# Gauss–Newton and Levenberg–Marquardt Algorithms

Roland Hostettler

September 26, 2018

# Recap

- Sequential linear least squares:

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} + \boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1})$$
$$\boldsymbol{P}_n = \mathrm{Cov}\{\hat{\boldsymbol{\theta}}_n\} = \boldsymbol{P}_{n-1} - \boldsymbol{L}_n(\boldsymbol{C}_n\boldsymbol{P}_{n-1}\boldsymbol{C}^\mathsf{T} + \boldsymbol{R}_n)\boldsymbol{L}_n^\mathsf{T}$$
$$\boldsymbol{L}_n = \boldsymbol{P}_{n-1}\boldsymbol{C}_n^\mathsf{T}(\boldsymbol{C}_n\boldsymbol{P}_{n-1}\boldsymbol{C}_n^\mathsf{T} + \boldsymbol{R}_n)^{-1}$$

- Regularized linear least squares:

$$\hat{\boldsymbol{\theta}} = \boldsymbol{m} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m})$$
$$\mathrm{Cov}\{\hat{\boldsymbol{\theta}}\} = \boldsymbol{P} - \boldsymbol{K}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^\mathsf{T} + \boldsymbol{R})\boldsymbol{K}^\mathsf{T}$$
$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{G}^\mathsf{T}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^\mathsf{T} + \boldsymbol{R})^{-1}$$

- Gradient descent:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} - \gamma\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^{(i)}}$$
$$\Delta\boldsymbol{\theta}^{(i+1)} = -\boldsymbol{G}_{\boldsymbol{\theta}}^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

# Intended Learning Outcomes

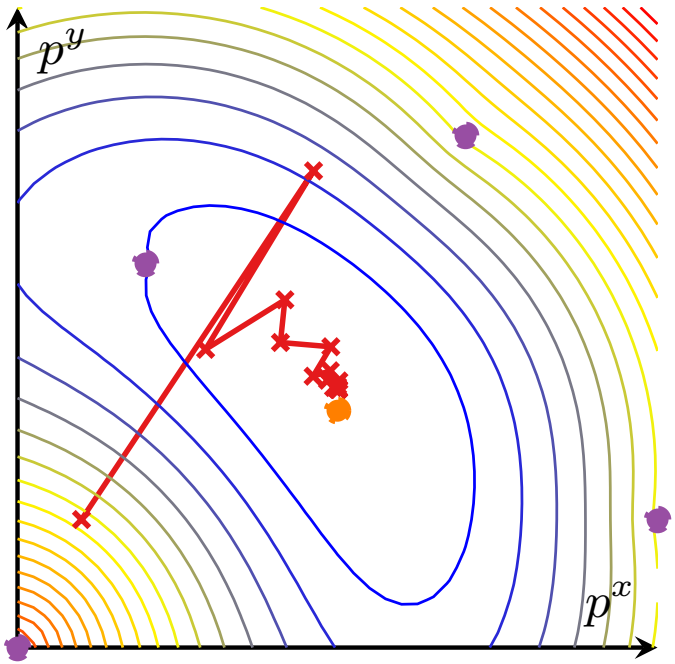
After this lecture, you will be able to:

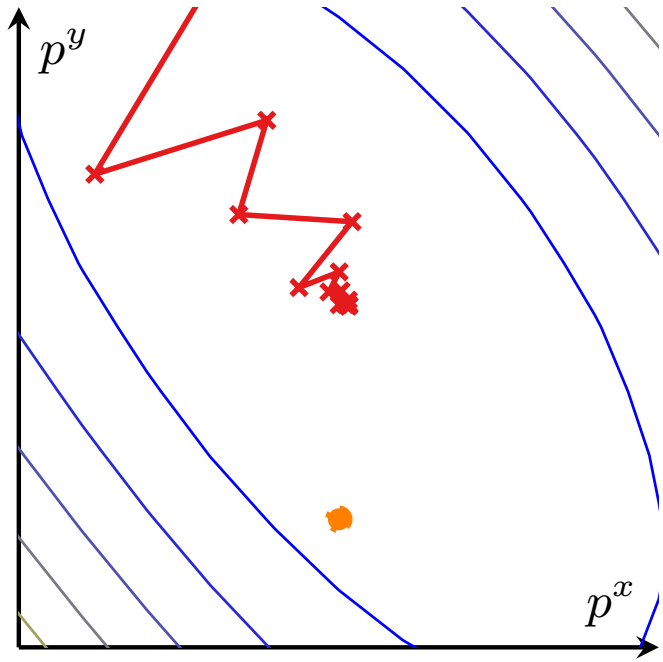
- Describe the Gauss–Newton and Levenberg–Marquardt algorithms;
- discuss the properties of the gradient descent, Gauss–Newton, and Levenberg–Marquardt algorithms;
- recall suitable ways of evaluating the convergence of numerical optimization methods.

