A Geometric Approach to Linear Control Theory Linear Time Invariant Systems

H. Priyadarshan

Department of Avionics
Indian Institute of Space Science and Technology (IIST)

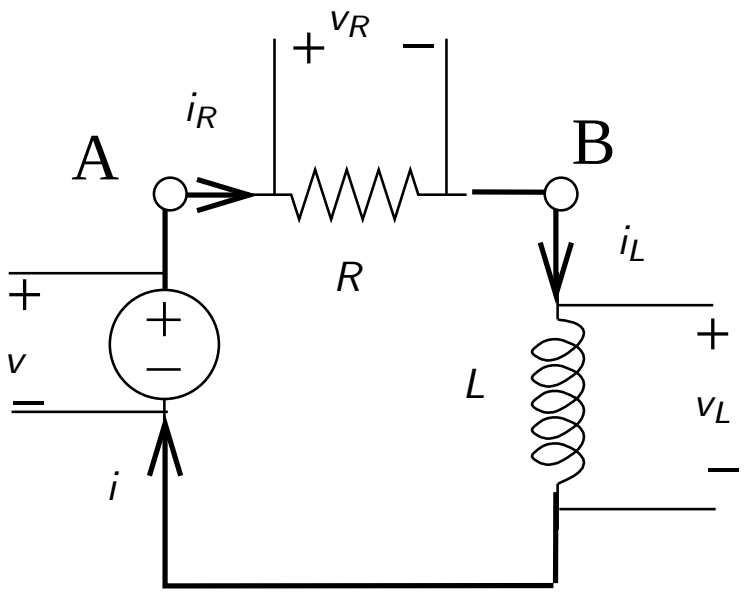
Applications of Linear Algebra in Control System – An Introduction
Kerala School of Mathematics

20-23 Oct., 2016

Outline

- State Space Modelling and Solutions
- Stable and Unstable Systems
- Stabilization using State Feedback and Controllability
- State Transfer Characterisation
- Parameterisation problem

Modelling of Physical Systems: RL Network



- Device characteristics

$$v_L(t) = L\frac{d}{dt}i_L(t)$$
$$v_R(t) = i_R(t)R$$

- Network Equations

$$v(t) = v_R(t) + v_L(t)$$
$$i(t) = i_R(t) = i_L(t)$$

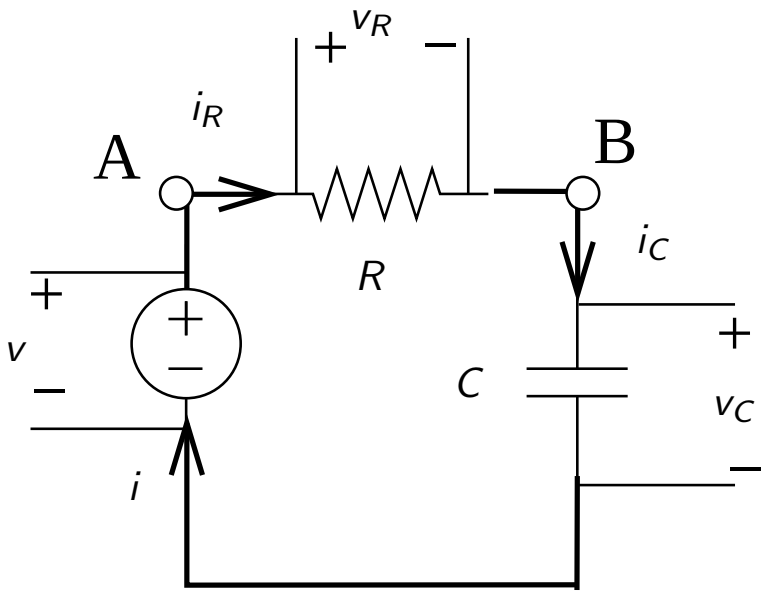
- State Space model

$$\frac{d}{dt}i_L(t) = -\frac{R}{L}i_L(t) + v(t)$$

- Output equations

$$v_R(t) = Ri_L(t)$$

Modelling of Physical Systems: RC Network



- Device characteristics

$$i_C(t) = C\frac{d}{dt}v_C(t)$$
$$v_R(t) = i_R(t)R$$

- Network Equations

$$v(t) = v_R(t) + v_C(t)$$
$$i(t) = i_R(t) = i_C(t)$$

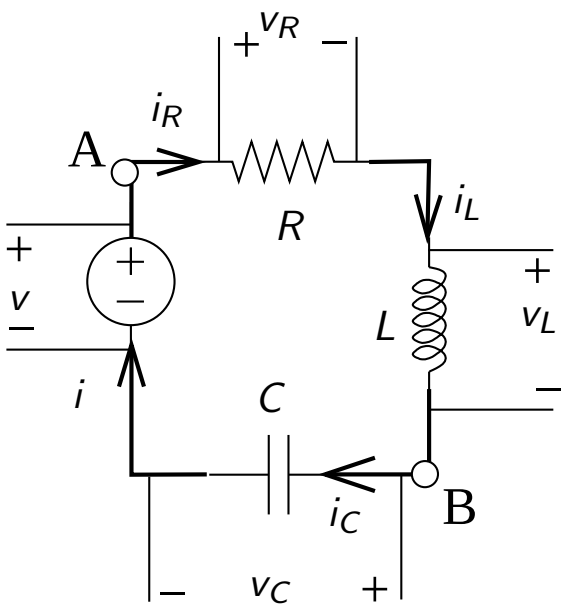
- State Space model

$$\frac{d}{dt}v_C(t) = -\frac{v_C(t)}{RC} + \frac{v(t)}{RC}$$

- Output equations

$$v_R(t) = -[1]v_C(t) + v(t)$$

Modelling of Physical Systems: RLC Network



- Device characteristics

$$i_C(t) = C\frac{d}{dt}v_C(t) \qquad v_L(t) = L\frac{d}{dt}i_L(t)$$
$$v_R(t) = i_R(t)R$$

- Network Equations

$$v(t) = v_R(t) + v_C(t) + v_L(t)$$
$$i(t) = i_R(t) = i_C(t) = i_R(t)$$

- State Space model

$$\begin{pmatrix} \frac{d}{dt}v_C(t) \\ \frac{d}{dt}i_L(t) \end{pmatrix} = \begin{bmatrix} \frac{1}{C} & 0 \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{pmatrix} v_C(t) \\ i_L(t) \end{pmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v(t)$$

- Output across AB

$$v_o(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{pmatrix} v_C(t) \\ i_L(t) \end{pmatrix} + [1]v(t)$$

General State Space Model (Linear Systems)

- State Equations

$$\begin{pmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \\ \vdots \\ \frac{dx_n(t)}{dt} \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1m} \\ b_{2m} & \dots & b_{2m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \dots & b_{nm} \end{bmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_n(t) \end{pmatrix}$$

- Output equations

$$\begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{pmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{p1} & c_{p2} & \dots & c_{pn} \end{bmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} + \begin{bmatrix} d_{11} & \dots & d_{1m} \\ d_{2m} & \dots & d_{2m} \\ \vdots & \ddots & \vdots \\ d_{p1} & \dots & d_{pm} \end{bmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_n(t) \end{pmatrix}$$

General Form: Matrix Notation

- State Equations and Output equations

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

  where vectors $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{u}(t) \in \mathbb{R}^m$, $\mathbf{y}(t) \in \mathbb{R}^p$ and matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$

- The variables $\mathbf{x}(t)$ are states and $\mathbf{u}(t)$ is called the inputs
- Why is it called the state?
- Answer: By knowing the initial values of the states and the inputs the solution of the system can be computed.
- What do we mean by solution of the system?
- Answer: Any trajectory $\mathbf{x}(t)$ which satisfies the above differential equation is called the solution.

Solution of (dynamic) state equations

- Scalar Equation

$$\dot{x}(t) = ax(t) + bu(t)$$
$$y(t) = cx(t) + du(t)$$

- Case 1: Without inputs

$$\dot{x}(t) = ax(t)$$
$$y(t) = cx(t)$$

Verify (how?) that the solution is

$$x(t) = e^{at}x(0)$$
$$y(t) = ce^{at}x(0)$$

Solution of (dynamic) state equations

- Scalar equation with input

$$\dot{x}(t) = ax(t) + bu(t)$$
$$y(t) = cx(t) + du(t)$$

- Verify that the solution is

$$x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau$$

$$y(t) = ce^{at}x(0) + c\int_0^t e^{a(t-\tau)}bu(\tau)d\tau + du(t)$$

- Note that using Leibnitz rule of differentiating under integration

$$\dot{x}(t) = ae^{at}x(0) + \int_0^t ae^{a(t-\tau)}bu(\tau)d\tau + bu(t)$$

Solution of vector state equations

- Basic equations

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

- Verify the solutions

$$\mathbf{x}(t) = e^{At}\mathbf{x}(0) + \int_0^t e^{A(t-\tau)}B\mathbf{u}(\tau)d\tau$$
$$\mathbf{y}(t) = Ce^{At}\mathbf{x}(0) + C\int_0^t e^{A(t-\tau)}b\mathbf{u}(\tau)d\tau + D\mathbf{u}(t)$$

