**Problem 2:** Using CST: Stability analysis and simulation of transfer functions

Table 1. Stability criterion for various transfer functions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Transfer Function** | **Poles** | **Plot of Pole** | **Step test** | **Impulse test** | **Stability Criterion** |
| 1 |  | -1 | [figure](figuresP2/P2_pole_1.png) | [figure](figuresP2/P2_step_1.png) | [figure](figuresP2/P2_impulse_1.png) | Asymptotically stable |
| 2 |  | -1 | [figure](figuresP2/P2_pole_2.png) | [figure](figuresP2/P2_step_2.png) | [figure](figuresP2/P2_impulse_2.png) | Asymptotically stable |
| 3 |  | 1 | [figure](figuresP2/P2_pole_3.png) | [figure](figuresP2/P2_step_3.png) | [figure](figuresP2/P2_impulse_3.png) | Unstable |
| 4 |  | -1  1 | [figure](figuresP2/P2_pole_4.png) | [figure](figuresP2/P2_step_4.png) | [figure](figuresP2/P2_impulse_4.png) | Unstable |
| 5 |  | 0 | [figure](figuresP2/P2_pole_5.png) | [figure](figuresP2/P2_step_5.png) | [figure](figuresP2/P2_impulse_5.png) | Marginally stable |
| 6 |  | 0  0  0 | [figure](figuresP2/P2_pole_6.png) | [figure](figuresP2/P2_step_6.png) | [figure](figuresP2/P2_impulse_6.png) | Marginally stable |
| 7 |  | -1 | [figure](figuresP2/P2_pole_7.png) | [figure](figuresP2/P2_step_7.png) | [figure](figuresP2/P2_impulse_7.png) | Asymptotically stable |
| 8 |  | -1 | [figure](figuresP2/P2_pole_8.png) | [figure](figuresP2/P2_step_8.png) | [figure](figuresP2/P2_impulse_8.png) | Asymptotically stable |
| 9 |  | -0.5000 + 0.8660i  -0.5000 - 0.8660i | [figure](figuresP2/P2_pole_9.png) | [figure](figuresP2/P2_step_9.png) | [figure](figuresP2/P2_impulse_9.png) | Asymptotically stable |
| 10 |  | 0.0000 + 1.0000i  0.0000 - 1.0000i | [figure](figuresP2/P2_pole_10.png) | [figure](figuresP2/P2_step_10.png) | [figure](figuresP2/P2_impulse_10.png) | Marginally  stable |
| 11 |  | 0  -1 | [figure](figuresP2/P2_pole_11.png) | [figure](figuresP2/P2_step_11.png) | [figure](figuresP2/P2_impulse_11.png) | Marginally  stable |

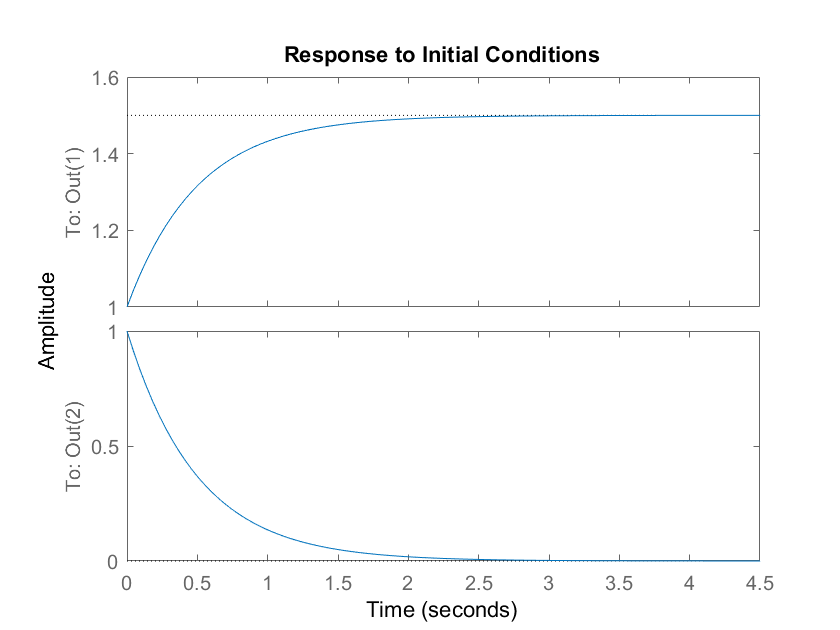
**Problem 3**: Stability analysis and simulation of a state space model

1. The state-space model is presented as the following. The values of A, B, C and D from the exercise 4.3 are used to build a SS - model in matlab. The problem is implemented in the file ‘Problem\_3.m’

The eigen values of the state space model can be obtained by using the ‘*eig*’ function in the CST toolbox. The eigen values obtained are as follows.

