**Compulsory exercise to lessons in Lecture 4**

**Systems theory**

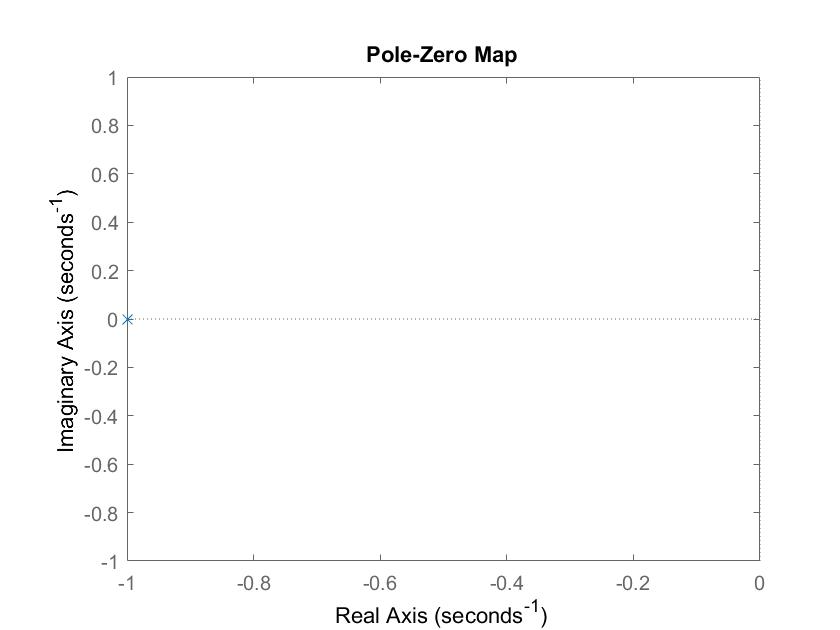
1.      **Learning Matlab Control System Toolbox (“CST”):** Work through the following sections of [this tutorial to the Matlab Control System Toolbox](http://techteach.no/publications/control_system_toolbox/): 1, 2, 3 - intro, 3.1, 3.2, 5. (You are not expected to report anything from this task.)

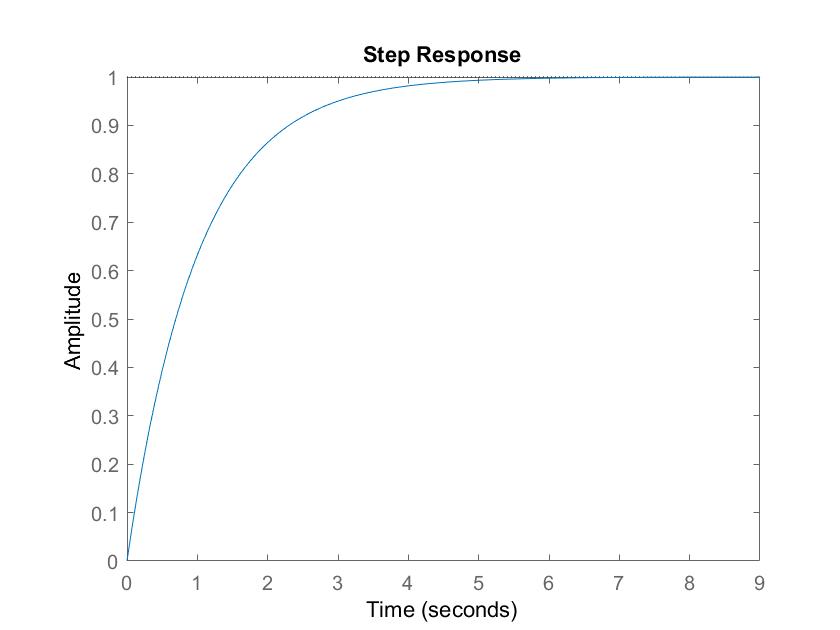
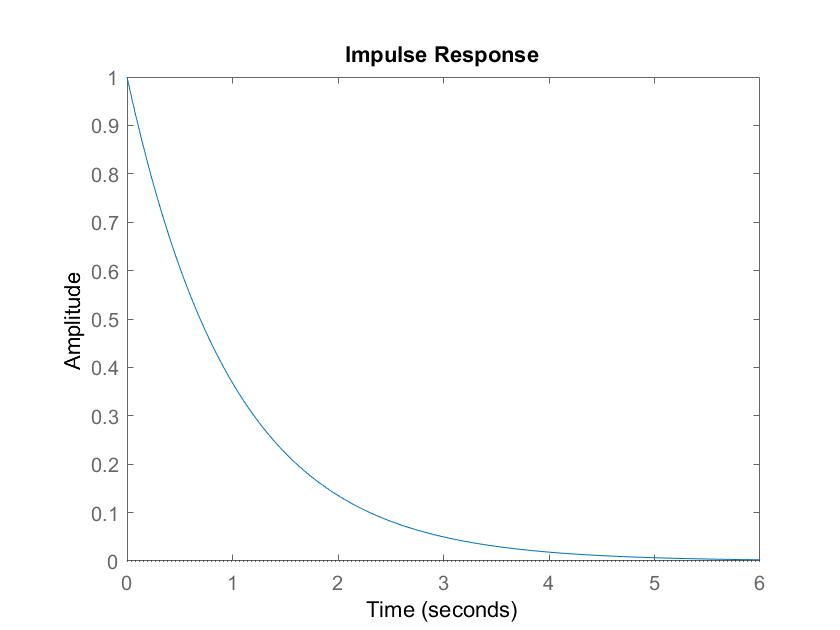
2.      **Using CST: Stability analysis and simulation of transfer functions:** See Exercise 4.2 in the [exercise book](http://techteach.no/publications/books/advanced_dynamics_and_control/advanced_dynamics_control_exercises.pdf). For each of the transfer functions: Calculate the poles, plot the poles, determine the stability property (“manually”), and simulate both the impulse response and the step response.

**Solution - 2. Exercise 4.2**

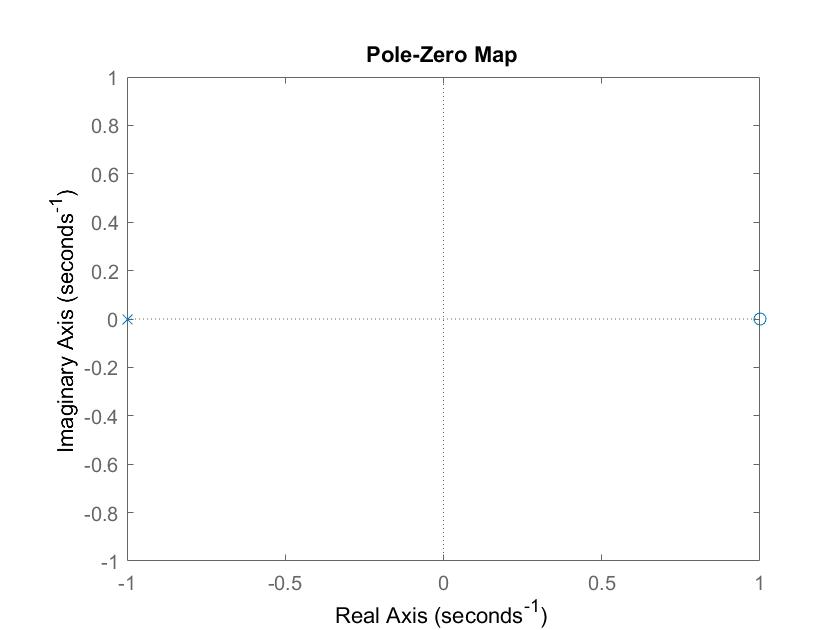
Transfer function

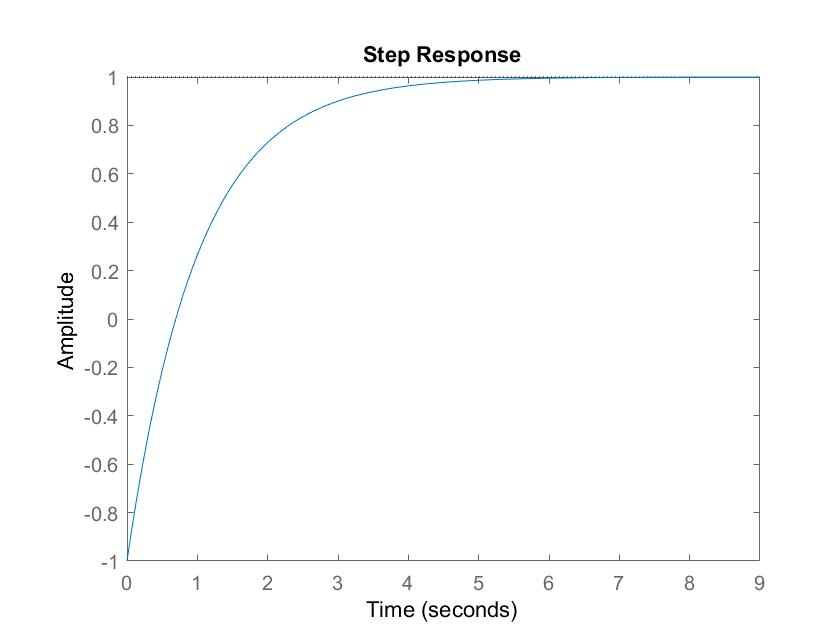
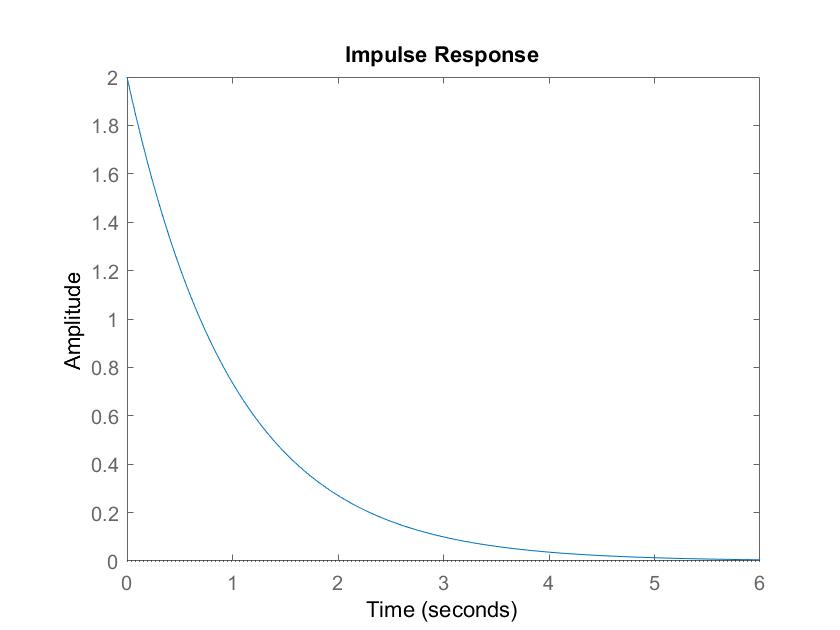
The system has pole , which is asymptotically stable.