Simulation Examples

- System without inputs

$$\dot{x}(t) = 2x(t)$$
$$y(t) = 3x(t)$$

- Solution

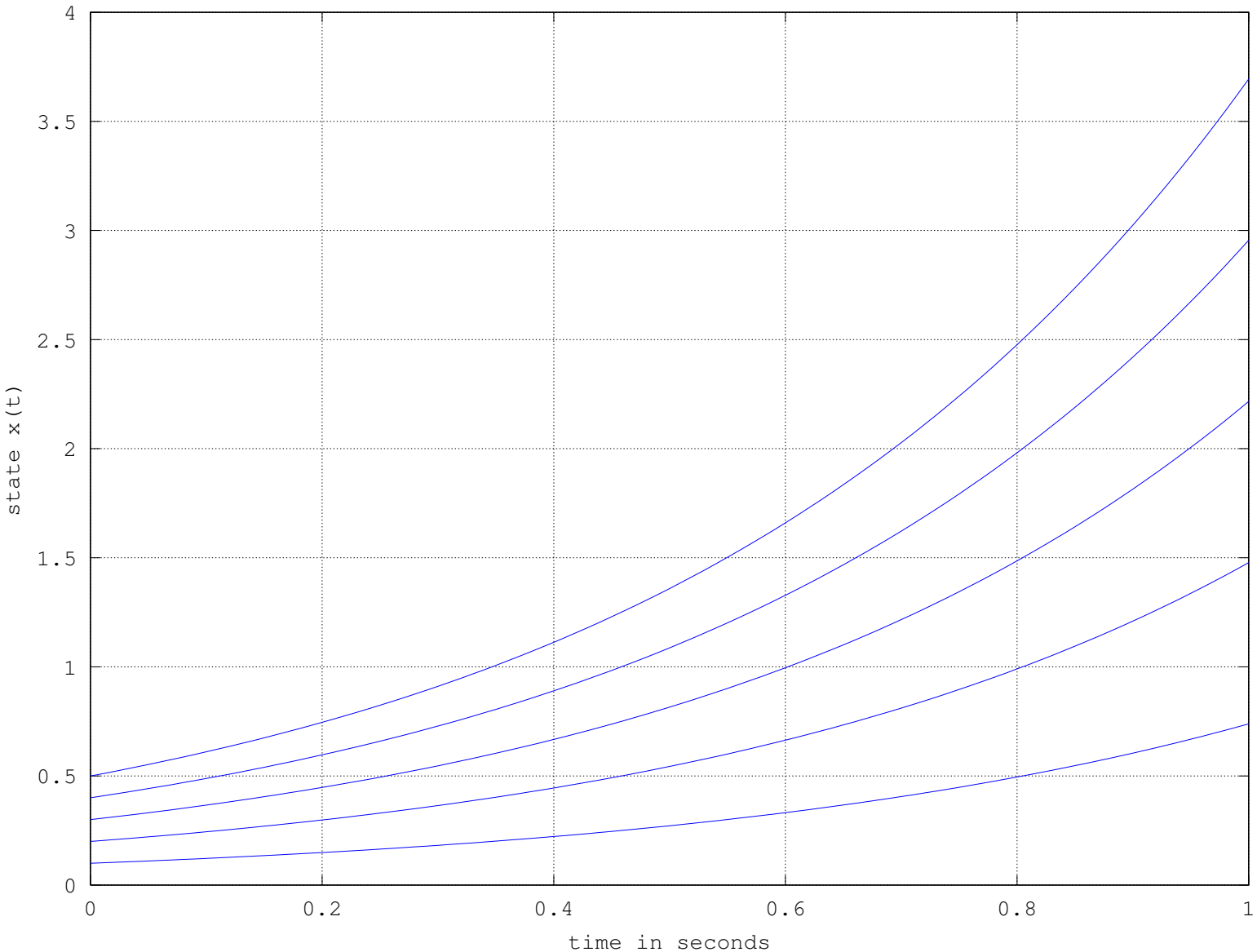
$$x(t) = e^{2t}x(0)$$
$$y(t) = 3e^{2t}x(0)$$

- Simulations for $A > 0$, $A < 0$, $A = 0$, different Cs and ts

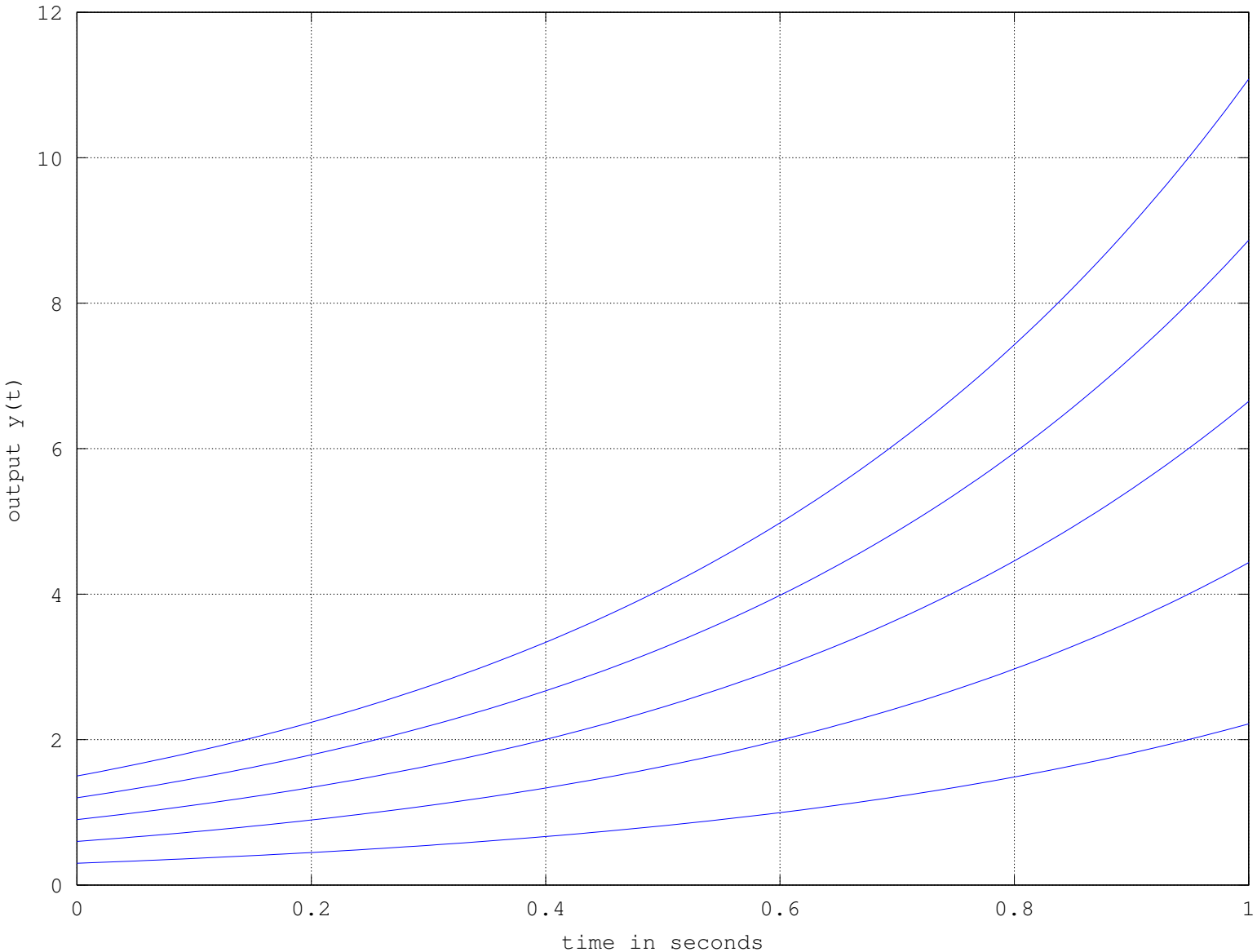
Matlab Program

```
A = [2];B = [0];C = [3];D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1; u = 0;

for i=1 : 5
[Y,T,X] =  lsim(sys1,u,t,[0.1*i ]);
x(i,:)=X;
y(i,:)=Y;
endfor

hold on; grid on;
for i=1:5  plot(t,x(i,:)); endfor;
xlabel("time in seconds");  ylabel("state x(t)");
figure;  hold on; grid on;
for i=1:5  plot(t,y(i,:)); endfor;
xlabel("time in seconds"); ylabel("output y(t)");
```

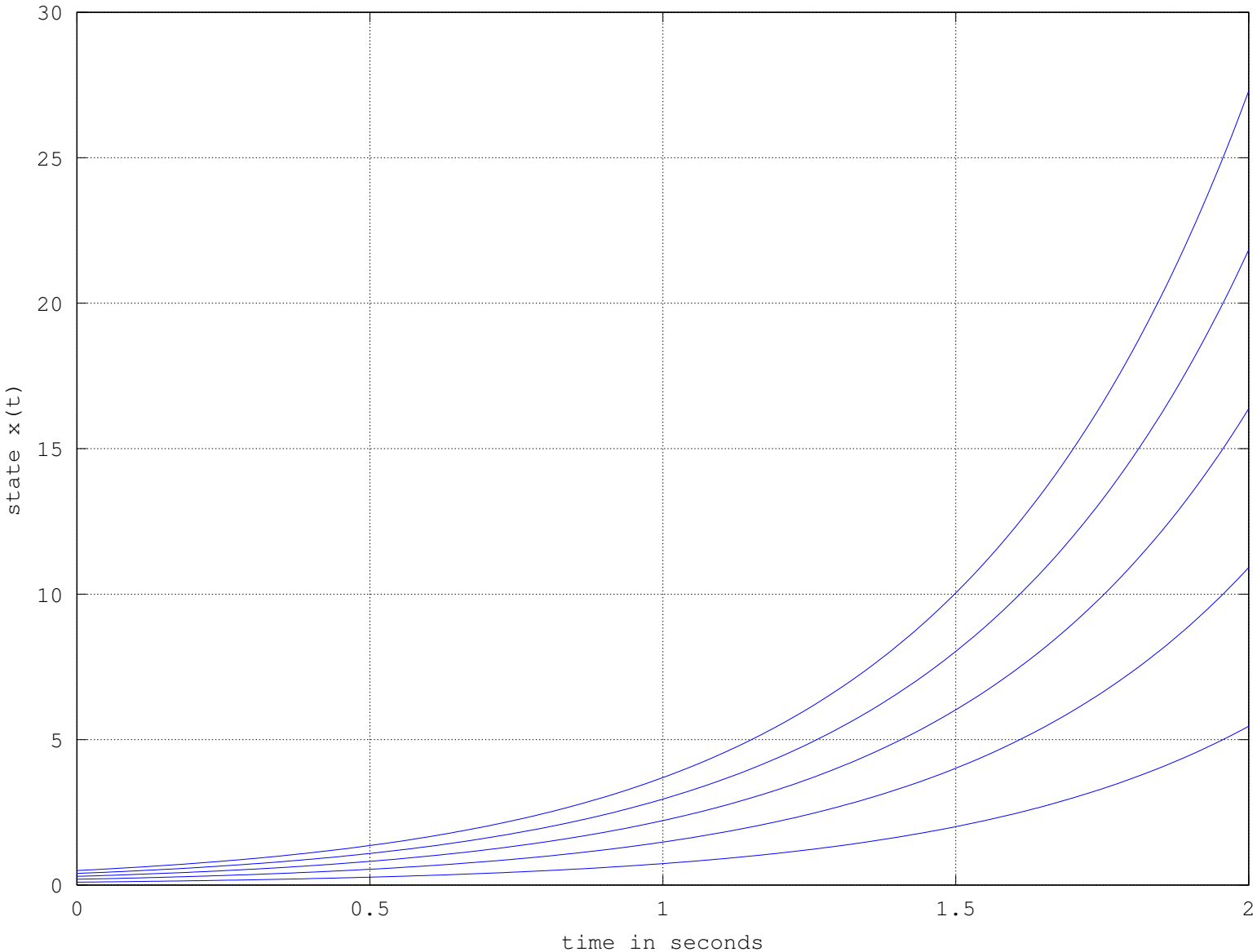
A = [2];B = [0];C = [3];D = [0]; t = 0:0.01:1;