*e1*= 0 *e2* = -2 therefore the system is marginally stable.

The initial values of the states are assumed to be [1;1]. Using these values as the initial response curve of the ss model is as represented in Figure 1



**Figure 1**. Response cure for the state space model for non-zero initial condition

1. The state-space model can be converted to a transfer function model by using the ‘*tf ’* command in the CST. The poles of the transfer function are obtained as

For X1 : *p1*= 0 *p2* = -2 for X2: P=-2

Therefore, the poles of the transfer functions are the same as the eigen values of the transfer function.

**Problem 4**: Tuning and simulation of a feedback control system

Using the values obtained from the problem 5.1 of the textbook the system transfer functions are as follows

The overall closed loop transfer function for the system can be described as

Since the controller used in this case is a PI controller the transfer function Hc is

The expression for the close loop transfer function is obtained by substituting the transfer function representation of a PI controller in HT

The code for Ziegler Nichols tuning is presented in the matlab script ‘*Problem4.m*’ (first part). The steady oscillations are obtained at Ku = 2 and give a period of oscillation P = 6.2s

Therefor the value of Kp and Ti from the Ziegler Nichols tuning method is obtained as Kp= 2\*0.45 = 0.9 and Ti= 6.2/1.2 = 5.16s

Therefore, the closed loop transfer function can be written as.

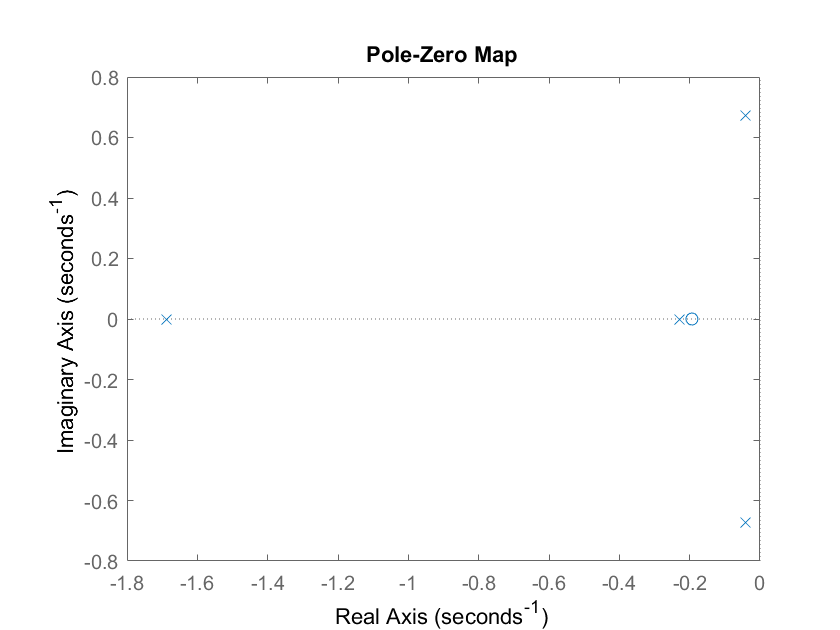
The Poles of the closed loop system is obtained as

