

89. Parameter estimation for ODE models


(a) Single equation, single parameter.

Consider ODE model

\[
\frac{dx}{dt} = f(x, t, a), \quad x(t_0) = x_0
\]

with parameter \(a\), which will be chosen to best approximate measured data set \((x_i, t_i), i = 1, 2, \ldots, N\).

- Recall what we did with algebraic model \(x = x(t, a)\):
  * Construct merit function
    \[
    X^2 = \sum_{i=1}^{N} \left( \frac{x_i - x(t_i, a)}{\sigma_i} \right)^2
    \]
  * Minimize \(X^2\) with respect to \(a\):
    \[
    \frac{\partial X^2}{\partial a} = 0 : G = -2 \sum_{i=1}^{N} \frac{x_i - x(t_i, a)}{\sigma_i} \left( \frac{\partial x}{\partial a} \right)_{t_i} = 0
    \]

Nonlinear algebraic equation to be solved by Newton–Raphson (\(\beta = -G/2\), \(\alpha = -\partial G/\partial a\)):

\[
\alpha \Delta a = -\beta; \quad \Delta x = \frac{\partial G}{\partial a} \approx \frac{1}{\sigma_i^2} \left[ \frac{\partial^2 x}{\partial a^2} \right]_{t_i} \Delta a
\]

with \(\frac{\partial^2 x}{\partial a^2}\) term ignored in Hessian matrix. Now start with guessed \(a\). improve it iteratively by correction \(\Delta a\).

- But with ODE model, we don't have \(x(t_i, a)\), and we don't have \(\partial x/\partial a|_{t_i}\). What to do?

* Get both through integrating/solving ODES.
Take \( \frac{\partial}{\partial a} \) of model \( \frac{dx}{dt} = f(x, t, a) \):

\[
\frac{\partial}{\partial a} \left( \frac{dx}{dt} \right) = \frac{\partial f}{\partial a} + \frac{\partial f}{\partial x} \cdot \frac{dx}{da} \quad \frac{d}{dt} \left( \frac{dx}{da} \right) = \frac{\partial f}{\partial a} + \frac{\partial f}{\partial x} \cdot \left( \frac{dx}{da} \right)
\]

Calling \( \frac{dx}{da} = y(t_*, t, a) \) a new unknown function, we have:

\[
\dot{x} = \begin{pmatrix} x \end{pmatrix}, \quad \frac{d}{dt} \begin{pmatrix} \dot{x} \end{pmatrix} = \begin{pmatrix} f \\ \frac{\partial f}{\partial a} + \frac{\partial f}{\partial x} \cdot y \end{pmatrix}
\]

\[
z(t_0) = \begin{pmatrix} x_0 \\ 0 \end{pmatrix} \quad \text{since } \frac{d}{da} (y(t_0, a)) = \frac{d}{da} (x_0) = 0.
\]

Therefore, we can get \( x(t) \) and \( \frac{dx}{dt} = y(t) \) at all \( t_i \) values, based on a guessed \( a \) values, and then solve the Newton-Raphson problem for \( da \). This is the Gauss-Newton Method.

- Algorithm:

1. (i) Guess a value
2. (ii) Integrate ODE's for \( x \) and \( y \). At each \( t_i \), evaluate \( x(t_i, a) \) and \( y(t_i, a) \)
3. (iii) Set up linear equation for \( da \):

\[
A \cdot \delta a = \beta
\]

\[
A = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left( \frac{\partial x}{\partial a} \right)_{t_i} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} y_i^2
\]

\[
\beta = +\sum_{i=1}^{N} \frac{x_i - x(t_i, a)}{\sigma_i^2} \cdot y_i \quad \text{[Left out factor of 2]}
\]

4. (iv) Update \( a \) by \( a + \delta a \rightarrow a \)

5. Check \( \chi^2(a + \delta a) \) to see if it's smaller refinement \* than \( \chi^2(a) \). Requires solving for \( x(t, a + \delta a) \)

\* If not half \( \delta a \): \( a + \frac{1}{2} \delta a \). Repeat.

6. (v) Iterate to convergence: \( |\delta a/a| < \varepsilon \)
(b) Single equation, multiple parameters.

\[ \frac{dx}{dt} = f(x, t, a_1, a_2, \ldots, a_M), \quad x(t_0) = x_0. \]

To match N data points \((x_i, t_i), \quad i = 1, \ldots, N.\)

Follow previous procedure; \(a = (a_1, \ldots, a_M)\)

\[
\beta = -\frac{1}{2} \frac{\partial^2 x}{\partial a_k^2} = \sum_{i=1}^{N} \frac{Z_i - x(t_i, a)}{\sigma_i^2} \frac{\partial x(t_i)}{\partial a_k}
\]

\[
\alpha_{kl} = -\frac{\partial \beta_k}{\partial a_l} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \frac{\partial x(t_i)}{\partial a_k} \frac{\partial x(t_i)}{\partial a_l} \quad \text{[Ignored]}
\]

\(\Rightarrow \quad \alpha_k \cdot \sigma_k = \beta, \quad M \times M \text{ linear system.}\)

But first need to get \(x(t_i, a)\) and \(y_k = \frac{\partial x(t_i, a)}{\partial a_k}.\)

\[
\begin{bmatrix}
  x_1 \\
  y_1 \\
  \vdots \\
  y_M
\end{bmatrix}
\]

\[ \frac{d^2}{dt^2} = \begin{bmatrix}
  f \\
  \frac{\partial f}{\partial a_1} + \frac{\partial f}{\partial x} y_1 \\
  \vdots \\
  \frac{\partial f}{\partial a_M} + \frac{\partial f}{\partial x} y_M
\end{bmatrix}, \quad \text{Mth ODEs, with initial conditions}
\]

Same algorithm as before, except that \(\sigma_k\) now involves a linear system, and the ODE solution now for \(M+1\) equations.

