EECS 127/227AT Optimization Models in Engineering
Spring 2019 Homework 4

Release date: 9/26/19.
Due date: 10/3/19, 23:00 (11 pm). Please LaTeX or handwrite your homework solution and submit an electronic version.

Submission Format
Your homework submission should consist of a single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

1. 3D reconstruction and pose estimation from two images

In this exercise, we will be learning about how to reconstruct a 3D object from two images taken from non-purely rotational views of the object. To do this, we will first estimate the rotation and translation (rigid body motion) between the two views. Next, we will use this to estimate the 3D position of points belonging to the object.

Note: You may skip the portions in color, although you are encouraged to read through them. They should give you sufficient background information to make this problem more meaningful and there may be a bonus midterm question on some of the concepts.

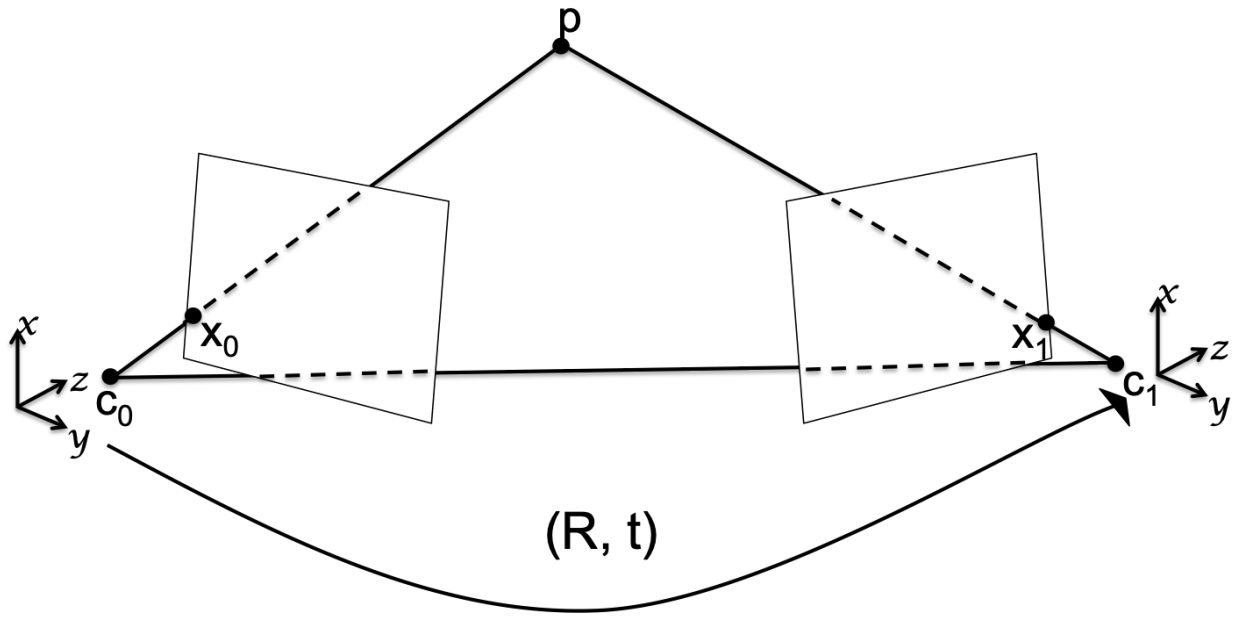


Figure 1: A 3D point $\mathbf{p}$, projected on two image planes. Projected points have coordinate $\mathbf{x}_0, \mathbf{x}_1$. The two views are related by the rotation R and translation t.

How a camera works:

Essentially, a camera works by taking a point in 3D space and projecting it onto a 2D image (plane). A common mathematical model for this camera projection model is the perspective camera model. Under this model, a camera projection can be viewed as follows:
Let $\mathbf{p} = [X, Y, Z]$ be a point in 3D space $\mathbb{R}^3$, then the 2D projection of this point via the camera is modeled via the equation:

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K\mathbf{p} \tag{1}$$

where $K \in \mathbb{R}^{3\times 3}$, known as the camera matrix is generally of the form

$$\begin{bmatrix} fs_x & fs_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

with the contents being physical properties of the camera.
Here, x, y are the coordinates of the projected point on the image plane. In what follows, we will denote $\mathbf{x} = [x, y, 1]^T$.

