE), Initial Value Problem (IVP)

$$\frac{dx}{dt}(t) = f(t, x(t), p) \qquad x(t_0) = x_0$$

Solution

$$x(t) = x(t, p) \qquad x \in \mathbb{R}^n,\ t \in \mathbb{R},\ p \in \mathbb{R}^{n_p}$$

Sensitivity

$$S_p(t) \triangleq \frac{\partial x}{\partial p}(t, p) \qquad S_p \in \mathbb{R}^{n \times n_p}$$

$$S_p(t) = \begin{bmatrix} \frac{\partial x_1}{\partial p_1}(t) & \frac{\partial x_1}{\partial p_2}(t) & \dots & \frac{\partial x_1}{\partial p_{n_p}}(t) \\ \frac{\partial x_2}{\partial p_1}(t) & \frac{\partial x_2}{\partial p_2}(t) & \dots & \frac{\partial x_2}{\partial p_{n_p}}(t) \\ \vdots & \vdots & & \vdots \\ \frac{\partial x_n}{\partial p_1}(t) & \frac{\partial x_n}{\partial p_2}(t) & \dots & \frac{\partial x_n}{\partial p_{n_p}}(t) \end{bmatrix}$$

Sensitivity Equation

Ordinary Differential Equation (ODE), Initial Value Problem (IVP)

$$\frac{dx}{dt}(t) = f(t, x(t), p) \qquad x(t_0) = x_0$$

Solution

$$x(t) = x(t, p) \qquad x \in \mathbb{R}^n,\ t \in \mathbb{R},\ p \in \mathbb{R}^{n_p}$$

Sensitivity

$$S_p(t) \triangleq \frac{\partial x}{\partial p}(t, p) \qquad S_p \in \mathbb{R}^{n \times n_p}$$

$$\begin{aligned}
\frac{dS_p}{dt}(t) = \frac{d}{dt}\left(\frac{\partial x}{\partial p}(t,p)\right) = \frac{\partial}{\partial p}\left(\frac{dx}{dt}(t,p)\right) &= \frac{\partial}{\partial p}\left(f(t, x(t,p), p)\right) \\
&= \frac{\partial f}{\partial x}(t, x(t,p), p)\frac{\partial x}{\partial p}(t,p) + \frac{\partial f}{\partial p}(t, x(t,p), p) \\
&= \frac{\partial f}{\partial x}(t, x(t,p), p) S_p(t) + \frac{\partial f}{\partial p}(t, x(t,p), p)
\end{aligned}$$

$$S_p(t_0) = \frac{\partial x}{\partial p}(t_0, p) = \frac{\partial x_0}{\partial p} = 0$$

Derivatives

$$\frac{dS_p}{dt}(t) = \frac{\partial f}{\partial x}(t, x(t,p), p) S_p(t) + \frac{\partial f}{\partial p}(t, x(t,p), p) \qquad S_p(t_0) = 0$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \qquad \frac{\partial f}{\partial p} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} & \cdots & \frac{\partial f_1}{\partial p_{n_p}} \\ \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} & \cdots & \frac{\partial f_2}{\partial p_{n_p}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial p_1} & \frac{\partial f_n}{\partial p_2} & \cdots & \frac{\partial f_n}{\partial p_{n_p}} \end{bmatrix}$$

$$S_p(t) = \begin{bmatrix} \frac{\partial x_1}{\partial p_1}(t) & \frac{\partial x_1}{\partial p_2}(t) & \cdots & \frac{\partial x_1}{\partial p_{n_p}}(t) \\ \frac{\partial x_2}{\partial p_1}(t) & \frac{\partial x_2}{\partial p_2}(t) & \cdots & \frac{\partial x_2}{\partial p_{n_p}}(t) \\ \vdots & \vdots & & \vdots \\ \frac{\partial x_n}{\partial p_1}(t) & \frac{\partial x_n}{\partial p_2}(t) & \cdots & \frac{\partial x_n}{\partial p_{n_p}}(t) \end{bmatrix}$$