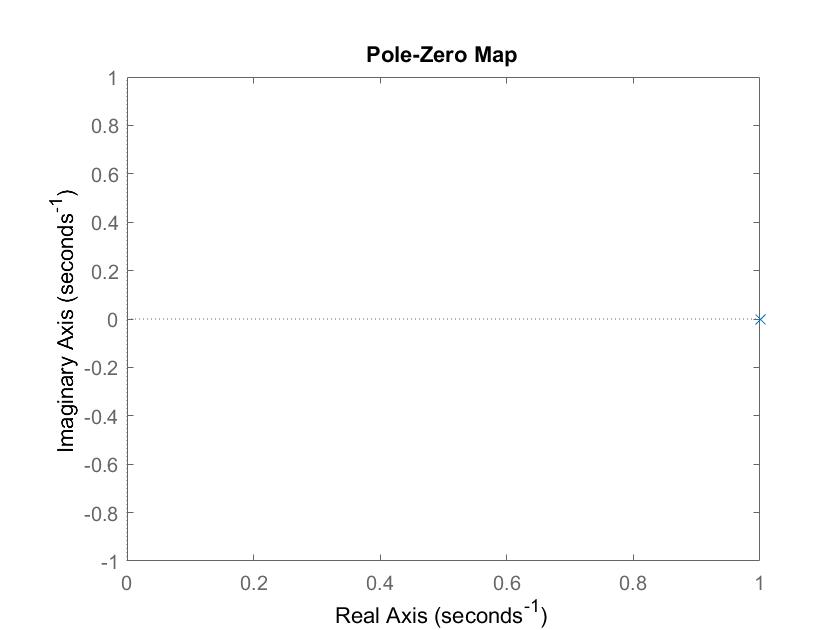
The step response and the impulse response of are

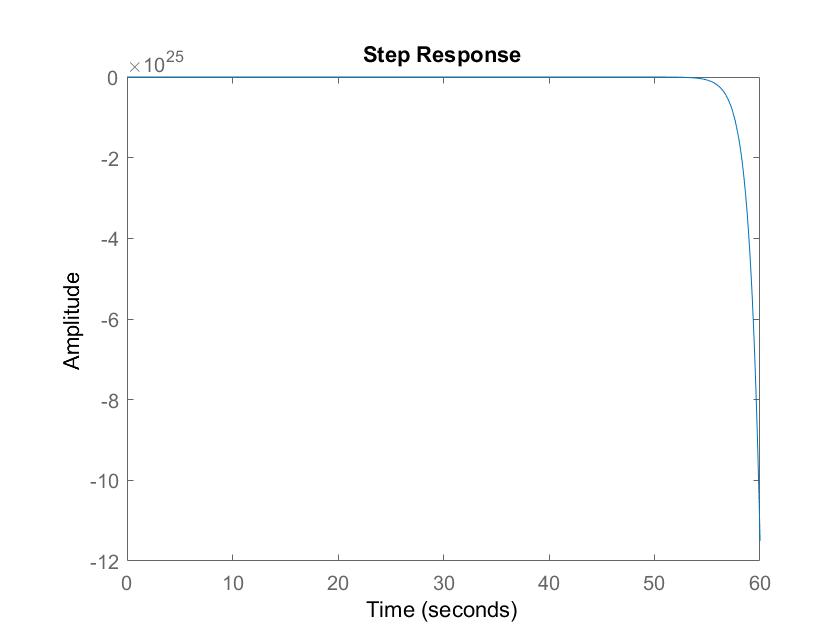
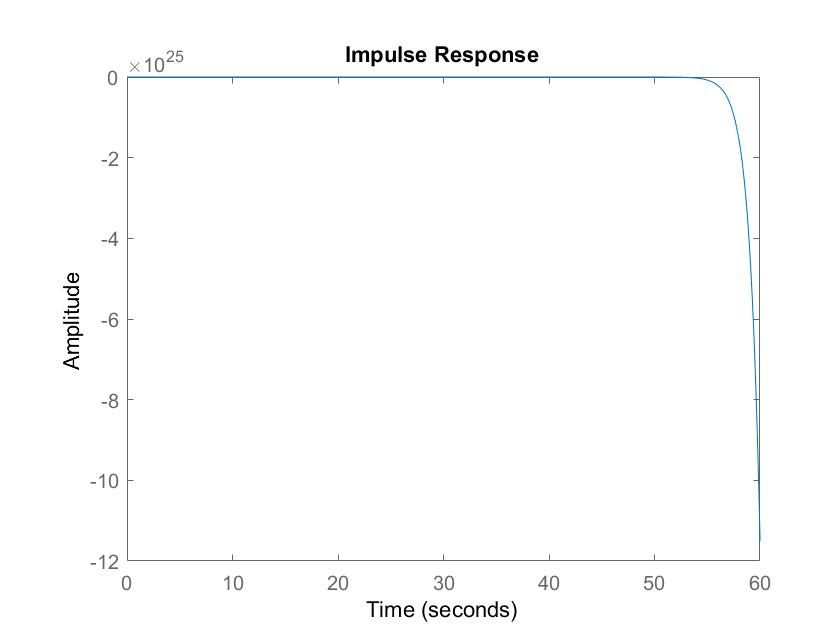
The system has pole , which is asymptotically stable.



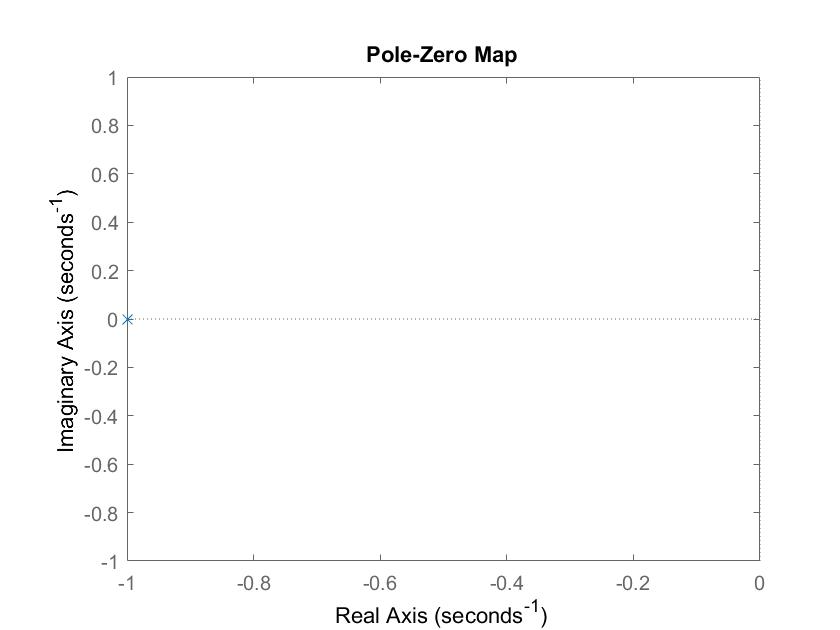
The step response and the impulse response of are

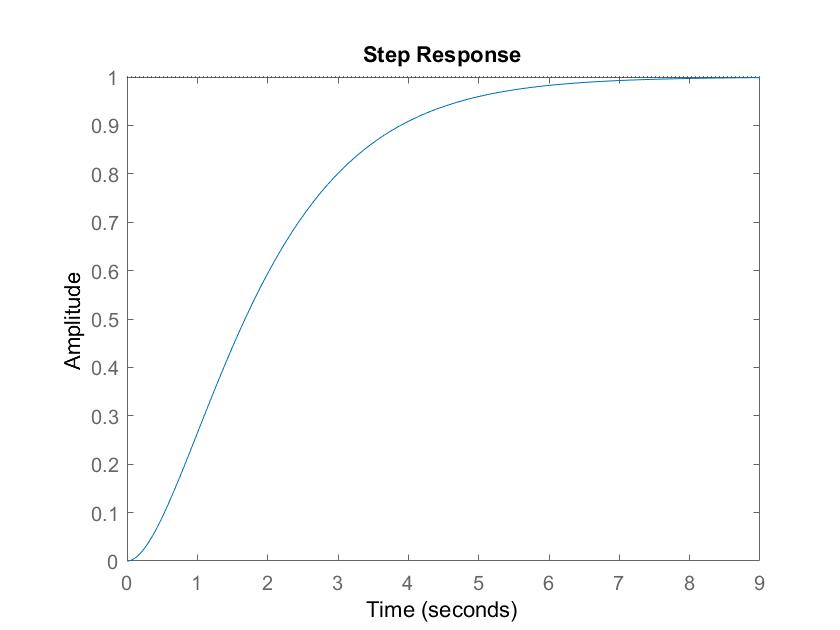
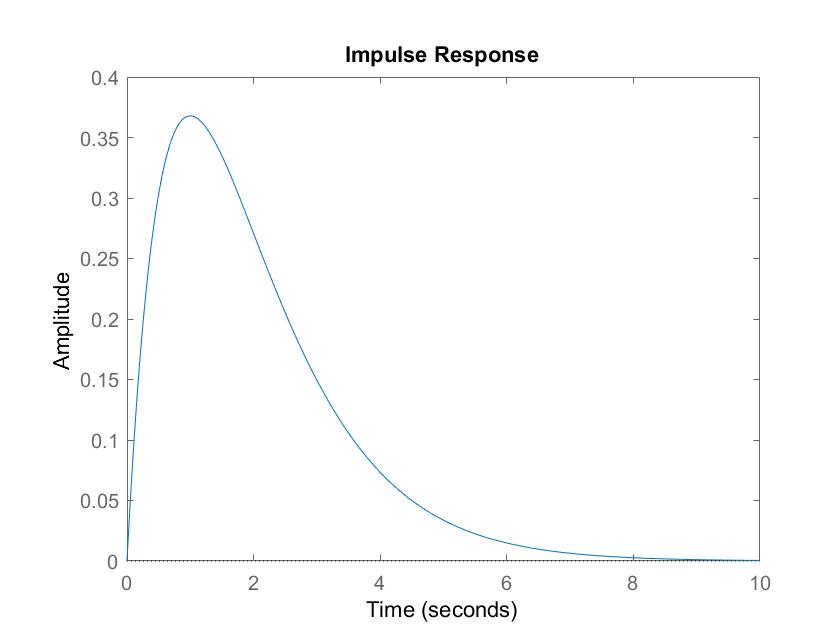
The system has pole , which is unstable.