Two views:

When we have two views of the same point, we can think of it simply as the same point viewed by two cameras at different locations c_1, c_2. The camera locations are related by the rigid motion R, t where R is a rotation matrix in SO(3) and t is a translation vector $\mathbb{R}^3$. SO(3) is the set of rotation matrices in $\mathbb{R}^{3\times 3}$, i.e. the set of all orthonormal matrices with determinant $= 1$ in $\mathbb{R}^{3\times 3}$

Pose estimation:

Putting both pieces together, we can derive an equation relating the coordinates of the projections of a point in both image planes. We will also assume that both cameras have the same camera matrix, K. Let $\mathbf{x}_0, \mathbf{x}_1$ be the coordinates of the projection of the 3D point $\mathbf{p}$ in camera views 0 and 1 respectively. We fix the global coordinate frame to be at the camera location in view 0, so that a point $\mathbf{p}$ in view 0 has coordinate $R\mathbf{p} + t$ in view 1. Then, we can write that:

$$Z_1\mathbf{x}_1 = K(R\mathbf{p} + t) = K(RZ_0K^{-1}\mathbf{x}_0 + t) \tag{2}$$

$$\implies Z_1K^{-1}\mathbf{x}_1 = RZ_0K^{-1}\mathbf{x}_0 + t \tag{3}$$

We assume that we know K, so we can write $K^{-1}\mathbf{x} = \tilde{\mathbf{x}}$ for any $\mathbf{x}$.

$$\implies Z_1\tilde{\mathbf{x}}_1 = Z_0R\tilde{\mathbf{x}}_0 + t \tag{4}$$

We may take the cross product of both sides with t, to obtain.

$$Z_1(t \times \tilde{\mathbf{x}}_1) = Z_0(t \times R\tilde{\mathbf{x}}_0) \tag{5}$$

Taking the dot product with $\tilde{\mathbf{x}}_1$, and noting that the dot product of othorgonal vectors is zero, we obtain

$$Z_1\tilde{\mathbf{x}}_1^T(t\times\tilde{\mathbf{x}}_1) = Z_0\tilde{\mathbf{x}}_1^T(t \times R\tilde{\mathbf{x}}_0) \tag{6}$$

$$\implies 0 = Z_0\tilde{\mathbf{x}}_1^T(t \times R\tilde{\mathbf{x}}_0) \tag{7}$$

$$\implies 0 = \tilde{\mathbf{x}}_1^T(t \times R\tilde{\mathbf{x}}_0) \tag{8}$$

$$= \tilde{\mathbf{x}}_1^T E\tilde{\mathbf{x}}_0. \tag{9}$$

where we define $E = t \times R$.

Problem setup:

Given a point $\mathbf{p} \in \mathbb{R}^3$, and its projection to images in two views $\tilde{\mathbf{x}}_0 = [\tilde{x}_0, \tilde{y}_0, 1]^T, \tilde{\mathbf{x}}_1 = [\tilde{x}_1, \tilde{y}_1, 1]^T$ respectively, and asumming both views are related through a rotational matrix $R \in \mathrm{SO}(3)$ (the 3D rotation group SO(3) is the group of rotation matrices [i.e. orthonormal and determinant = 1] in $\mathbb{R}^{3\times3}$, corresponding to rotations about the origin in $\mathbb{R}^3$) and translation vector $t \in \mathbb{R}^3$. The geometric relationship between the projections in the two views, is given by:

$$\tilde{\mathbf{x}}_1^T E \tilde{\mathbf{x}}_0 = 0, \tag{10}$$

where

$$E = [t]_\times R. \tag{11}$$

The matrix $E = [t]_\times R$ is known as the *essential matrix*. The cross product matrix $[t]_\times$ is a skew -symmetric matrix (i.e. $[t]_\times^\top = -[t]_\times$) representing the cross product of t with any appropriately sized vector. It returns the $\mathbf{0}$ vector when pre- and post-multiplied by the same vector, since the cross product is orthogonal to its factors. Concretely, given a vector $t = [t_0, t_1, t_2]^T$,

$$t \times x = [t]_\times x = \begin{bmatrix} 0 & -t_2 & t_1 \\ t_2 & 0 & -t_0 \\ -t_1 & t_0 & 0 \end{bmatrix} x \tag{12}$$

for any appropriately sized vector x.

