Solving Non-Linear Least Squares using the Levenberg-Marquardt Algorithm

Gnana Prakash Punnavajhala

What exactly is Non-Linear Least Squares?

What exactly is Non-Linear Least Squares?

- Defined as follows:

$$\epsilon = \sum_{i=1}^{n} ||y_i - \hat{f}(x_i)||^2$$

where y is defined as a function of x by the function f as $y = f(x)$

- Our goal is to arrive at a function $\hat{f}$ that best resembles f and hence minimise ϵ
- We want to devise a method for any function f which need not necessarily be a linear one (e.g., cos x)

Let's start off with a linear function f

- We define $f : \mathbb{R}^n \to \mathbb{R}^m$ as a linear mapping given by

$$f(X) = A_{m \times n} X_{n \times 1}$$

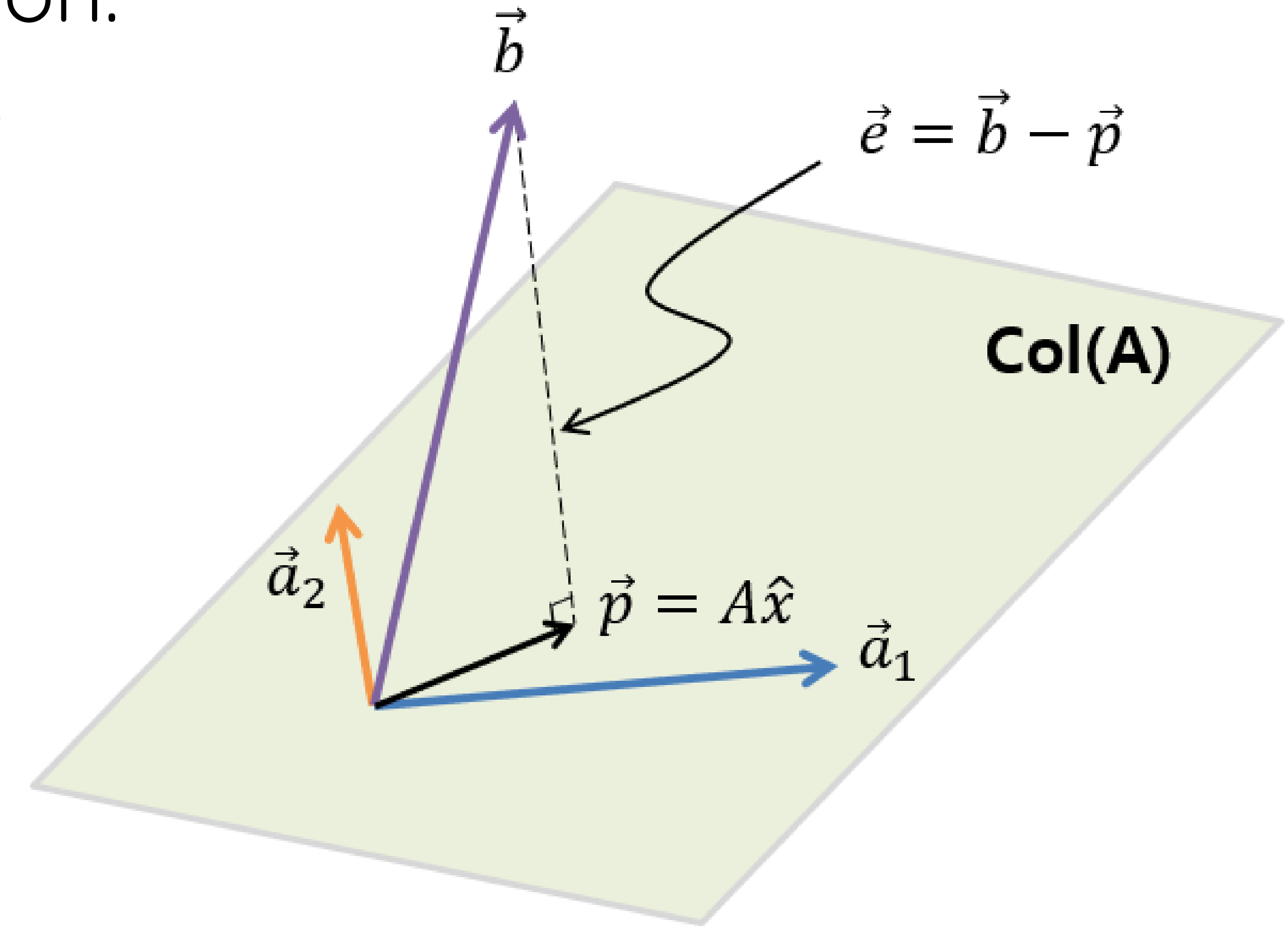
- Here the matrix A is generally obtained from the constraint equations we have from the sampled data points
- The construction of A is based on the following assumptions:
  - **m >> n** - Constraints v/s No of unknowns
  - **A is full rank** - Fair to assume due to noisy measurements

