




Homework series 1 for Data Assimilation course (WI4475 2025)

This is the first homework series for the data assimilation course (WI4475 2025). There are 2 questions with 5 subquestions each. For each of these 10 subquestions you can earn one point, which directly sums to the grade for this homework series. You are free to select your programming language and editor of choice, but we can only process pdf-files for grading. However, the interactive elements in this document will only work with Pluto.jl and it's probably

easiest to add the answers after each question and generate a pdf from there using the  on the top right of the screen.

The files for this homework series are all available [here](#).

Julia and Pluto

This homework was created in Pluto. Pluto.jl is a notebook system where one can combine text and code into one document that contains a list of cells. Each cell can contain code or text. Unlike the commonly used jupyter the order of execution of the cells is inferred automatically, and when you change a cell, then all dependent cells will be run again. The consequence is that at any time, the results that you see are the output of the current inputs, which is not the case in Jupyter. For installation go to the [install at Pluto.jl site](#)

The Julia language is high-level like python, yet fast like c or fortran. Here's a tiny example:

```
ratio = 2* $\pi$ *x0 #Greek symbols are allowed and are typed as \pi<TAB> and x\_{0}<TAB>  
a=randn(2,100) #some random Gaussian points  
scatter(a[1,:),a[2,:]) #make a scatter plot
```

With 'ctrl-m' you turn a cell into a text cell. The text is written in [markdown format](#)

In the Pluto notebook the code can be hidden and shown again by clicking on the eye symbol:



Just try to show the markdown code for this block of text now.

```
1 # Packages used in this notebook. Packages will be downloaded automatically
  which can take a bit of time when you first start the notebook
2 using Plots, DataFrames, CSV, HTTP, DifferentialEquations, Statistics, Dates
```

Question 1: Non-linear oscillator

This question is about a non-linear spring-mass system. You have probably encountered the linear variant of this equation, where the force of the spring is linear in the position. Here we consider the case where the force is non-linear. We have slightly simplified the equation to:

$$x''(t) = -x - rx^3 + F$$

where $x(t)$ is the position and r is a parameter describing the non-linearity. For $r = 0$ the spring is linear, for $r > 0$ the spring is hardening and for $r < 0$ the spring is softening. F is an external force that is constant over time.



1a: Equilibrium

Show that for $r \geq 0$ and $F > 0$ there is one equilibrium. Note that the question is not to try to solve the equation analytically with the Cardano formula. Also show that for $r < 0$ and $0 < F < F_{critical}$ the equation has three real roots. And compute $F_{critical}$.

Answer

Formulas go between dollar signs: $\sqrt{\text{latex}} \beta \mathbf{a} \beta \mathbf{a} x = \frac{1}{2}$

A list in Markdown notation:

- point 1
- point 2

```

1 md"""
2 __Answer__
3
4 Formulas go between dollar signs:
5  $\sqrt{\text{latex}} \sim \mathbf{\beta a \beta a} x = \frac{1}{2}$ 
6
7 A list in Markdown notation:
8 - point 1
9 - point 2
10 """

```

1b: Find root numerically

Let's try to find a good approximation of the equilibrium for x numerically. For this we use newton iterations, but in a slightly different form:

- substitute $x_i + \Delta x_i$ for x in the equations for the equilibrium and solve for Δx_i while ignoring terms in Δx_i higher than $O(\Delta x_i)$.
- solve for Δx_i
- implement
- apply for $F = 0.2, r = -0.1$

Answer

TODO

```

1 md"""
2 __Answer__
3
4 TODO
5 """

```

```

1 begin
2     #settings
3     F=0.2
4     r=-0.1
5     i_max=100
6     abs_tol=1.0e-8
7     #initialize loop
8     x_i=0.0
9     Δx_i=1.0
10    for i=1:i_max
11        global Δx_i, x_i
12        if abs(Δx_i)<abs_tol
13            break
14        end
15        Δx_i=0.0 #TODO HERE
16        x_i=x_i+Δx_i
17    end
18    @show x_i, Δx_i
19 end

```

```

1 begin
2     #in case that you want to plot something
3     h(x)=sin(x) #function to plot
4     plot(h,xlims=(-4.0,4.0),xlabel="x",label="h(x)")
5 end

```

1c

Next, we want to write the system in the standard form, i.e.:

$$\mathbf{x}'(t) = \mathbf{M}(\mathbf{x})$$

where $\mathbf{x}(t)$ is the state vector. Here we use x and x' as elements of the state vector. Write in this form and make an analysis (analytical) of the stability. How would you characterize the behaviour of the system?

