02610
Optimization and Data Fitting
Week 7: Nonlinear Least-Squares Problems

Yiqiu Dong

DTU Compute
Technical University of Denmark

19 October 2020

Lecture Material

- J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd Edition, Springer.
  - Chapter 10: Least-squares problems.
  - We cover: 10.3.
- P. C. Hansen, V. Pereyra and G. Scherer, *Least Squares Data Fitting with Applications*, Johns Hopkins University Press.
  - Chapter 8: Nonlinear least squares problems.
  - We cover: 8.1 and 8.2.
  - Chapter 9: Algorithms for solving nonlinear LSQ problems.
  - We cover: 9.1, 9.2, and 9.3.

Nonlinearity

- A parameter α of the function appears nonlinearly, if the derivative $\partial f / \partial \alpha$ is a function of α.
- The model $\phi(\mathbf{x}, t)$ is nonlinear, if at least one of the parameters in $\mathbf{x}$ appear nonlinearly.

Nonlinearity

- A parameter α of the function appears nonlinearly, if the derivative $\partial f/\partial \alpha$ is a function of α.
- The model $\phi(\mathbf{x}, t)$ is nonlinear, if at least one of the parameters in $\mathbf{x}$ appear nonlinearly.

Example: Consider the exponential decay model

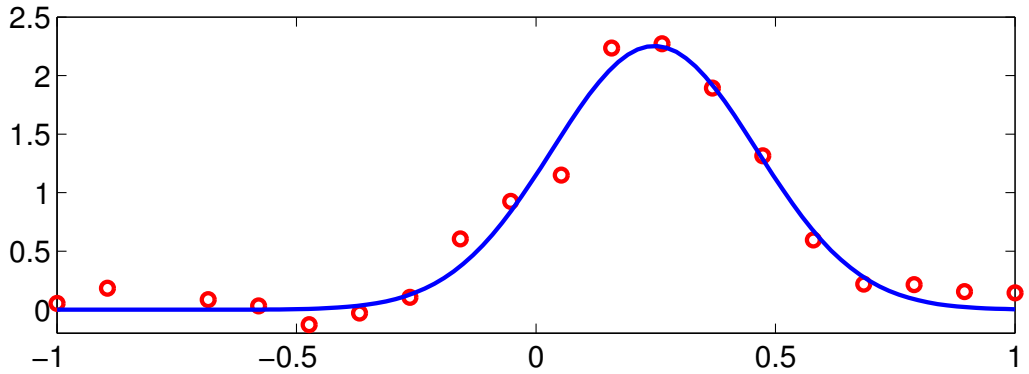
$$\phi(x_1, x_2, t) = x_1 e^{-x_2 t}.$$

We have:

- $\partial\phi/\partial x_1 = e^{-x_2 t}$ which is independent of x_1,
- $\partial\phi/\partial x_2 = -t\, x_1 e^{-x_2 t}$ which depends on x_2.

Thus ϕ is a nonlinear model with the parameter x_2 appearing nonlinearly.

Fitting with a Gaussian Model



The non-normalized Gaussian function:

$$\phi(\mathbf{x}, t) = x_1 e^{-(t-x_2)^2/(2x_3^2)}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

where x_1 is the amplitude, x_2 is the time shift, and x_3 determines the width of the Gaussian function.
The parameters x_2 and x_3 appear nonlinearly in this model.
Gaussian models also arise in many other data fitting problems.

Nonlinear Least Squares Problem

Find a parameter vector $\mathbf{x}^*$ which minimizes the sum-of-squares of the residuals:

$$\boxed{\min_{\mathbf{x}} f(\mathbf{x}) \equiv \min_{\mathbf{x}} \tfrac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2 = \min_{\mathbf{x}} \tfrac{1}{2} \sum_{i=1}^{m} (y_i - \phi(\mathbf{x}, t_i))^2,}$$

where $\mathbf{x} \in \mathbb{R}^n$ and, as usual,

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} r_1(\mathbf{x}) \\ \vdots \\ r_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} y_1 - \phi(\mathbf{x}, t_1) \\ \vdots \\ y_m - \phi(\mathbf{x}, t_m) \end{bmatrix} \in \mathbb{R}^m,$$

- Here y_i are the measured data corresponding to t_i.
- The nonlinearity arises **only** from $\phi(\mathbf{x}, t)$.

