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. Eeach cell can contain code or text. Unlike the commonly used jupyter the order of execution of the cells is infered 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*π*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**

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.

```julia
# Packages used in this notebook. Packages will be downloaded automatically which can take a bit of time when you first start the notebook
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 and 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 and the equation has three real roots. And compute .

Answer

Formulas go between dollar signs:

A list in Markdown notation:

- point 1
- point 2

```
md"""
__Answer__

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

A list in Markdown notation:
- point 1
- point 2
"""
```

1b: Find root numerically

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

- substitute for in the equations for the equilibrium and solve for while ignoring terms in higher than .
- solve for
- implement
- apply for ,

Answer

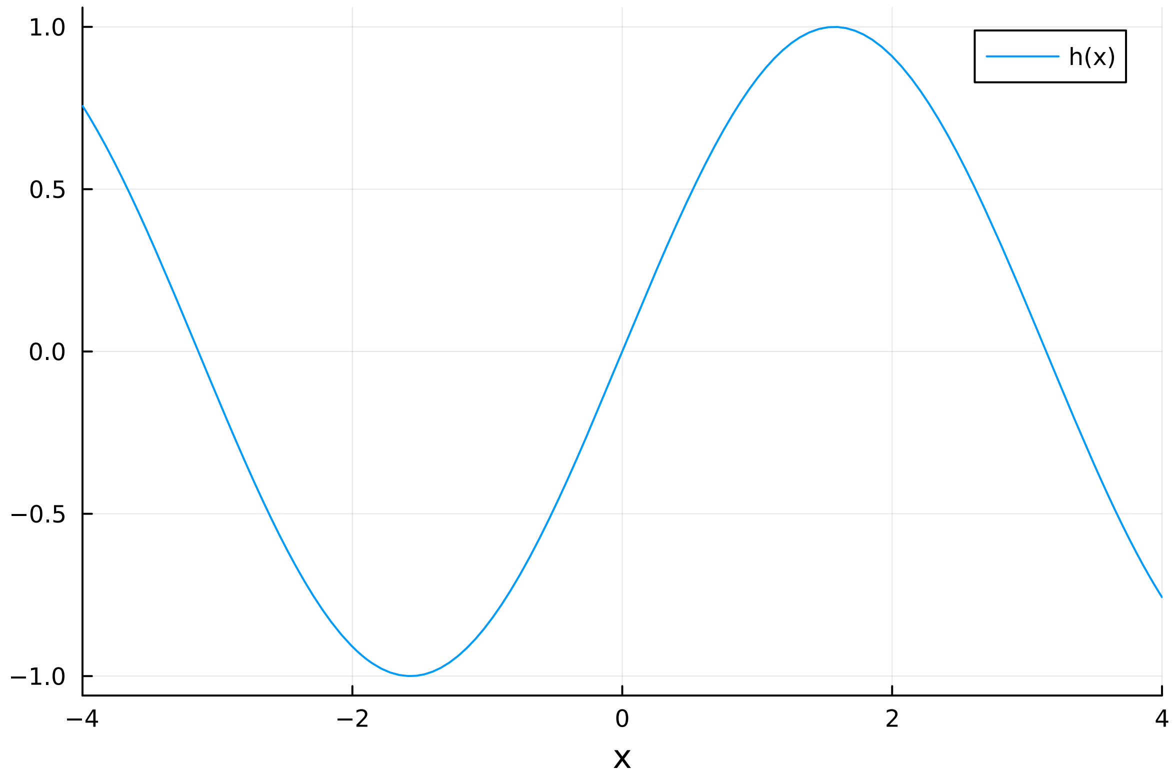
TODO

```
md"""
__Answer__

TODO
"""
```

```
(0.0, 0.0)
```

```
begin
    #settings
    F=0.2
    r=-0.1
    i_max=100
    abs_tol=1.0e-8
    #initialize loop
    x_i=0.0
    Δx_i=1.0
    for i=1:i_max
```

