




Homework series 1 for Data Assimilation course (WI4475)

This is the first homework series for the data assimilation course (WI4475 2024). 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*π #Greek symbols are allowed and are typed as \pi<TAB>  
a=randn(2,100) #some random 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, StaticArrays
```

Question 1: Unfair coins

The idea for this question comes from the Deep Learning book by Bishop (Chapter 2) where a curved coin is shown as an unfair probability between heads and tails. Based on this, I modified a few coins and made a 100 throws with each one.



For each coin the thicknes (in mm) minus the thickness of a flat coin was measured. A sign was added to distinguish between the two bendig directionss. Positive values show tails in the figure. The assumption is that the rounded side on the table is somewhat preferred. Let's study that later.

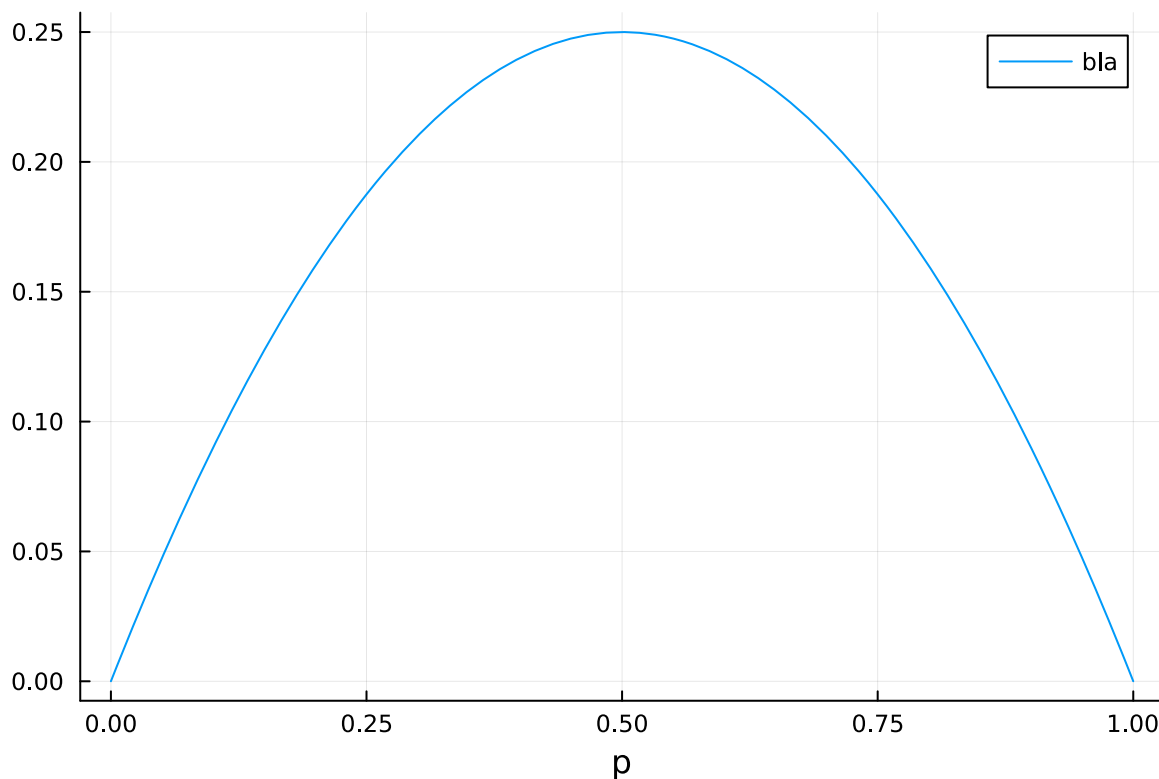
1a: Bayes rule

Assume an uninformative uniform prior where the probability for tails p has itself a uniform distribution between zero and one $X_0 \sim U(0, 1)$. The first throw of the coin can have value $y_1 = 0$ for heads and $y_1 = 1$ for tails. The random variable Y_1 has a conditional distribution $P(Y_1 = y_1 | X_0 = p) = p^{y_1} (1 - p)^{1-y_1}$. Use Bayes' rule to derive $P(X_0 = p | Y_1 = y_1)$ for both values of y_1 and plot them together with the prior distribution.