# Example: Localizing a Target

- Parameters: Target position $\boldsymbol{\theta} = \begin{bmatrix} p_t^x & p_t^y \end{bmatrix}^\mathsf{T}$
- Measurements model (with $\Delta \boldsymbol{p}_n = \boldsymbol{p}_t - \boldsymbol{p}_n$):

$$y_n = |\Delta \boldsymbol{p}_n| + r_n = \sqrt{(\Delta p_n^x)^2 + (\Delta p_n^y)^2} + r_n$$



# Gauss–Newton Algorithm: Derivation (1/3)

- Idea: Given $\hat{\boldsymbol{\theta}}^{(i)}$, we can linearize the nonlinear measurement model around that point
- Linearized measurement model:

$$g(\boldsymbol{\theta}) \approx g(\hat{\boldsymbol{\theta}}^{(i)}) + \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})$$

- Cost function approximation:

$$\begin{aligned} J_{\text{WLS}}(\boldsymbol{\theta}) &= (\boldsymbol{y} - g(\boldsymbol{\theta}))^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\boldsymbol{\theta})) \\ &\approx (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}))^{\mathsf{T}} \boldsymbol{R}^{-1} \\ &\quad \times (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})) \\ &= (\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}))^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})) \end{aligned}$$

# Gauss–Newton Algorithm: Derivation (2/3)

- Cost function approximation:

$$J_{\text{WLS}}(\boldsymbol{\theta}) \approx (\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}))^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}))$$

- Gradient of the cost function approximation w.r.t. $\boldsymbol{\theta}$:

$$\begin{aligned}\frac{\partial J_{\text{WLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &\approx \frac{\partial}{\partial \boldsymbol{\theta}} (\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}))^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})) \\ &= -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} + 2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\end{aligned}$$

- Solving for $\boldsymbol{\theta}$:

$$\begin{aligned}\boldsymbol{\theta} &= \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} \\ &= \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))\end{aligned}$$

# Gauss–Newton Algorithm: Derivation (3/3)

- Solution of the linearized cost function:

$$\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

- Gauss–Newton update:

$$\begin{aligned} \hat{\boldsymbol{\theta}}^{(i+1)} &= \hat{\boldsymbol{\theta}}^{(i)} + \gamma \Delta \hat{\boldsymbol{\theta}}^{(i+1)}, \\ \Delta \hat{\boldsymbol{\theta}}^{(i+1)} &= (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})) \end{aligned}$$

- The step length $\gamma$ can be selected using a line search with step direction $\Delta \boldsymbol{\theta}^{(i+1)}$

