Multivariable Control Problems

Coursework

Ashutosh Mukherjee

26.02.2023

Contents

1 Topology and Mobility Analysis **3**

2 Geometry and Kinematics **4**

3 Dynamics **5**

4 Control **6**

4.1 Exercise 11 6

4.1.1 Trade-offs between Euler Integrator and Fourth-order Runge-Kutta 6

4.1.2 Linearization of the System about the upper equilibrium point . . 7

4.1.3 Linear Quadratic Regulator Design 8

4.1.4 Strategy for Swing-up and Balancing 8

1 Topology and Mobility Analysis

2 Geometry and Kinematics

3 Dynamics

4 Control

Exercise 11

In order to make use of the convenience of inbuilt tools already available, the *ode45* solver from MATLAB® is made use of to emulate the fourth-order Runge-Kutta scheme. The implementation of the same along with the euler integrator can be found in *Pendulum.m*.

Trade-offs between Euler Integrator and Fourth-order Runge-Kutta

1. The euler integrator implemented is the forward euler method which is an explicit method and is highly sensitive to the time step-size. Very large step-sizes can lead to numerical instabilities. On the other hand, the fourth-order Runge-Kutta handles smaller step-sizes much better.

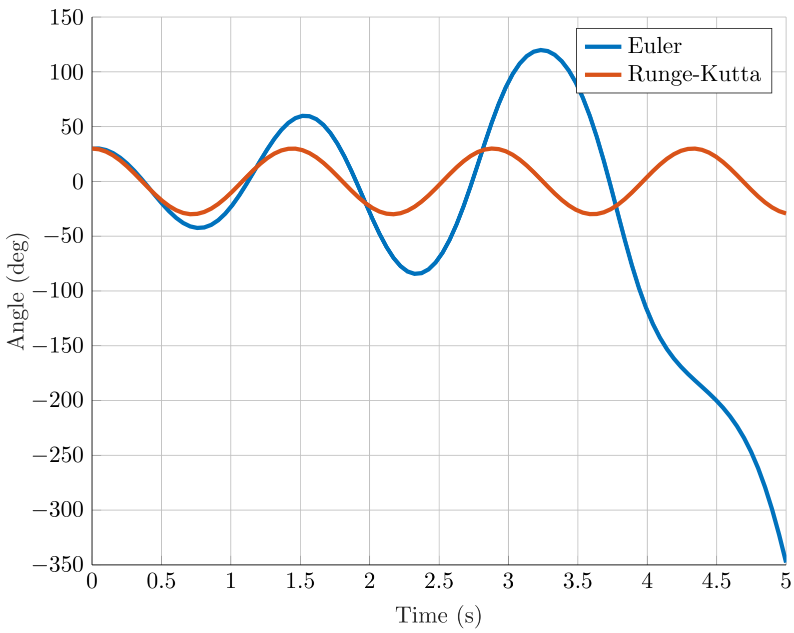

Figure 4.1: Solution of the open-loop pendulum from Euler-Integrator and Runge-Kutta for time step-size = 0.05

Figure 4.1 demonstrates the diverging behavior of the euler integrator for step-size of 0.05 seconds. On the other hand, the two solutions are a lot closer for 0.005 seconds as demonstrated by Figure 4.2.

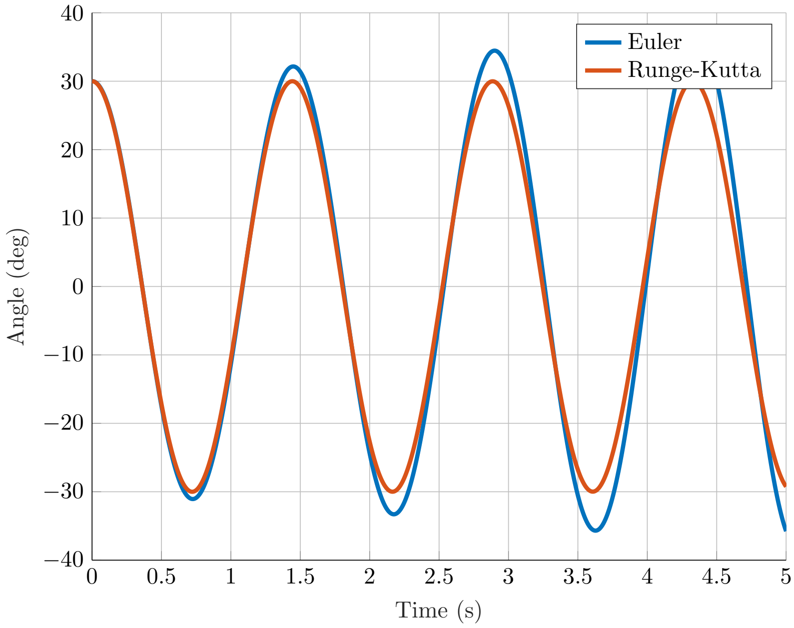

Figure 4.2: Solution of the open-loop pendulum from Euler-Integrator and Runge-Kutta for time step-size = 0.005

2. Euler integrator is easier to implement compared to the fourth-order Runge-Kutta since the fourth-order Runge-Kutta requires four approximations of the derivative and Euler integrator requires only one.