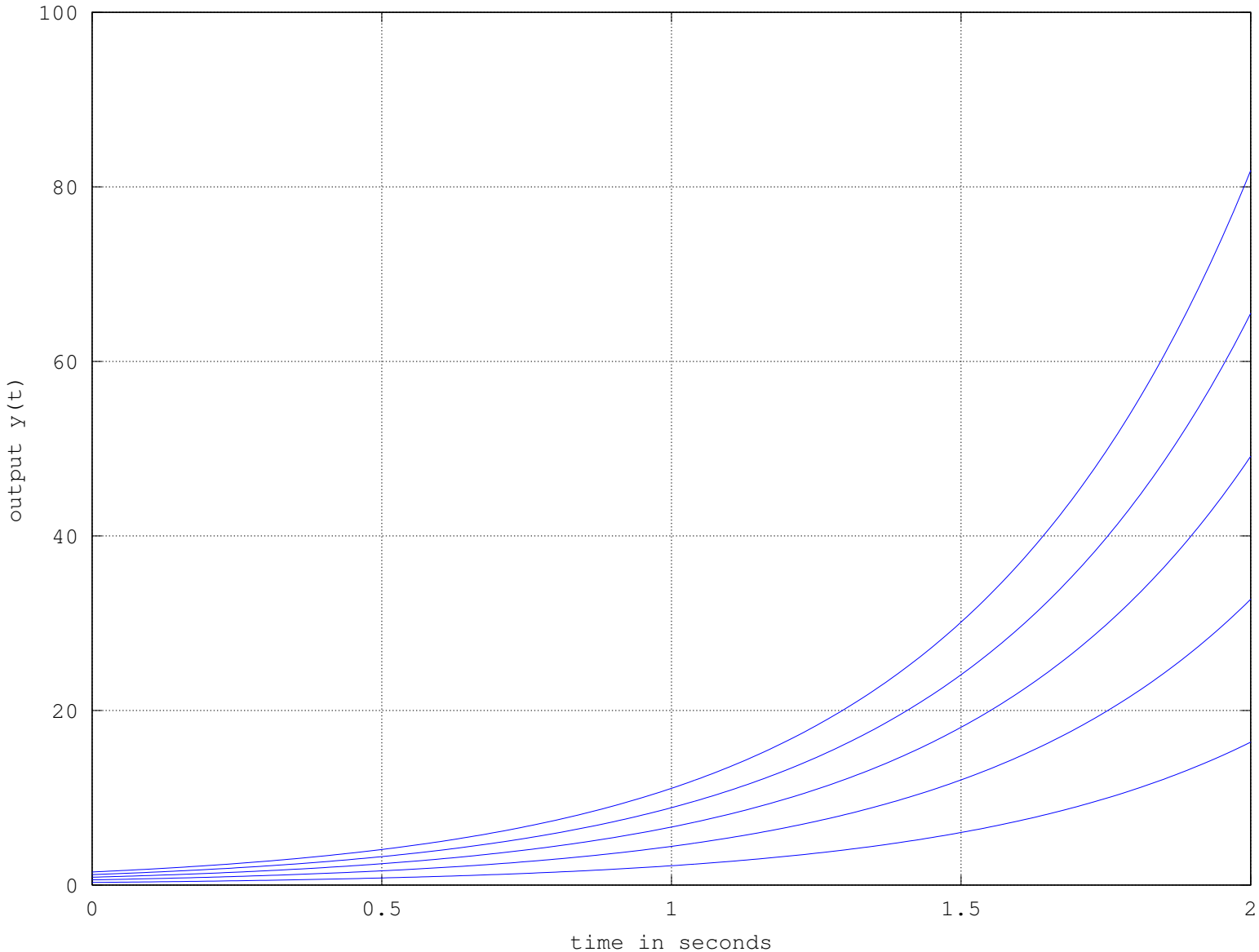
A = [2];B = [0];C = [3];D = [0]; t = 0:0.01:1;



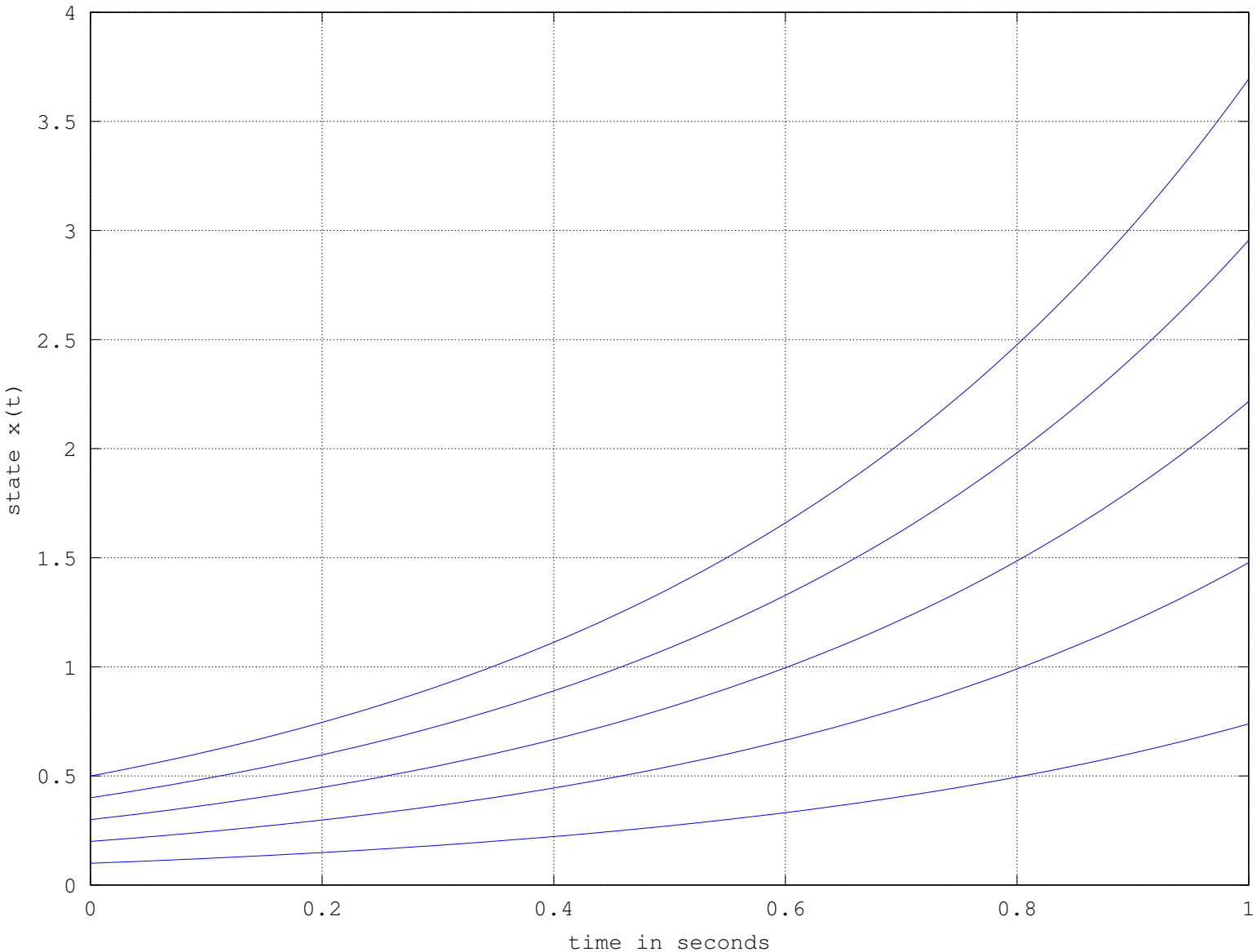
A = [2];B = [0];C = [3];D = [0]; t = 0:0.01:2;



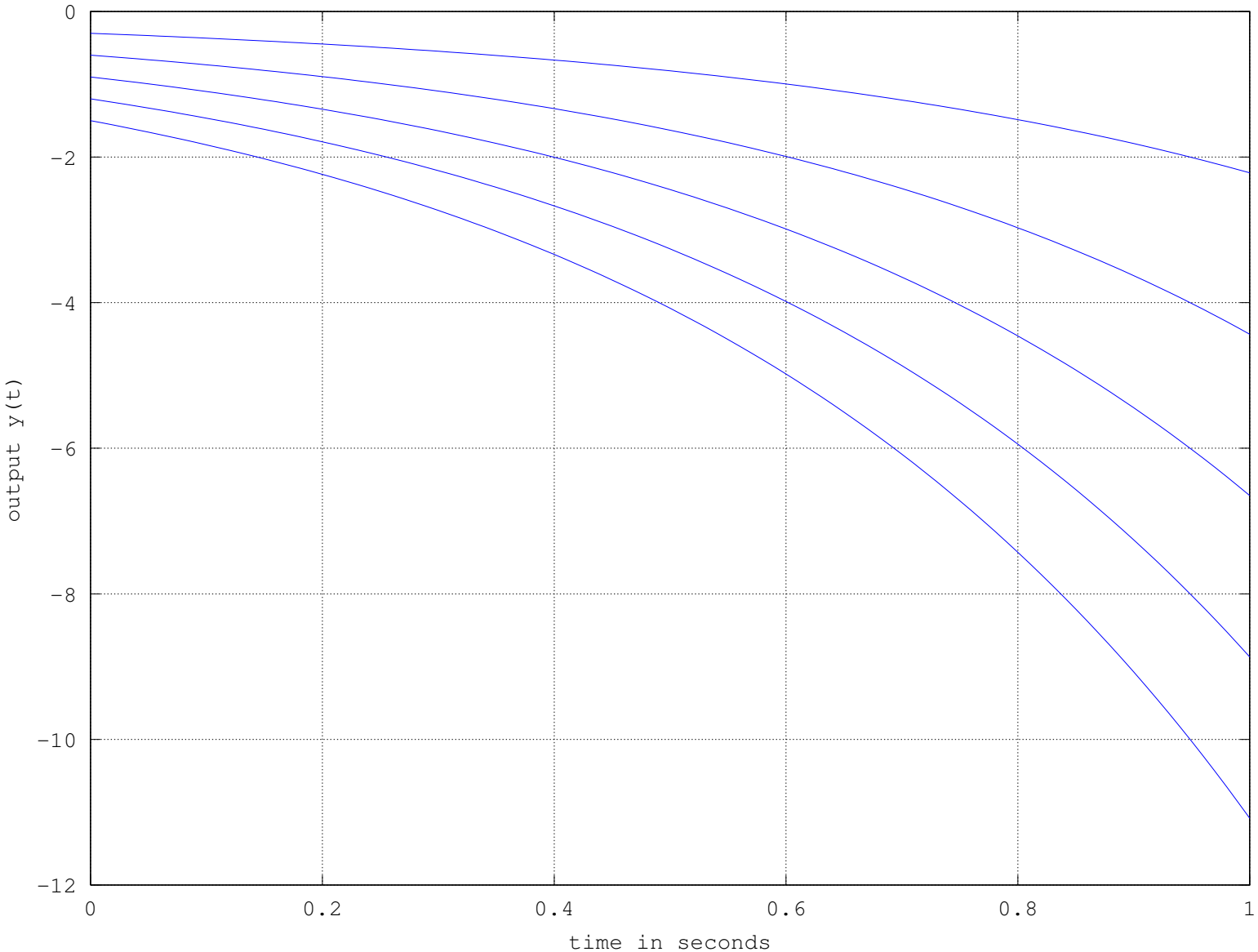
A = [2];B = [0];C = [3];D = [0]; t = 0:0.01:2;



