

Exercise 3 - Gauss-Newton

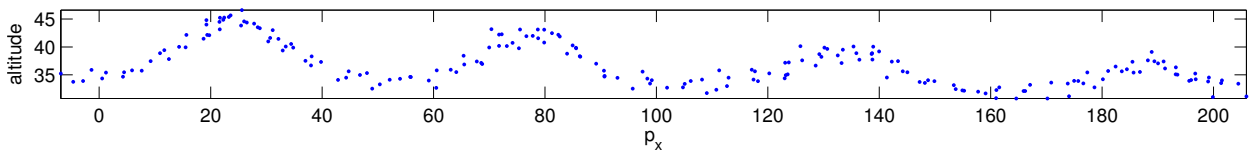
Prof. Dr. Moritz Diehl and Greg Horn

The aim of this exercise is to solve some parameter estimation problems with own solver instead of `fmincon`. You will warm up by fitting a model which can be solved analytically with least squares. You will then use Gauss-Newton to fit a simple model where derivative information is available. Last, you will solve the airplane parameter estimation problem from exercise 2 using your Gauss-Newton solver.

1 Paper Airplane Modeling

We will use the same data set as in exercise 2, which is available at

https://github.com/ghorn/OCE-2014/blob/master/exercise2/airplane_data.m



As before, this data set contains noisy position measurements $\hat{p}_{x,k}$ and $\hat{p}_{z,k}$ but not velocity.

Tasks

1. Linear least squares

Assuming the solution trajectory has a polynomial form in time t :

$$\bar{p}_{x,k} = \sum_{j=0}^{10} a_j t_k^j \quad (1)$$

$$\bar{p}_{z,k} = \sum_{j=0}^{10} b_j t_k^j \quad (2)$$

The optimization problem is:

$$\min_{a_0, \dots, a_{10}, b_0, \dots, b_{10}} \sum_{k=0}^{N-1} (\bar{p}_{x,k}(a, b) - \hat{p}_{x,k})^2 + (\bar{p}_{z,k}(a, b) - \hat{p}_{z,k})^2$$

Formulate the problem as a linear least squares problem, that is:

$$\min_x \|Ax + b\|_2^2$$

and solve using the well-known formula:

$$x_{\text{LS}} = (A^T A)^{-1} A^T b$$

You should expect an imperfect fit, and possibly a badly conditioned matrix inverse. It may be better to use the MATLAB `mldivide` function (which is written as $A \backslash b$) instead of the least squares formula.

Plot p_x vs $-p_z$, $-p_z$ vs time, and p_x vs time.

2. Gauss-Newton on simple model

Assume the solution trajectory has the form:

$$\bar{p}_{x,k}(\theta_x) = \theta_{x,1} + t_k \theta_{x,2} + \theta_{x,3} \sin(\theta_{x,4} + t_k \theta_{x,5}) e^{-\theta_{x,6} t_k} \quad (3)$$

$$\bar{p}_{z,k}(\theta_z) = \theta_{z,1} + t_k \theta_{z,2} + \theta_{z,3} \sin(\theta_{z,4} + t_k \theta_{z,5}) e^{-\theta_{z,6} t_k} \quad (4)$$

The optimization problem is:

$$\min_{\theta_x, \theta_z} \sum_{k=0}^{N-1} (\bar{p}_{x,k}(\theta_x, \theta_z) - \hat{p}_{x,k})^2 + (\bar{p}_{z,k}(\theta_x, \theta_z) - \hat{p}_{z,k})^2$$

Write down the objective function in the form of Gauss-Newton, that is:

$$\min_{\theta} \frac{1}{2} F(\theta)^T F(\theta) \quad (5)$$

Linearize $F(\theta)$ analytically to solve for F_0 , J , where:

$$F(\theta) \approx F_0 + J \Delta \theta$$

Use Newton's method with the Gauss-Newton Hessian approximation to solve (5).

Plot p_x vs $-p_z$, $-p_z$ vs time, and p_x vs time. It will probably be very useful in debugging to plot each iteration of the algorithm.

3. Gauss-Newton on simulation model

For exercise 2, you used a RK4 integrator to minimize measurement errors, estimating α and initial state. Now put that problem in the Gauss-Newton form. Only write a MATLAB function for F , not J . A function for computing J from F is provided at:

https://github.com/ghorn/OCE-2014/blob/master/exercise3/finite_difference_jacob.m

You will call this function with a command something like:

```
[F0, J] = finite_difference_jacob(@(x) Fobj(x), x0);
```

Solve this problem using Newton's method with the Gauss-Newton Hessian approximation. For initial guess, use any initial state you want, and use $\alpha = 3^\circ$.

Plot p_x vs $-p_z$, $-p_z$ vs time, and p_x vs time. It will probably be very useful in debugging to plot each iteration of the algorithm.