1.1. Properties of the Essential Matrix:

A nonzero matrix $E \in \mathbb{R}^{3\times3}$ is said to be an essential matrix if and only if

(a) It has a singular value decomposition $E = U\Sigma V^T$, with

$$\Sigma = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(b) and $U, V \in \mathrm{SO}(3)$, i.e. U, V are rotation matrices. Note that in general, the U, V matrices of an SVD are only required to be orthorgonal, the extra requirement here is the the $\det(U) = \det(V) = 1$, making U, V rotation matrices.

The space of matrices with the properties above is known as the *essential space*.

1.2. Estimating the Essential Matrix:

The equation (9) is linear in E. To see this, given the projections $\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1$ of a point $\mathbf{p}$ in view 0 and view 1 respectively, where $\tilde{\mathbf{x}}_0 = [\tilde{x}_0, \tilde{y}_0, 1]^T$ and $\tilde{\mathbf{x}}_1 = [\tilde{x}_1, \tilde{y}_1, 1]^T$ and the essential matrix E is written as

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}, \tag{13}$$

we may re-write equation (9) as $\mathbf{a}^T E^s = 0$, with:
$\mathbf{a} = [\tilde{x}_1\tilde{x}_2, \tilde{x}_1\tilde{y}_2, \tilde{x}_1, \tilde{y}_1\tilde{x}_2, \tilde{y}_1\tilde{y}_2, \tilde{y}_1, \tilde{x}_2, \tilde{y}_2, 1]^T$
$E^s = [e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}]^T$

Given n points $p_0, \ldots, p_{n-1}$ and their projections in both views, we may construct a system of linear equations:

$$\mathcal{X} E^s = 0 \tag{14}$$

with $\mathcal{X} \coloneqq [\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{n-1}]^T$. For the basic algorithm, we will need at least eight of these points. Once we have solved for E^s, we may rearrange to components to obtain E.

In practice, due to noise in measurements of the points and their projections, $\mathcal{X}$ may not have a non-trivial nullspace and we will have to be satisfied with solving

$$\min_{E^s} ||\mathcal{X} E^s||_2^2 \tag{15}$$

instead of (14). In general, the solution to this minimization problem need not lie in the essential space — once we find a solution F to the above problem, we will need to project it onto the Essential Space.

(a) **Projection onto the Essential Space** Given a matrix $F \in \mathbb{R}^{3\times3}$, with SVD $F = U\Sigma V^T$, and $U, V \in \mathrm{SO}(3)$, find the solution E^* to the following problem:

$$\operatorname*{argmin}_{E\in\mathcal{E}} ||E - F||_f^2. \tag{16}$$

Here $||.||_f$ is the Frobenius norm, and $\mathcal{E}$ is the essential space. You should assume that $E^* = U\Sigma_\sigma V^T$, i.e. F and E^* share the same matrices $U, V \in \mathrm{SO}(3)$ matrices.
Hint: Σ_σ should be of the form

$$\begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(b) **Recovering the translation vector:** Recall that $E = [t]_\times R$. Show that the translation vector t is in the left null space of E.

(c) Unfortunately, we cannot say more than this, given just the essential matrix E. We may estimate the direction of t as a unit norm vector in the left null space of E. In addition, this procedure gives us two solutions, since for any unit norm vector $\hat{t}$ in the left null space of E, $-\hat{t}$ is also a valid solution. Later we will see how to recover the proper sign.

Write a procedure in the IPython notebook to recover the two directions of translation from the essential matrix E using SVD. Normalize the resulting translation vector before returning it.

(d) **Recovering the Rotation matrix:** Next, we will attempt to recover the rotation between both views. Because we only know the estimate of the essential matrix up to sign, $\pm E$ are both solutions to our minimization problem (15), and also because typical SVD algorithms

do not provide a guarantee that the U, V matrices are rotation matrices, we will have four different solutions for our rotation matrix, from which only two will be true rotation matrices, i.e. $\det(R) = 1$.