- We now define the cost function as $F(x) = ||Ax - b||_2^2$ (different from f)
  - We get the minimum possible value of F when Ax = b (not always feasible)
  - Computing $x = A^{-1}b$ fails when A is rectangular

Geometrical intuition for the solution

- The solution is given by $\hat{x} = \arg\min_{x} ||Ax - b||_2^2$

- We run into two cases for the existence of a solution:
  - No solution exists when b is not in colspace(A)
  - Unique solution when b is in colspace(A)



https://angeloyeo.github.io/2020/11/11/pseudo_inverse_en.html

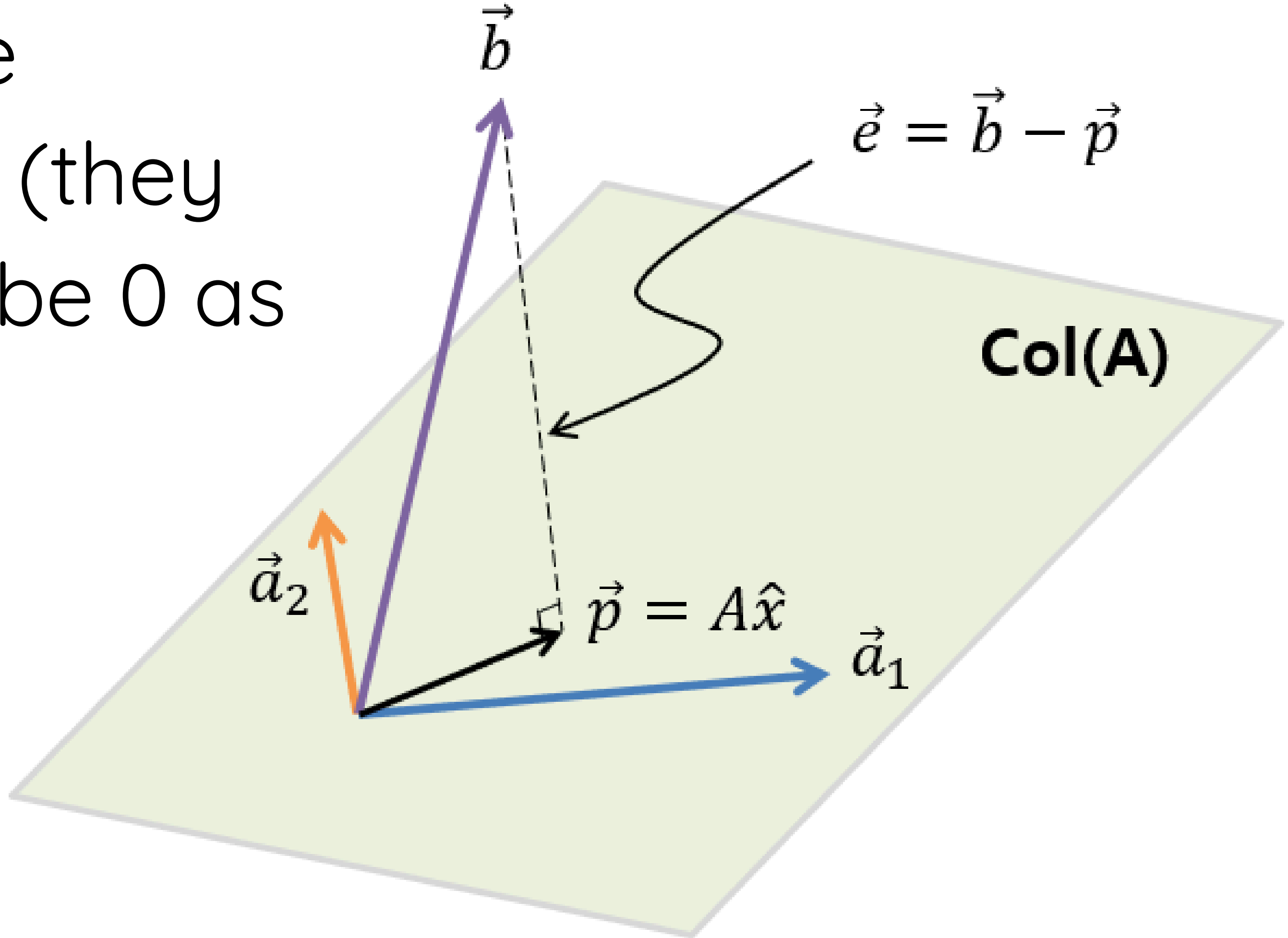
Geometrical intuition for the solution

- Not always possible to find some x such that $||Ax - b||_2^2 = 0$
- We'll try to obtain $\hat{x}$ such that the cost function is minimised
- $A\hat{x}$ will be the projection of b in the column space
- Hence, the dot product of the column vectors of A (they span the column space of A) and the vector e will be 0 as they are perpendicular