The Jacobian and the Gradient of $f(\mathbf{x})$

The Jacobian $J(\mathbf{x})$ of the vector function $\mathbf{r}(\mathbf{x})$ is defined as the matrix with elements

$$[J(\mathbf{x})]_{ij} = \frac{\partial r_i(\mathbf{x})}{\partial x_j} = -\frac{\partial \phi(\mathbf{x}, t_i)}{\partial x_j}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

The ith row of $J(\mathbf{x})$ equals the transpose of the gradient of $r_i(\boldsymbol{x})$:

$$[J(\mathbf{x})]_{i,:} = \nabla r_i(\mathbf{x})^T = -\nabla \phi(\mathbf{x}, t_i)^T, \qquad i = 1, \ldots, m.$$

Thus the elements of the gradient of $f(\mathbf{x})$ are given by

$$[\nabla f(\mathbf{x})]_j = \frac{\partial f(\mathbf{x})}{\partial x_j} = \sum_{i=1}^{m} r_i(\mathbf{x}) \frac{\partial r_i(\mathbf{x})}{\partial x_j}$$

and it follows that the gradient is the vector

$$\boxed{\nabla f(\mathbf{x}) = J(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \ .}$$

The Hessian Matrix of $f(\mathbf{x})$

The elements of the Hessian of f, denoted $\nabla^2 f(\mathbf{x})$, are given by

$$[\nabla^2 f(\mathbf{x})]_{k\ell} = \frac{\partial^2 f(\mathbf{x})}{\partial x_k \partial x_\ell} = \sum_{i=1}^{m} \frac{\partial r_i(\mathbf{x})}{\partial x_k} \frac{\partial r_i(\mathbf{x})}{\partial x_\ell} + \sum_{i=1}^{m} r_i(\mathbf{x}) \frac{\partial^2 r_i(\mathbf{x})}{\partial x_k \partial x_\ell},$$

and it follows that the Hessian can be written as

$$\boxed{\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}),}$$

where

$$\begin{aligned} \left[\nabla^2 r_i(\mathbf{x})\right]_{k\ell} &= -\left[\nabla^2 \phi(\mathbf{x}, t_i)\right]_{k\ell} \\ &= -\frac{\partial^2 \phi(\mathbf{x}, t_i)}{\partial x_k \partial x_\ell}, \qquad k, \ell = 1, \ldots, n\ . \end{aligned}$$

The Optimality Conditions

The conditions for $\mathbf{x}^*$ to be a local minimum of a twice continuously differentiable function f are

- **First-order necessary condition**:

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^T \mathbf{r}(\mathbf{x}) = \mathbf{0} \ .$$

- **Second-order sufficient condition**:

$$\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}) \quad \text{is positive definite.}$$

The Optimality Conditions

The conditions for $\mathbf{x}^*$ to be a local minimum of a twice continuously differentiable function f are

- **First-order necessary condition**:

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^T \mathbf{r}(\mathbf{x}) = \mathbf{0} \ .$$

- **Second-order sufficient condition**:

$$\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}) \quad \text{is positive definite.}$$

Remark

- The first term of the Hessian contains only $J(\mathbf{x})$, i.e., only first derivatives!
- If the residuals are close to affine or are small near the solution, the second term will be very small.
- When the first term is dominant, one gets the main part of the Hessian "for free".

Newton's Method

If $f(\mathbf{x})$ is twice continuously differentiable, based on the second-order Taylor series approximation to $f(\mathbf{x}_k + \mathbf{p})$, we get the Newton iteration: For $k = 0, 1, 2, \ldots$

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \left(\nabla^2 f(\mathbf{x}_k)\right)^{-1} \nabla f(\mathbf{x}_k) \\ &= \mathbf{x}_k - \left(J(\mathbf{x}_k)^T J(\mathbf{x}_k) + S(\mathbf{x}_k)\right)^{-1} J(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k), \end{aligned}$$

where $S(\mathbf{x}_k)$ denotes the matrix

$$S(\mathbf{x}_k) = \sum_{i=1}^{m} r_i(\mathbf{x}_k) \nabla^2 r_i(\mathbf{x}_k).$$

Convergence. Locally quadratic convergence, but expensive – requires mn^2 derivatives to evaluate $S(\mathbf{x}_k)$.