Answer

```

1 md"""
2 __Answer__
3
4
5 """

```

1d

Probably the simplest method for time integration is Euler-forward. Below you can see the result. Write the system in standard form for discrete-time, and analyse the stability by computing the eigenvalues of the Jacobian. Why is this not satisfactory?

```

1 # simulation with Euler-forward
2 begin
3     # parameters
4     #r=-0.5 #defined above already
5     #F=0.35
6     p=(r, F) #pack parameters
7     x0 = [0.0] # initial condition is a vector
8     dx0 = [0.0]
9     tspan = (0.0, 50.0)
10
11     function nonlinearoscillator(ddx, dx, x, p, t)
12         r, F = p # unpack parameters
13         # second derivative of x, ie x'' = ddx
14         @. ddx = -x - r*x^3 + F #avoiding [1] index with '@.' macro
15     end
16     prob_euler = SecondOrderODEProblem(nonlinearoscillator, dx0, x0, tspan, p)
17     sol_euler = solve(prob_euler, Euler(), dt=0.01)
18     plot(sol_euler, vars=(0,2), linewidth=2, label="x(t)")
19 end

```

Answer

```

1 md"""
2 __Answer__
3
4
5 """

```

1e

An improved second-order method is known as Heun. Make an analysis of the stability of the resulting discrete-time system. In what ways is this method better than Euler? What is the difficulty, so what must be going on in the background of the Heun time-integration in the example below?

```

1 # solver using Heun
2 begin
3     prob_heun = SecondOrderODEProblem(nonlinearoscillator, dx0, x0, tspan, p)
4     sol_heun = solve(prob_heun, Heun(), dt=0.01)
5     plot(sol_euler, vars=(0,2), linewidth=2, label="x(t) Euler")
6     plot!(sol_heun, vars=(0,2), linewidth=2, label="x(t) Heun")
7 end

```

Answer

```

1 md"""
2 __Answer__
3
4
5 """

```

Question 2: Linear regression / least-squares

In this question we'll study sealevel rise at the location Vlissingen in the south-western part of

the Netherlands. Vlissingen has one of the longest tide gauge records in the country. A common way to study sealevel rise considers the annual averages of the measured data. In practice also corrections are needed e.g. to compensate for subsidence of the tide gauge. The PSMSL aims to maintain a worldwide collection of tide gauge measurements exactly for this purpose. Let's download and plot the data first.

```

1 # download annual mean sealevel in Vlissingen
2 begin
3     station_label="vlissingen"
4     station_code=20
5     function psmsl_yearly_data(station_code)
6         # https://psmsl.org/data/obtaining/rlr.annual.data/20.rlrdata
7         url = "https://www.psmsl.org/data/obtaining/rlr.annual.data/"
8             $(station_code).rlrdata"
9         println(url)
10        df = CSV.File(HTTP.get(url).body, delim=';', header=false) |> DataFrame
11        # Assign meaningful column names
12        rename!(df, [:Year, :SeaLevel, :Col3, :Col4])
13        return df
14    end
15
16    # get data
17    series=psmsl_yearly_data(station_code)
18    series=filter(row -> row.Year>=1950, series) #keep only 1950 onwards
19    series.SeaLevel=series.SeaLevel.-6800
20    #plot data
21    # series.Year is a Vector containing years [1950, 1951, ... ,]
22    plot(series.Year,series.SeaLevel,xlabel="year",ylabel="sealevel [mm]",
23         title="Annual mean sealevel at $(station_label)",label=false)
24 end

```

2a: Weighted Least Squares

Let's consider the following linear model with noise:

$$z(k) = z_{2000} + \alpha(t(k) - 2000) + \epsilon(k)$$

Where $z(k)$ is the measured sealevel, $t(k)$ is the year, and $\epsilon(k)$ is assumed to be white Gaussian noise with mean zeros and standard-deviation σ .