-1.6872 + 0.0000i

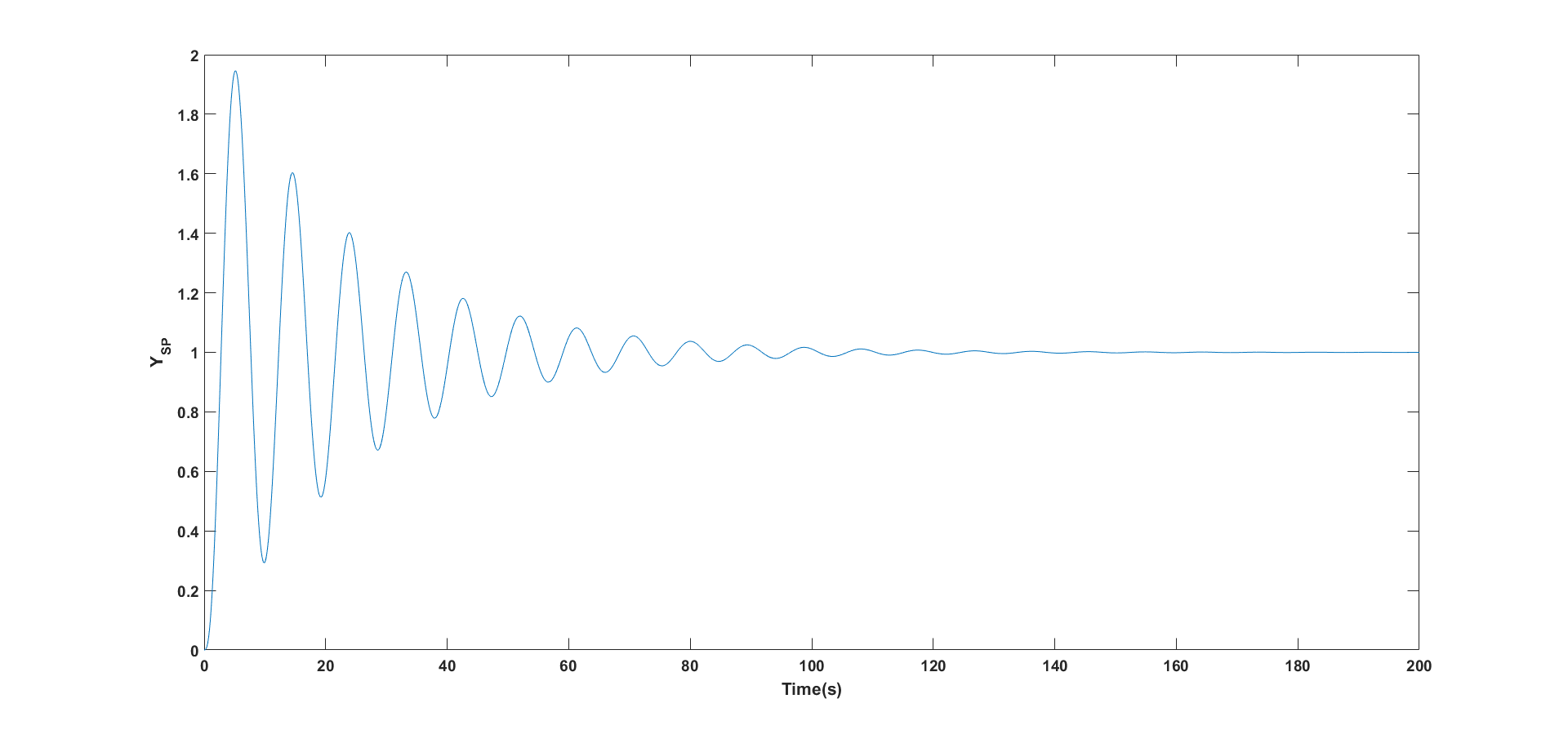
-0.0424 + 0.6716i

-0.0424 - 0.6716i

-0.2280 + 0.0000i



**Figure 2 :** Pole-zero map of the transfer function

****

**Figure 3 T**he setpoint-step response of the system

From the poles and the response curve we can say that the system Is asymptotically stable

**Problem 5**: Discretization of a continuous-time model

The discretized state space model representation is as the follows

The continuous ss-model presented in exercise 4.3 is converted in a discrete form using the ZOH method. The matlab code for this conversion is presented in ‘*Problem\_5.m*’. The values of Ad, Bd, Cd and Dd obtained from the discretization of the continuous equation is presented below.

The eigen values of the discrete model are e1=1 and e2=0.8187. Therefore, the system is considered as marginally stable.

Although the eigen values of the discrete system is different from the continuous system, the stability criteria appears to be the same.

**Problem 6**: Symbolic linearization with Matlab Symbolic Toolbox

The system is explained as the following equations

(1)

(2)

The A and B matrices of the linearized model can be obtained by finding the Jacobian of the non linear function as follows.

(3)

The values of A and B derived manually by differentiating the equation 3 is

A = B =

The code for the linearization of the model is in file ‘*Problem\_6.m*’. The values obtained from the matlab code is the same as the value which is derived manually.

A = [-(K\_v1\*g\*rho)/(2\*A\_1\*G\*((g\*h\_1\*rho)/G)^(1/2)) 0

(K\_v1\*g\*rho)/(2\*A\_2\*G\*((g\*h\_1\*rho)/G)^(1/2)) (K\_v2\*g\*rho\*u\_2)/(2\*A\_2\*G\*((g\*h\_2\*rho)/G)^(1/2))]

B= [K\_p/A\_1 0]

0 -(K\_v2\*((g\*h\_2\*rho)/G)^(1/2))/A\_2]

**OPTIMISATION**

**Rosenbrock optimization problem (“ROP”)**

**Problem 1,2,3:**

The optimization problem explained in the question has the following standard form.

