Course of Numerical Methods for Engineering
Lab 13

Luca Bonaventura, Tommaso Benacchio

MOX - Politecnico di Milano

PHYS-ENG, A.Y. 2020-21

9/12/2020

Topic of this session:

- **Nonlinear least square problems**

Nonlinear least square problem

Definition

Let $\mathrm{f} : \mathrm{R}^n \to \mathrm{R}^m$ **be a vector field** $\mathrm{x} \in \mathrm{R}^n$ **and** $\mathrm{b} \in \mathrm{R}^m$**, with** $m > n$**. The vector** $\mathrm{x}^* \in \mathrm{R}^n$ **is called a solution of the nonlinear least square problem associated with** f **and** b **if it is such that**

$$\|\mathrm{f}(\mathrm{x}^*) - \mathrm{b}\|_2 = \min_{\mathrm{x}\in\mathrm{R}^n} \|\mathrm{f}(\mathrm{x}) - \mathrm{b}\|_2 = \min_{\mathrm{x}\in\mathrm{R}^n} \phi(\mathrm{x}).$$

Definition

*(***Gauss-Newton method***)* **Let** $\mathrm{x}^{(0)}$ **an initial approximation for the minimum of** ϕ**. The Gauss-Newton method is defined by**

$$\begin{aligned} \mathrm{J_f}^T(\mathrm{x}^{(k)})\mathrm{J_f}(\mathrm{x}^{(k)})\boldsymbol{\delta}\mathrm{x}^{(k)} &= \mathrm{J_f}^T(\mathrm{x}^{(k)})(\mathrm{b} - \mathrm{f}(\mathrm{x}^{(k)})) \\ \mathrm{x}^{(k+1)} &= \mathrm{x}^{(k)} + \boldsymbol{\delta}\mathrm{x}^{(k)} \quad k \geq 0. \end{aligned}$$

Levenberg-Marquardt method

Definition

Let $x^{(0)}$ an initial approximation for the minimum of ϕ and λ_0 an initial approximation for the damping term. The Levenberg-Marquardt method is defined by the following procedure:

- **for $k \geq 0$, compute**

$$\begin{aligned} \left[J_f^T(x^{(k)}) J_f(x^{(k)}) + \lambda_k I \right] \boldsymbol{\delta} x^{(k)} &= J_f^T(x^{(k)})(b - f(x^{(k)})) \\ x^{(k+1)} &= x^{(k)} + \boldsymbol{\delta} x^{(k)} \end{aligned}$$

- **if $\phi(x^{(k+1)}) < \phi(x^{(k)})$, determine $\lambda_{k+1} \leq \lambda_k$**
- **if $\phi(x^{(k+1)}) > \phi(x^{(k)})$, determine $\lambda_{k+1} > \lambda_k$ and repeat iteration k.**

MATLAB/Octave:

```
options_lm = optimoptions(@lsqnonlin,'Algorithm','levenberg-marquardt');
x_lm = lsqnonlin(f(x)-b,x0,[],[],options_lm)
```

Trust region minimization method

Definition

Given initial estimate $x^{(0)}, \Delta_0$ a trust region method to find a minimum of $\phi(x)$ builds a sequence of approximate objective functions $\phi_k(x)$, approximate minima $x^{(k)}$ and of trust regions $\Omega_k = \{d | \|d\| \leq \Delta_k\}$ of radius Δ_k such that $x^{(k+1)} = x^{(k)} + d^{(k)}$ and

$$\phi_k(x^{(k)} + d^{(k)}) = \min\left\{\phi_k(x^{(k)} + d) | d \in \Omega_k\right\}.$$

MATLAB/Octave:

```
x_base = lsqnonlin(f(x)-b,x0,[],[])
```

Exercise 1

Consider the nonlinear function:

$$f(x,y) = 15 + x + 2y - \frac{80}{L_x^2}\left[\left(x - \frac{L_x}{2}\right)^2 + \left(y - \frac{L_y}{2}\right)^2\right] + \frac{300}{L_y^4}\left[\left(x - \frac{L_x}{2}\right)^4 + \left(y - \frac{L_y}{2}\right)^4\right]$$

in the domain $x, y \in [0, L_x] \times [0, Ly]$**. For** $L_x = L_y = 2$**, initial guess** $x = 0.5, y = 0.5$**, and up to a** 10^{-8} **tolerance, find the global minimum of** f**:**

(a) **Using the steepest descent method with fixed step length** $\gamma = 0.001$**;**

(b) **Using the modified gradient method;**

(c) **Using Newton's method with exact Jacobian.**

Compare the number of iterations and time to convergence for the three cases. Experiment with different values of the tolerance and different initial guesses.