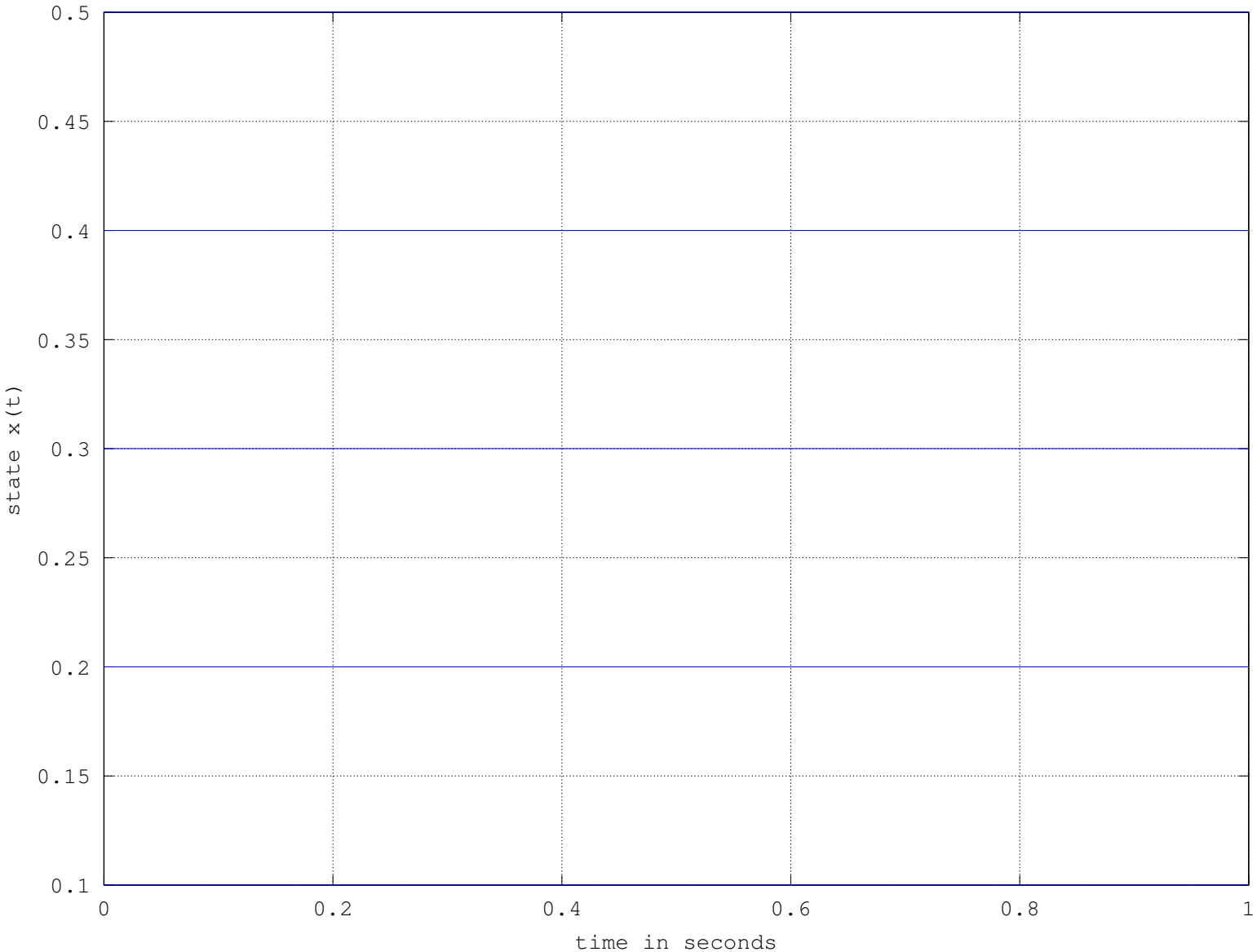
A = [2];B = [0];C = [-3];D = [0]; t = 0:0.01:1;



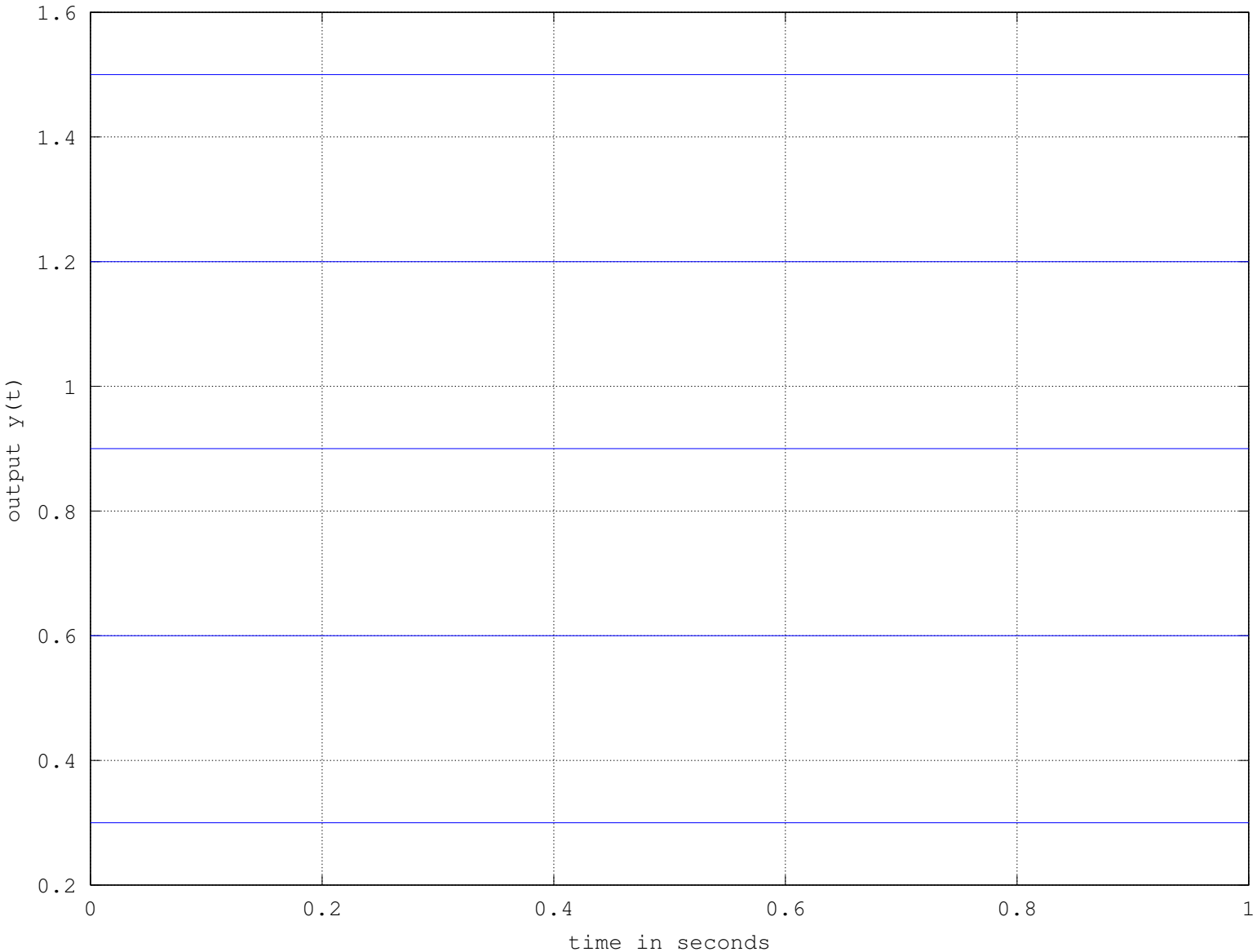
A = [2];B = [0];C = [-3];D = [0]; t = 0:0.01:1;



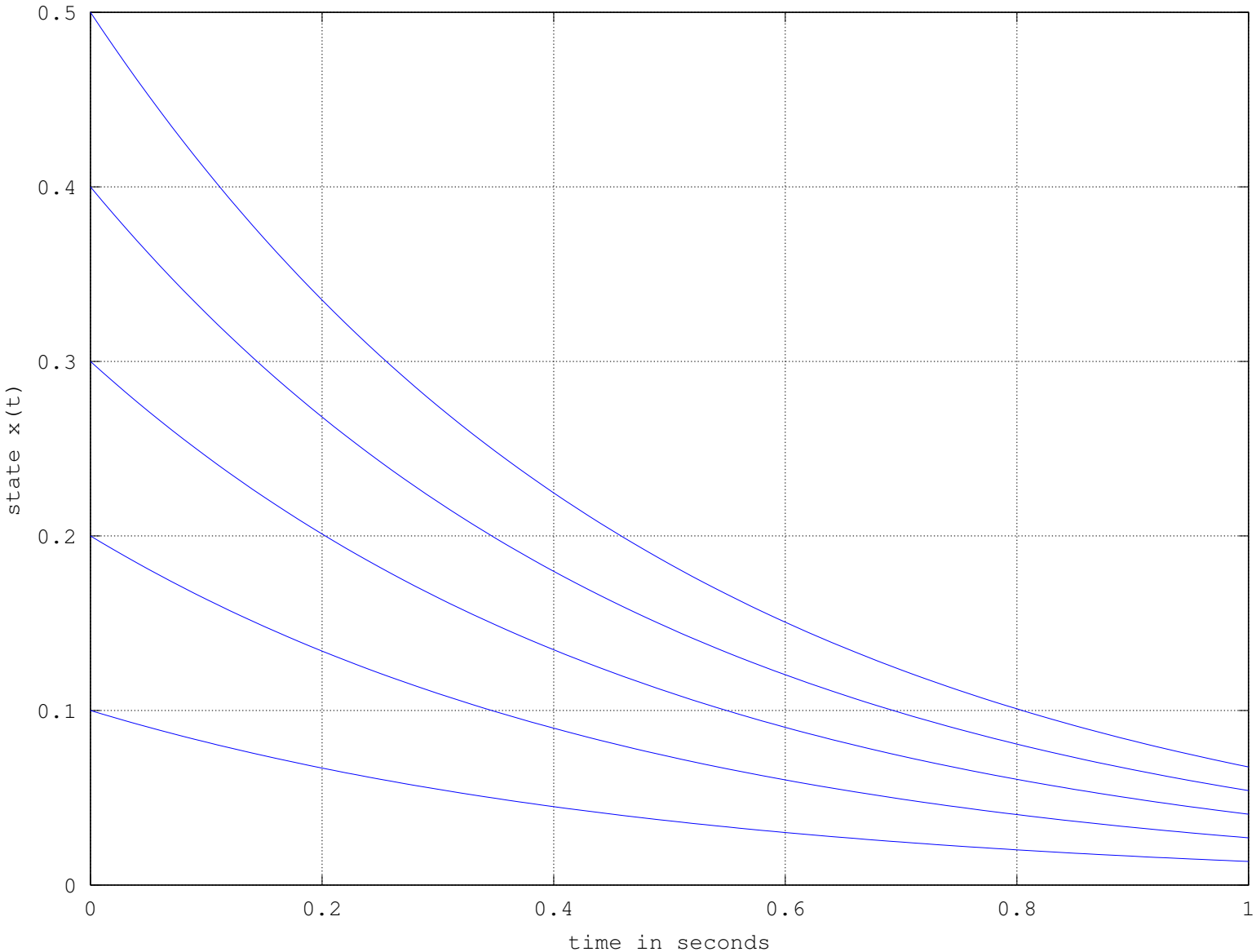
A = [0];B = [0];C = [3];D = [0]; t = 0:0.01:1;