```
        global Δx_i, x_i
        if abs(Δx_i)<abs_tol
            break
        end
        Δx_i=0.0 #TODO HERE
        x_i=x_i+Δx_i
    end
    @show x_i, Δx_i
end
```




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

1c

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

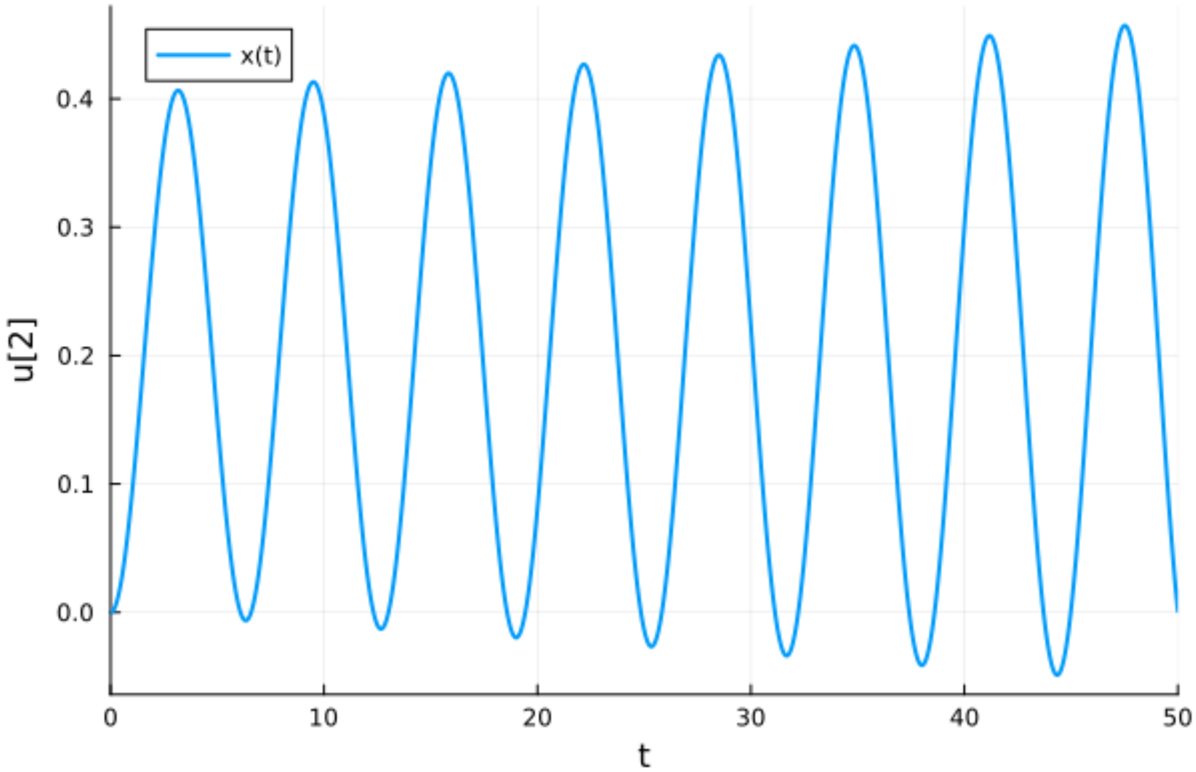
where is the state vector. Here we use and as elements of the state vector. Write in this form and make an analysis (analytical) of the stability. How would you characterize the behaviour of the system?

Answer