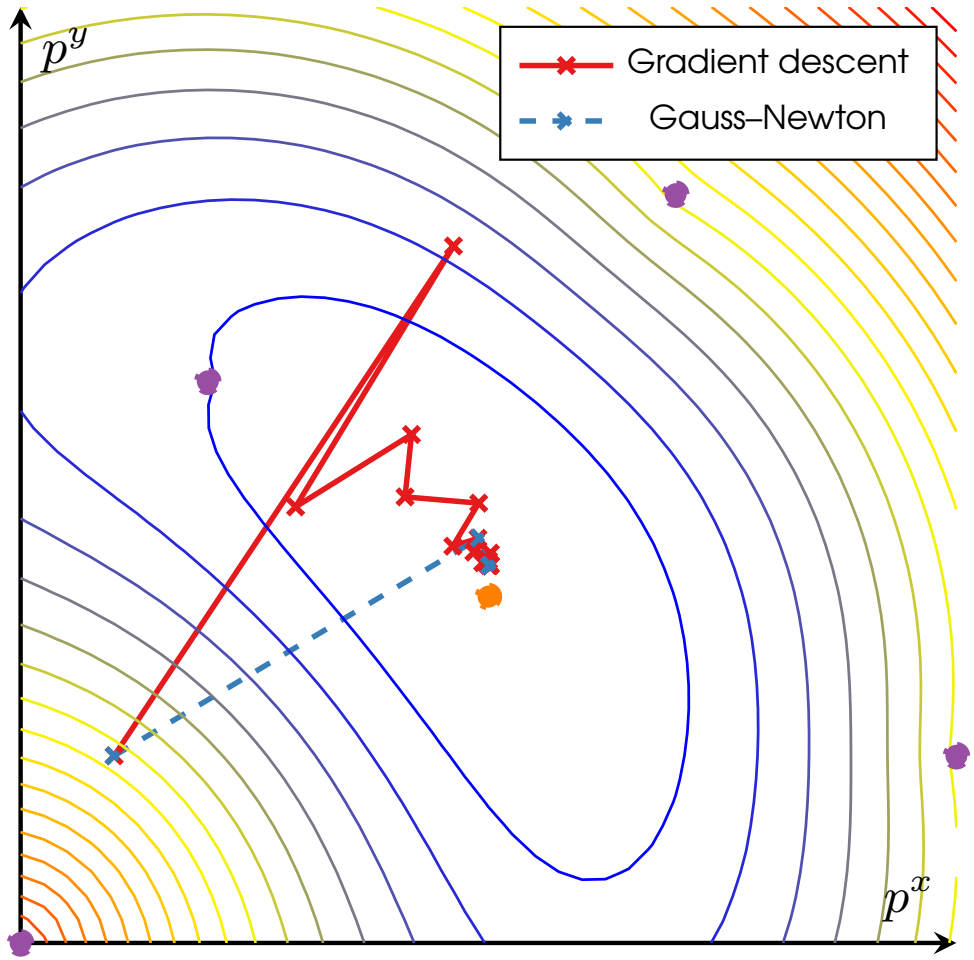
# Gauss–Newton Algorithm: Algorithm

**Algorithm 1** Gauss–Newton Algorithm with Line Search

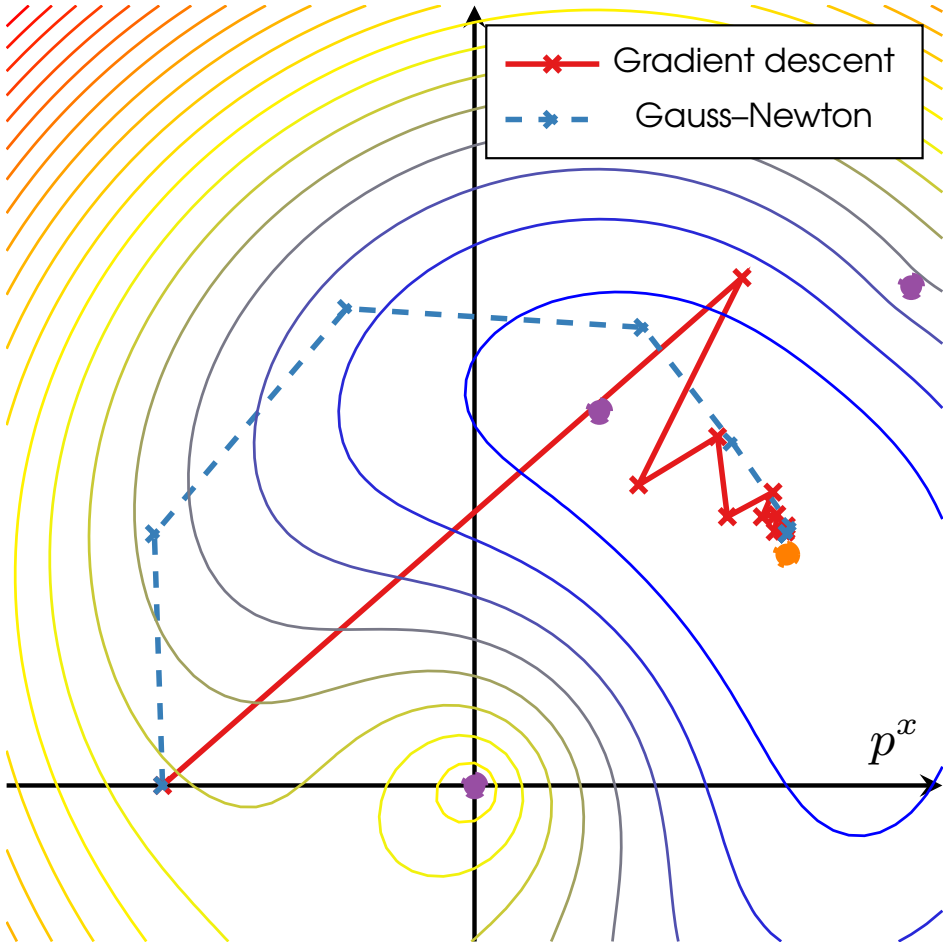
**Input:** Initial guess $\hat{\boldsymbol{\theta}}^{(0)}$, data $\boldsymbol{y}$, function $g(\boldsymbol{\theta})$, Jacobian $\boldsymbol{G}_{\boldsymbol{\theta}}$
**Output:** Parameter estimate $\hat{\boldsymbol{\theta}}_{\text{WLS}}$