The Gauss-Newton Method

If the residuals are close to affine or very small when near to the solution, a good alternative is to neglect the second term $S(\mathbf{x}_k)$ of the Hessian. It turns out to the Gauss-Newton method, where $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k^{\mathrm{GN}}$ with $\mathbf{p}_k^{\mathrm{GN}}$ obtained by solving

$$\left(J(\mathbf{x}_k)^T J(\mathbf{x}_k)\right) \mathbf{p}_k^{\mathrm{GN}} = -J(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k).$$

If $J(\mathbf{x}_k)$ has full rank

- The linear system of $\mathbf{p}_k^{\mathrm{GN}}$ is actually the normal equations for the linear least squares problem

$$\min_{\mathbf{p}_k^{\mathrm{GN}}} \left\| J(\mathbf{x}_k)\, \mathbf{p}_k^{\mathrm{GN}} - (-\mathbf{r}(\mathbf{x}_k)) \right\|_2^2 .$$

- $\mathbf{p}_k^{\mathrm{GN}}$ is a descent step.

Local Linear Approximation

If we introduce a Taylor expansion to approximate the nonlinear LSQ problem, then

$$\begin{aligned}\min_{\Delta \mathbf{x}_k} f(\mathbf{x}_k + \Delta \mathbf{x}_k) &= \min_{\Delta \mathbf{x}_k} \frac{1}{2}\|\mathbf{r}(\mathbf{x}_k + \Delta \mathbf{x}_k)\|_2^2 \\ &\approx \min_{\Delta \mathbf{x}_k} \frac{1}{2}\|\mathbf{r}(\mathbf{x}_k) + J(\mathbf{x}_k)\Delta \mathbf{x}_k\|_2^2.\end{aligned}$$

It is the same as the Gauss-Newton step in the full-rank case. Hence, the Gauss-Newton method is solving a linear approximation of the original nonlinear LSQ problem.

Damped Gauss-Newton Method

Implementations of the G-N method usually perform a line search in the direction $\mathbf{p}_k^{\mathrm{GN}}$, e.g., requiring the step length α_k to satisfy the Armijo condition:

$$\begin{aligned} f(\mathbf{x}_k + \alpha_k \mathbf{p}_k^{\mathrm{GN}}) &< f(\mathbf{x}_k) + c_1\, \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k^{\mathrm{GN}} \\ &= f(\mathbf{x}_k) + c_1\, \alpha_k \mathbf{r}(\mathbf{x}_k)^T J(\mathbf{x}_k) \mathbf{p}_k^{\mathrm{GN}}, \end{aligned}$$

with a constant $c_1 \in (0, 1)$.

This ensures that the reduction is (at least) proportional to both the parameter α_k and the directional derivative $\nabla f(\mathbf{x}_k)^T \mathbf{p}_k^{\mathrm{GN}}$.

Line search make the algorithm (often) globally convergent.
Convergence. Can be quadratic if the neglected term in the Hessian is small. Otherwise it is linear.

Gauss-Newton Algorithm

Set the starting point $\mathbf{x}_0$

loop

Solve

$$\min_{\mathbf{p}} \|J(\mathbf{x}_k)\,\mathbf{p} + \mathbf{r}(\mathbf{x}_k)\|_2$$

to obtain the search direction $\mathbf{p}_k^{\mathrm{GN}}$;

Choose a step length α_k so that there is enough descent;

Calculate the new iterate: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k^{\mathrm{GN}}$;

Check for convergence;

end loop

Output $\mathbf{x}_{k+1}$.

Example

We fit the one-parameter model $\phi(x,t) = e^{xt}$ to the data set

$$(t_1, y_1) = (1, 2), \quad (t_2, y_2) = (2, 4), \quad (t_3, y_3) = (3, y),$$

where y can take one of the four values 8, 3, -1, -8. The least squares problem is, for every y, to determine the single parameter x by minimizing

$$f(x) = \frac{1}{2}\sum_{i=1}^{3} r_i(x)^2 = \frac{1}{2}\left[(e^x - 2)^2 + (e^{2x} - 4)^2 + (e^{3x} - y)^2\right].$$

Then, the two terms in the Hessian are

$$J(x)^\top J(x) = \sum_{i=1}^{3}(t_i e^{xt_i})^2 \qquad S(x) = \sum_{i=1}^{3} r_i(x) t_i^2 e^{xt_i}.$$

Example

y	x^*	$f(x^*)$	$J(x^*)^T J(x^*)$	$S(x^*)$
8	0.6932	0	644.00	0
3	0.4401	1.639	151.83	9.0527
-1	0.0447	6.977	17.6492	8.3527
-8	-0.7915	41.145	0.4520	2.9605

		Newton		Gauss-Newton	
y	x_0	# iter.	rate	# iter.	rate
8	1	7	quadratic	5	quadratic
	0.6	6	quadratic	4	quadratic
3	1	9	quadratic	12	linear
	0.5	5	quadratic	9	linear
-1	1	10	quadratic	34	linear (slow)
	0	4	quadratic	32	linear (slow)
-8	1	12	quadratic	no convergence	
	-0.7	4	quadratic	no convergence	