```
md"""
__Answer__


"""
```

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?



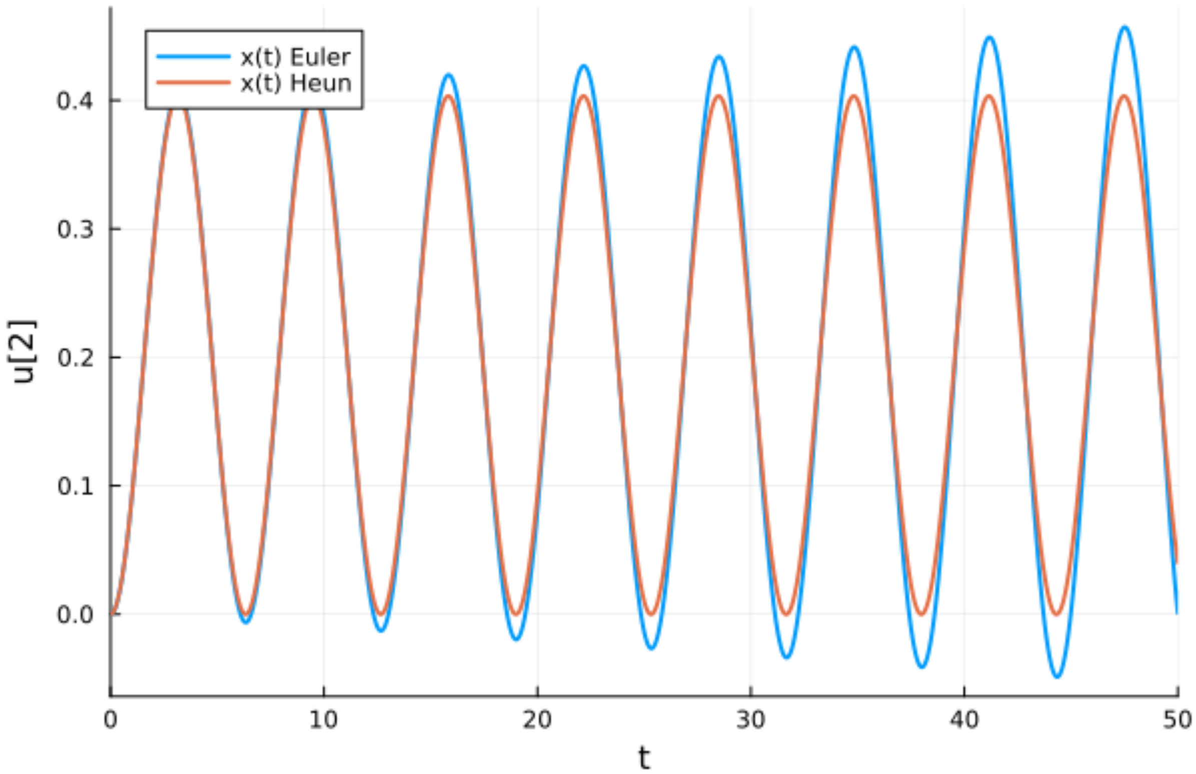

```
# simulation with Euler-forward
begin
    # parameters
    #r=-0.5 #defined above already
    #F=0.35
    p=(r, F) #pack parameters
    x₀ = [0.0] # initial condition is a vector
    dx₀ = [0.0]
    tspan = (0.0, 50.0)

    function nonlinearoscillator(ddx, dx, x, p, t)
        r, F = p # unpack parameters
        # second derivative of x, ie x'' = ddx
        @. ddx = -x - r*x^3 + F #avoiding [1] index with `@.` macro
    end
    prob_euler = SecondOrderODEProblem(nonlinearoscillator, dx₀, x₀, tspan, p)
    sol_euler = solve(prob_euler, Euler(), dt=0.01)
    plot(sol_euler, vars=(0,2), linewidth=2, label="x(t)")
end
```

Answer

```
md"""
__Answer__


"""
```

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 teh example below?



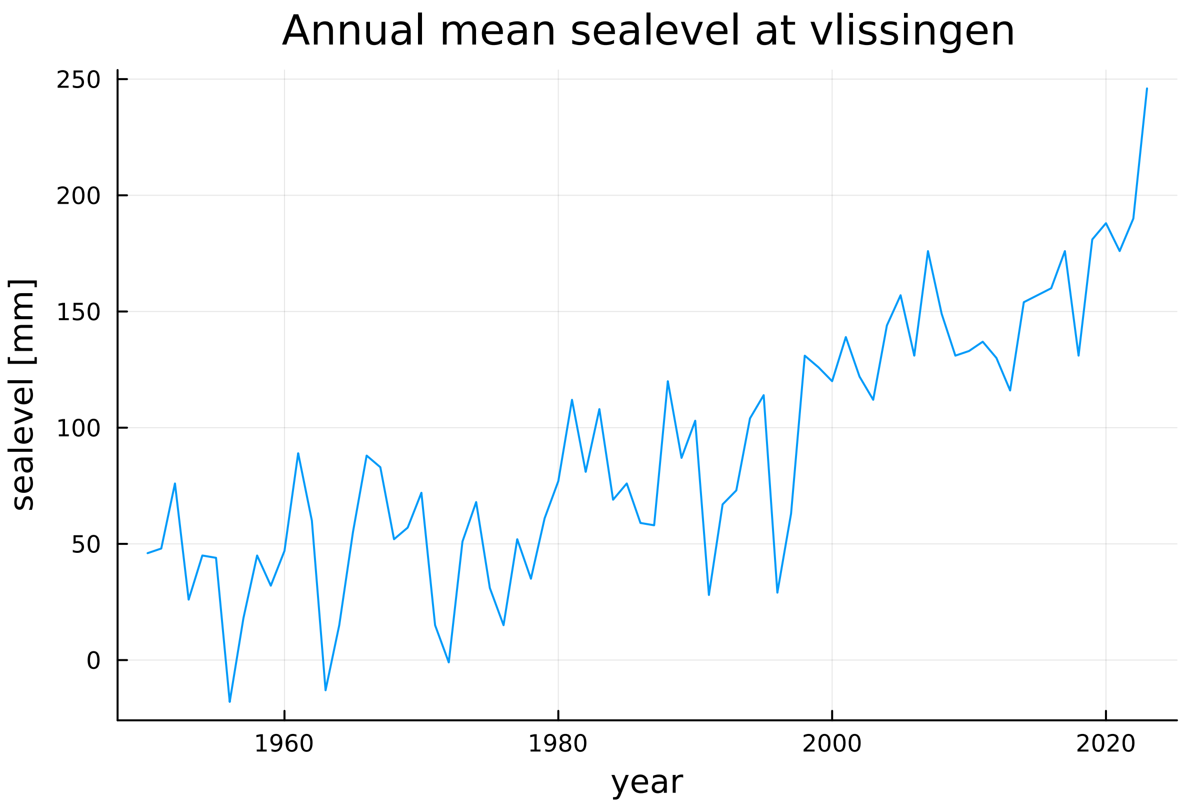