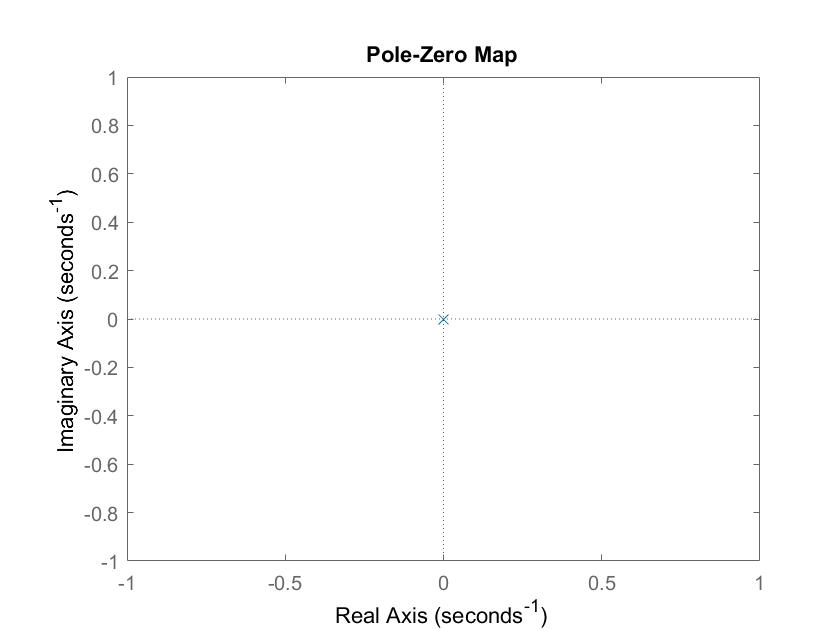
The step response and the impulse response of are

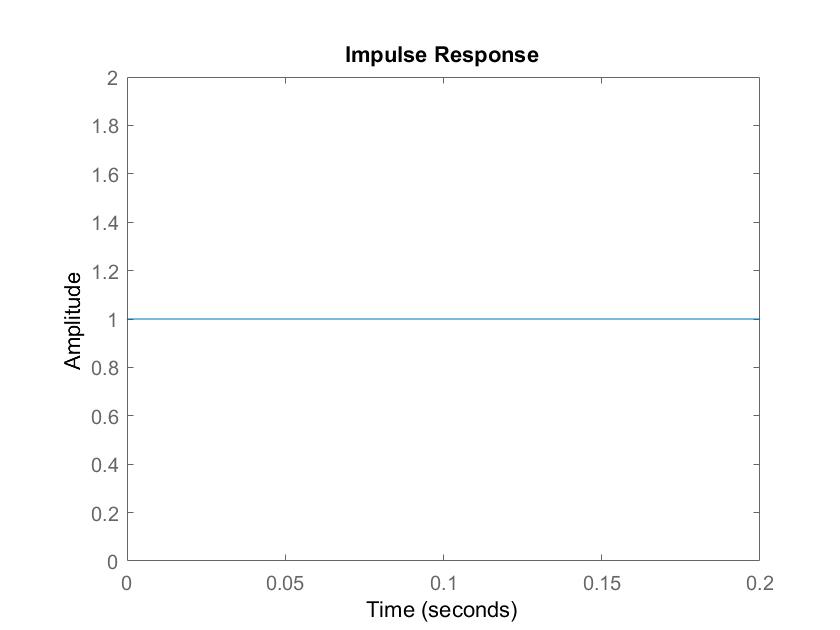
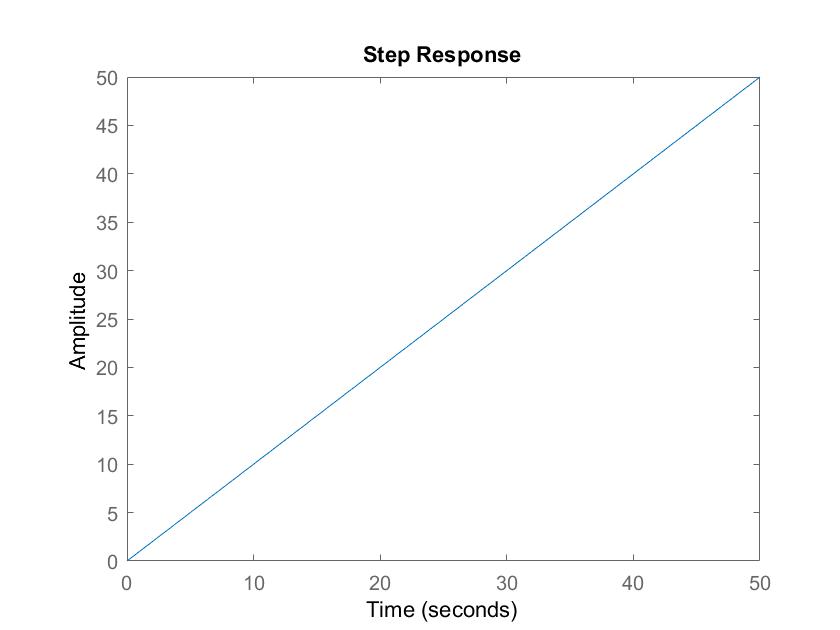
The system has pole , which is asymptotically stable.



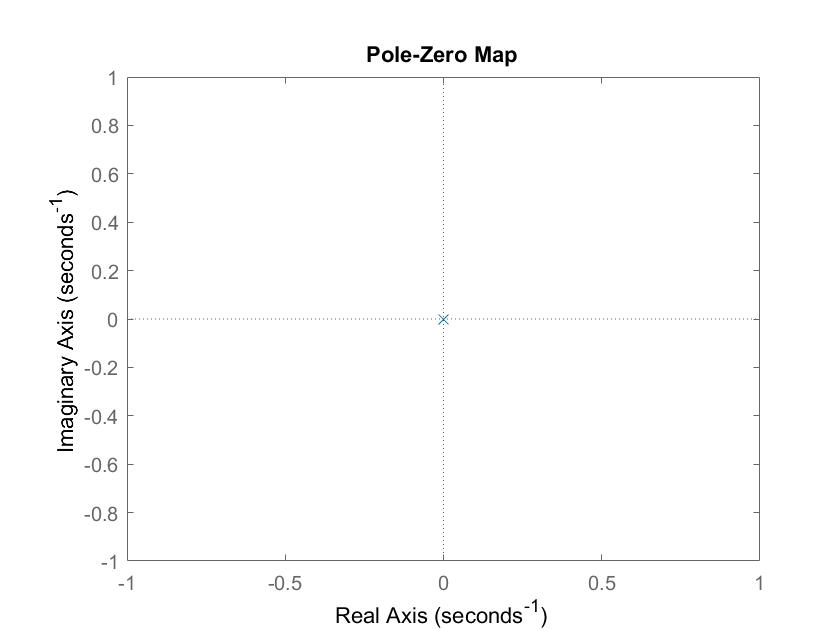
The step response and the impulse response of are

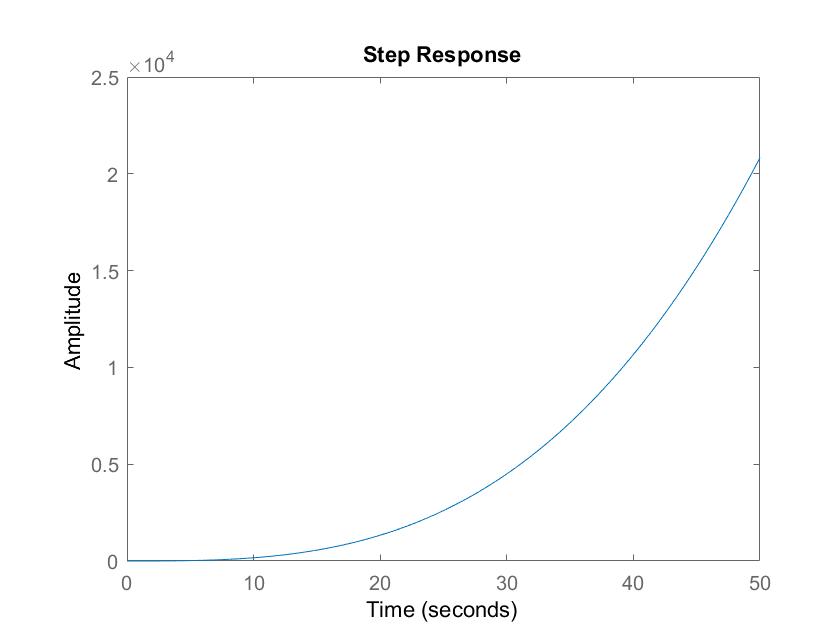
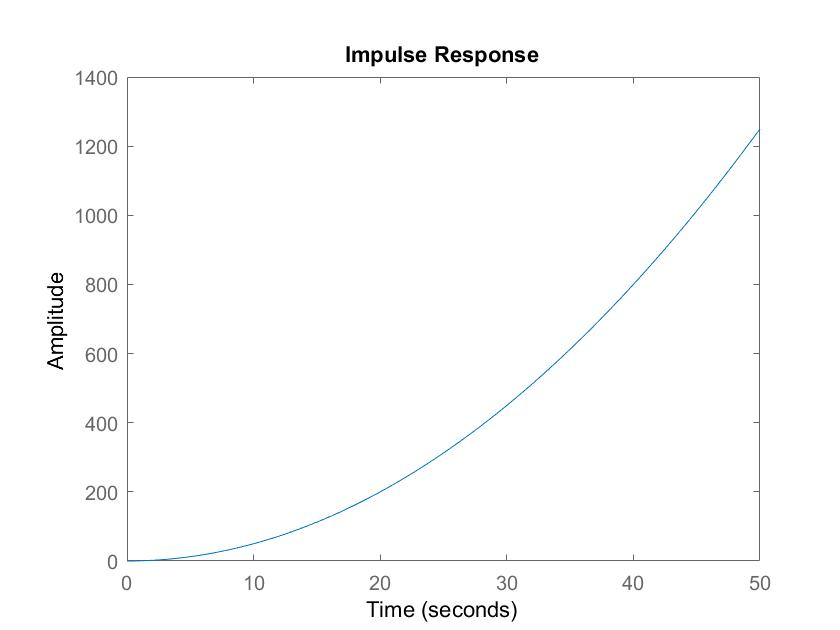
The system has pole , which is marginally stable.