Sensitivity Equation

Ordinary Differential Equation (ODE), Initial Value Problem (IVP)

$$\frac{dx}{dt}(t) = f(t, x(t), p) \qquad x(t_0) = x_0$$

Solution

$$x(t) = x(t, p) \qquad x \in \mathbb{R}^n,\ t \in \mathbb{R},\ p \in \mathbb{R}^{n_p}$$

Sensitivity

$$S_p(t) \triangleq \frac{\partial x}{\partial p}(t, p) \qquad S_p \in \mathbb{R}^{n \times n_p}$$

$$\frac{dS_p}{dt}(t) = \frac{\partial f}{\partial x}(t, x(t, p), p) S_p(t) + \frac{\partial f}{\partial p}(t, x(t, p), p) \qquad S_p(t_0) = 0$$

Sensitivity of Outputs

$$\hat{y}(t) = g(x(t), p)$$

$$x(t) = x(t, p)$$

$$\hat{y}(t, p) = g(x(t, p), p)$$

$$\begin{aligned} \frac{\partial \hat{y}}{\partial p}(t, p) &= \frac{\partial g}{\partial x}(x(t, p), p) \frac{\partial x}{\partial p}(t, p) + \frac{\partial g}{\partial p}(x(t, p), p) \\ &= \frac{\partial g}{\partial x}(x(t, p), p) S_p(t) + \frac{\partial g}{\partial p}(x(t, p), p) \end{aligned}$$

Summary - Computation of Derivatives

The model predictions are computed by

$$\frac{dx}{dt}(t) = f(t, x(t); p) \qquad x(t_0) = x_0$$
$$\hat{y}(t) = g(x(t), p)$$

and the derivatives w.r.t. p are computed simultaneously by

$$\frac{dS_p}{dt}(t) = \frac{\partial f}{\partial x}(t, x(t,p), p) S_p(t,p) + \frac{\partial f}{\partial p}(t, x(t,p), p) \qquad S_p(t_0) = 0$$
$$\frac{\partial \hat{y}}{\partial p}(t,p) = \frac{\partial g}{\partial x}(x(t,p), p) S_p(t) + \frac{\partial g}{\partial p}(x(t,p), p)$$

Numerical Solution of ODEs

$$\frac{dx}{dt}(t) = f(t, x(t)) \qquad x(t_0) = x_0$$

Non-stiff systems of differential equations - ode45

```
ODE45  Solve non-stiff differential equations, medium order method.
   [TOUT,YOUT] = ODE45(ODEFUN,TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates
   the system of differential equations y' = f(t,y) from time T0 to TFINAL
   with initial conditions Y0. ODEFUN is a function handle. For a scalar T
   and a vector Y, ODEFUN(T,Y) must return a column vector corresponding
   to f(t,y). Each row in the solution array YOUT corresponds to a time
   returned in the column vector TOUT.  To obtain solutions at specific
   times T0,T1,...,TFINAL (all increasing or all decreasing), use TSPAN =
   [T0 T1 ... TFINAL].

   [TOUT,YOUT] = ODE45(ODEFUN,TSPAN,Y0,OPTIONS) solves as above with default
   integration properties replaced by values in OPTIONS, an argument created
   with the ODESET function. See ODESET for details. Commonly used options
   are scalar relative error tolerance 'RelTol' (1e-3 by default) and vector
   of absolute error tolerances 'AbsTol' (all components 1e-6 by default).
   If certain components of the solution must be non-negative, use
   ODESET to set the 'NonNegative' property to the indices of these
   components.
```

$$\frac{dx}{dt}(t) = f(t, x(t), p) \qquad x(t_0) = x_0$$

Model

```
function xdot = model(t,x,p)
...
...
```

Solve Model

```
t0 = ...          // Initial time
tf = ...          // Final time
x0 = ...          // Initial states
p  = ...          // Parameter vector

[T,X] = ode45(@model,[t0 tf],x0,[],p)
```