Rewrite this problem, such that we get the cost-function:

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{z} - \mathbf{H}\mathbf{x})'\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\mathbf{x})$$

where $\mathbf{x} = [z_{2000}, \alpha]'$ is a vector with the parameters of the model and \mathbf{z} is a vector that contains all the measurements. Finally, find the optimal \mathbf{x} . Assume $\sigma = 1.0$ to get started.

Answer

```

1 md"""
2 __Answer__
3 """

```

```

1 #code
2 begin
3     z=series.SeaLevel
4     t=series.Year
5     n=length(t)
6     #TODO
7 end

```

2b: Estimate uncertainty

Next estimate σ using the residuals and use the Hessian of the cost-function to estimate the error covariance of the estimated parameters.

Answer

```

1 md"""
2 __Answer__
3
4
5 """

```

```

1 #code
2 begin
3     #TODO
4 end

```

2c: Compute uncertainty of the fit for each time

Use the result of the previous question to compute the standard deviation of the error of the fitted model for each time t . Plot the original series, the model fit and the error bars at each time.

Answer

```

1 md"""
2 __Answer__
3
4
5 """

```

2d: Correction for effect of local wind and pressure

The sealevel at a coastal location also varies with the wind and air-pressure. It is possible to use a model (numerical or statistical) to compensate for these variations. Here we use data from the Global Tide Surge Model as also used by the "sealevelmonitor"

Fit the model to the corrected series. Estimate the resulting σ . What is a benefit of this approach?

```

1 #code
2 begin
3   function get_gtsm_data(station_name)
4     url = "https://raw.githubusercontent.com/Deltares-research/
5           sealevelmonitor/refs/heads/main/data/deltares/gtsm/
6           gtsm_surge_annual_mean_main_stations.csv"
7     println(url)
8     surge_series = CSV.File(HTTP.get(url).body, header=3,dateformat="yyyy-mm-
9                             dd") |> DataFrame # download data and parse
10    surge_series.Year = Dates.year.(surge_series.t) # get year from date
11    surge_series=filter(row -> row.name==station_name, surge_series) #
12    select station
13    surge_series=filter(row -> row.Year<2024,surge_series) #ignore 2024
14    surge_series.surge=surge_series.surge.*1000.0 #meter to mm
15    return surge_series
16  end
17  surge_series=get_gtsm_data("Vlissingen")
18  year_surge=surge_series.Year
19  surge=surge_series.surge
20  z_corrected=z.-surge # subtract surge from measured annual means
21
22  # plot
23  p1=plot(year_surge,surge,xlabel="year",ylabel="surge
24          [mm]",label="correction", title="Surge correction at $(station_label)")
25  p2=plot(surge_series.Year,surge_series.SeaLevel,xlabel="year",ylabel="sealevel [mm]",
26          title="Annual mean sealevel at $(station_label)",label="measured")
27  plot!(p2,surge_series.Year,z_corrected,label="corrected")
28  plot(p1,p2,layout=(2,1))
29 end

```

Answer

```

1 md"""
2 __Answer__
3 """

```

```

1 begin
2   #TODO
3   @show year_surge, surge, z_corrected
4 end

```

2e Monte carlo estimate of uncertainty

In this question the model is quite simple, so the covariances of the errors can be computed directly, which is hard or impossible for more complex models. In those cases a Monte Carlo approach can be useful.

Use your estimates of σ , z_{2000} and α to generate a synthetic dataset with the model. Then estimate the parameters again and repeat this many (e.g. 100) times. The covariance of the parameters can be computed from these samples.

Answer

```
1 md"""
2 __Answer__
3
4 """
```

```
1 begin
2     #TODO
3 end
```