```
1 md"""
2 ### 1a: Bayes rule
3
4 Assume an uninformative uniform prior where the probability for tails $p$ has
  itself a uniform distribution between zero and one  $X_0 \sim U(0,1)$ . The first
  throw of the coin can have value  $y_1=0$  for heads and  $y_1=1$  for tails. The
  random variable  $Y_1$  has a conditional distribution  $P(Y_1=y_1|X_0=p)=p^{y_1}$ 
5  $(1-p)^{1-y_1}$ .
  Use Bayes' rule to derive  $P(X_0=p|Y_1=y_1)$  for both values of  $y_1$  and plot
6 them together with the prior distribution.
  """
```

Answer *latex blabla* $x = \frac{1}{2}$

```
1 md"""
2 __Answer__
3 $latex ~ blabla ~ x=\frac{1}{2}$
4 """
```



```
1 #sample figure
2 begin
3 f(p)=(1-p)*p #rubbish
4 plot(f,0.0,1.0,label="bla") # p ∈ [0,1]
5 xlabel!("p")
6 end
```

1b: Bayes applied in sequence

After the result for $P(X_0 = p|Y_1 = y_1)$, we throw the coin again to obtain Y_2 . Calculate $P(X_0 = p|Y_1 = y_1, Y_2 = y_2)$ for all combinations of (y_1, y_2) . The result should build on $P(X_0 = p|Y_1 = y_1)$ and not start from the prior directly. Note that there are now 4 possibilities for y_1, y_2 . Does the order of the observations Y_1 and Y_2 affect the result?

1 Enter cell code...

1c: Bayes in one batch

Now we apply Bayes in one go to a vector with N values (y_1, y_2, \dots, y_N) and we try to find the maximum for the parameter p . Note that the denominator, that normalizes the probability, does not depend on p , thus we can also omit it for the optimization. Derive an expression for the estimator \hat{p} .

Next apply the estimator to the data in file `coin.csv` for curvature $c = 0.0$, which is the flat coin. The file contains the results after throwing each of the $M = 5$ coins $N = 100$ times. The example below shows how to read and work with the data.

(55, 100)

```
1 begin
2   read_remote_csv(url) = DataFrame(CSV.File(HTTP.get(url).body))
3   coins = read_remote_csv("https://raw.githubusercontent.com/robot144/
    /wi4475_2024/main/coin.csv")
4   #select where curvalue is 0.0, and assume that this is a fair coin
5   fair=coins[coins.curvature .== 0.0, :] # flat coins where curvature is 0.0
6   count_tails=sum(fair[:, :tails])
7   N=length(fair[:, :tails])
8   @show count_tails, N
9 end
```

(count_tails, N) = (55, 100)



1d: Data distribution or bootstrapping

One create an approximate model for the distribution of the data by random sampling with replacement from the dataset. Next one can apply the estimator to the generated samples (observations). This process can be repeated many times to compute the mean, standard deviation (or other statistics) and in that way estimate the uncertainty resulting from the finite sample size. Apply this method to the data of the previous question to compute a standard deviation for \hat{p} .

```

1 begin #sample to get you started.
2   for i=1:3
3     n = rand(1:5)
4     @show n
5   end
6 end

```