Sensitivity Equations

The states and their parameter sensitivities are computed by

$$\begin{aligned} \frac{dx}{dt}(t) &= f(t, x(t); p) && x(t_0) = x_0 \\ \frac{dS_p}{dt}(t) &= \frac{\partial f}{\partial x}(t, x(t,p), p) S_p(t,p) + \frac{\partial f}{\partial p}(t, x(t,p), p) && S_p(t_0) = 0 \end{aligned}$$

Define

$$\begin{aligned} x &= \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^T \\ s_p &= \begin{bmatrix} S_{p,1,1} & \dots & S_{p,n,1} & S_{p,1,2} & \dots & S_{p,n,2} & \dots & S_{p,1,n_p} & \dots & S_{p,n,n_p} \end{bmatrix}^T \\ z &= \begin{bmatrix} x^T & s_p^T \end{bmatrix}^T \end{aligned}$$

$$\frac{dz}{dt}(t) = F(t, z, p) \qquad z(t_0) = z_0$$

Outline of Matlab Implementation

$$\frac{dz}{dt}(t) = F(t, z, p) \qquad z(t_0) = z_0$$

Model and Sensitivities

```
function zdot = ModelAndSensitivity(t,z,p,n,np)

x = z(1:n,1);                      // Unpack states
sp = z(n+1:end,1);                 // Unpack sensitivities
Sp = reshape(sp,n,np);             // Sensitivities as a matrix

"xdot = f(t,x,p);"                 // Evaluate the model equations
"[dfdx,dfdp] = Derivatives(t,x,p);" // Evaluate the derivatives

Spdot = dfdx*Sp + dfdp;

zdot = [xdot; Spdot(:)];           // Return derivatives as a vector
```

Solution

```
[T,Z] = ode45(@ModelAndSensitivity,[t0 tf],z0,[],p,n,np);
```

Considerations

- Accuary/precision of ODE solution
- Termination criteria for nonlinear optimization
- Compatibility of these accuracies / precisions

Least Squares Estimation

Residual

$$r_i = r_i(p) = \hat{y}(t_i;p) - y_i = -e_i \qquad e_i \sim N(0,\sigma^2) \qquad i = 1,2,\ldots,m$$

$$\frac{\partial r_i}{\partial p_j}(p) = \frac{\partial \hat{y}}{\partial p_j}(t_i;p)$$

$$r(p) = \begin{bmatrix} r_1(p) \\ r_2(p) \\ \vdots \\ r_m(p) \end{bmatrix} \qquad J(p) = \begin{bmatrix} \frac{\partial \hat{y}}{\partial p_1}(t_1;p) & \frac{\partial \hat{y}}{\partial p_2}(t_1;p) & \ldots & \frac{\partial \hat{y}}{\partial p_{n_p}}(t_1;p) \\ \frac{\partial \hat{y}}{\partial p_1}(t_2;p) & \frac{\partial \hat{y}}{\partial p_2}(t_2;p) & \ldots & \frac{\partial \hat{y}}{\partial p_{n_p}}(t_2;p) \\ \vdots & \vdots & & \vdots \\ \frac{\partial \hat{y}}{\partial p_1}(t_m;p) & \frac{\partial \hat{y}}{\partial p_2}(t_m;p) & \ldots & \frac{\partial \hat{y}}{\partial p_{n_p}}(t_m;p) \end{bmatrix}$$