Set $i \leftarrow 0$
**repeat**
    Calculate $\Delta\boldsymbol{\theta}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$
    Set $\gamma^{(i+1)} \leftarrow 1$
    **repeat** ▷ Line Search
        Calculate $\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma^{(i+1)}\Delta\boldsymbol{\theta}^{(i+1)}$
        Set $\gamma^{(i+1)} \leftarrow \gamma^{(i+1)}/2$
    **until** $J_{\text{WLS}}(\hat{\boldsymbol{\theta}}^{(i+1)}) < J_{\text{WLS}}(\hat{\boldsymbol{\theta}}^{(i)})$
    Set $i \leftarrow i + 1$
**until** Converged

# Example: Localizing a Target (2)



# Example: Localizing a Target (3)



# Levenberg–Marquardt Algorithm: Motivation

- Gradient descent: Quickly moves to low cost area, creeps to minimum
- Gauss–Newton: Straight to minimum, may take a detour
- Can we have the best of both worlds?
- …kind of, the Levenberg–Marquardt algorithm

# Levenberg–Marquardt Algorithm: Derivation (1/2)

- The Levenberg–Marquardt algorithm can be seen as a regularized version of the Gauss–Newton algorithm
- Cost function approximation:

$$J(\boldsymbol{\theta}) \approx \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})$$

  with $\boldsymbol{e}^{(i)} = \boldsymbol{y} - g(\boldsymbol{\theta}^{(i)})$
- Gradient of the cost function:

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &\approx -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{e}^{(i)} + 2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) + 2\lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\ &= -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{e}^{(i)} + 2(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\end{aligned}$$

# Levenberg–Marquardt Algorithm: Derivation (2/2)

- Gradient of the cost function:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \approx -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{e}^{(i)} + 2(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})$$

- Solving for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{e}^{(i)}$$

- Levenberg–Marquardt update:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \Delta\hat{\boldsymbol{\theta}}^{(i+1)},$$
$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})).$$

# Levenberg–Marquardt Algorithm: Damping (1/2)

- Levenberg–Marquardt update:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \Delta\hat{\boldsymbol{\theta}}^{(i+1)},$$
$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})).$$

- How should the *damping parameter* $\lambda$ be chosen?
- If $\lambda \to 0$:

$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} \to (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

- If $\lambda \to \infty$:

$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} \to \frac{1}{\lambda}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})).$$

# Levenberg–Marquardt Algorithm: Damping (2/2)

- The predicted (linearized) reduction in cost is:

$$\begin{aligned}\Delta J(\boldsymbol{\theta}) &= J(\hat{\boldsymbol{\theta}}^{(i)}) - J(\boldsymbol{\theta}) \\ &\approx 2(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})) \\ &\quad - (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\ &\triangleq \Delta \tilde{J}(\boldsymbol{\theta})\end{aligned}$$