```

n = 1
n = 5
n = 4

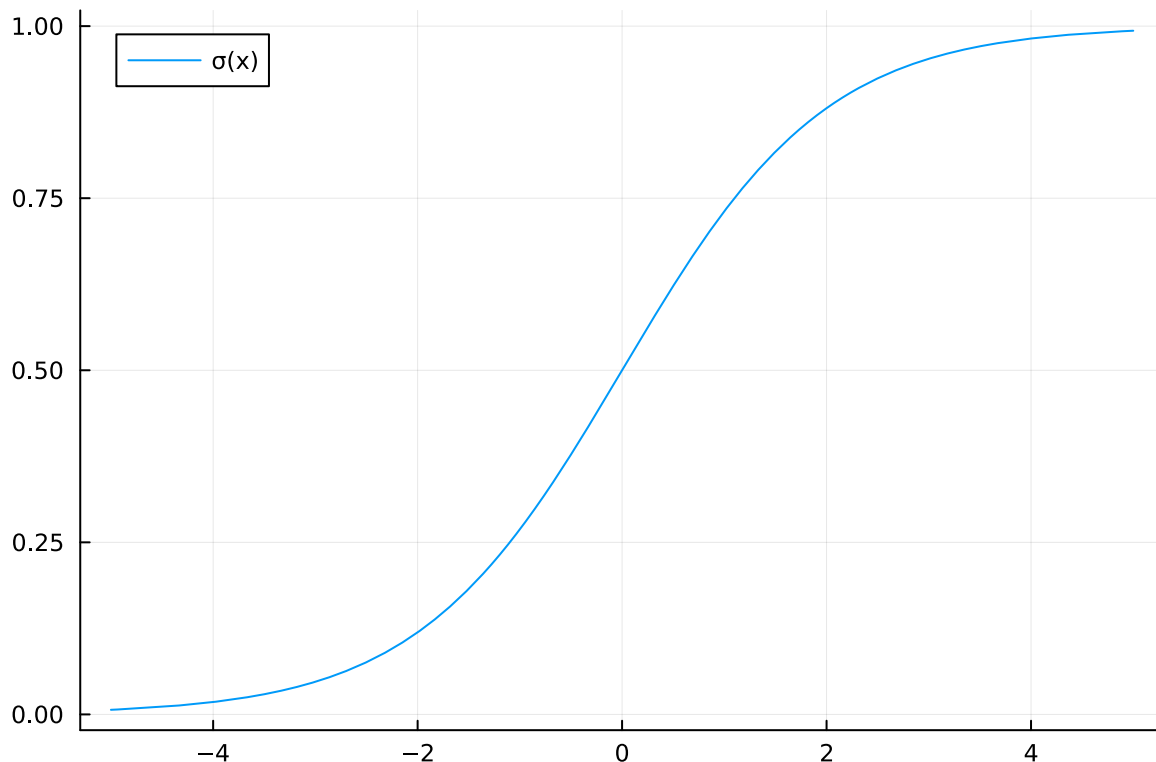
```



1e: Logistic model

We can also try to model the probability p as a function of the curvature c with a logistic function $p(c) = \sigma(a * c - b)$ with $\sigma(x) = 1/(1 + e^{-x})$ and parameters a and b . For each sample in the data we know which coin was used, so the data contains pairs (c_i, y_i) .

Derive an expression for $-\log(p(Y_i|(a, b)))$ and find the optimal parameters. Note that for one observation y_k we have $p(y_k|a, b) = p_{a,b}(c_k)^{y_k}(1 - p_{a,b}(c_k))^{1-y_k}$. Finally, try to find values a and b with a low log-likelihood. Let's keep this part simple and just try many values. Don't waste too much time on an efficient optimizer, we'll look into gradients in the second homework.



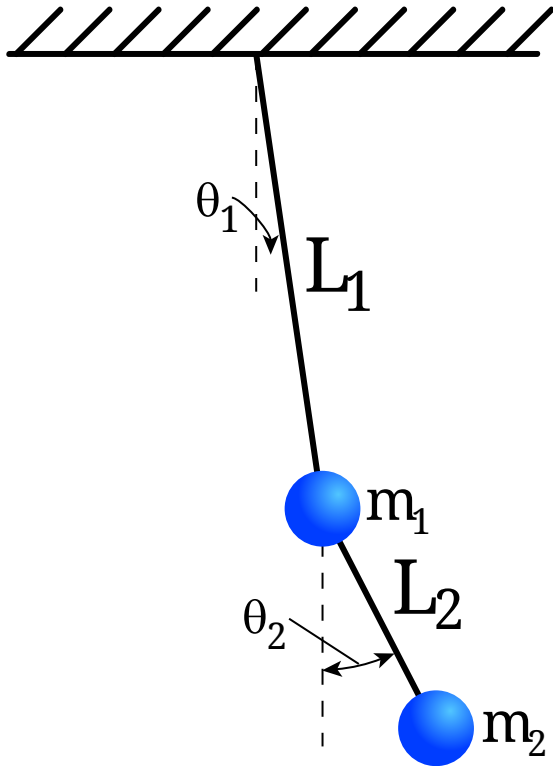
```