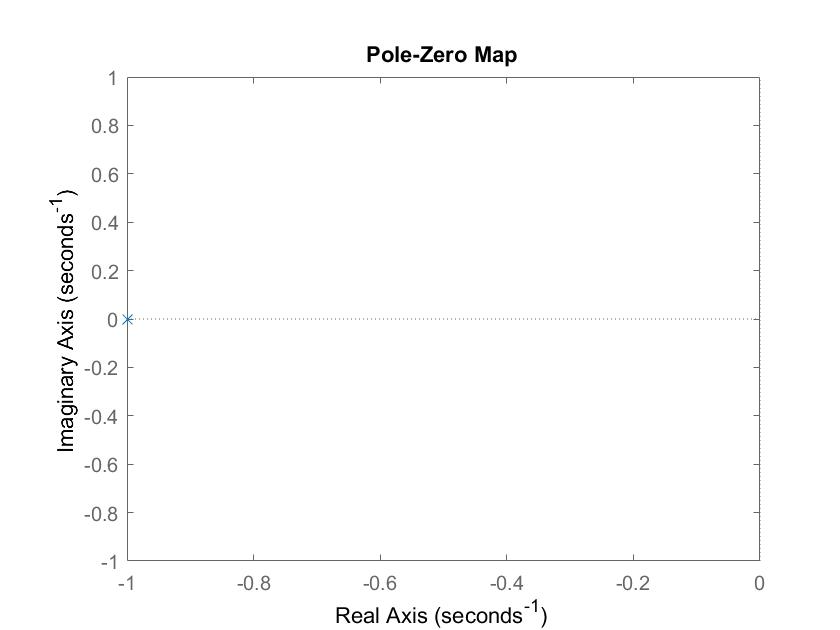
The step response and the impulse response of are

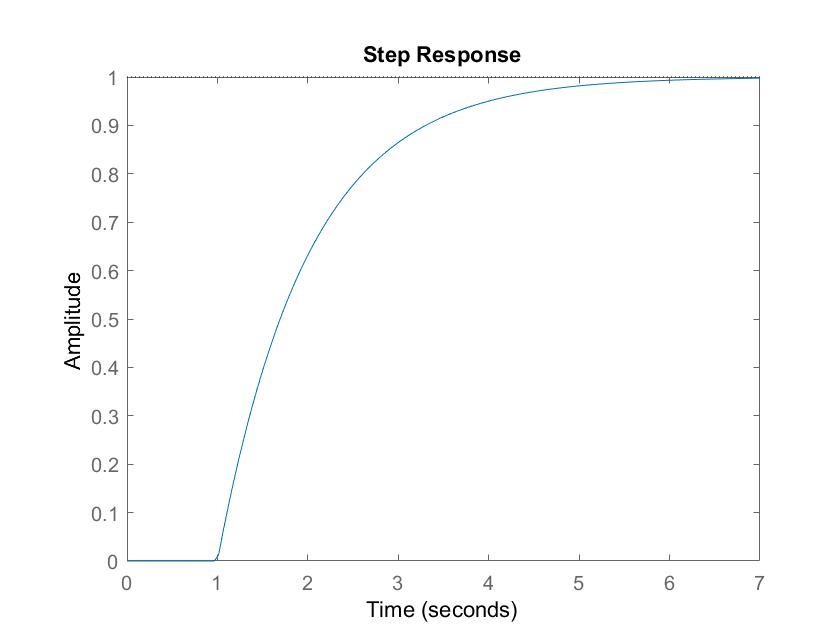
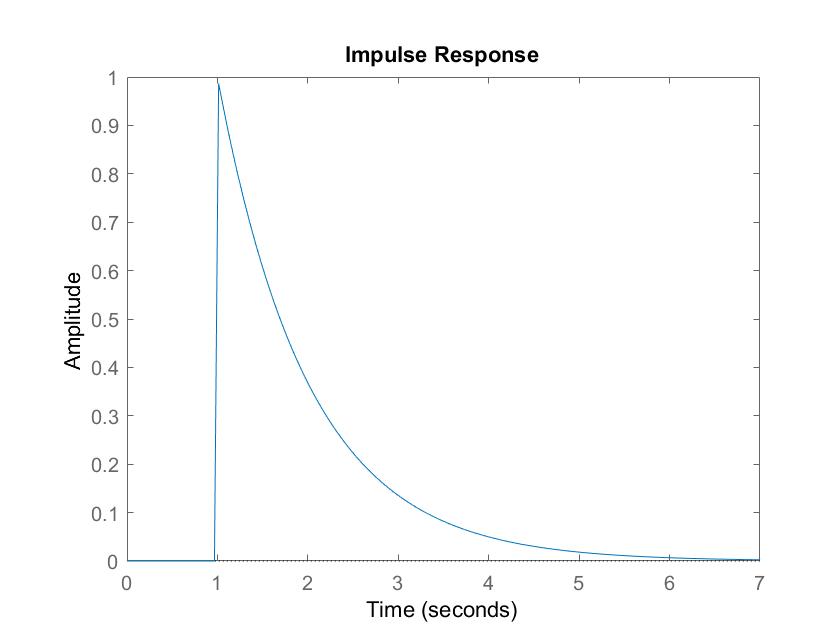
The system has pole , which is marginally stable.



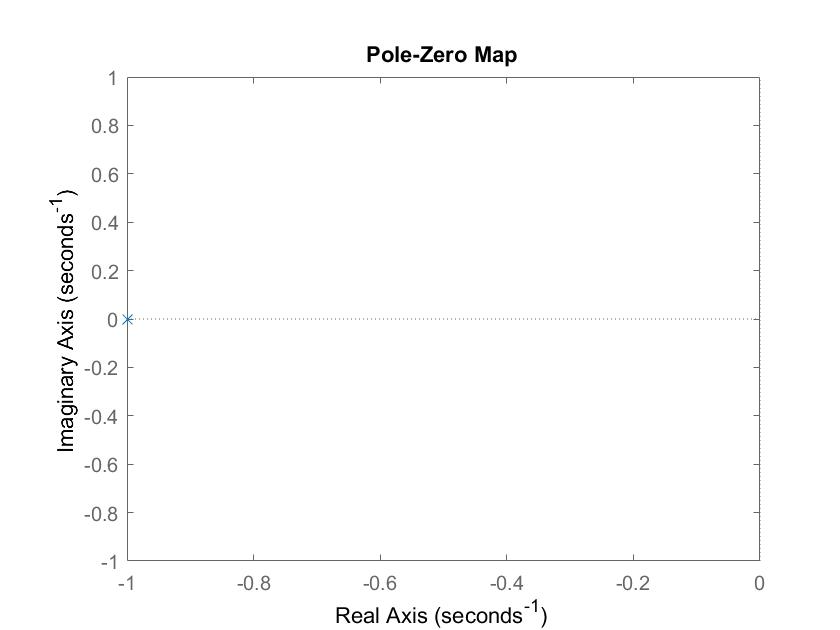
The step response and the impulse response of are

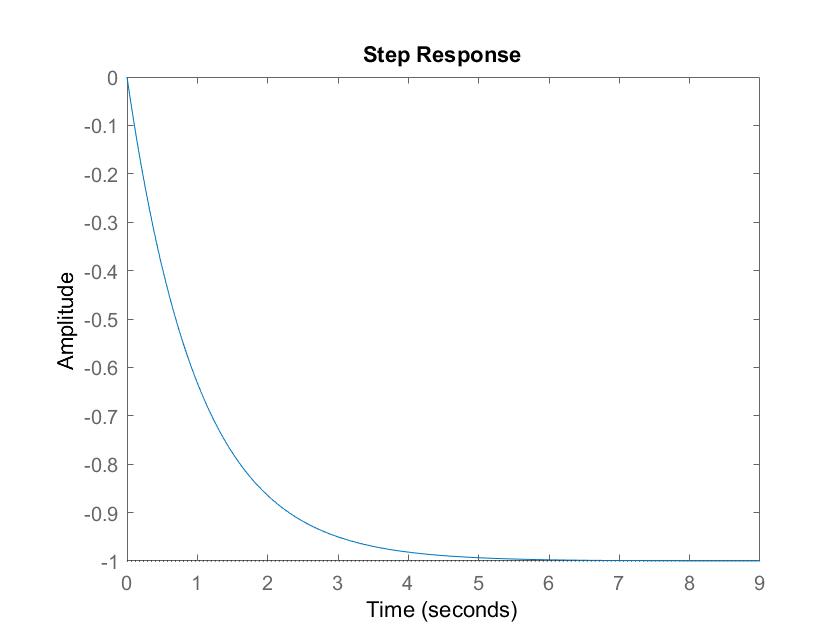
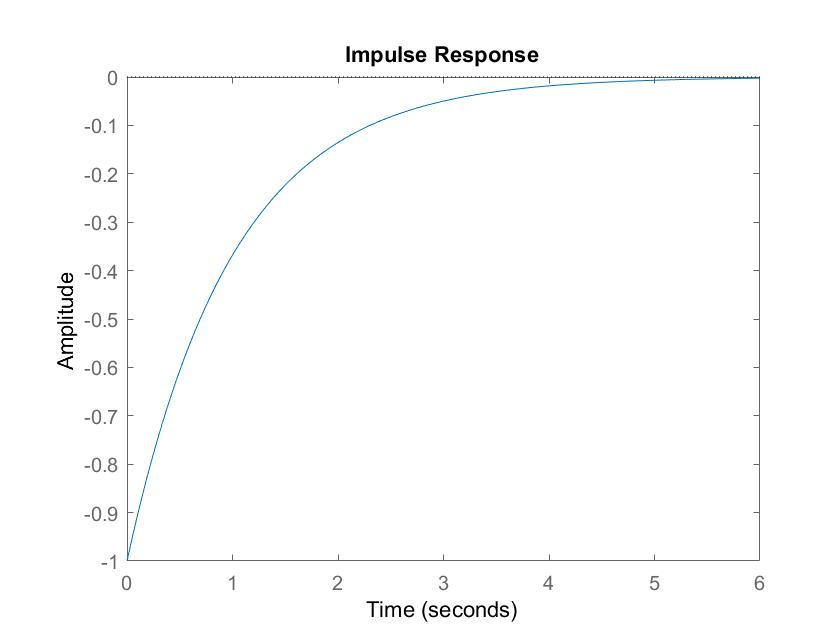
The system has pole , which is asymptotically stable.