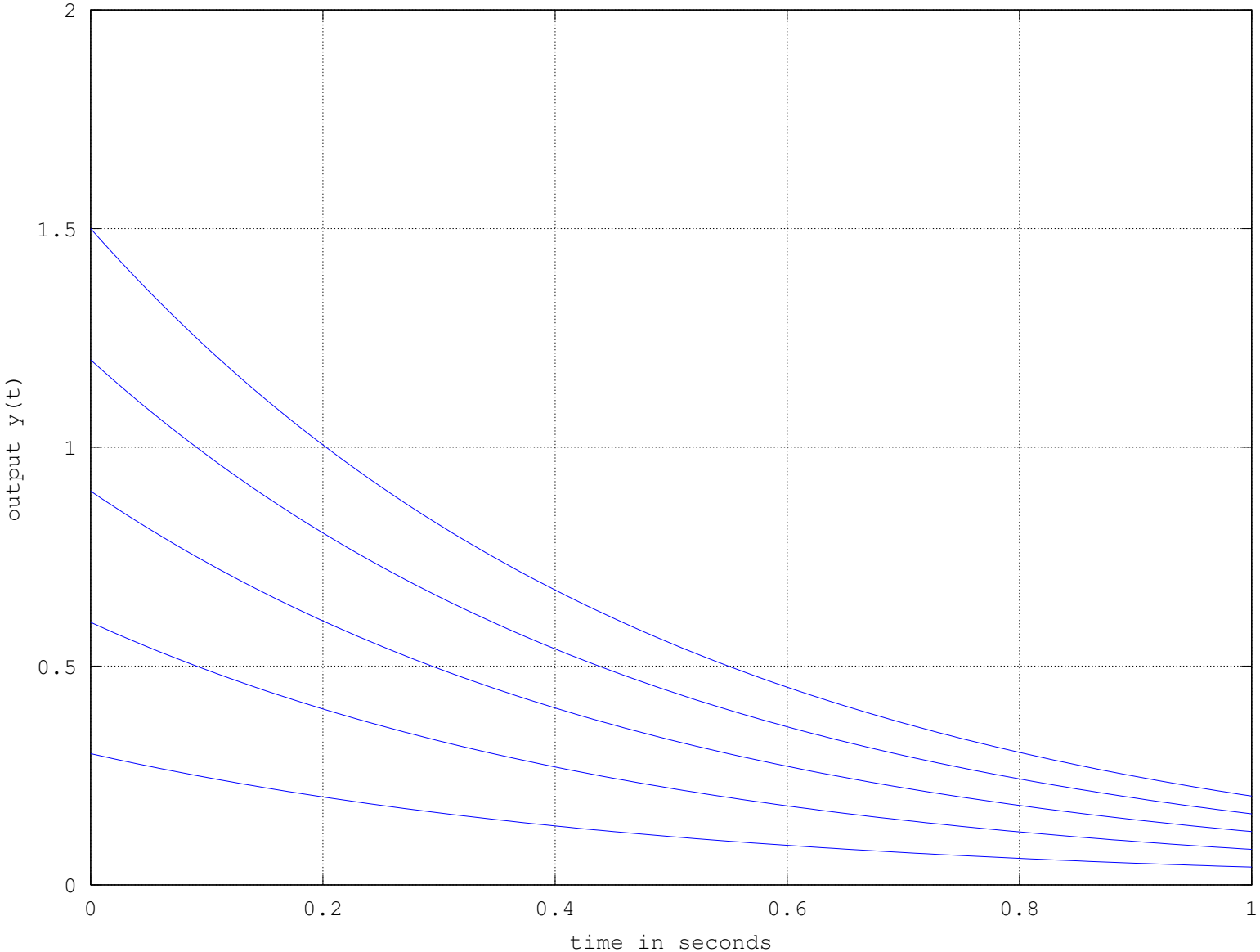
A = [0];B = [0];C = [3];D = [0]; t = 0:0.01:1;



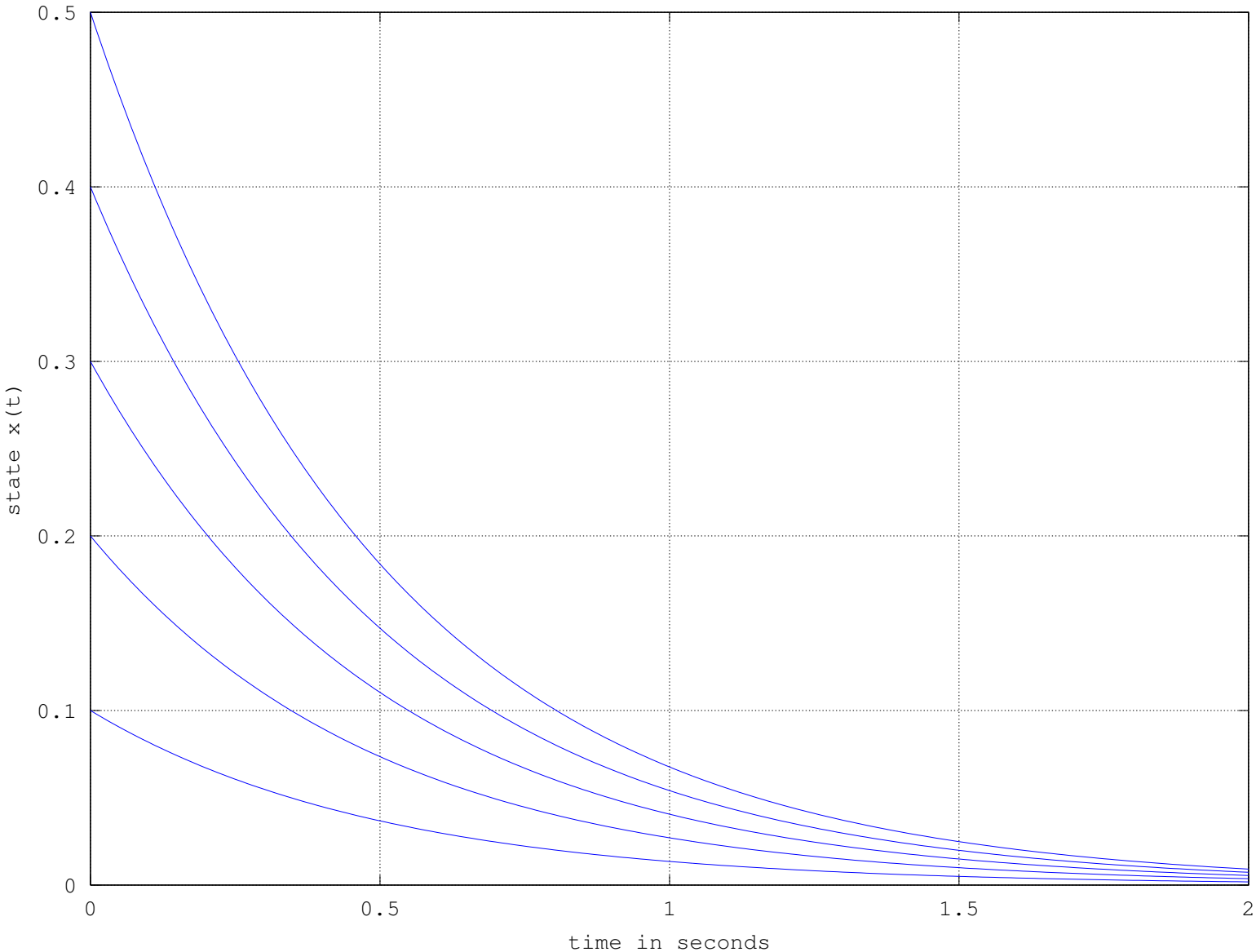
A = [-2];B = [0];C = [3];D = [0]; t = 0:0.01:1;



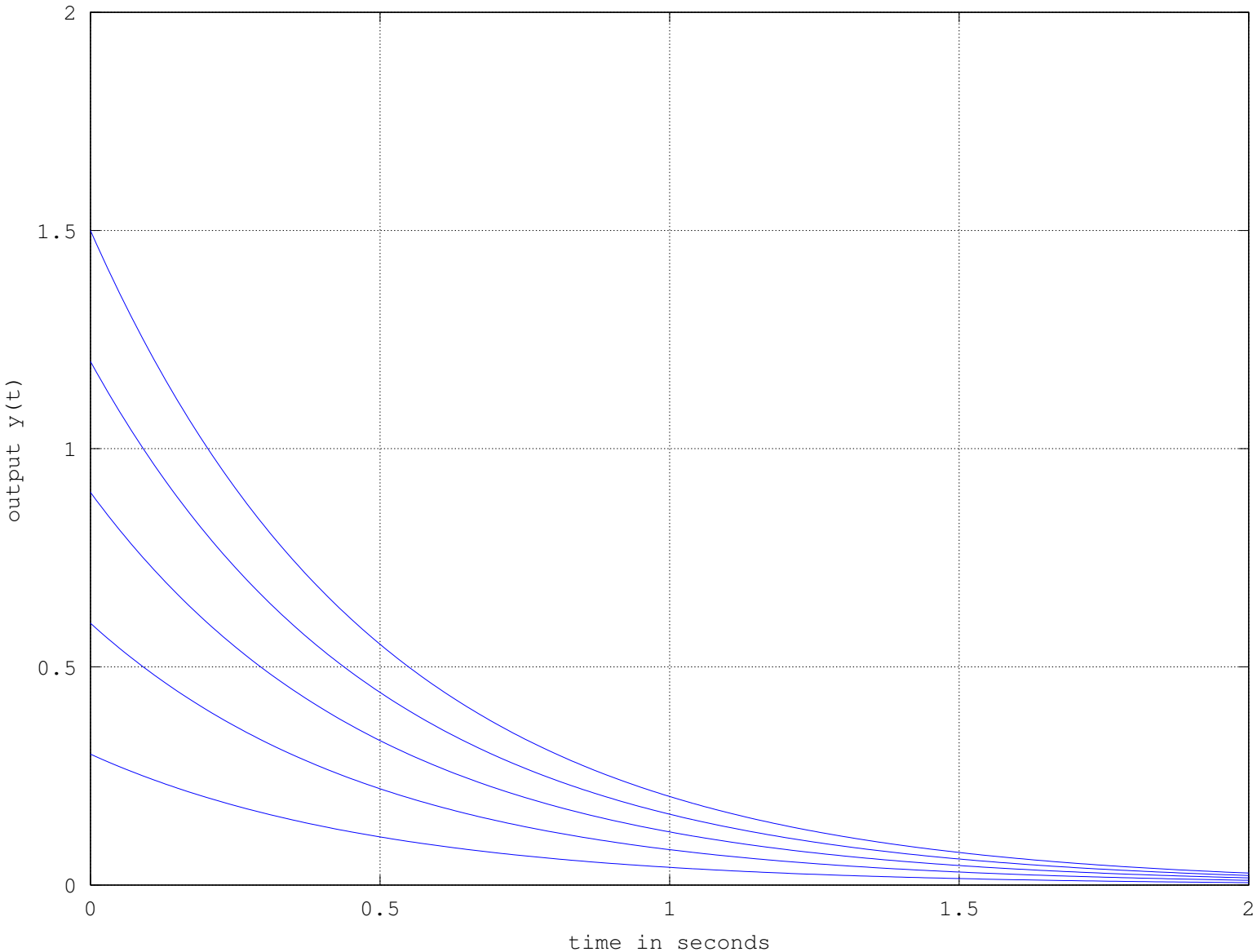
A = [-2];B = [0];C = [3];D = [0]; t = 0:0.01:1;