1 begin
2   sigma(x)=1/(1+exp(-x))
3   plot(sigma,-5.0,5.0,label="sigma(x)")
4 end

```

Question 2: Double pendulum

A double pendulum is a simple well known mechanical device that can show chaotic behaviour. The dynamics of the system can be derived based on the conservation of momentum. The image below shows the notation for the variables. For more info you can visit [Double Pendulum on Wikipedia](#) and [double-pendulum on myphysicslab.com](#).



The dynamics can be described by:

$$\theta_1' = \omega_1$$

$$\theta_2' = \omega_2$$

$$\omega_1' = \frac{-g(2m_1 + m_2)\sin(\theta_1) - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2)m_2(\omega_2^2 l_2 + \omega_1^2 l_1 \cos(\theta_1 - \theta_2))}{l_1(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2))}$$

$$\omega_2' = \frac{2\sin(\theta_1 - \theta_2)(\omega_1^2 l_1(m_1 + m_2) + g(m_1 + m_2)\cos(\theta_1) + \omega_2^2 m_2 l_2 \cos(\theta_1 - \theta_2))}{l_2(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2))}$$

In the exercises here we'll use the following values for simplicity: $l_1 = 1$, $l_2 = 1$, $m_1 = 1$, $m_2 = 1$, $g = 10$.

The code below shows how to plot a double pendulum and how to solve the system numerically.

plot_double_pendulum (generic function with 1 method)

```

1 #Plot a double pendulum
2 function plot_double_pendulum(θ1, θ2, l1, l2; line_color=:red, line_width=3,
   marker_color=:blue, marker_size=10)
3     x1 = l1 * sin(θ1)
4     y1 = -l1 * cos(θ1)
5     x2 = x1 + l2 * sin(θ2)
6     y2 = y1 - l2 * cos(θ2)
7     plot([0, x1, x2], [0, y1, y2], aspect_ratio=:equal, xlims=(-2.2, 2.2), ylims=
   (-2.2, 2.2), legend=false, lw=line_width, color=line_color, axes=false)
8     scatter!([x1, x2], [y1, y2], color=marker_color, ms=marker_size)
9 end

```

animate_double_pendulum (generic function with 3 methods)

```

1 function animate_double_pendulum(sol, l1, l2, timestep=0.05, fps=30)
2     # create an animation
3     times = sol.t[1]:timestep:sol.t[end] # times at which to plot the pendulum
4     n = length(times)
5     anim = @animate for i in 1:n
6         u=sol(times[i])
7         θ1, θ2, ω1, ω2 = u
8         plot_double_pendulum(θ1, θ2, l1, l2)
9     end
10    return gif(anim, fps = fps)
11 end