Least Squares Estimation

$$\hat{p} = \arg\min_p \, \phi(p) = \frac{1}{2}\,\|r(p)\|_2^2$$

Least Squares Estimation - Hessian Matrix Approximation

Least Squares Estimation

$$\hat{p} = \arg\min_{p} \phi(p) = \frac{1}{2} \|r(p)\|_2^2$$

Derivatives

$$\nabla\phi(p) = J(p)^T r(p)$$

$$\nabla^2\phi(p) = J(p)^T J(p) + \sum_{i=1}^{m} r_i(p)\nabla^2 r_i(p)$$

Approximation in numerical algorithms for LS problems

$$r_i(p) \approx 0 \qquad i = 1, 2, \ldots, m$$

$$H = J(p)^T J(p) \approx \nabla^2\phi(p)$$

```
Nonlinear minimization of functions.
  fminbnd      - Scalar bounded nonlinear function minimization.
  fmincon      - Multidimensional constrained nonlinear minimization.
  fminsearch   - Multidimensional unconstrained nonlinear minimization,
                 by Nelder-Mead direct search method.
  fminunc      - Multidimensional unconstrained nonlinear minimization.
  fseminf      - Multidimensional constrained minimization, semi-infinite
                 constraints.
  ktrlink      - Multidimensional constrained nonlinear minimization
                 using KNITRO(R) third-party libraries.

Nonlinear minimization of multi-objective functions.
  fgoalattain  - Multidimensional goal attainment optimization
  fminimax     - Multidimensional minimax optimization.

Linear least squares (of matrix problems).
  lsqlin       - Linear least squares with linear constraints.
  lsqnonneg    - Linear least squares with nonnegativity constraints.

Nonlinear least squares (of functions).
  lsqcurvefit  - Nonlinear curvefitting via least squares (with bounds).
  lsqnonlin    - Nonlinear least squares with upper and lower bounds.

Nonlinear zero finding (equation solving).
  fzero        - Scalar nonlinear zero finding.
  fsolve       - Nonlinear system of equations solve (function solve).

Minimization of matrix problems.
  bintprog     - Binary integer (linear) programming.
  linprog      - Linear programming.
  quadprog     - Quadratic programming.

Controlling defaults and options.
  optimset     - Create or alter optimization OPTIONS structure.
  optimget     - Get optimization parameters from OPTIONS structure.
```

Solution of Nonlinear Least Squares Problems in Matlab

```
LSQNONLIN solves non-linear least squares problems.
   LSQNONLIN attempts to solve problems of the form:
   min  sum {FUN(X).^2}    where X and the values returned by FUN can be
    X                      vectors or matrices.

   LSQNONLIN implements two different algorithms: trust region reflective and
   Levenberg-Marquardt. Choose one via the option Algorithm: for instance, to
   choose Levenberg-Marquardt, set OPTIONS = optimset('Algorithm','levenberg-marquardt'),
   and then pass OPTIONS to LSQNONLIN.

   X = LSQNONLIN(FUN,X0) starts at the matrix X0 and finds a minimum X to
   the sum of squares of the functions in FUN. FUN accepts input X
   and returns a vector (or matrix) of function values F evaluated
   at X. NOTE: FUN should return FUN(X) and not the sum-of-squares
   sum(FUN(X).^2)). (FUN(X) is summed and squared implicitly in the
   algorithm.)

   X = LSQNONLIN(FUN,X0,LB,UB) defines a set of lower and upper bounds on
   the design variables, X, so that the solution is in the range LB <= X
   <= UB. Use empty matrices for LB and UB if no bounds exist. Set LB(i)
   = -Inf if X(i) is unbounded below; set UB(i) = Inf if X(i) is
   unbounded above.

   X = LSQNONLIN(FUN,X0,LB,UB,OPTIONS) minimizes with the default
   optimization parameters replaced by values in the structure OPTIONS,
   an argument created with the OPTIMSET function. See OPTIMSET for details.
   Use the Jacobian option to specify that FUN also returns a second output
   argument J that is the Jacobian matrix at the point X. If FUN returns a
   vector F of m components when X has length n, then J is an m-by-n matrix
   where J(i,j) is the partial derivative of F(i) with respect to x(j).
   (Note that the Jacobian J is the transpose of the gradient of F.)
```

LSQNONLIN in Matlab's Optimization Toolbox

$$\begin{aligned} \min_{p \in \mathbb{R}^{n_p}} \quad & \phi(p) = \|r(p)\|_2^2 \\ s.t. \quad & l \leq p \leq u \end{aligned}$$

- Read about LSQNONLIN using `doc lsqnonlin`
- Read about optimization options using `doc optimset`
- Also see the options by typing `optimset()` in the work space
- Figure out how to supply the Jacobian, $J(p)$, to `lsqnonlin`
- Compare the case (CPU time) when you supply the Jacobian, $J(p)$, with the case when the Jacobian is evaluated numerically.