The Levenberg-Marquardt Method

Similar to Gauss-Newton method, except that we replace the line search with a trust-region strategy where the norm of the step is bounded by the trust-region radius Δ_k.

$$\min \|J(\mathbf{x}_k)\,\mathbf{p} + r(\mathbf{x}_k)\|_2^2 \qquad \text{subject to} \qquad \|\mathbf{p}\|_2 \leq \Delta_k.$$

The Levenberg-Marquardt Method

Similar to Gauss-Newton method, except that we replace the line search with a trust-region strategy where the norm of the step is bounded by the trust-region radius Δ_k.

$$\min \|J(\mathbf{x}_k)\,\mathbf{p} + r(\mathbf{x}_k)\|_2^2 \qquad \text{subject to} \qquad \|\mathbf{p}\|_2 \leq \Delta_k.$$

We can handle such an inequality constraint via the use of a Lagrange multiplier, and it turn out the step in Levenberg-Marquardt's method:

$$\mathbf{p}_k^{\mathrm{LM}} = \operatorname{argmin}_{\mathbf{p}} \left\{ \|J(\mathbf{x}_k)\,\mathbf{p} + \mathbf{r}(\mathbf{x}_k)\|_2^2 + \lambda_k \|\mathbf{p}\|_2^2 \right\}$$

where $\lambda_k > 0$ is a so-called *Lagrange parameter* for the constraint at the kth iteration.

The Levenberg-Marquardt Method

The optimization problem in each iteration

$$\min_{\mathbf{p}} \left\{ \|J(\mathbf{x}_k)\,\mathbf{p} + \mathbf{r}(\mathbf{x}_k)\|_2^2 + \lambda_k \|\mathbf{p}\|_2^2 \right\}$$

is equivalent to two forms:

$$\min_{\mathbf{p}} \left\| \begin{bmatrix} J(\mathbf{x}_k) \\ \sqrt{\lambda_k} I \end{bmatrix} \mathbf{p} - \begin{bmatrix} -\mathbf{r}(\mathbf{x}_k) \\ \mathbf{0} \end{bmatrix} \right\|_2^2$$

and

$$\left(J(\mathbf{x}_k)^\top J(\mathbf{x}_k) + \lambda_k I \right) \mathbf{p} = -J(\mathbf{x}_k)^\top \mathbf{r}(\mathbf{x}_k).$$

This method is more robust in case of an ill-conditioned Jacobian, and can also handle a rank-deficient Jacobian.

Levenberg-Marquardt Algorithm

Set the starting point $\mathbf{x}_0$

loop

Choose the Lagrange parameter λ_k;

Solve the linear LSQ problem

$$\min_{\mathbf{p}} \left\| \begin{bmatrix} J(\mathbf{x}_k) \\ \sqrt{\lambda_k} I \end{bmatrix} \mathbf{p} - \begin{bmatrix} -\mathbf{r}(\mathbf{x}_k) \\ \mathbf{0} \end{bmatrix} \right\|_2^2$$

to obtain the step $\mathbf{p}_k^{\mathrm{LM}}$.

Calculate the new iterate: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k^{\mathrm{LM}}$;

Check for convergence;

end loop

Output $\mathbf{x}_{k+1}$.

Note: there is no line search (i.e., no α_k-parameter), its role is taken over by the Lagrange parameter λ_k.

The Role of Lagrange Parameter

Consider the Levenberg-Marquardt step, which we formally write as:

$$\mathbf{p}_k^{\mathrm{LM}} = -\left(J(\mathbf{x}_k)^T J(\mathbf{x}_k) + \lambda_k I\right)^{-1} J(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k).$$

The parameter λ_k influences both the direction and the length of the step.

- If λ_k is close to 0, then $\mathbf{p}_k^{\mathrm{LM}}$ tends to the Gauss-Newton step.
- If λ_k is very large, then $\mathbf{p}_k^{\mathrm{LM}}$ is a short step approximately in the steepest descent direction.

How to Choose Lagrange Parameter

A strategy developed by Marquardt. The underlying principles are:

1. The initial value $\lambda_0 \approx \|J(\mathbf{x}_0)^T J(\mathbf{x}_0)\|_2$.
2. For subsequent steps, an improvement ratio is defined as:

$$\rho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})}{\frac{1}{2}(\mathbf{p}_k^{\mathrm{LM}})^\top(\lambda_k \mathbf{p}_k^{\mathrm{LM}} - J(\mathbf{x}_k)^\top \mathbf{r}(\mathbf{x}_k))}.$$

Here, the denominator is the reduction in f predicted by the local linear model.