A = [-2];B = [0];C = [3];D = [0]; t = 0:0.01:2;



A = [-2];B = [0];C = [3];D = [0]; t = 0:0.01:2;



Simulation without Inputs Summary

- System without inputs

$$\dot{x}(t) = Ax(t)$$
$$y(t) = Cx(t)$$

- Solution

$$x(t) = e^{At}x(0)$$
$$y(t) = Ce^{At}x(0)$$

- Simulations for $A > 0$, $A < 0$, $A = 0$, different Cs and ts
- $A > 0$ trajectory explodes, $A < 0$ trajectory decays to zero
- Stable systems: State trajectory decays to zero
- In many systems it is preferable to have decaying trajectories

Simulation with Inputs Examples

- System with input

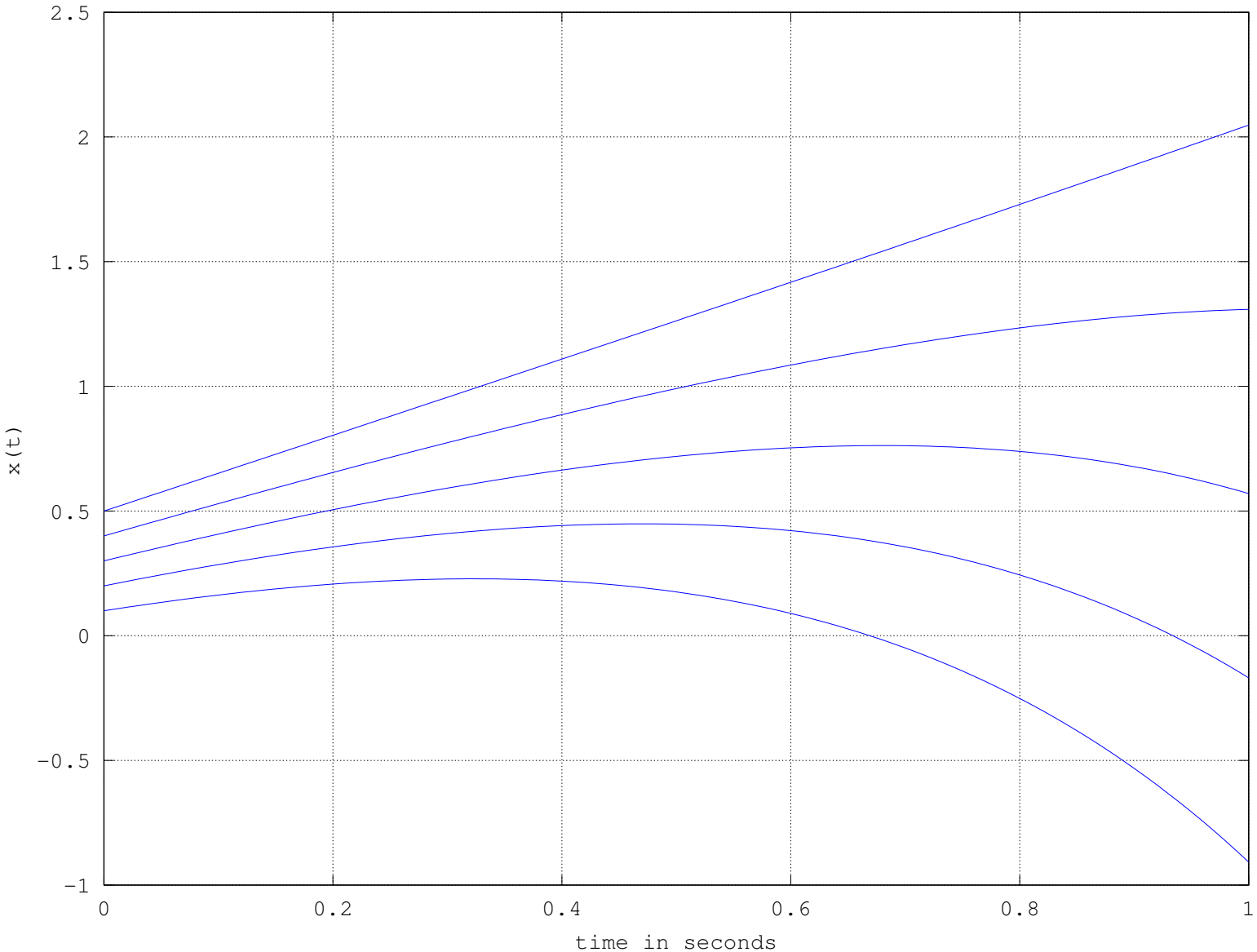
$$\dot{x}(t) = 2x(t) + 3u(t)$$
$$y(t) = 3x(t) - u(t)$$

- Solution

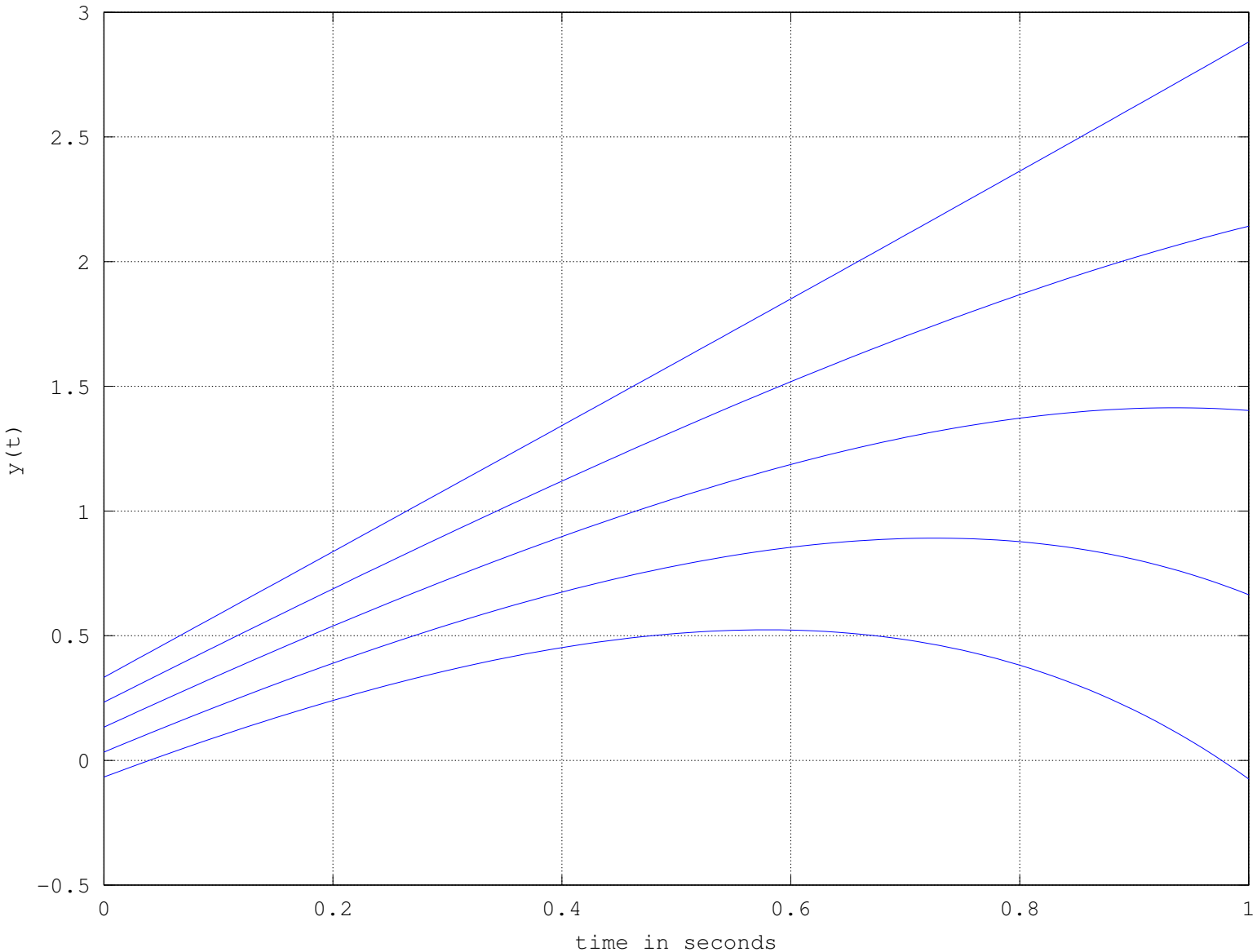
$$x(t) = e^{2t}x(0) + \int_0^t e^{2(t-\tau)}3u(\tau)d\tau$$
$$y(t) = 3e^{2t}x(0) + 3\int_0^t e^{2(t-\tau)}3u(\tau)d\tau - u(t)$$