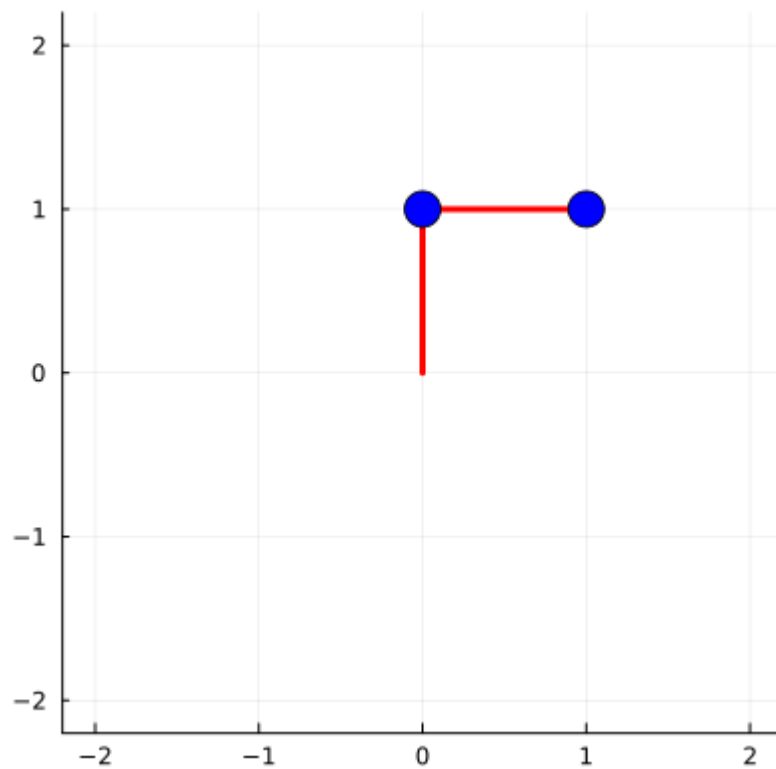
```

double_pendulum (generic function with 1 method)

```

1 function double_pendulum(u, p, t)
2     # compute f(x,p,t) for ode dx/dt=f(x,p,t)
3     θ1, θ2, ω1, ω2 = u
4     m1, m2, l1, l2, g = p
5     # equations taken from https://www.myphysicslab.com/pendulum/double-pendulum-en.html
6     dθ1dt = ω1
7     dθ2dt = ω2
8     dω1dt = ( -g*(2*m1+m2)*sin(θ1) -m2*g*sin(θ1-2*θ2) - 2*sin(θ1-θ2)*m2*
   (ω2^2*l2+ω1^2*l1*cos(θ1-θ2)) )/( l1*(2*m1 + m2 - m2*cos(2*θ1 - 2*θ2)) )
9     dω2dt = ( 2*sin(θ1-θ2)*(ω1^2*l1*(m1+m2) + g*(m1+m2)*cos(θ1)
   +ω2^2*m2*l2*cos(θ1-θ2)) )/( l2*(2*m1 + m2 - m2*cos(2*θ1 - 2*θ2)) )
10    return @SVector [dθ1dt, dθ2dt, dω1dt, dω2dt] # the @SVector creates a short-
   lived vector that's faster. It will work without, but a bit slower.
11 end

```

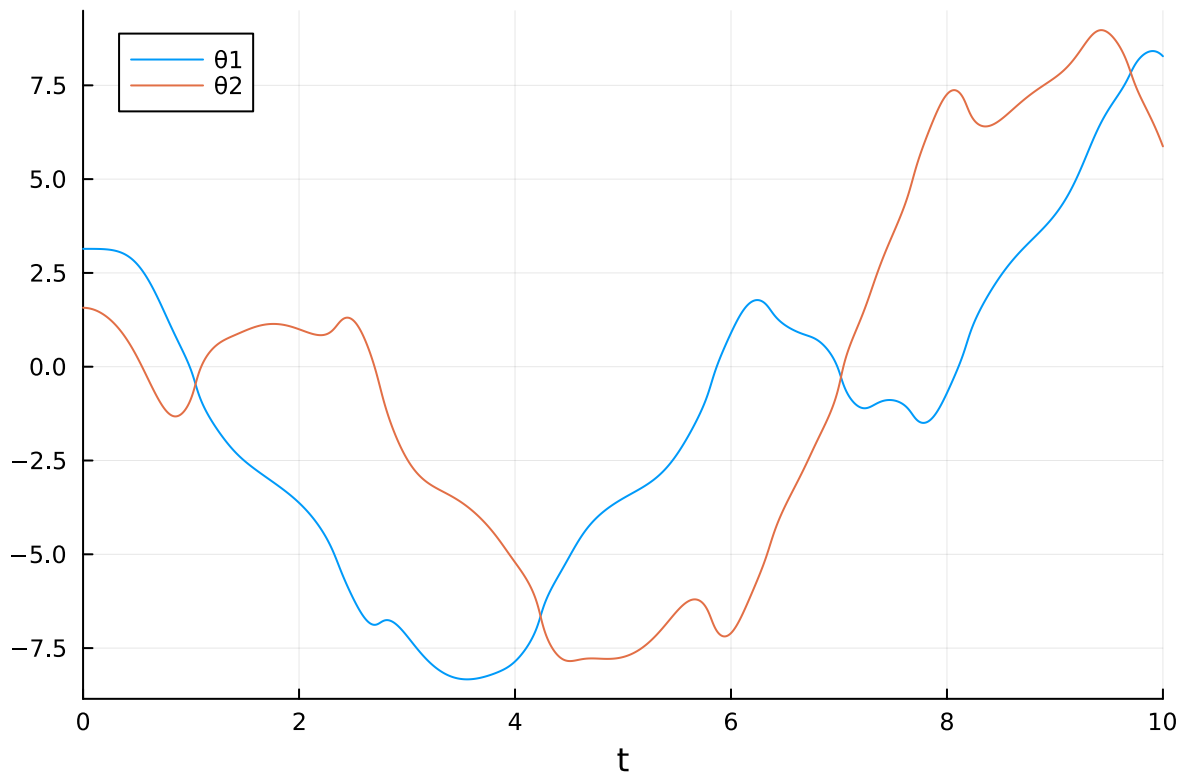
```