```
# solver using Heun
begin
    prob_heun = SecondOrderODEProblem(nonlinearoscillator, dx₀, x₀, tspan, p)
    sol_heun = solve(prob_heun, Heun(), dt=0.01)
    plot(sol_euler,vars=(0,2), linewidth=2, label="x(t) Euler")
    plot!(sol_heun, vars=(0,2), linewidth=2, label="x(t) Heun")
end
```

Answer

```
md"""
__Answer__


"""
```

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 guage measurements exactly for this purpose. Let's download and plot the data first.




```
# download annual mean sealevel in Vlissingen
begin
    station_label="vlissingen"
    station_code=20
    function psmsl_yearly_data(station_code)
    # https://psmsl.org/data/obtaining/rlr.annual.data/20.rlrdata
    url = "https://www.psmsl.org/data/obtaining/rlr.annual.data/
$(station_code).rlrdata"
    println(url)
    df = CSV.File(HTTP.get(url).body, delim=';', header=false) |> DataFrame
    # Assign meaningful column names
    rename!(df, [:Year, :SeaLevel, :Col3, :Col4])
    return df
end

    # get data
    series=psmsl_yearly_data(station_code)
    series=filter(row -> row.Year>=1950, series) #keep only 1950 onwards
    series.SeaLevel=series.SeaLevel.-6800
    #plot data
    # series.Year is a Vector containing years [1950, 1951, ... ,]
    plot(series.Year,series.SeaLevel,xlabel="year",ylabel="sealevel [mm]",
title="Annual mean sealevel at $(station_label)",label=false)
end
```

2a: Weighted Least Squares

Let's consider the followind linear model with noise:

Where is the measured sealevel, is the year, and is assumed to be white Gaussian noise with mean zeros and standard-deviation .

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

where is a vector with the parameters of the model and is a vector that contains all the measurements. Finally, find the optimal . Assume to get started.

Answer

```
md"""
__Answer__
"""
```

74

```
#code
begin
    z=series.SeaLevel
    t=series.Year
    n=length(t)
    #TODO
end
```

2b: Estimate uncertainty

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

Answer