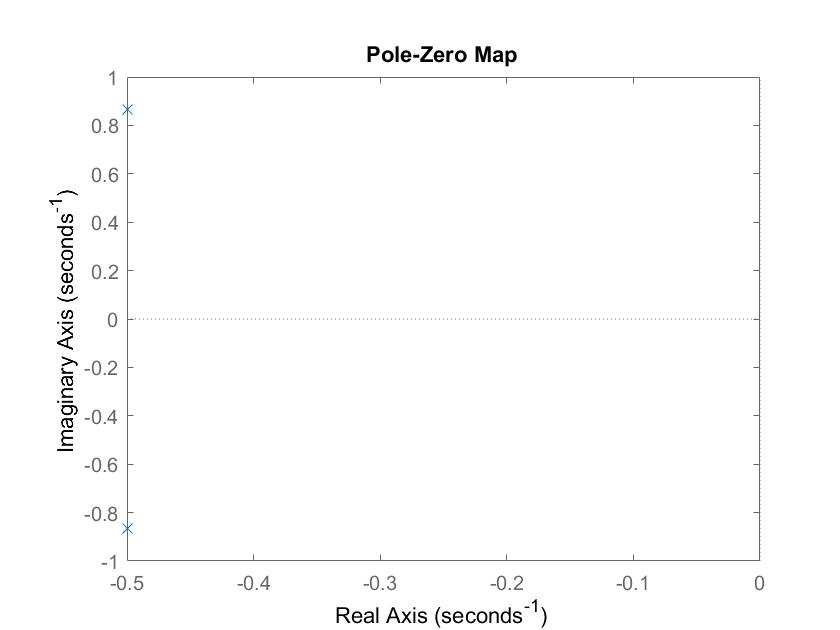
The step response and the impulse response of are

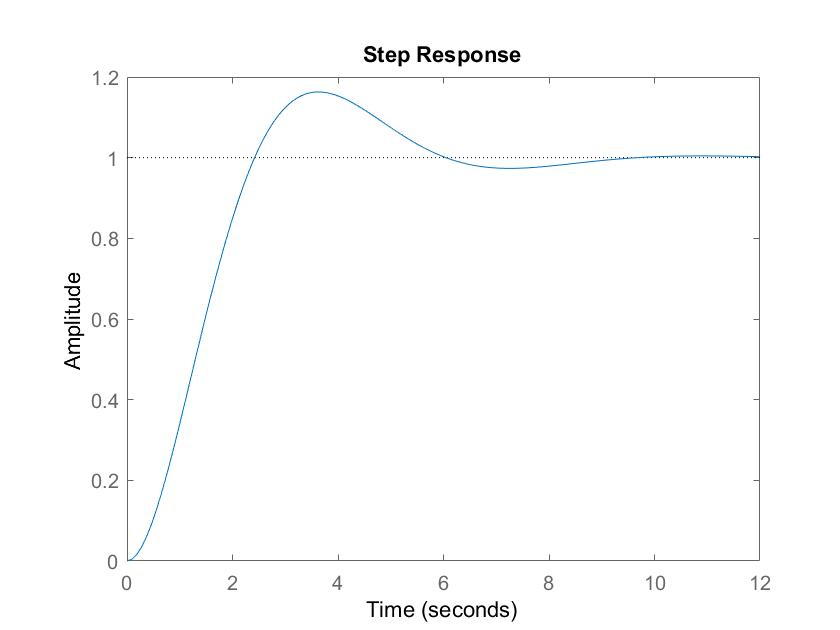
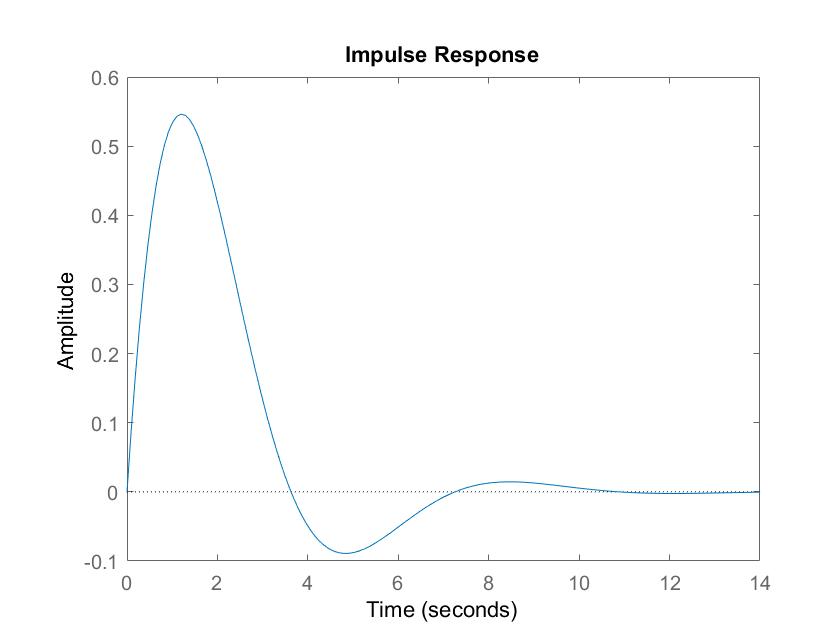
The system has pole , which is asymptotically stable.



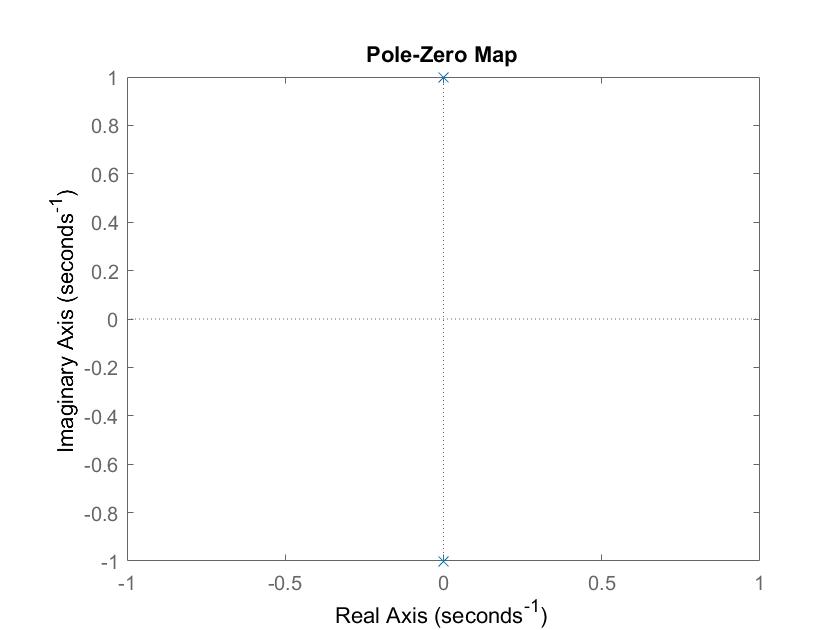
The step response and the impulse response of are