- The relative gain in cost is then:

$$\rho(\boldsymbol{\theta}) = \frac{J(\hat{\boldsymbol{\theta}}^{(i)}) - J(\boldsymbol{\theta})}{\Delta \tilde{J}(\boldsymbol{\theta})}$$

- Adaption strategy:
  - If $\rho > 0$, accept $\Delta\hat{\boldsymbol{\theta}}^{(i+1)}$ and decrease the damping

$$\lambda \leftarrow \lambda \max(1/3, 1 - (2\rho - 1)^3), \ \nu = 2,$$

  - otherwise, discard $\Delta\hat{\boldsymbol{\theta}}^{(i+1)}$ increase the damping by setting

$$\lambda \leftarrow \nu\lambda, \ \nu \leftarrow 2\nu.$$

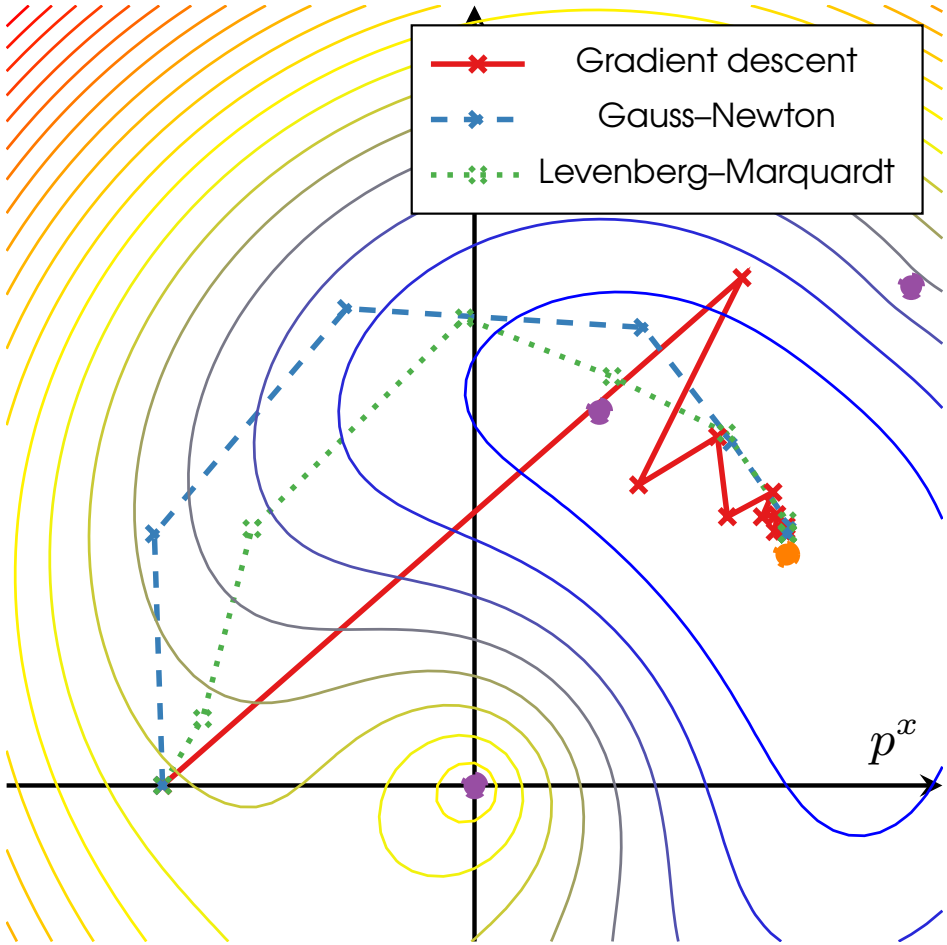
# Levenberg–Marquardt Algorithm: Algorithm