```
md"""
__Answer__


"""
```

```
#code
begin
    #TODO
end
```

2c: Compute uncertainty of the fit for each time

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

Answer

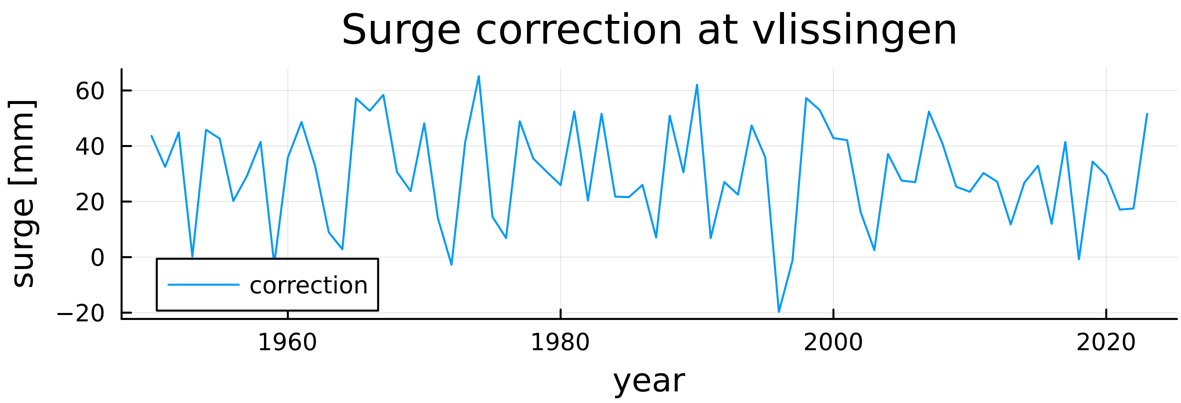
```
md"""
__Answer__

```

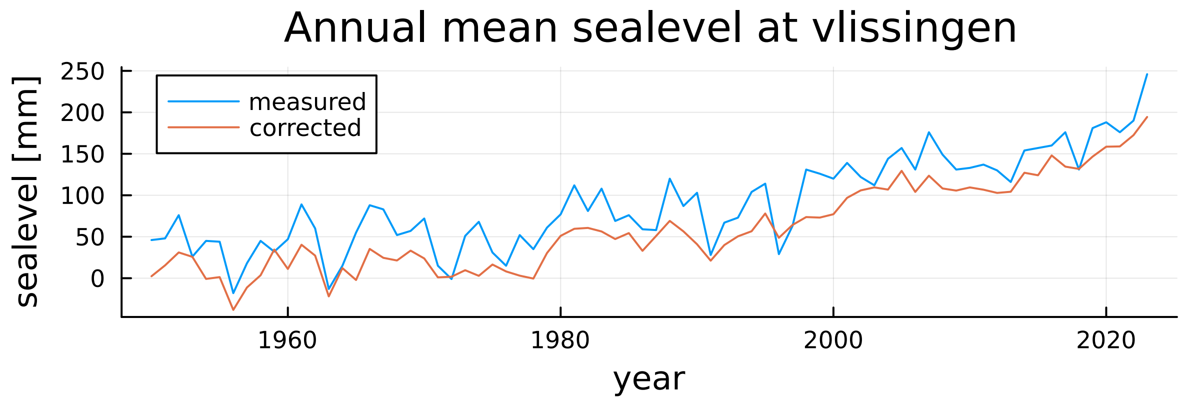
```
"""
```

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 . What is a benefit of this approach?






```
#code
begin
    function get_gtsm_data(station_name)
        url = "https://raw.githubusercontent.com/Deltares-research/
sealevelmonitor/refs/heads/main/data/deltares/gtsm/
gtsm_surge_annual_mean_main_stations.csv"
        println(url)
        surge_series = CSV.File(HTTP.get(url).body, header=3,dateformat="yyyy-mm-
dd") |> DataFrame # download data and parse
        surge_series.Year = Dates.year.(surge_series.t) # get year from date
        surge_series=filter(row -> row.name==station_name, surge_series) # select
station
        surge_series=filter(row -> row.Year<2024,surge_series) #ignore 2024
        surge_series.surge=surge_series.surge.*1000.0 #meter to mm
        return surge_series
    end
    surge_series=get_gtsm_data("Vlissingen")
    year_surge=surge_series.Year
    surge=surge_series.surge
    z_corrected=z.-surge # subtract surge from measured annual means

    # plot
```

```
    # plot
    p1=plot(year_surge,surge,xlabel="year",ylabel="surge [mm]",label="correction", title="Surge correction at $(station_label)")
    p2=plot(series.Year,series.SeaLevel,xlabel="year",ylabel="sealevel [mm]", title="Annual mean sealevel at $(station_label)",label="measured")
    plot!(p2,series.Year,z_corrected,label="corrected")
    plot(p1,p2,layout=(2,1))
end
```

Answer

```
md"""
__Answer__
"""
```

```
begin
    #TODO
    @show year_surge, surge, z_corrected
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 , and 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

```
md"""
__Answer__

"""
```

```
begin
    #TODO
end
```