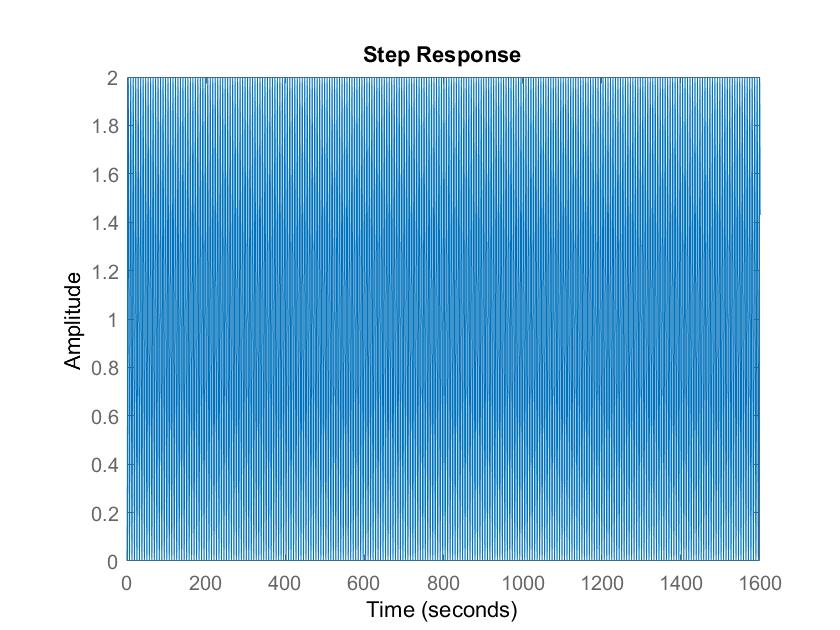
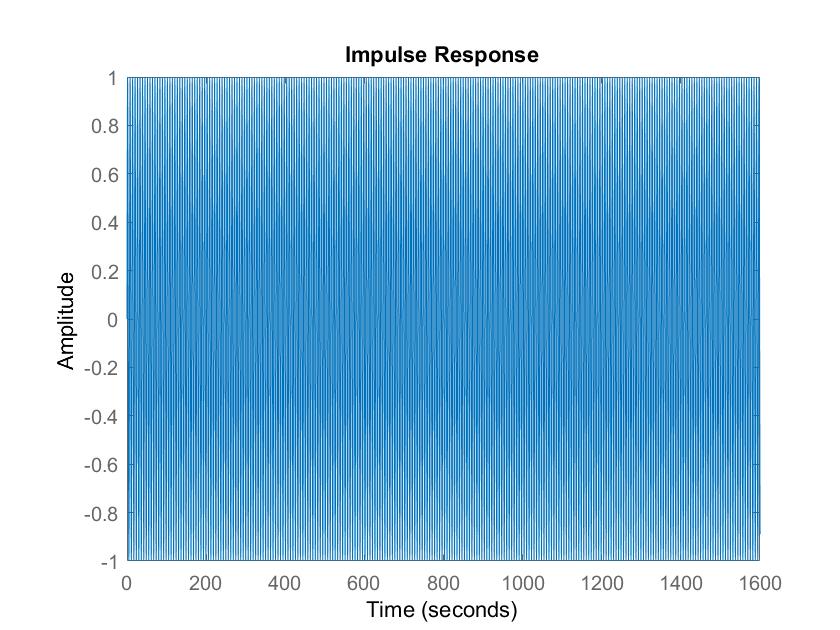
The system has poles , , which is asymptotically stable.



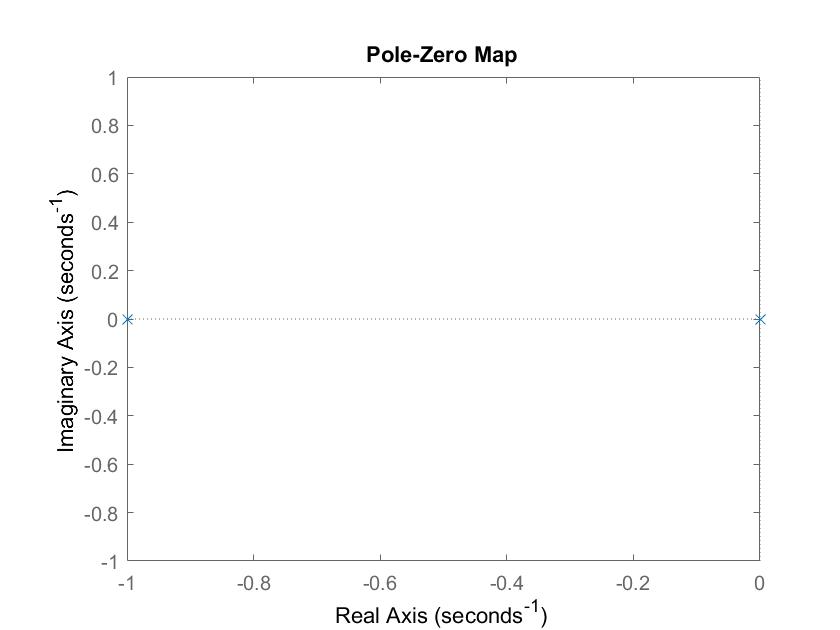
The step response and the impulse response of are

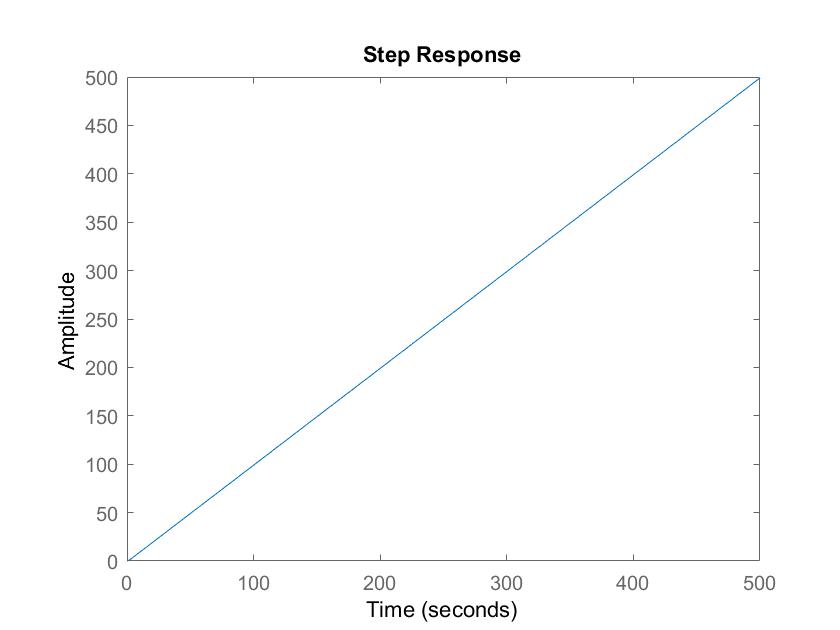
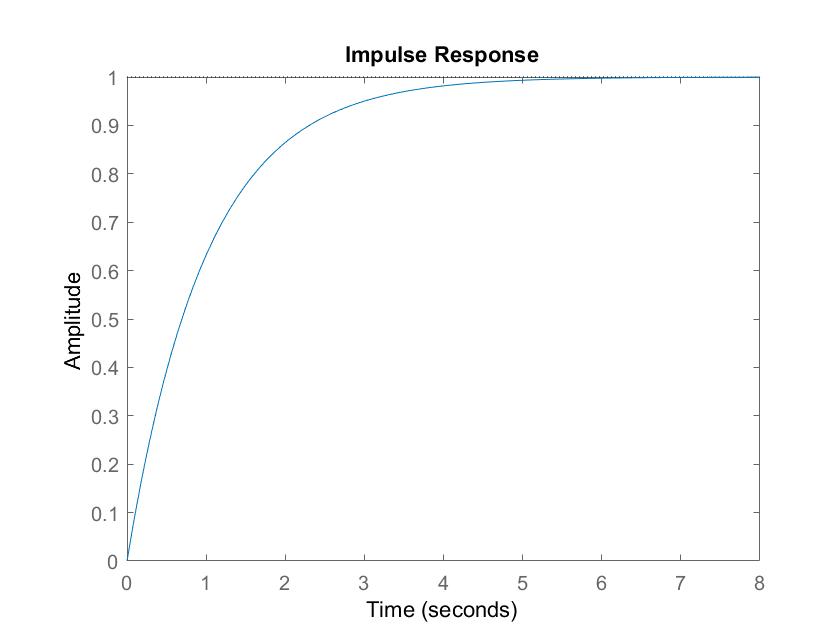
The system has poles , , which is marginally stable.



The step response and the impulse response of are

The system has poles , , which is marginally stable.



The step response and the impulse response of are

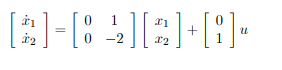
3.      **Using CST: Stability analysis and simulation of a state space model:** See Exercise 4.3 in the [exercise book](http://techteach.no/publications/books/advanced_dynamics_and_control/advanced_dynamics_control_exercises.pdf).

a.       Determine the stability property of the system from its eigenvalues. Simulate an initial state response of the state space model, i.e. assume some non-zero initial state, and simulate the responses in the states.

b.      Find the transfer function from u to y, and calculate its poles. Are these poles the same as the eigenvalues of the state space model?

**Solution - 3. Exercise 4.3**

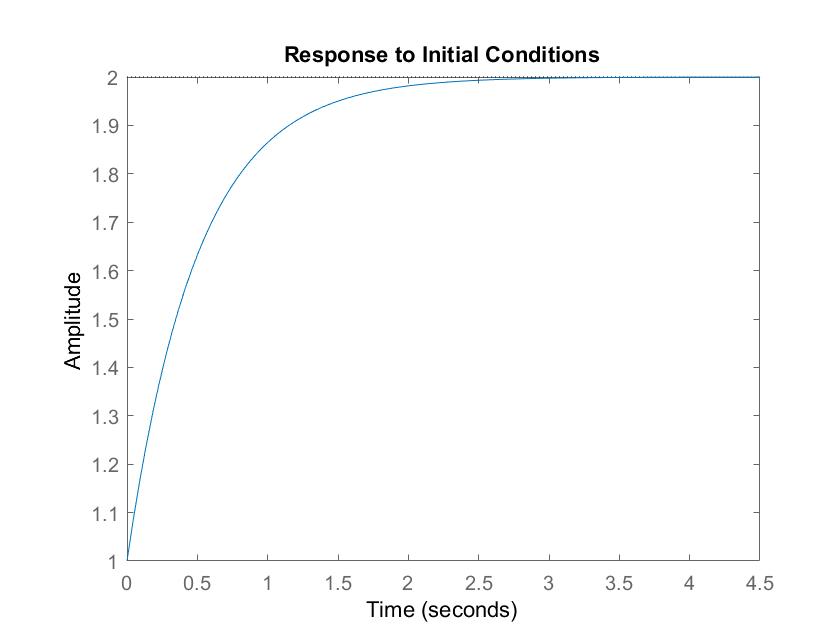
State space model is



1. System matrix and input matrix .

Eigenvalues are and , the system is marginally stable.