Estimation of Covariance

Let $\hat{p}$ be the estimated parameter vector such that the estimated model is

$$\hat{y} = \hat{y}(t; \hat{p}) \approx \hat{y}(t; p^*) + J(p^*)(\hat{p} - p^*)$$

An unbiased estimate of the noise covariance is

$$\hat{\sigma}^2 = \frac{1}{m - n_p} \sum_{i=1}^{m} (y_i - \hat{y}(t_i; \hat{p}))^2$$

Distribution of the parameter vector estimate, $\hat{p}$ (under a linear approximation)

$$\hat{\boldsymbol{p}} \sim N(p^*, \hat{\sigma}^2 H^{-1})$$

$$H = H(\hat{p}) = J(\hat{p})^T J(\hat{p})$$

p^* is the true (unknown) value of the parameter vector

Confidence Intervals - Parameters

Statistical results

$$a^T \hat{\boldsymbol{p}} \sim N(a^T p^*, \hat{\sigma}^2 a^T H^{-1} a)$$

$$T = \frac{a^T \hat{\boldsymbol{p}} - a^T p^*}{\hat{\sigma}(a^T H^{-1} a)^{1/2}} \sim t_{m-n_p}$$

An approximate $100(1-\alpha)\%$ confidence interval is

$$a^T \hat{p} \pm t_{m-n_p}(\alpha/2)\hat{\sigma}(a^T H^{-1} a)^{1/2}$$

$$\hat{p}_i \pm t_{m-n_p}(\alpha/2)\hat{\sigma}\sqrt{C_{ii}} \qquad C_{ii} = [H^{-1}]_{ii}$$

Confidence Interval - Predictions

Statistical results (asymptotic, under a linear approximation)

$$\hat{y}(t;\hat{p}) \approx \hat{y}(t;p^*) + \frac{\partial \hat{y}}{\partial p}(t;p^*)(\hat{p} - p^*)$$

$$\hat{y}(t;\hat{p}) \sim N\left(\hat{y}(t;p^*), \hat{\sigma}^2 \left[\frac{\partial \hat{y}}{\partial p}(t;\hat{p})\right]^T H^{-1} \left[\frac{\partial \hat{y}}{\partial p}(t;\hat{p})\right]\right)$$

An approximate $100(1-\alpha)\%$ confidence interval is

$$a^T \hat{p} \pm t_{m-n_p}(\alpha/2)\hat{\sigma}(a^T H^{-1} a)^{1/2}$$

$$\hat{y}(t;\hat{p}) \pm t_{m-n_p}(\alpha/2)\hat{\sigma}\left(\left[\frac{\partial \hat{y}}{\partial p}(t;\hat{p})\right]^T H^{-1} \left[\frac{\partial \hat{y}}{\partial p}(t;\hat{p})\right]\right)^{1/2}$$

Computation of Student's t-distribution

$$T = \frac{Z}{\sqrt{V/\nu}} \sim t_\nu \quad Z \in N(0,1) \quad V \in \chi(\nu)$$

Probability density function

$$p(x;\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Γ: the Gamma function.

Cumulative probability density function

$$P(x;\nu) = \int_{-\infty}^{x} p(t;\nu)dt = 1 - \frac{1}{2} I_{x(t)}\left(\frac{\nu}{2}, \frac{1}{2}\right) \qquad x > 0$$

I: the regularized incomplete beta function

Computation of $t_{m-n_p}(\alpha/2)$: Compute x such that $P(x; m-n_p) = \alpha/2$

- $t_{m-n_p}(\alpha/2) = \texttt{tinv}(\alpha/2, m-n_p)$ (statistical toolbox)