**Algorithm 2** Levenberg–Marquardt Algorithm

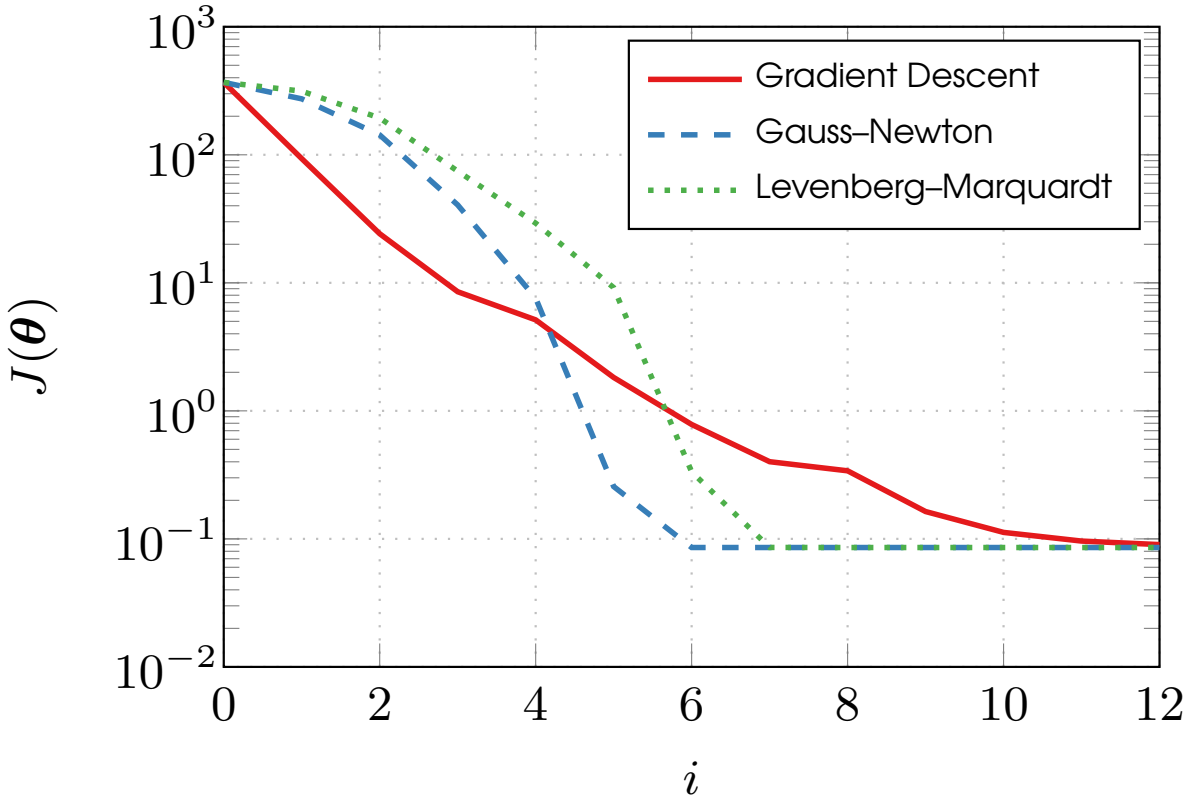
**Input:** Initial guess $\hat{\boldsymbol{\theta}}^{(0)}$, data $\boldsymbol{y}$, function $g(\boldsymbol{\theta})$, Jacobian $\boldsymbol{G}_{\boldsymbol{\theta}}$
**Output:** Parameter estimate $\hat{\boldsymbol{\theta}}_{\text{WLS}}$

Set $i \leftarrow 0$, $\lambda = \max(\text{diag}(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}))$, and $\nu \leftarrow 2$
**repeat**
    **repeat**
        Calculate $\Delta\hat{\boldsymbol{\theta}}^{(i+1)}$ and $\rho$
        **if** $\rho > 0$ **then**
            Set $\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i+1)} + \Delta\hat{\boldsymbol{\theta}}^{(i+1)}$
            Set $\lambda \leftarrow \lambda \max(1/3, 1 - (2\rho - 1)^3)$, $\nu = 2$
        **else**
            Set $\lambda \leftarrow \nu\lambda$ and $\nu \leftarrow 2\nu$
        **end if**
    **until** $\Delta\boldsymbol{\theta}^{(i+1)}$ is accepted
    Set $i \leftarrow i + 1$
**until** Converged