- Octave Simulations

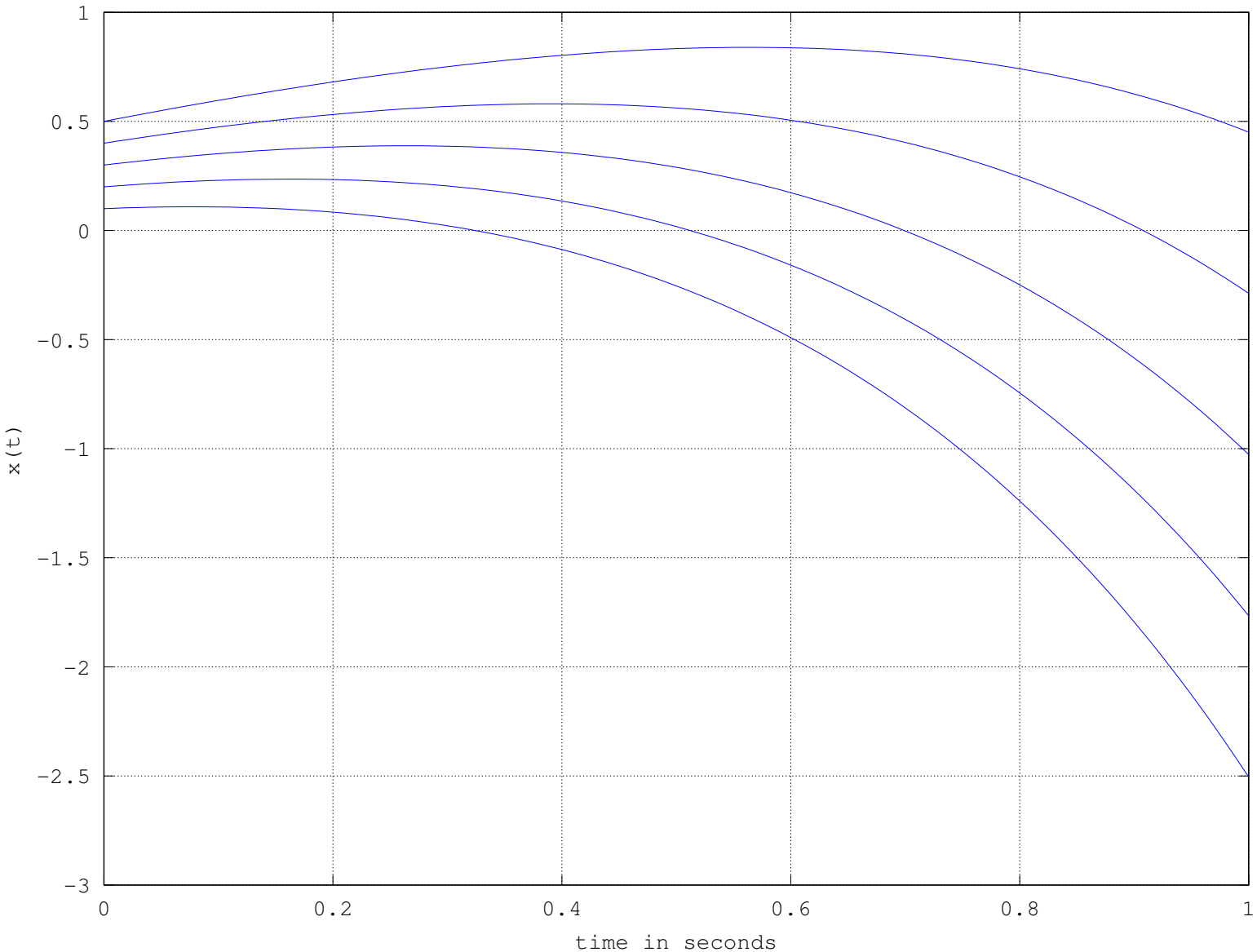
A = [2];B = [3];C = [1];D = [-1]; u=(0.5-3t)/3;t = 0-1;



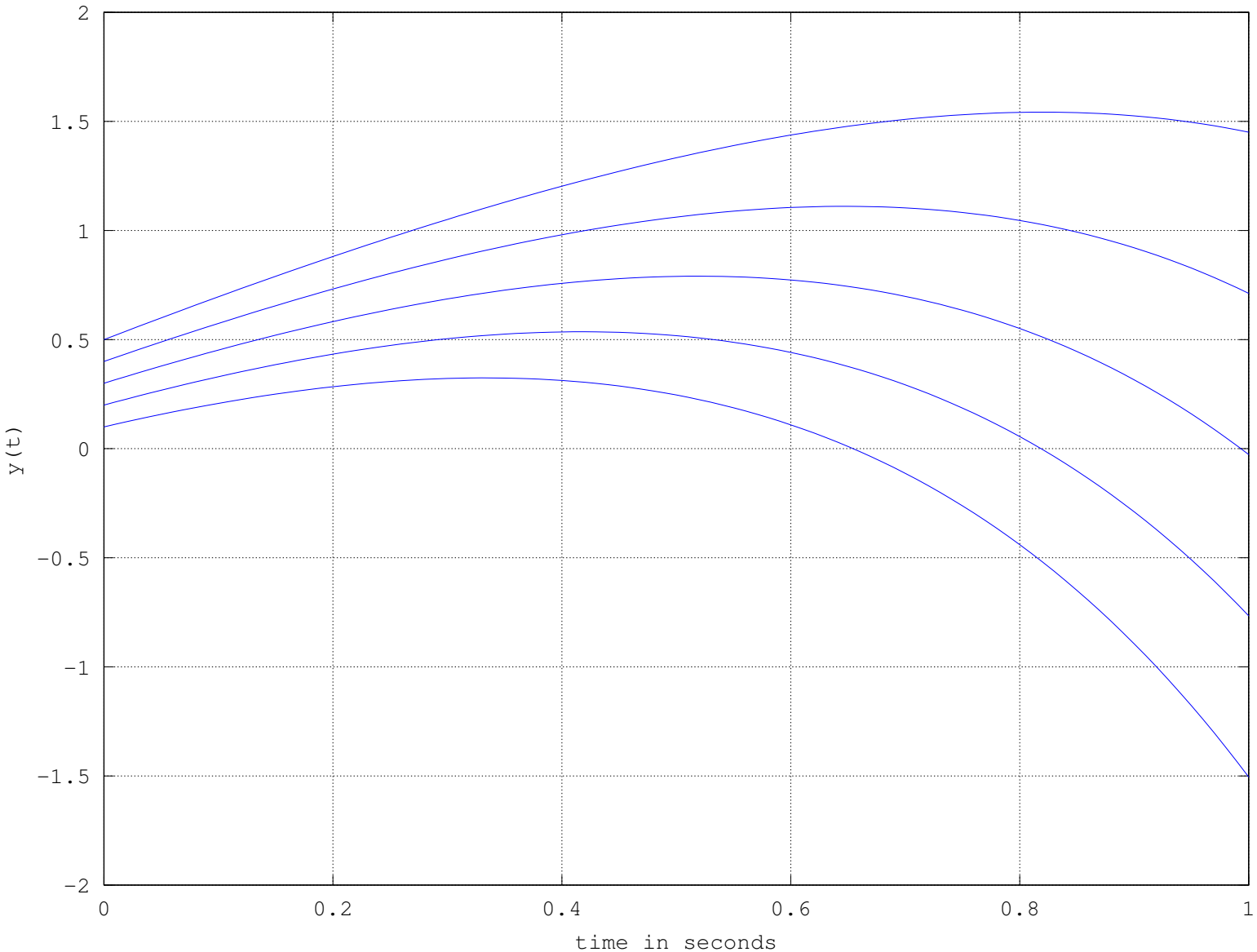
A = [2];B = [3];C = [1];D = [-1]; u=(0.5-3t)/3;t = 0-1;



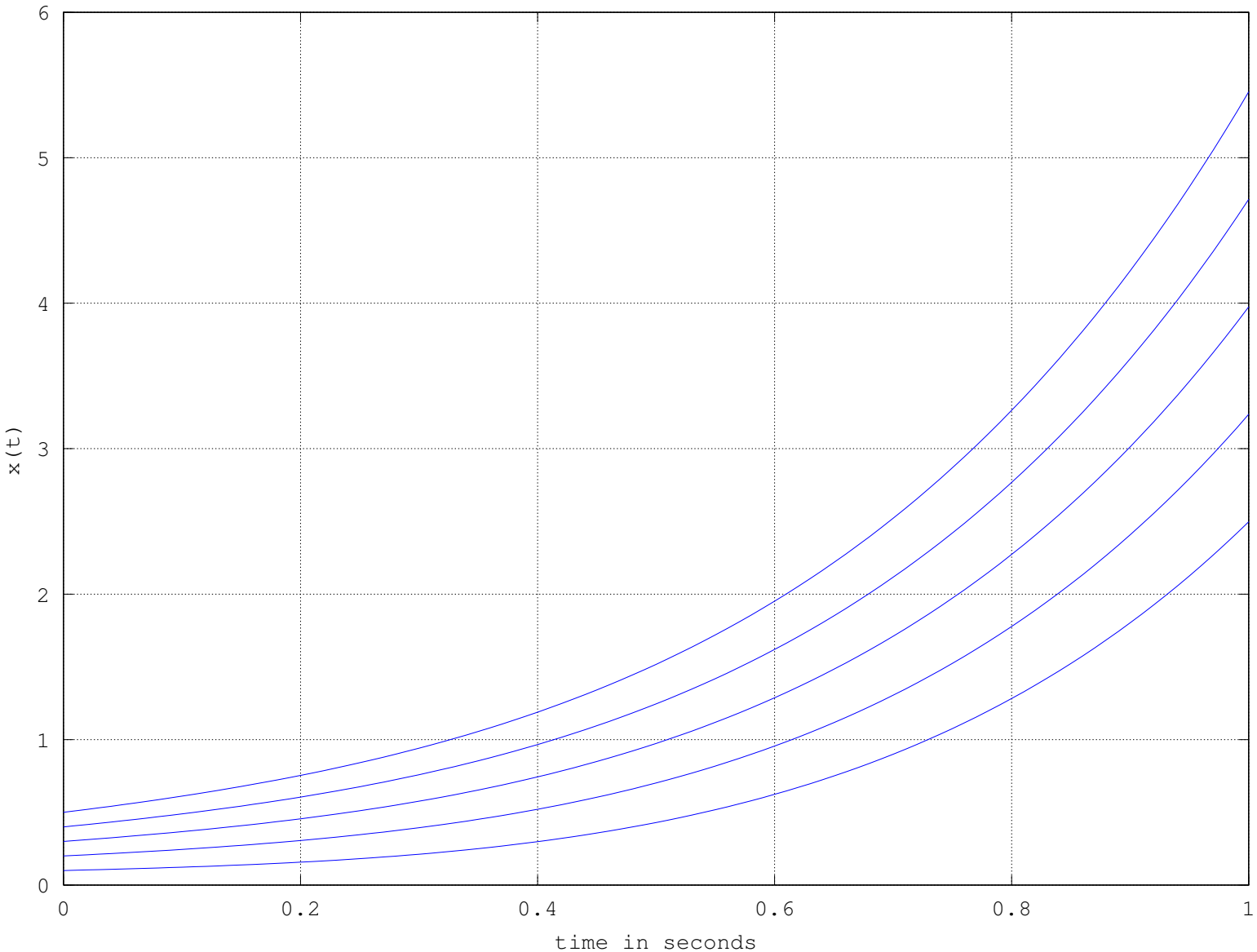
A = [2];B = [3];C = [1];D = [-1]; u=-t;t = 0-1;