1 begin
2   θ1= π; θ2=π/2; ω1=0.0; ω2=0.0
3   u0 = @SVector [θ1, θ2, ω1, ω2 ]
4   m1 = 1.0; m2 = 1.0; l1 = 1.0; l2 = 1.0; g = 10.0
5   p = [m1, m2, l1, l2, g]
6   tspan = (0.0, 10.0)
7   prob = ODEProblem(double_pendulum, u0, tspan, p)
8   @time sol = solve(prob, Tsit5(), reltol=1e-6, abstol=1e-6)
9   animate_double_pendulum(sol, l1, l2)
10 end

```

Saved animation to /tmp/jl_eDUHOXad8x.gif

0.386868 seconds (781.66 k allocations: 52.859 MiB, 5.30% gc time, 99. 93% compilation time) ?



```

1 begin
2     plot(sol, idxs=[1,2], labels=["θ1" "θ2"]) #plot θ1 and θ2
3 end

```

2a: Equilibrium points

Find the 4 equilibrium points of the double-pendulum system.

```
StaticArraysCore.SVector{4, Float64}: [1.0, 1.0, -4.3542, -2.92691]
```

```

1 begin
2     # compute f(x) for system dx/dt=f(x),
3     f_x = double_pendulum([π/5, π/4, 1.0, 1.0], p, 0.0)
4     @show f_x
5 end

```

```
f_x = [1.0, 1.0, -4.354196087652785, -2.926913568466086]
```



2b: Stability

Perturb the initial state slightly for each of the 4 equilibria and simulate the system to find the most stable equilibrium.

```

1 md"""
2 ### 2b: Stability
3
4 Perturb the initial state slightly for each of the 4 equilibria and simulate the
  system to find the most stable equilibrium.
5 """

```