# Example: Localizing a Target (4)



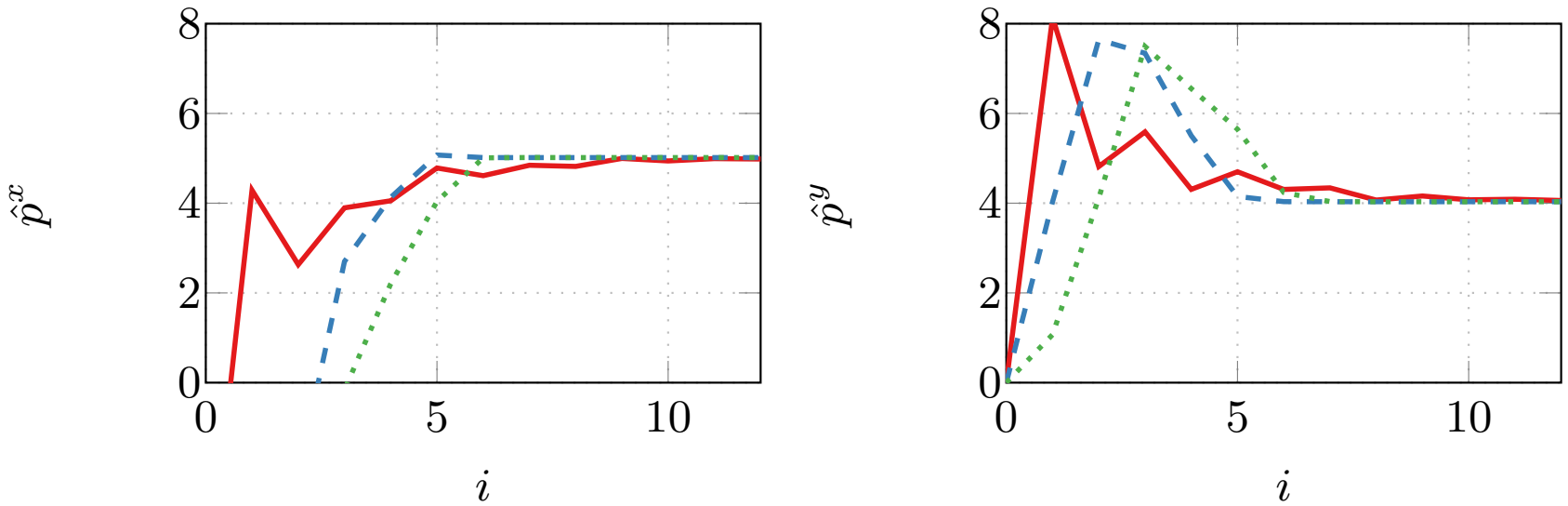
# Convergence Criteria: Decrease in Cost




- Relative change in cost:

$$\frac{J(\hat{\boldsymbol{\theta}}^{(i+1)}) - J(\hat{\boldsymbol{\theta}}^{(i)})}{J(\hat{\boldsymbol{\theta}}^{(i)})}$$

# Convergence Criteria: Change in Parameters




- Relative change in parameter magnitude:

$$\frac{|\hat{\boldsymbol{\theta}}^{(i+1)} - \hat{\boldsymbol{\theta}}^{(i)}|}{|\hat{\boldsymbol{\theta}}^{(i)}|}$$

# Convergence Criteria: Overview

Terminate the numerical search when:

- The relative change in cost

$$\frac{J(\hat{\boldsymbol{\theta}}^{(i+1)}) - J(\hat{\boldsymbol{\theta}}^{(i)})}{J(\hat{\boldsymbol{\theta}}^{(i)})}$$

  falls below a threshold $\epsilon$,
- the relative change in parameter magnitude

$$\frac{|\hat{\boldsymbol{\theta}}^{(i+1)} - \hat{\boldsymbol{\theta}}^{(i)}|}{|\hat{\boldsymbol{\theta}}^{(i)}|}$$

  falls below a threshold $\epsilon$,
- the maximum number of iterations $I_{\text{max}}$ is reached.

# Summary

- Gauss–Newton algorithm:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$

- Levenberg–Marquardt algorithm:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$