Given an essential matrix of the form $E = U\Sigma V^T$, and defining

$$R_Z^T(\frac{\pi}{2}) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_Z^T(-\frac{\pi}{2}) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

we can obtain the four estimates for the rotation matrix R as,

$$R = \pm U R_Z^T(\pm\frac{\pi}{2}) V^T.$$

.

We choose the two R's with positive determinants to be our estimates for R. We will see how to settle on the final answer later. Write a procedure in the Ipython notebook to return the valid rotation matrices from a given essential matrix E.

(e) **Estimating 3D position from 2D matches: Triangulation**

Our next step is to estimate the 3D position of a point $\mathbf{p}$, given its projection onto the two views $\mathbf{x}_0 = [x_0, y_0, 1]^T, \mathbf{x}_1 = [x_1, y_1, 1]^T$, an estimate of the rotation R and translation t between the two views, and camera parameters in the camera matrix K (recall that $K^{-1}\mathbf{x} = \tilde{\mathbf{x}}$. For clarity, $\mathbf{x}, \tilde{\mathbf{x}}$ are both projections of a 3D point, but in different units).

The full derivation of this solution is beyond the scope of this course; however, to obtain the 3D position of the point $\mathbf{p}$, we start by constructing the matrices

$$P_0 = K[I_{3\times3}, \mathbf{0}_{3\times1}] = \begin{bmatrix} \leftarrow P_0^{1T} \rightarrow \\ \leftarrow P_0^{2T} \rightarrow \\ \leftarrow P_0^{3T} \rightarrow \end{bmatrix} \in \mathbb{R}^{3\times4}, \;\; P_1 = K[R, t] = \begin{bmatrix} \leftarrow P_1^{1T} \rightarrow \\ \leftarrow P_1^{2T} \rightarrow \\ \leftarrow P_1^{3T} \rightarrow \end{bmatrix} \in \mathbb{R}^{3\times4}. \tag{17}$$

Note that the notation $P_*^{1T}, P_*^{2T}, P_*^{3T}$ are just the rows of P_*, they have nothing to do with $\mathbf{p}$.

Then we construct the matrix

$$A = \begin{bmatrix} x_0(P_0^{3T}) - P_0^{1T} \\ y_0(P_0^{3T}) - P_0^{2T} \\ x_1(P_1^{3T}) - P_1^{1T} \\ y_1(P_1^{3T}) - P_1^{2T} \end{bmatrix} \in \mathbb{R}^{4\times4}. \tag{18}$$

Then the following equation helps us find the 3D point $\mathbf{p}$

$$A \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \mathbf{0}_{4\times1}. \tag{19}$$

Here, we are simply looking for an element in the nullspace of A with 1 as the fourth component. To find $\mathbf{p}$, we may simply find a (non-trivial) element in the null space of A , divide this vector by its 4th component and take the first 3 components of the resulting vector as $\mathbf{p}$.

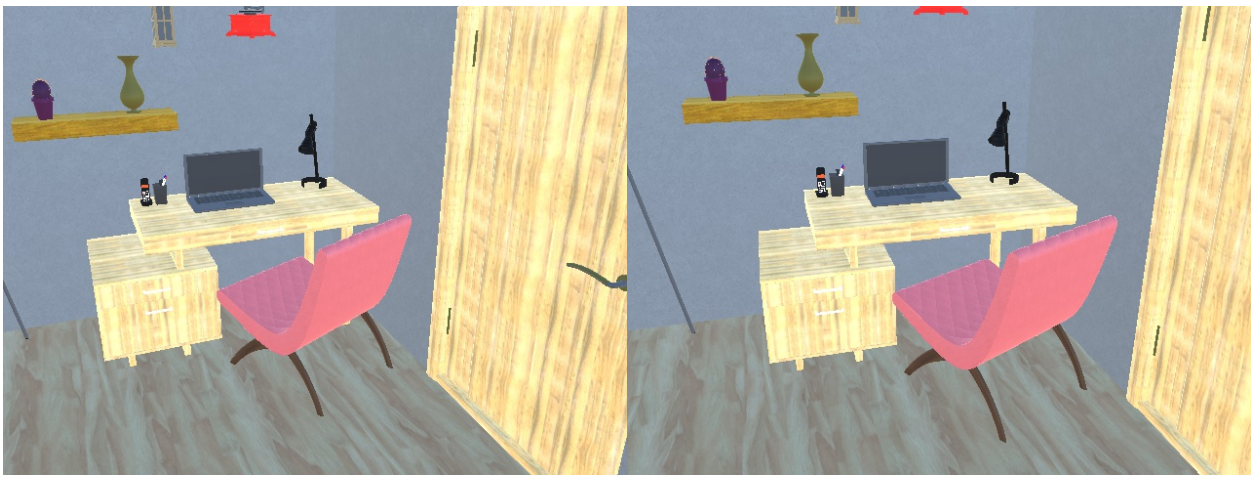
Write a procedure in your Ipython notebook to triangulate a point $\mathbf{p}$, given its projection onto both images $\mathbf{x}_0, \mathbf{x}_1$ an estimate of P_0, P_1.