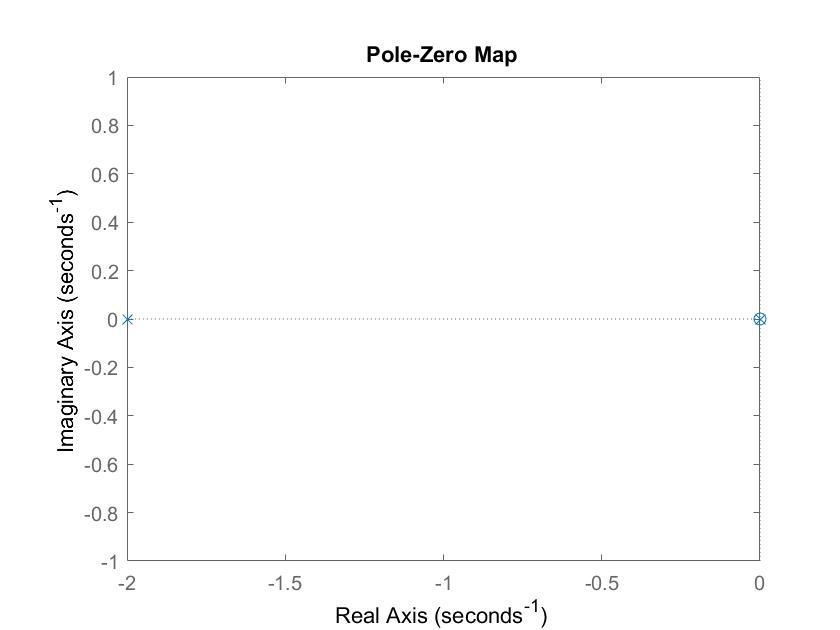
Initial condition is then initial condition response:



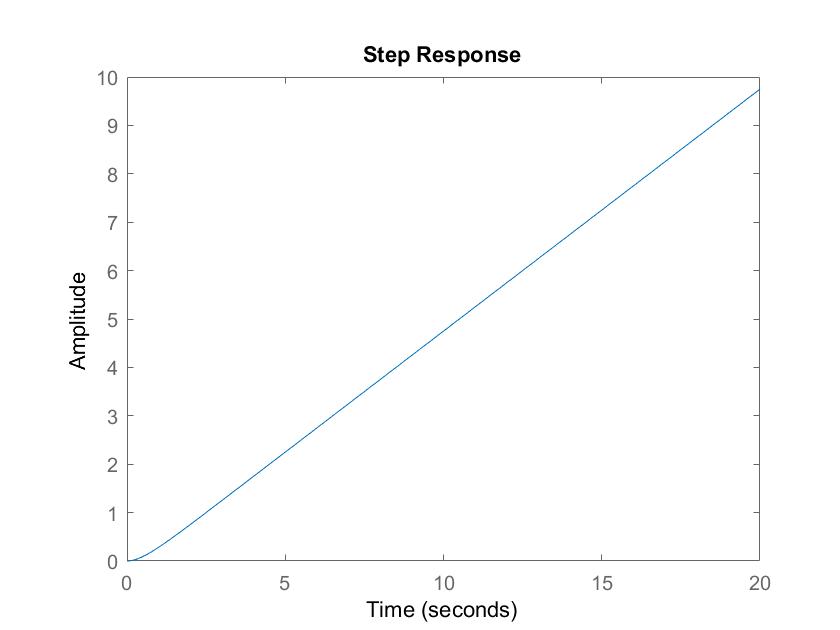
1. Output equation is , where and it is .

The transfer function is

Poles are and , those poles are the same as eigenvalues of the state space model.

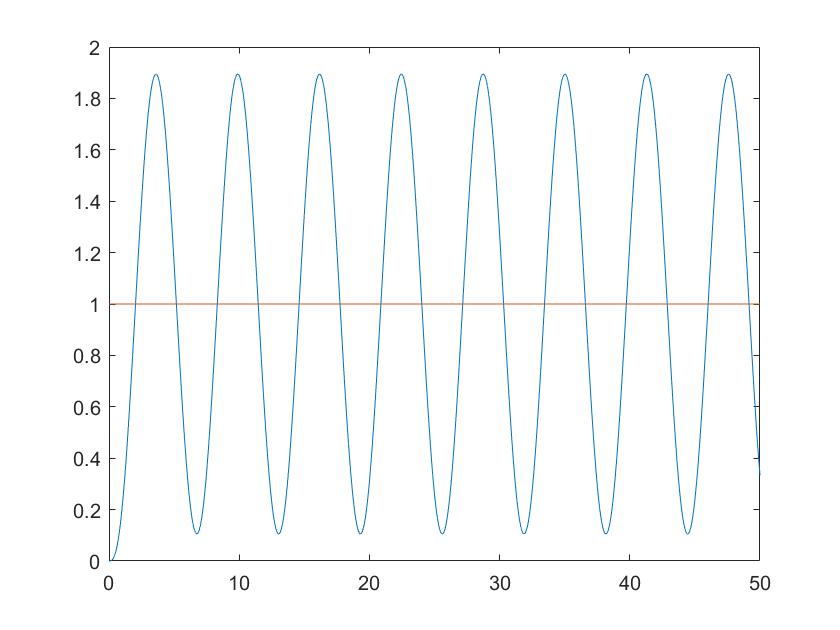


Step response of the system is

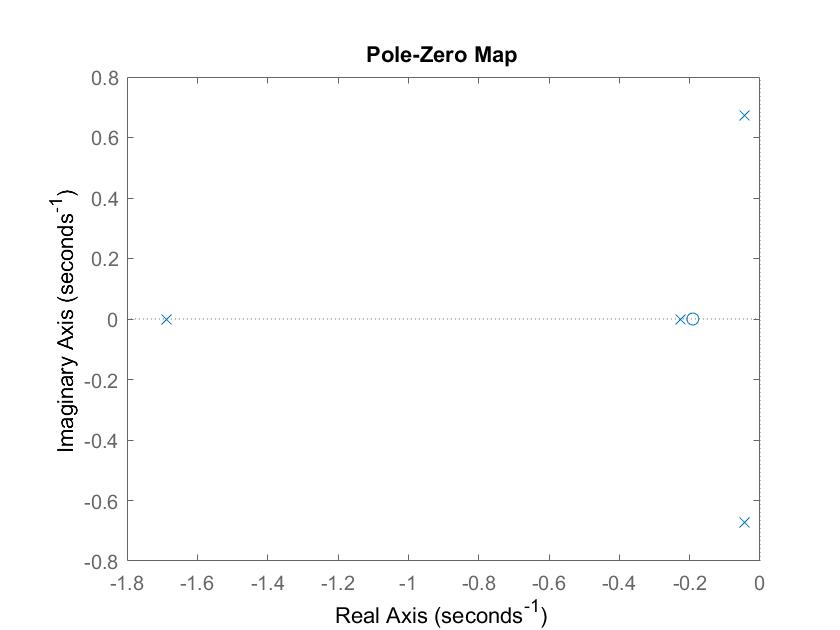


4.      **Using CST: Tuning and simulation of a feedback control system:** See Example 5.1 in [the text book](http://techteach.no/publications/books/advanced_dynamics_and_control/advanced_dynamics_control_textbook.pdf). Assume the controller is a PI controller. Tune the controller using the Ziegler-Nichols’ method. Plot the closed loop poles (i.e. the poles of the closed loop, i.e. the feedback control system), and simulate the setpoint-step response after the tuning. (Tip: Create a transfer function model of the control system using the feedback function of CST.)

Critical parameters:

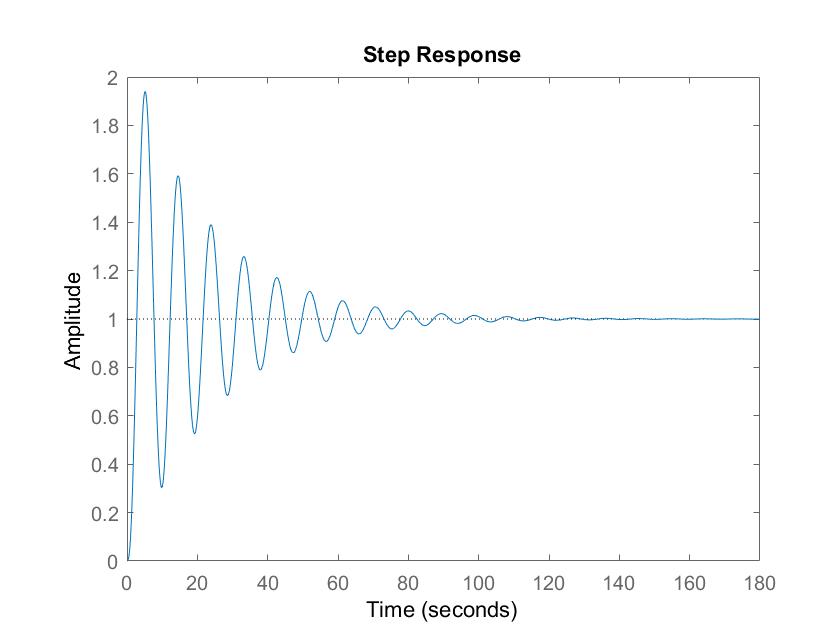


Ziegler Nicolson parameters tuning for PI controller:



The close loop system is asymptotically stable, where poles are

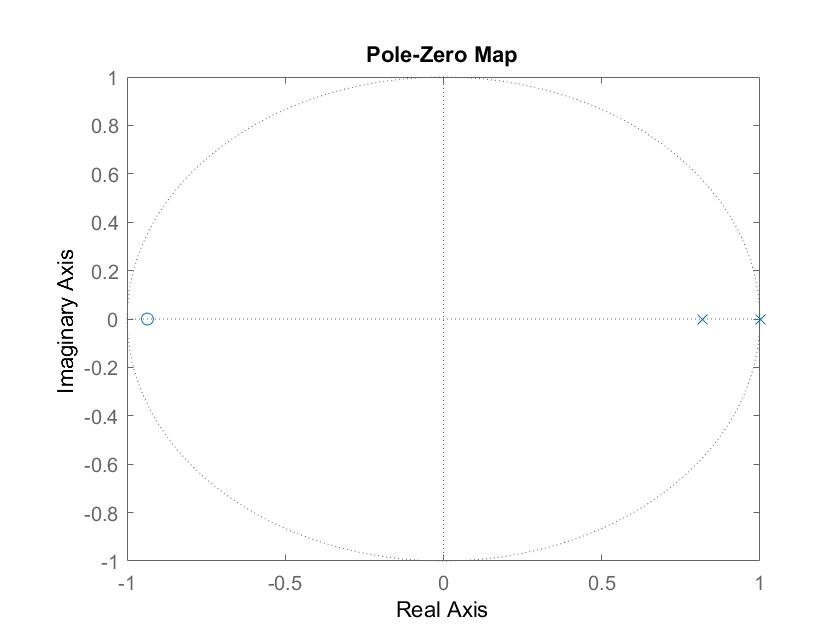
, ,



5.      **Using CST: Discretization of a continuous-time model:** See Exercise 4.3 in the [exercise book](http://techteach.no/publications/books/advanced_dynamics_and_control/advanced_dynamics_control_exercises.pdf). Discretize the given continuous-time state space model using the ZOH method with time-step 0.1 s. Calculate the eigenvalues of the resulting discrete-time model, and conclude about its stability property. Is the stability property the same as for the original continuous-time model, cf. Problem 3 above.