```
1 Enter cell code...
```

2c: Linearized stability

Compute the Jacobian and study the stability at this equilibrium using the eigenvalues. You get half the points if you solve this numerically instead of analytically. Can we conclude that the equilibrium is stable?

1 Enter cell code...

2d: Conservation of Energy

The energy of the double pendulum consists of kinetic and potential energy. Derive and expression for the energy $E = E_{kinetic} + E_{potential}$ and compute it at several times along a trajectory. Assuming that energy should be conserved exactly for this system, the differences over time can be interpreted as numerical error how large are these?

To get you started

$$x_1 = l_1 \sin(\theta_1)$$

$$y_1 = -l_1 \cos(\theta_1)$$

$$x_2 = x_1 + l_2 \sin(\theta_2)$$

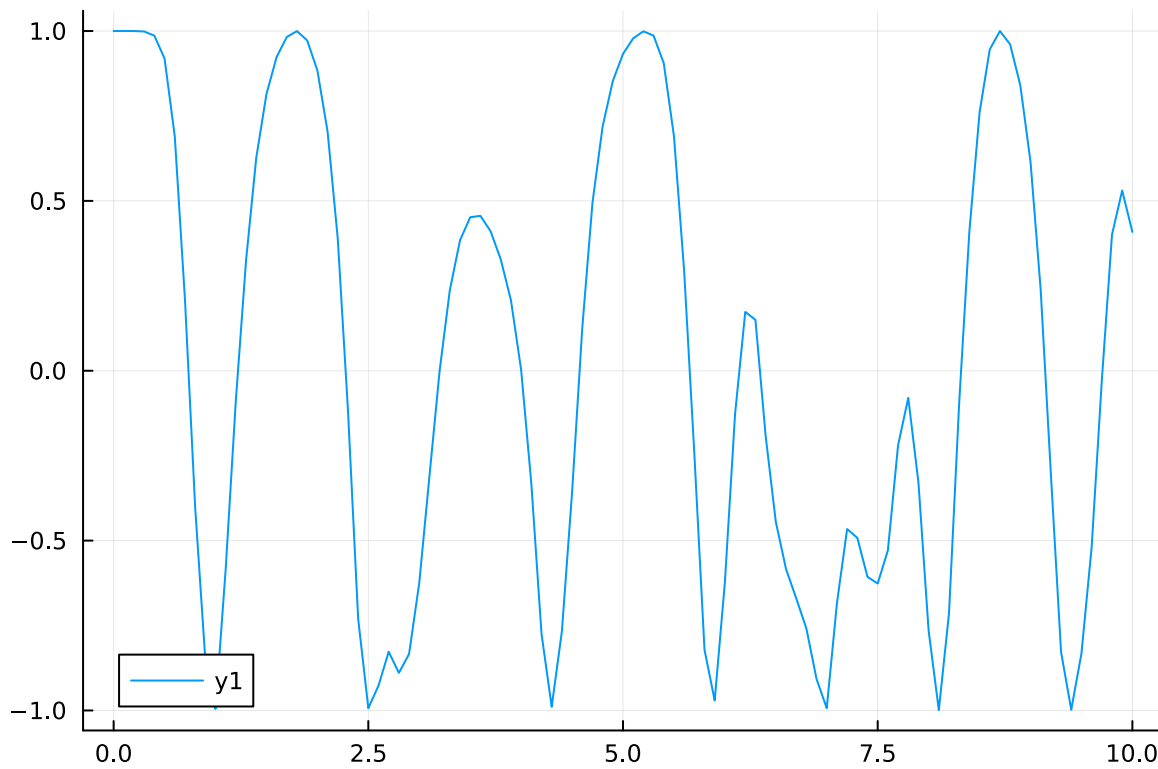
$$y_2 = y_1 - l_2 \cos(\theta_2)$$

$$E = E_{kinetic} + E_{potential} = \dots$$

```
1 md"""
2 To get you started
3
4 $x_1 = l_1 \sin(\theta_1)$
5 $y_1 = -l_1 \cos(\theta_1)$
6 $x_2 = x_1 + l_2 \sin(\theta_2)$
7 $y_2 = y_1 - l_2 \cos(\theta_2)$
8 $E = E_{kinetic} + E_{potential} = \dots$
9
10 """
```

E (generic function with 1 method)