3. Error accumulation over time is faster in Euler as already demonstrated in Figure 4.1.

Linearization of the System about the upper equilibrium point

The non-linear state-space is given by

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \frac{u}{ml^2} - \frac{b\dot{\theta}}{ml} - \frac{g\sin\theta}{l} \end{bmatrix} = \boldsymbol{f}\left(\boldsymbol{x}, u\right)$$

The linear states will be given by

$$\begin{bmatrix} z \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \pi - \theta \\ -\dot{\theta} \end{bmatrix} = \begin{bmatrix} \delta\theta \\ \delta\dot{\theta} \end{bmatrix}$$

The first-order taylor series approximation around $\boldsymbol{x}_E = (\pi, 0)$ is given by

$$
\begin{aligned}
\begin{bmatrix} \delta\dot{\theta} \\ \delta\ddot{\theta} \end{bmatrix} &\approx \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=(\pi,0)} \begin{bmatrix} \delta\theta \\ \delta\dot{\theta} \end{bmatrix} \\
\implies \begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{b}{ml} \end{bmatrix} \begin{bmatrix} \pi - \theta \\ -\dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u \\
\implies \begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{b}{ml} \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u
\end{aligned}
$$

Hence, for the linear system around the upper equilibrium point,

$$
\begin{aligned}
\boldsymbol{A} &= \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{b}{ml} \end{bmatrix} \\
\boldsymbol{b} &= \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}
\end{aligned}
$$

Linear Quadratic Regulator Design

A LQR is designed assuming full-state feedback. Assuming an infinite-time horizon, the algebraic Riccati equation is solved in order to obtain the LQR gain matrix $\boldsymbol{K}_{\mathrm{LQR}} \in \mathbb{R}^{1\times 2}$. Positive definite and symmetric matrices $\boldsymbol{Q} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{R} \in \mathbb{R}$ are used to prioritize the minimization of the transient energy (states reach reference quicker) and the minimization of the actuator energy (actuator limits are prioritized) respectively.

The closed-loop linear system with LQR is formulated as

$$
\begin{aligned}
\dot{\boldsymbol{z}} &= \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}u & u &= -\boldsymbol{K}_{\mathrm{LQR}}\boldsymbol{z} \\
\dot{\boldsymbol{z}} &= \left(\boldsymbol{A} - \boldsymbol{b}\boldsymbol{K}_{\mathrm{LQR}}\right)\boldsymbol{z} & \boldsymbol{A}_{\mathrm{CL}} &= \boldsymbol{A} - \boldsymbol{b}\boldsymbol{K}_{\mathrm{LQR}}
\end{aligned}
$$

Strategy for Swing-up and Balancing

1. Perform swing-up using energy error till a close value to 180 degrees is reached.

2. Define a threshold (ϵ) for linear behaviour of the pendulum about its upright equilibrium point. For this implementation, $\epsilon = 10$ degrees.

3. Switch to the LQR Controller once the linear threshold is reached in order to balance the pendulum at its upright position.

For this implementation, $\boldsymbol{Q} = 0.1\boldsymbol{I}_{2\times 2}$ and $R = 1$. The values of $\boldsymbol{Q}$ and R can naturally be fine-tuned in order to ensure a balance between faster response time and the actuator limit of $1Nm$.

This strategy implementation can be found in *swingUp_Balance_Pend.m*.

Figure 4.3 clearly shows the decent performance of the implemented strategy where the upright position is maintained as compared to when only the energy based controller is used.

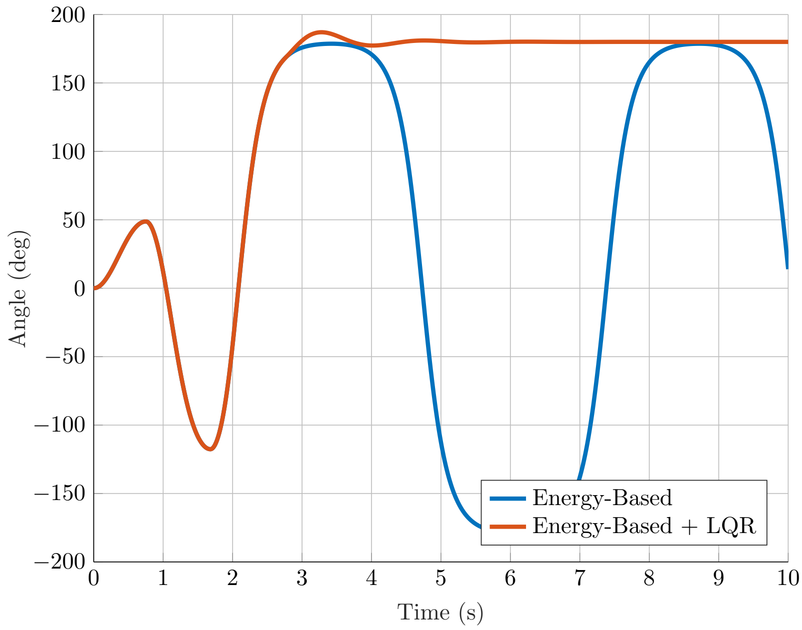

Figure 4.3: Swing-up and Balance Response