$$\implies A^T(b - A\hat{x}) = 0$$

$$\implies A^Tb = A^TA\hat{x}$$

$$\implies \hat{x} = (A^TA)^{-1}A^Tb$$



https://angeloyeo.github.io/2020/11/11/pseudo_inverse_en.html

- Note: $||x||^2 = x \cdot x = x^Tx$

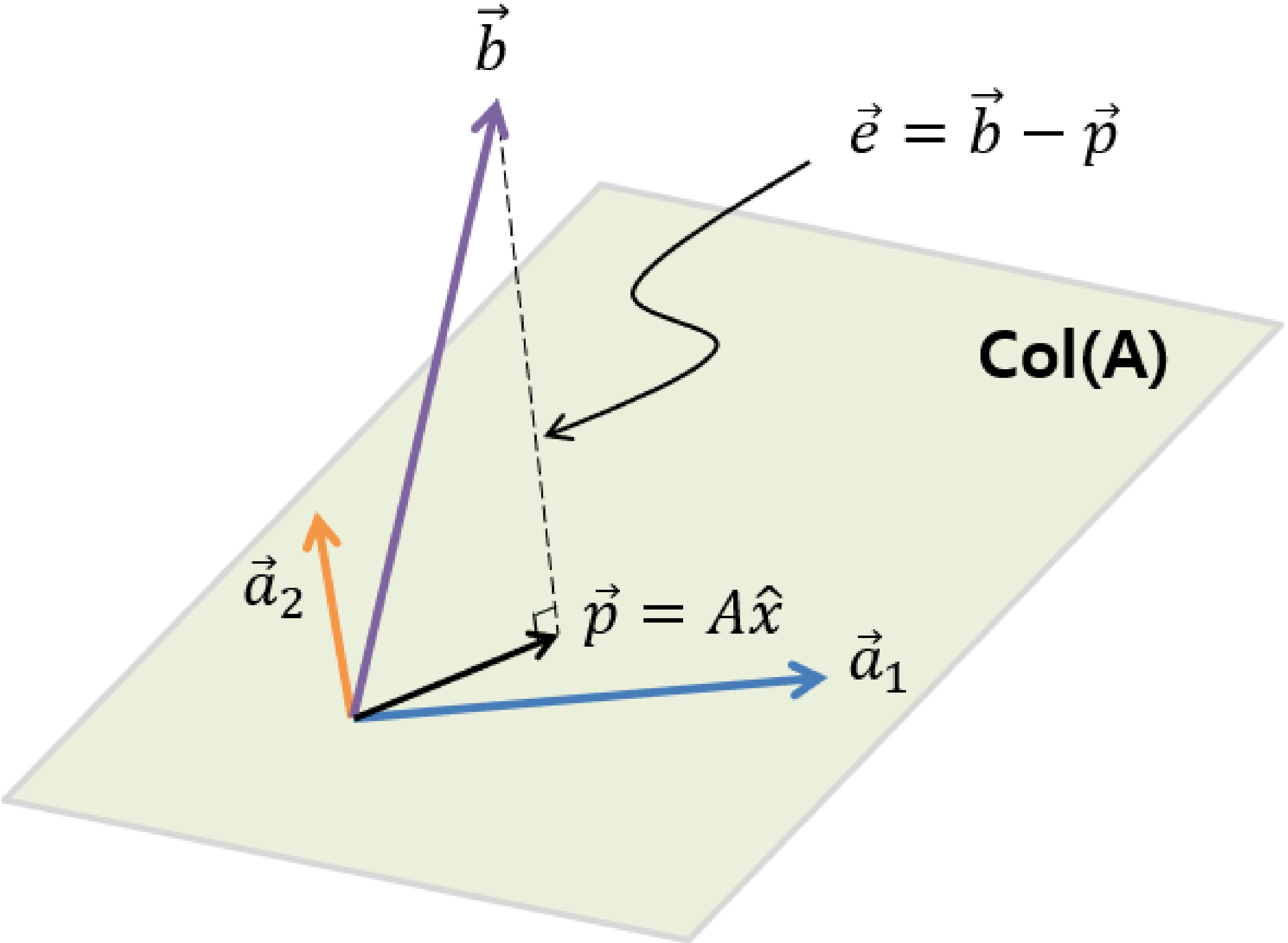
Geometrical intuition for the solution

$$\implies A^T(b - A\hat{x}) = 0$$

$$\implies A^T b = A^T A\hat{x}$$

$$\implies \hat{x} = (A^T A)^{-1} A^T b$$

- The previously discussed conditions about the construction of A (no of rows >> cols; full-rank) come into play here to ensure the existence of $(A^T_{n\times m}\, A_{m\times n})^{-1}_{n\times n}$



https://angeloyeo.github.io/2020/11/11/pseudo_inverse_en.html

Analytical approach for the solution

- Consider the following:

$$\begin{aligned}F(x) &= ||Ax-b||_2^2\\ &= (Ax-b)\cdot(Ax-b)\\ &= (Ax-b)^T(Ax-b)\\ &= (x^TA^T-b^T)(Ax-b)\\ &= x^TA^TAx - x^TA^Tb - b^TAx + b^Tb\end{aligned}$$