How to Choose Lagrange Parameter

A strategy developed by Marquardt. The underlying principles are:

1. The initial value $\lambda_0 \approx \|J(\mathbf{x}_0)^T J(\mathbf{x}_0)\|_2$.
2. For subsequent steps, an improvement ratio is defined as:

$$\rho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})}{\frac{1}{2}(\mathbf{p}_k^{\mathrm{LM}})^\top(\lambda_k \mathbf{p}_k^{\mathrm{LM}} - J(\mathbf{x}_k)^\top \mathbf{r}(\mathbf{x}_k))}.$$

Here, the denominator is the reduction in f predicted by the local linear model.

- If ρ_k is large, then the pure Gauss-Newton model is good enough, so λ_{k+1} can be made smaller than at the previous step.
- If ρ_k is small (or even negative), then a short steepest descent step should be used, i.e., λ_{k+1} should to be increased.

Marquardt's Parameter Updating

Algorithm

if $\rho_k > 0.75$ **then**
  Set $\lambda_{k+1} = \lambda_k/3$;
else if $\rho_k < 0.25$ **then**
  Set $\lambda_{k+1} = 2\,\lambda_k$;
else
  Use $\lambda_{k+1} = \lambda_k$;
end if

if $\rho_k > 0$ **then**
  Accept the update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \mathbf{p}_k^{\mathrm{LM}}$;
else
  Reject the update: $\mathbf{x}_{k+1} = \mathbf{x}_k$;
end if

Convergence. The Levenberg-Marquardt method is (often) globally convergent. It can reach quadratic convergent rate, if the neglected term in the Hessian is small. Otherwise, it is linear.

Marquardt's Parameter Updating

Algorithm

if $\rho_k > 0.75$ **then**
  Set $\lambda_{k+1} = \lambda_k/3$;
else if $\rho_k < 0.25$ **then**
  Set $\lambda_{k+1} = 2\,\lambda_k$;
else
  Use $\lambda_{k+1} = \lambda_k$;
end if

if $\rho_k > 0$ **then**
  Accept the update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \mathbf{p}_k^{\mathrm{LM}}$;
else
  Reject the update: $\mathbf{x}_{k+1} = \mathbf{x}_k$;
end if

Convergence. The Levenberg-Marquardt method is (often) globally convergent. It can reach quadratic convergent rate, if the neglected term in the Hessian is small. Otherwise, it is linear.