Discretized state space model

Eigenvalues of the discretized system are



Stability of the discretized system is marginally stable as continuous system, because one of the eigenvalues is equal 1.

6.      **Symbolic linearization with Matlab Symbolic Toolbox**: Linearize the nonlinear state-space model in Problem 1.3 in the exercise book. (Tip: Use the jacobian function.) Compare with (your) manual result.

**Manual results**:

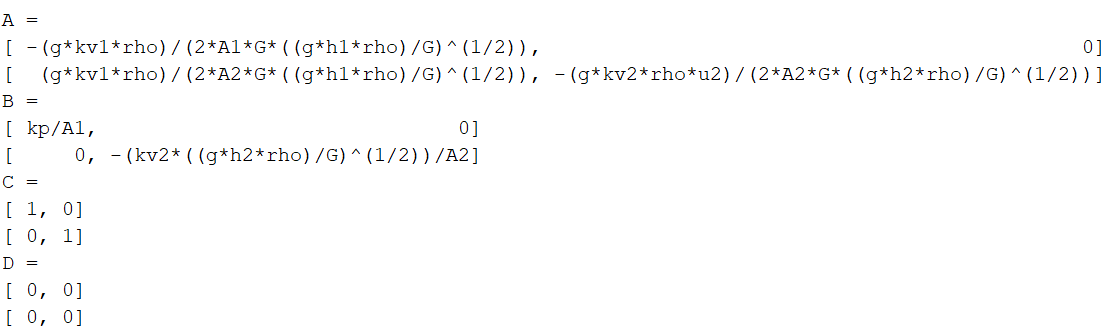
Nonlinear state space model:

System equation

Output equation:

Linearization:

**Matlab result**



The manual result is the same as the matlab result.

### Optimization

The Rosenbrock optimization problem (“ROP”) is a “standard” optimization problem, cf. <https://se.mathworks.com/help/optim/examples/banana-function-minimization.html>.

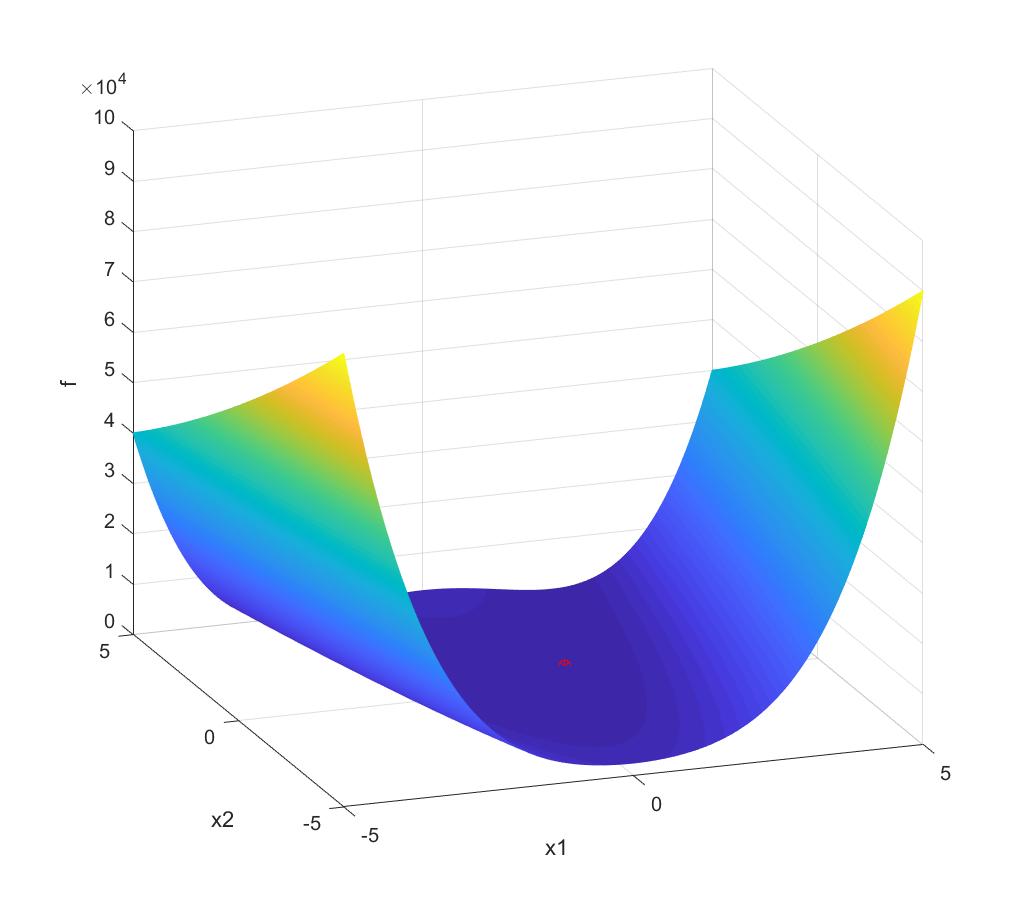
1. **Grid search:** Solve the ROP using the grid search method. The initial guess can be set to [-1.9,2] as in the above reference, and you may allow for a search of the optimal solution within -5 and 5 for both x1 and x2.

Function to minimize:

Optimal solution:

Optimal

Optimal



1. **fmincon:** The same as the above problem, but now by using fmincon.

Optimal solution:

Optimal

Optimal

1. **Newton search**: The same as the above problem, but now by implementing the Newton search method, from scratch. You may derive the gradient function and the Hessian function symbolically with the gradient() and the hessian() functions in the Symbolic Toolbox. ([Template for solution, similar to script presented in lecture 22 Feb.](http://techteach.no/fag/process_control_nmbu_2018/optim/optim_newton_240218.m))  
   Newton search: x\_kp1 = x\_k - inv[Hessian\_f(x\_k)]\*Gradient\_f(x\_k), where “kp1” means “k plus 1”.

Function to minimaze:

Newton search step:

Gradient:

Hessian:

Optimal solution:

Optimal

Optimal

4.      **Grid search, with constraints:** Same as the grid problem above, but now include the constraint x2 >= x1+1.

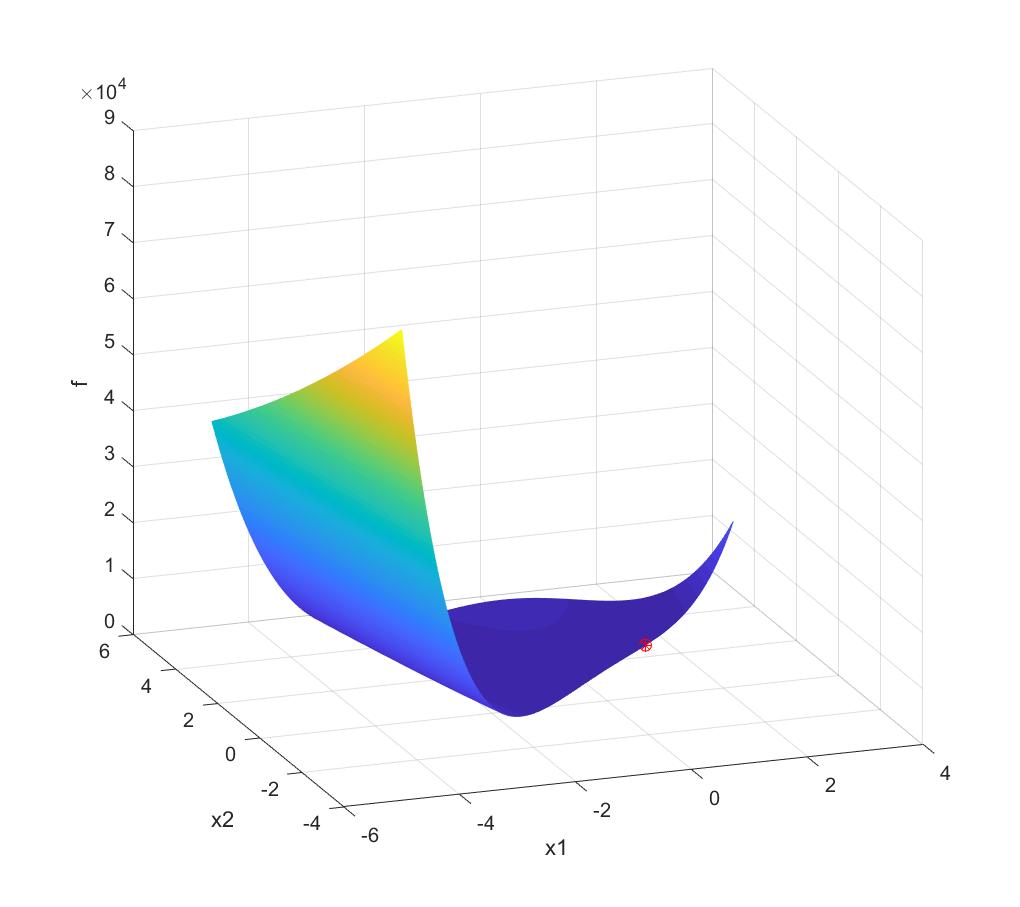
Function to minimize:

With inequality constraint:

Optimal solution:

Optimal

Optimal



5.      **fmincon, with constraints:** Same as the fmincon problem above, but now include the constraint x2 >= x1+1.

Function to minimize:

With inequality constraint:

Optimal solution:

Optimal

Optimal