```
1 function E(x,p)
2     θ1, θ2, ω1, ω2 = x
3     m1, m2, l1, l2, g = p
4     # TODO
5     return -l1*cos(θ1) # not correct, but just to get you started
6 end
```



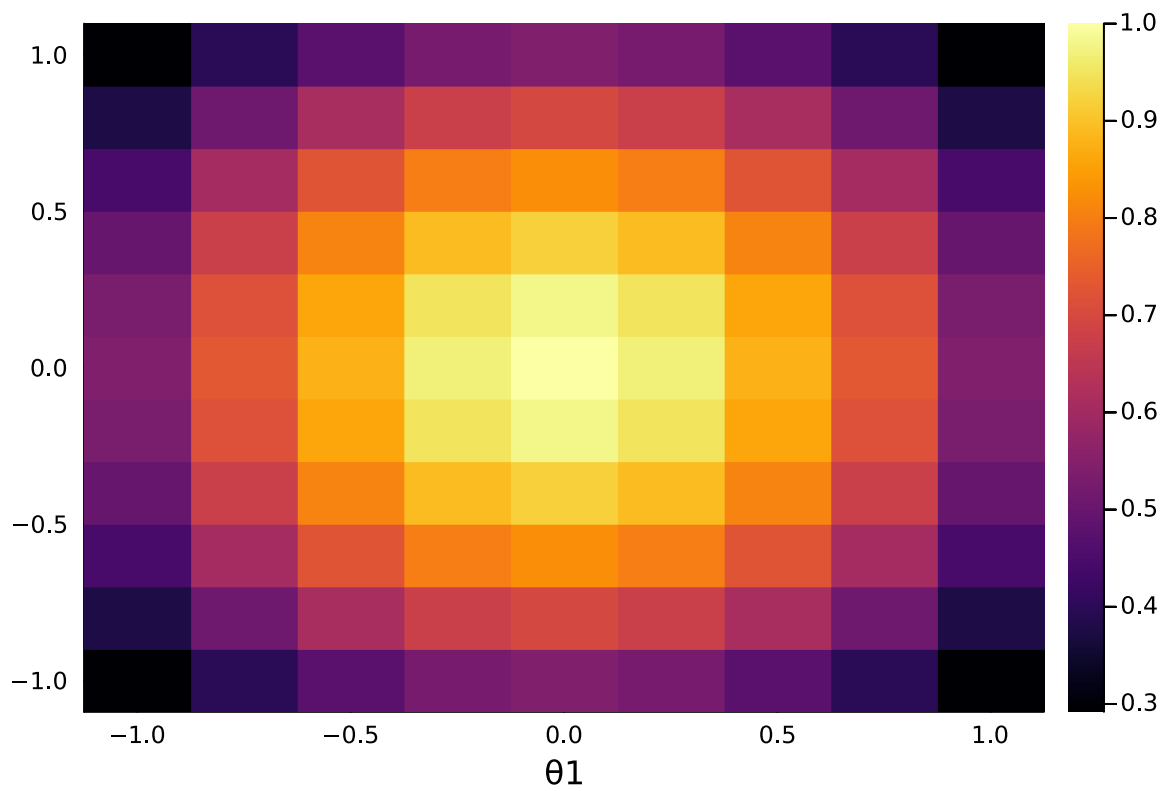
```

1 begin
2   dt=0.1
3   t_range=tspan[1]:dt:tspan[2]
4   Et = [E(sol(t),p) for t in t_range]
5   plot(t_range,Et)
6 end

```

2e: Stability and energy

Plot the energy in a region around the most stable equilibrium with zero velocity, ie $\omega_1 = 0$ and $\omega_2 = 0$. Consider the set of points $S = \{(\theta_1, \theta_2) \text{ where } E([\theta_1, \theta_2, 0, 0]) \leq E_0 + \epsilon\}$ where epsilon is some small positive number and E_0 is the energy in the equilibrium. Can a trajectory that starts in $[\theta_1, \theta_2, 0, 0]$ with $(\theta_1, \theta_2) \in S$ leave the set? What does this say about the stability?



```
1 begin
2    $\theta_1$ _range = -1.0:0.25:1.0
3    $\theta_2$ _range = -1.0:0.2:1.0
4   values=[cos( $\theta_1$ )*cos( $\theta_2$ ) for  $\theta_1$  in  $\theta_1$ _range,  $\theta_2$  in  $\theta_2$ _range] #nonsens
5   heatmap( $\theta_1$ _range, $\theta_2$ _range,values')
6   xlabel!(" $\theta_1$ ")
7 end
```

1 Enter cell code...