(f) Now that you have completed the triangulation function, we will make use of it to select the optimal R, t from the 4 different combinations (2 for R, and 2 for t) of R, t we had previously.

Essentially, we will triangulate a set of 3D points from the projections of those points and choose the combination of R, t that gives us most points in front of the camera, i.e. the z-coordinates (in a camera system the z direction points outwards) of the points are positive. False combinations would give us a lot of points behind the camera, but we wouldn't have been able to see those points if this was the case.

We have written most of this function for you, but want you to choose the best combination of R, t given a count of the number of points in front of the camera. Complete this part of the function in your IPython notebook.

(g) **Reconstructing a chair from two views**



(a) View 0 (b) View 1

Figure 2: Two views of the same scene

Finally, we are able to put all these ideas together to reconstruct an object. We have chosen to reconstruct the chair in the scenes below. Run all the cells of the Ipython notebook. Do you see a 3D chair? Take a screenshot of the chair with all parts visible and attach to your write up.

2. Higher dimension least-square

You are a world renowned animal specialist who has been studying the movement of animals to answer the pivotal question of our generation: can we approximate the movement of animals as a mathematical function? You controversially believe this is possible, and decide to test your hypothesis by observing the hunt of a tiger. You go to the best place to observe tigers, the Ranthambore National Park in Rajasthan, India, and set up a place where you can observe these majestic creatures hunt.

After a week in the wilderness, you have observed the tiger and realize that when it hunts, it moves according to a cubic function. However, your qualitative observations are insufficient if you are to convince the rest of the animal behavior scientists that your hypothesis holds true. You need to give them the exact function that describes how the tiger moves.

(a) Luckily, you wrote down your exact observations of the tiger at 4 distinct points in the form of (x, y) coordinates, given below. As you know thanks to CS70, we have enough information to solve for a cubic polynomial. Using the points given, setup a system of linear equations in the form $\mathbf{A}\vec{x} = b$ that you can use to solve for the polynomial. Hint: Any cubic function can be described as $ax^3 + bx^2 + cx + d$ for some a, b, c, d.

x	y
1	8
2	12
3	10
4	-4

(b) Decompose the $\mathbf{A}$ matrix you constructed in the previous part using QR decomposition. How could you use this decomposition to solve for $\vec{x}$? Note: You may use a calculator (including Python or other scripting languages) to speed up algebraic work. Round to 3 decimal places and make sure to show your work. If using Python etc., please calculate everything explicitly (don't just call a library!).

Unfortunately, you need to leave the beauty of the wilderness and return to Berkeley for your EECS127 midterm (how inconvenient!). However, you would like to continue collecting data and work on your hypothesis remotely. So, you leave behind some sensors that can track the tiger hunt for you. Now that we have sensors, we have imperfect readings (because the sensors will have some noise and inaccuracies). Rather than take 4 exact measurements, we take many measurements to try to approximate the tiger's path.

(c) How would you update your setup in part (a)? How can we solve this system?

(d) (Jupyter Notebook) Follow the instructions in the Jupyter notebook in order to solve the new system in part (c)

3. Dynamical system as linear equation

In this exercise, we consider a linear dynamical system as a matrix multiplication. We then apply our findings to compute the solution of the pendulum.