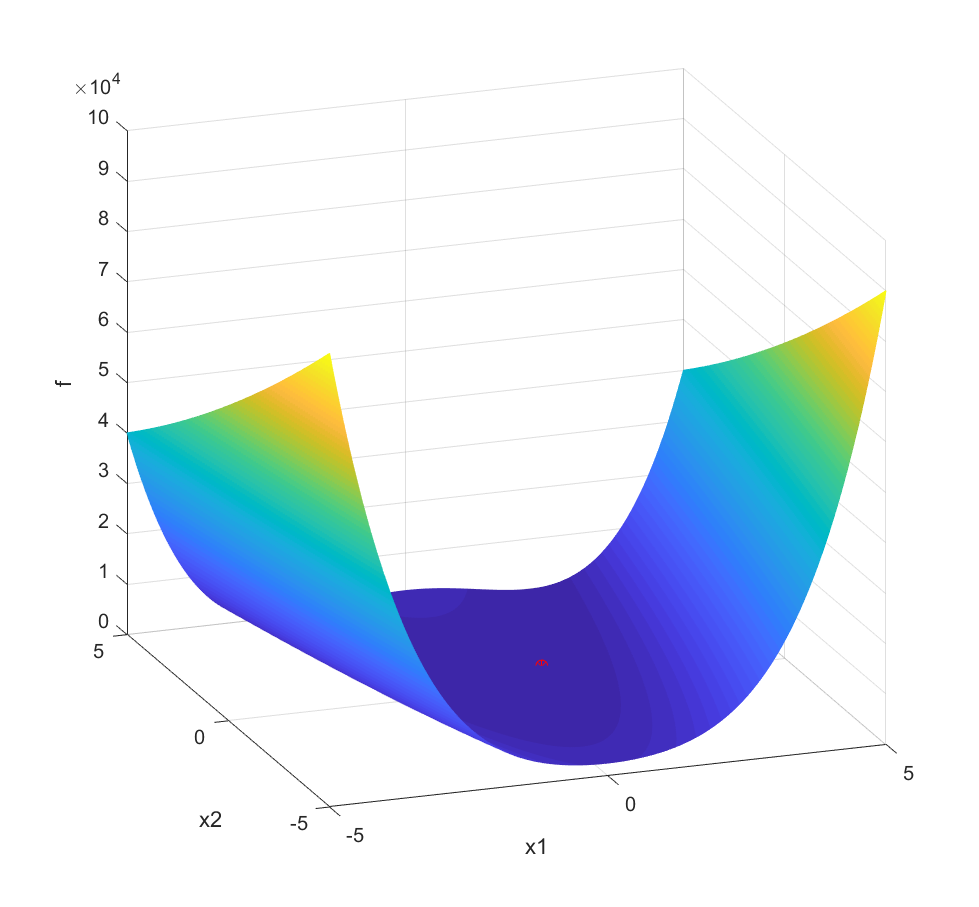
Subject to

The optimization problem is solved using three different optimization algorithms. The grid search method, interior point method and newton’s method. All the three algorithms are coded in matlab and can be found in the files named ‘*Optimisation1\_gridsearch.m*’, ‘*Optimisation2\_fmincon.m*’ and ‘*Optimisation3\_newton.m*’. The solution for the optimization problem is presented in Table 2.

Table 2. Comparison of between different optimization methods (without nonlinearly constraint)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Grid search method** | **Interior point method (fmincon)** | **Newtons method** |
| Optimal values | 0.0016 | 2.0474e-11 | 3.8654e-29 |
| Objective function | X1 = 0.9960  X2 = 0.9960 | X1 = 1  X2 = 1 | X1 = 1  X2 = 1 |
| No. Iterations |  | 21 | 4 |
|  |  |  |  |
| Simulation time  (seconds) | 1.2686 | 0.754858 | 0.007554 |

The symbolic representation of gradient and the hessian is obtained by using the matlab function and the code is presented in the matlab file ‘*grad\_hess.m*’. From the Table 2 it can be observed that the newton’s method converges to the solution very quick and it also has the least amount of time and the number of steps required to reach the solution.