Exercise 2

Consider the nonlinear least square problem in which the data vector b **contains the values of the function** $h(z) = z^3 + z^2 - 1$ **sampled on a uniform mesh of step 0.01 on the interval** $[1,3]$ **and to which a Gaussian noise of mean zero and standard deviation** 0.5 **has been added. Use as fitting function** $\mathrm{f}(\mathrm{x})$ **the vector field whose components are**

$$f_i(x_1, x_2, x_3) = z_i^{x_1} + z_i^{x_2} + x_3,$$

where z_i **denotes the** $i-$**th component of the vector** $[1, 0.01, \ldots, 3]^T$**. Solve the problem:**

(a) **with the MATLAB command** `lsqnonlin`**, using the default algorithm and** $\mathrm{x}_0 = [1,1,1]^T$**;**

(b) **with the MATLAB command** `lsqnonlin`**, using the Levenberg-Marquardt algorithm and the same** x_0**;**

(c) **solving the nonlinear equation** $\mathrm{g}(\mathrm{x}) = \mathrm{J_f}^T(\mathrm{x})(\mathrm{f}(\mathrm{x}) - \mathrm{b}) = 0$ **with the MATLAB command** `fsolve`**,** $\mathrm{x}_0 = [2.5, 1, 1]^T$ **and a** 10^{-8} **tolerance on** x**.**

(d) **solving the nonlinear equation** $\mathrm{g}(\mathrm{x}) = \mathrm{J_f}^T(\mathrm{x})(\mathrm{f}(\mathrm{x}) - \mathrm{b}) = 0$ **with the Newton method with finite difference approximate Jacobian of** $\mathrm{g}(\mathrm{x})$ **with increment** $\delta = 0.01$**, a** 10^{-8} **tolerance, and** $\mathrm{x}_0 = [2.8, 1, 1]^T$**.**

In each case, display in a single plot the function h**, the data vector** b **and the function** $\mathrm{f}(\mathrm{x}_s)$ **where** x_s **is the solution. Check what happens when** $\mathrm{x}_0 = [1,1,1]^T$ **in points (c) and (d).**

Exercise 3

Consider the nonlinear least square problem in which the data vector b **contains the values of the function** $h(z) = \cos(\pi z) - 2\cos(5\pi z) + \cos(6\pi z)$ **sampled on a uniform mesh of step 0.001 on the interval** $[0, 2]$ **and to which a Gaussian noise of mean zero and standard deviation** 0.5 **has been added. Use as fitting function** $\mathrm{f}(\mathrm{x})$ **the vector field whose components are**

$$f_i(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 \cos(x_2 \pi z_i) + x_3 \cos(x_4 \pi z_i) + x_5 \cos(x_6 \pi z_i),$$

where z_i **denotes the** $i-$**th component of the vector** $[0, 0.001, \ldots, 2]^T$**. Solve the problem:**

(a) **with the MATLAB command** `lsqnonlin`**, using the default algorithm**

(b) **with the MATLAB command** `lsqnonlin`**, using the Levenberg-Marquardt algorithm;**

(c) **solving the nonlinear equation** $\mathrm{J_f}^T(\mathrm{x})(\mathrm{f}(\mathrm{x}) - \mathrm{b}) = 0$ **with the MATLAB command** `fsolve`**, with a** 10^{-8} **tolerance on** x**;**

(d) **solving the nonlinear equation** $\mathrm{g}(\mathrm{x}) = \mathrm{J_f}^T(\mathrm{x})(\mathrm{f}(\mathrm{x}) - \mathrm{b}) = 0$ **with the Newton method with finite difference approximate Jacobian of** $\mathrm{g}(\mathrm{x})$ **with increment** $\delta = 0.001$ **and** 10^{-8} **tolerance.**

In each case, use as initial guess the vector $[1, 1, 1, 1, 1, 1]^T$**. and display in a single plot the function** $h(z)$**, the data vector** b **and the function** $\mathrm{f}(\mathrm{x}_s)$ **where** x_s **is the solution. Repeat the computation using as initial guesses the vectors** $[1, 1, 1, 1, 1, 5]^T$ **and** $[1, 1, 1, 1, -1, 5]^T$**.**