Example: Consider estimation of rate constants \(k_1, k_2\) in Bodenstein-Linder model for homogeneous gas phase reaction \(2NO + O_2 \leftrightarrow 2NO_2:\)

\[ \frac{dx}{dt} = k_1 (x - x)^2 (\beta - x) - k_2 x^2, \quad x(0) = 0, \]

with \(\alpha = 126.2, \quad \beta = 91.9, \quad x\) being concentration of \(NO_2\).

Choose \(k_1\) and \(k_2\) to match data set:

\[
\begin{array}{cccccccc}
  t_i & 0 & 1 & 2 & \cdots & 24 & 29 & 39 \\
  x_i & 0 & 1.4 & 6.3 & \cdots & 41.6 & 43.5 & 45.3
\end{array}
\]
Solution: Now \( f = k_1(\alpha-x)(\beta-x)^2 - k_2 x^2 \) has 2 parameters. Note that \( \alpha, \beta \) are 2 numbers, different from our Hessian matrix.

\[
\begin{align*}
Z &= \begin{pmatrix} \frac{\partial}{\partial k_1} x \vspace{1mm} \\
\frac{\partial}{\partial k_2} \end{pmatrix}, \quad \frac{\partial f}{\partial x} = -k_1(\beta-x)^2 - 2k_1(\alpha-x)(\beta-x) - 2k_2 x \\
\frac{\partial f}{\partial k_1} &= (\alpha-x)(\beta-x)^2, \quad \frac{\partial f}{\partial k_2} = -x^2
\end{align*}
\]

\[
\frac{d^2 f}{dt} = \begin{bmatrix}
k_1(\alpha-x)(\beta-x)^2 - k_2 x^2 \\
(\alpha-x)(\beta-x)^2 + \frac{\partial f}{\partial x} \cdot y_1 \\
x^2 + \frac{\partial f}{\partial x} \cdot y_2
\end{bmatrix}, \quad y_1 = \frac{\partial x}{\partial k_1}, \quad y_2 = \frac{\partial x}{\partial k_2}
\]

Write code to do this problem \( \implies \) get best estimation for \( k_1, k_2 \). [New assignment?]

(c) Multiple equations with multiple parameters.

\[
X = (x_1, \ldots, x_n) \to \text{index } j = 1, 2, \ldots, n
\]
\[
\alpha = (\alpha_1, \ldots, \alpha_m) \to \text{index } k = 1, 2, \ldots, m
\]
\[ (x_i, t_i) : N \text{ data points} \to \text{index } i = 1, 2, \ldots, N \]

\[
\frac{dx}{dt} = f(x, t, \alpha), \ n \text{ equations}; \ x(t) = x_0
\]

\[
J = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ (x_i - x(t_i; \alpha))^2 \right] = \frac{1}{\sigma_i^2} \left[ x_i - x(t_i) \right] \cdot \left[ x_i - x(t_i) \right]^T
\]

\[
\frac{\partial^2 J}{\partial \alpha_k} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ \frac{\partial x_i}{\partial \alpha_k} \right] \cdot 2 \left[ x_i - x(t_i) \right] \left( \frac{\partial x_i}{\partial \alpha_k} \right)
\]

Define \( n \times m \) matrix \( G_{kj} = \frac{\partial x_j}{\partial \alpha_k} \), "Sensitivity matrix"

Then \( \left( \frac{\partial x_i}{\partial \alpha_k} \right) = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ x_i - x(t_i) \right] \cdot G, \quad \text{\( 1 \times m \) vector} \)

Now \( \beta_k = -\frac{1}{2} \frac{\partial x_i}{\partial \alpha_k} \sum_{i=1}^{N} \frac{1}{\sigma_i} \left[ x_i - x(t_i) \right] \cdot G \)

\[
\alpha_{kj} = -\frac{\partial x_k}{\partial \alpha_k} = \sum_{i=1}^{N} \frac{1}{\sigma_i} G^T G, \ \text{\( m \times m \) matrix} \)
Again we're going to solve for $\delta a$ by 

$$\varepsilon \cdot \delta a = \beta$$

But first, we need to integrate ODE's for $x(t, a)$ and $G(t, a)$:

$$\frac{dx}{dt} = \frac{df}{da} + \frac{df}{dx} \cdot \frac{dx}{da} = \frac{df}{da} + \frac{df}{dx} \cdot \delta$$

$$G(t_0) = 0.$$ 

Writing $G = [g_1, g_2, \ldots, g_m] = [\frac{dx}{da_1}]$ into $m$ columns of $n$-dimensional vectors, then append $x$ to the left:

$$\bar{z} = [x \mid g_1 \mid \cdots \mid g_m]$$

such that

$$\left\{ \begin{array}{l}
\frac{d\bar{z}}{dt} = \left[ \frac{df}{da_1} + \frac{df}{dx} \cdot g_1, \ldots, \frac{df}{da_k} + \frac{df}{dx} \cdot g_k, \ldots, \frac{df}{da_n} + \frac{df}{dx} \cdot g_m \right] \\
\bar{z}(t_0) = \left[ x_0, 0, \ldots, 0 \right]
\end{array} \right.$$