**Problem 4,5:**

The problem statement is reformulated by adding an additional inequality constraint.

Subject to

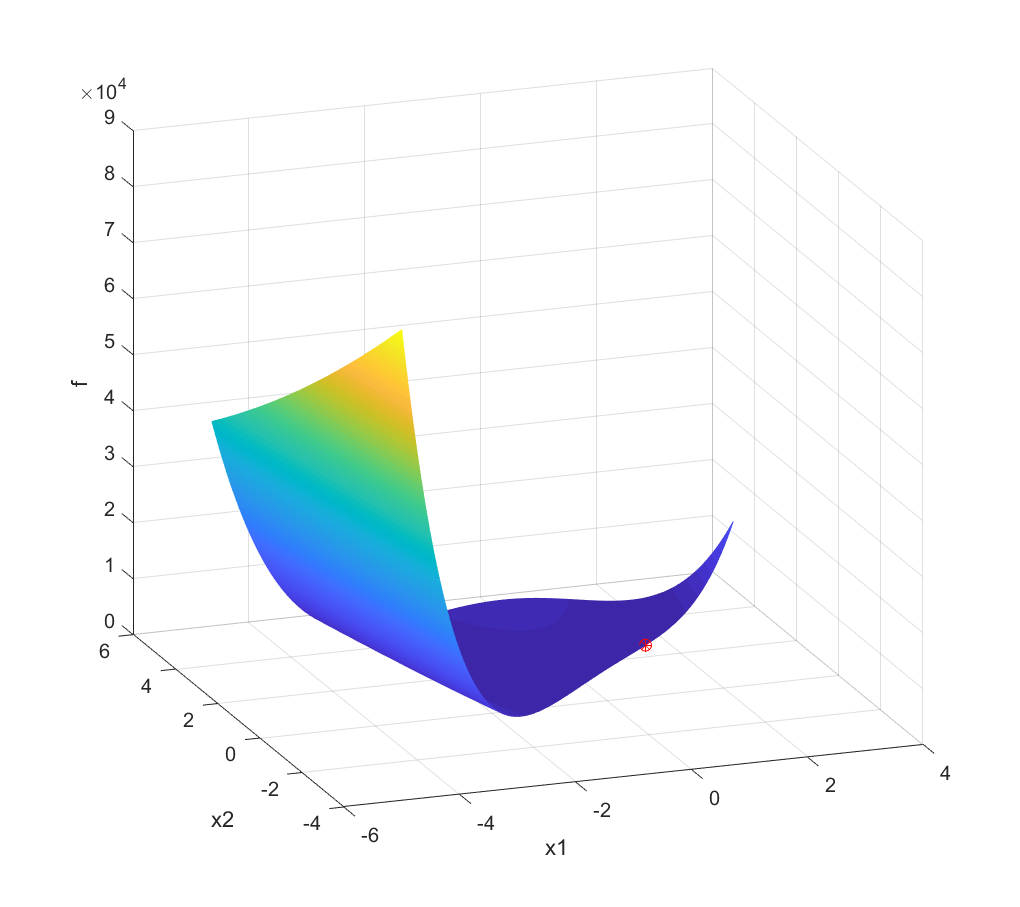
The optimizations problem is solved using grid search method and the fmincon function using interior point algorithm. The code for solving the problem is presented in the matlab files ‘*Optimisation1\_gridsearch\_constrain.m*’, ‘*Optimisation2\_fmincon\_constrain.m*’ respectively. Table presents a comparison between both the solver algorithms.

When the constraint of is included, the value of xopt = [1, 1] doesn’t fit the constraints. Therefore the optimizer searches for a new optimal value in the solution space. The new value of the X for which the function reaches a minimum is xopt = [1.61, 2.61]. which lies in the solution space for the constrained optimization problem.

Table 3. Comparison of between different optimization methods (with nonlinearly constraint)

|  |  |  |
| --- | --- | --- |
|  | **Grid search method** | **Interior point method (fmincon)** |
| Optimal values | 0.3823 | 0.3812 |
| Objective function | X1 = 1.6183  X2 = 2.6188 | X1 = 1.6168  X2 = 2.6168 |
| No. Iterations |  | 25 |
|  |  |  |
| Simulation time  (seconds) | 8.9063 | 0.808245 |

From the values presented in Table 3 it can be observed that the grid search looks for the global minimum in the solution space, but the simulation time is much higher compared to the interior point algorithm (using fmincon). The interior point algorithm is faster and would give the same solution if the optimization problem is convex.

****