- To minimise F(x), we set the derivative of F(x) wrt x to 0, i.e.,

$$\frac{d}{dx}\Big[F(x)\Big] = 0$$

Matrix Calculus 101

$$\frac{d}{dx}(Ax) = A$$

$$\frac{d}{dx}(y^T Ax) = y^T A$$

$$\frac{d}{dy}(y^T Ax) = (Ax)^T = x^T A^T$$

Analytical approach for the solution

- To minimise F(x), we set the derivative of F(x) wrt x to 0, i.e.,

$$
\begin{aligned}
\frac{d}{dx}\Big[F(x)\Big] &= \frac{d}{dx}\Big[x^TA^TAx - x^TA^Tb - b^TAx + b^Tb\Big] = 0 \\
&= \Big[(A^TAx)^T + x^TA^TA - (A^Tb)^T - b^TA\Big] = 0 \\
&= x^TA^TA + x^TA^TA - b^TA - b^TA = 0 \\
&\implies x^TA^TA = b^TA \implies (x^TA^TA)^T = (b^TA)^T \\
&\implies A^TAx = A^Tb \\
&\implies x = (A^TA)^{-1}A^Tb
\end{aligned}
$$

More Math!

- The jacobian of a function f $J_f = \nabla_x f(x)$, where ∇_x is the gradient taken wrt the variable x, in the most general case of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is given by:

$$f(x_1, x_2, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

The Jacobian matrix has a shape of mxn, i.e., no of equations x no of variables

https://www.sefidian.com/2022/03/02/understand-jacobian-and-hessian-matrices-with-example/

More Math!

- The hessian of a function f $H_f = \nabla_x J_f$, where ∇_x is the gradient taken wrt the variable x, in the most general case of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is given by:

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

The Hessian matrix has a shape of nxn, i.e., no of variables x no of variables

Note the difference!

https://www.sefidian.com/2022/03/02/understand-jacobian-and-hessian-matrices-with-example/

More Math!

- The Taylor series expansion for univariate functions is given by

$$f(x+\delta x) = f(x) + \frac{f'(x)(\delta x)}{1!} + \frac{f''(x)(\delta x)^2}{2!} + - - -$$

- The Taylor series expansion for multivariate functions is given by

$$f(x+\delta x) = f(x) + \frac{J_f^T(\delta x)}{1!} + \frac{(\delta x)^T H_f \delta x}{2!} + - - -$$

Questions Break

Gradient Descent

- For the function from our model f, we define the loss function F, for input x, with expected output y as $F(x) = ||f(x) - y||_2^2$
- Our task is to find $\arg\min_x F(x)$, i.e., the value at which the loss is minimum
- Trust Region:
  - Interval of x within which our approximated model function, i.e., f predicts the output very close to the actual value, i.e., y
- The accuracy of f depends on:
  - Number of terms upto which the function was expanded in the Taylor series (More terms indicate better approximation)
  - Trust region window (difficult to perform well in smaller windows)

Gradient Descent

- Now consider

$$
\begin{aligned}
F(x) &= ||f(x) - y||_2^2 \\
&= (f(x) - y)^T (f(x) - y) \\
&= f^T f - f^T y - y^T f + y^T y \\
\nabla_x F(x) &= \nabla_x (f^T f - f^T y - y^T f + y^T y) \\
J_F &= 2f^T J_f - y^T J_f - y^T J_f \\
\implies \quad J_F &= 2f^T J_f - 2y^T J_f
\end{aligned}
$$

- Note the different Jacobians in the last equation!

Gradient Descent

- We have that $J_F = 2f^T J_f - 2y^T J_f$
- From Taylor series approximation, $F(x + \delta x) = F(x) + J_F^T \delta x$
- The only term we can minimise in the above approximation is the jacobian term, which is essentially a dot product
  - (a) · (b) is minimum at cosθ = -1, i.e., (a) · (b) = - ||a|| ||b||
- Hence, we set $\delta x = -\lambda J_F$, where lambda is a fixed scalar
- The disadvantage with gradient descent is that lambda is fixed throughout the process and there is no concept of adaptation based on the neighbourhood. Hence, it might not be guaranteed that we can reach the minima with this approach