Let us consider the following dynamical system:

$$\dot{x} = Ax + Bu$$

with u a control variable and x the state of the system.

We want to approximate the solution $x(t)$ at time $t > 0$, given $x(0) = x_0$.

We divided the time interval $[0, t]$ in n time step of $\Delta t = \frac{t}{n}$: $[0, t/n], [t/n, 2t/n], \dots [(n-1)t/n, t]$.

We approximate the linear equation by: $\frac{x_{k+1} - x_k}{\Delta t} = Ax_k + Bu$ where $x_k = x(k\Delta t)$ and $0 \leq k < n$.

First we assume that $u = 0$, so $x_{k+1} = (A\Delta t + I)x_k$.

(a) Assuming that A is diagonalizable show that $\exists P,\ \lambda_1 \geq \dots \geq \lambda_m$, such that

$$x_{k+1} = P \begin{bmatrix} 1 + \lambda_1 \Delta t & 0 & \dots & 0 \\ 0 & 1 + \lambda_2 \Delta t & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & 1 + \lambda_m \Delta t \end{bmatrix} P^{-1} x_k$$

(b) Show that

$$x(t) = P \begin{bmatrix} 1 + \lambda_1 \frac{t}{n} & 0 & \dots & 0 \\ 0 & 1 + \lambda_2 \frac{t}{n} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & 1 + \lambda_m \frac{t}{n} \end{bmatrix}^n P^{-1} x_0$$

(c) Show that, for $n \in \mathbb{N}$ big enough, $\forall a \in \mathbb{R}$, $(1 + \frac{a}{n})^n = e^{n \ln(1 + \frac{a}{n})}$. State the values of n for which the equality is true.

Hint: Recall that $\ln(x)$ is not defined for some value of x.

(d) Using the fact that $\ln(1 + \frac{a}{n}) \sim \frac{a}{n}$ when $n \to \infty$, show that $\lim_{n \to \infty} (1 + \frac{a}{n})^n = e^a$.

(e) Show that

$$\lim_{n \to \infty} x(t) = P \begin{bmatrix} e^{\lambda_1 t} & 0 & \dots & 0 \\ 0 & e^{\lambda_2 t} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & e^{\lambda_m t} \end{bmatrix} P^{-1} x_0$$

Now assume that $u \neq 0$. We are interested in the existence of an equilibrium point x_{eq} such that $0 = Ax_{eq} + Bu$.

(f) Show that an equilibrium point exists if and only if $\forall x \in N(A^\top),\ x^\top Bu = 0$.

Hint: You might show first that if $Ax_{eq} = -Bu$ then $\forall x \in N(A^\top), x^\top Bu = 0$. Then, you might use the fundamental theorem of linear algebra.

(g) Let u and x_{eq} such that $0 = Ax_{eq} + Bu$. Let $x(0) = x_0$ and $\lambda_1 < 0$. Show that $\lim_{t \to \infty} x(t) = x_{eq}$.

Some applications.

(h) Let consider $A = (-1)$, $x_0 = 10$ and $u = 0$, what is $x(t)$ (using e.)?

The pendulum.

Let consider the following system: $\ddot{y} = -\omega^2 y$. Let $X = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$. We have that $\dot{X} = \begin{bmatrix} \dot{y} \\ \ddot{y} \end{bmatrix}$.

(i) Give A a 2×2 matrix such that $\dot{X} = AX$. Also, write the matrix in the notebook "Dynamical system.ipynb".

(j) Assume that λ is an eigenvalue of A, show that it implies that $\lambda^2 = -\omega^2$.

We can diagonalize matrices in $\mathbb{C}$. For this specific case of A then $\lambda_1 = j\omega$ and $\lambda_2 = -j\omega$. Also, the result of question e. is still true and $e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$. The solution of the system is $y = y_0 \cos(\omega t) + \frac{\dot{y}_0}{\omega} \sin(\omega t)$

(k) Complete this question in the notebook by filling in X in terms of n, t, X_0, and the eigenvalues/eigenvectors of A.

Bonus. With respect to this exercise explain the quote of Iron Man in Avengers Endgame: "the singular value decomposition of the inverted Mobius strip will give you an eigenvalue that will bring you back in the past."