Example

We fit the one-parameter model $M(x, t) = e^{xt}$ to the data set

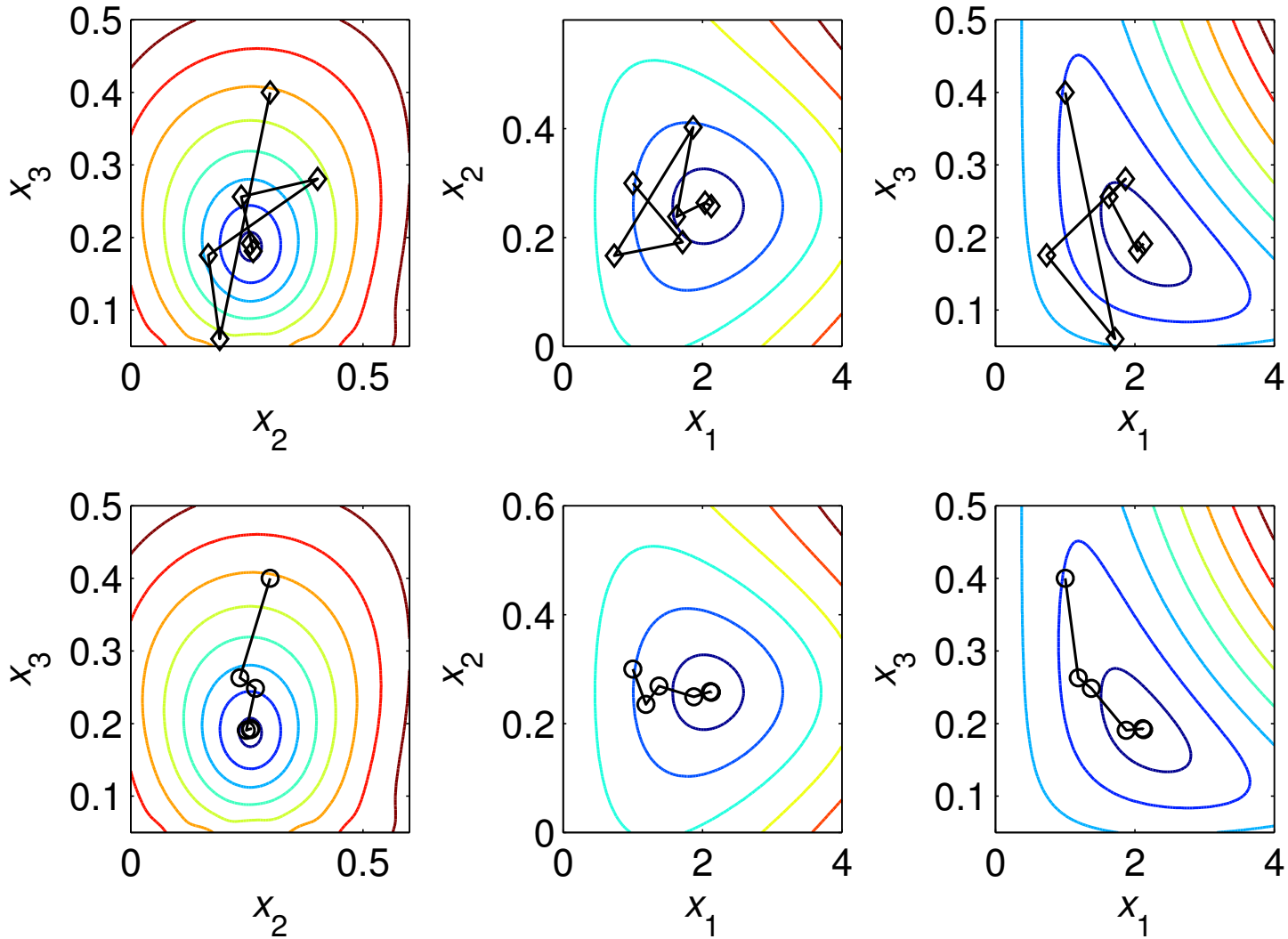
$$(t_1, y_1) = (1, 2), \quad (t_2, y_2) = (2, 4), \quad (t_3, y_3) = (3, y).$$

The least squares problem is minimizing

$$f(x) = \frac{1}{2}\sum_{i=1}^{3} r_i(x)^2 = \frac{1}{2}\left[(e^x - 2)^2 + (e^{2x} - 4)^2 + (e^{3x} - y)^2\right].$$

		Gauss-Newton		Levenberg-Marquardt	
y	x_0	# iter.	rate	# iter.	rate
8	1	5	quadratic	10	quadratic
	0.6	4	quadratic	7	quadratic
3	1	12	linear	13	linear
	0.5	9	linear	10	linear
-1	1	34	linear (slow)	26	linear (slow)
	0	32	linear (slow)	24	linear (slow)
-8	1	no convergence		125	linear (very slow)
	-0.7	no convergence		120	linear (very slow)

G-N no damping (top) vs. L-M (bottom)



Conclusion

Characteristics	Newton	G-N	L-M
Ill-conditioned Jacobian	yes	yes (but slow)	yes
Rank-deficient Jacobian	yes	no	yes
Convergence $S(\boldsymbol{x}_k) = 0$	quadratic	quadratic	quadratic
Convergence $S(\boldsymbol{x}_k)$ small	quadratic	linear	linear
Convergence $S(\boldsymbol{x}_k)$ large	quadratic	slow or none	slow or none

MATLAB Optimization Toolbox: `lsqnonlin`

`[x,resnorm] = lsqnonlin(fun, x0, lb, ub, options)` solves nonlinear LSQ the problem

$$\min_{\mathbf{x}} \|\mathbf{f}(\mathbf{x})\|_2^2 = \min_{\mathbf{x}} \left(f_1(\mathbf{x})^2 + \cdots + f_m(\mathbf{x})^2\right)$$

Inputs:

- `fun`: a function to compute the *vector-valued* function

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \cdots, f_m(\mathbf{x})]^T;$$

- `x0`: an starting point ;
- `lb` and `ub`: lower and upper bounds on $\mathbf{x}$, so that $\mathrm{lb} \leq \mathbf{x} \leq \mathrm{ub}$.
- `options`: optimization options. For example, change display format, tolerance, algorithms, user given Jacobian, etc.