Gauss-Newton Method

- We tried approximating F in Gradient Descent as $F(x + \delta x) = F(x) + J_F^T \delta x$
- What happens when we approximate f? $f(x + \delta x) = f(x) + J_f^T \delta x$

$$
\begin{aligned}
F(x + \delta x) &= ||f(x + \delta x) - y||_2^2 \\
&= ||f(x) + J_f^T \delta x - y||_2^2 \\
&= ||F(x) + J_f^T \delta x||_2^2 \\
&= (F(x) + J_f^T \delta x)^T (F(x) + J_f^T \delta x)
\end{aligned}
$$

$$
F(x + \delta x) = F^T F + F^T J_f^T \delta x + (\delta x)^T J_f F + (\delta x)^T J_f J_f^T \delta x
$$

Gauss-Newton Method

- We have $F(x+\delta x) = F^T F + F^T J_f^T \delta x + (\delta x)^T J_f F + (\delta x)^T J_f J_f^T \delta x$
- We now take derivative (or gradient) wrt δx and set it to 0

$$\begin{aligned}
\nabla_{\delta x} F(x+\delta x) &= 0 + F^T J_f^T + F^T J_f^T + 2(\delta x)^T J_f J_f^T = 0 \\
(\delta x)^T J_f J_f^T &= -F^T J_f^T \\
J_f J_f^T \delta x &= -J_f F \\
\delta x &= -(J_f J_f^T)^{-1} J_f F
\end{aligned}$$

- As it is apparent from the above equation, the magnitude of the step changes with every iteration as the Jacobian of f is different at each of the x
- What's the issue then? The Jacobian multiplication can blow up and we risk overshooting the minima

Gauss-Newton - Hessian Approximation

- We have $F(x+\delta x) = F^T F + F^T J_f^T \delta x + (\delta x)^T J_f F + (\delta x)^T J_f J_f^T \delta x$
- We consequently have $\nabla_{\delta x} F(x+\delta x) = 2F^T J_f^T + 2(\delta x)^T J_f J_f^T$
- Taking derivative one more time with δx we have

$$\nabla_{\delta x}\left(\nabla_{\delta x} F(x+\delta x)\right) = 2J_f J_f^T \implies H_F = 2J_f J_f^T$$

- Hence, in general we have $H_F = \lambda J_f J_f^T$
- Ok, and? We do this because computing the Hessian using double derivatives is very computationally intensive. So, we approximate it using first order gradients

Is there no one good method then?

Is there no one good method then?

Levenberg-Marquardt Algorithm gives us the best of both worlds

Levenberg-Marquardt Algorithm

$$(J_f J_f^T + \lambda I)\delta x = -J_f F \implies \delta x = -(J_f J_f^T + \lambda I)^{-1} J_f F$$

- As is evident, the algorithm is a mix of both Gauss-Newton method (from the Jacobian term) and the Gradient Descent method (from the lambda term)
- Higher values of lambda make the algorithm behave like Gradient Descent and for lower lambda values, Gauss-Newton behaviour is observed
- A frequently observed modification to this algorithm is done by scaling the identity matrix so that all the terms adapt to the curvature at that point. It is defined as follows

$$[J_f J_f^T + \lambda\, diag(H_F)]\delta x = -J_f F \implies \delta x = -[J_f J_f^T + \lambda\, diag(H_F)]^{-1} J_f F$$

- Note that LM is not a holy grail solution. It can also get stuck in bad local minima if the initialisation is too far off from the optimal solution. One trick that is used is to have a decay/blowup for the lambda parameter based on the trend of change in loss

Questions Break

References

- MITOCW 18.02 course
- Muti view geometry by Hartley and Zisserman
- Convex optimization by Stephen Boyd
- Notes by Hal Daume
- Khan Academy Notes and Videos
- Basic Knowledge on Visual SLAM From Theory to Practice by Gao Xiang et. al. - Ch-5 Non-Linear Optimisation
- RRC Summer School 2023
- Wikipedia
- For the curious, also look up dog leg method - an alternate to the LM algorithm

Code Demo!