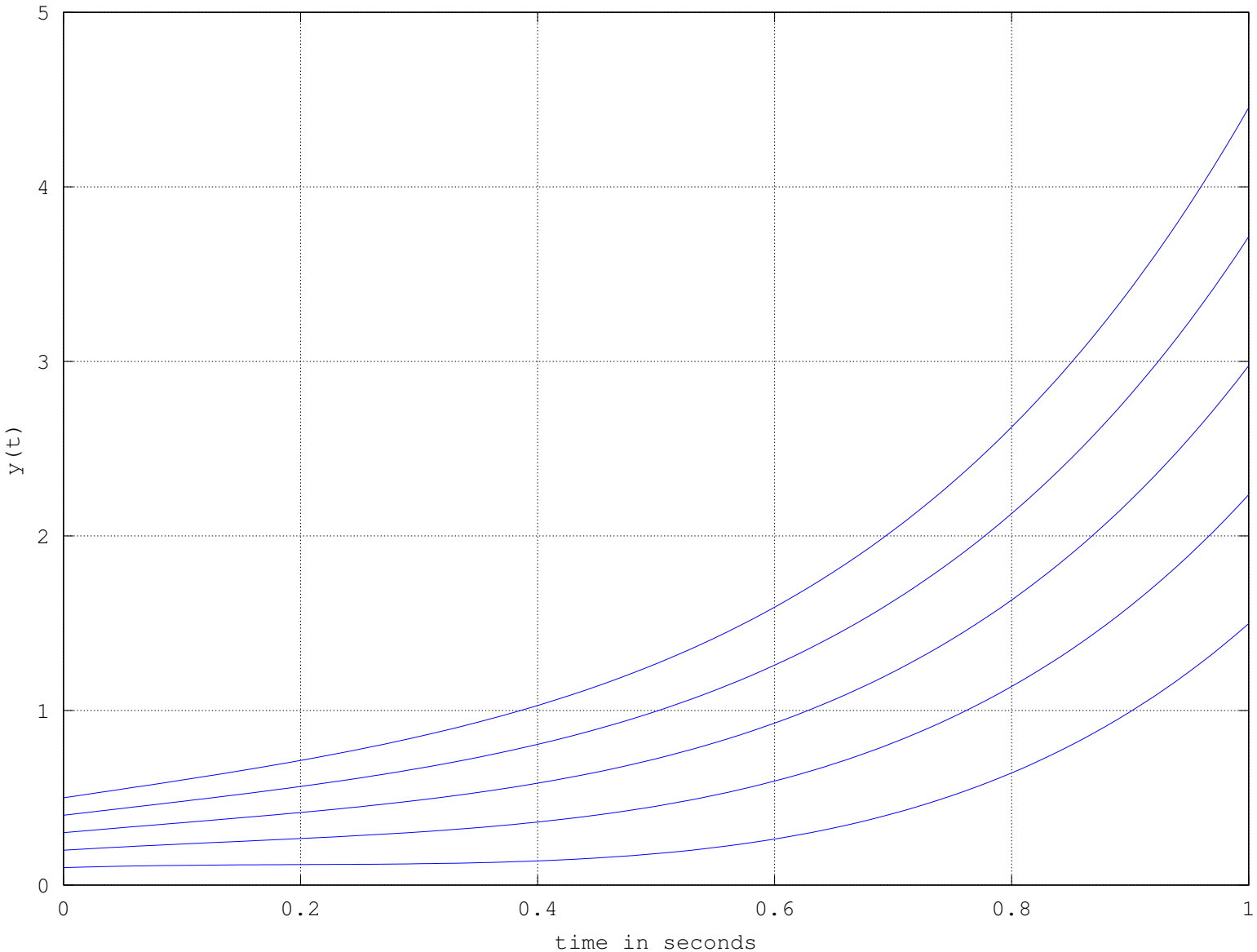
A = [2];B = [3];C = [1];D = [-1]; u=-t;t = 0-1;



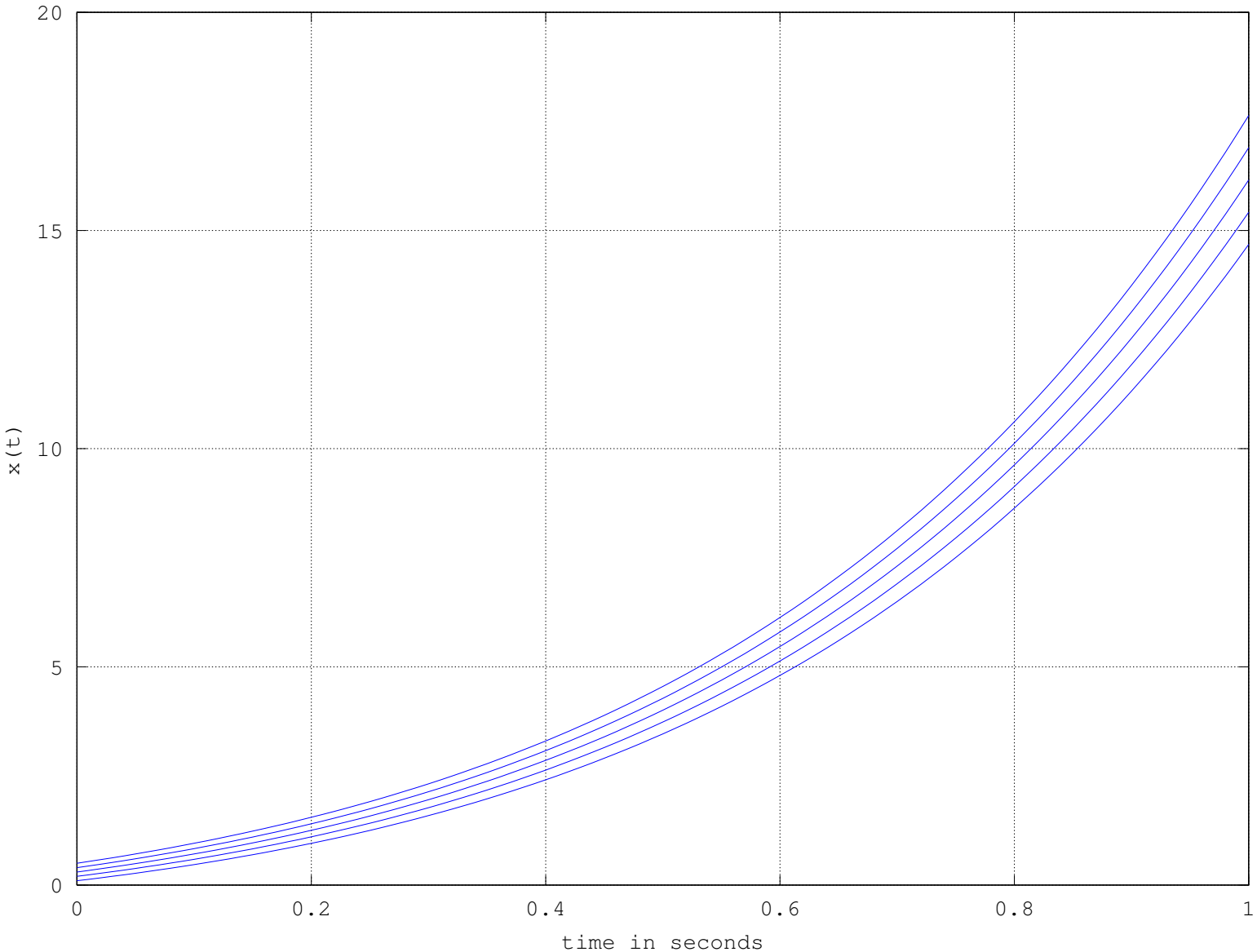
A = [2];B = [3];C = [1];D = [-1]; u=t*t;t = 0-1;



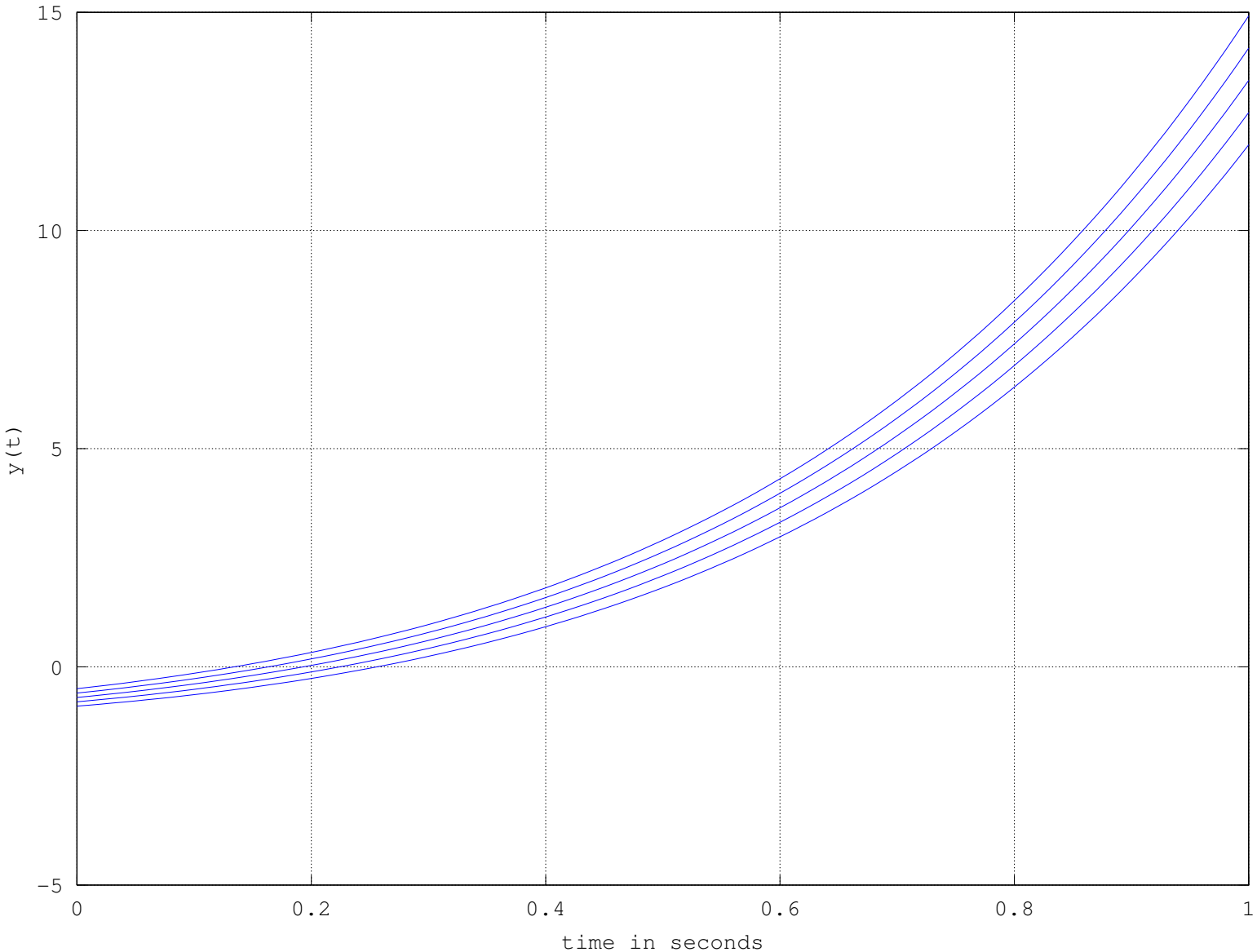
A = [2];B = [3];C = [1];D = [-1]; u=t*t;t = 0-1;



A = [2];B = [3];C = [1];D = [-1]; u=exp(t);t = 0-1;



A = [2];B = [3];C = [1];D = [-1]; u=exp(t);t = 0-1;



Simulation with Inputs Examples Summary

- System with input

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

- Solution

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

$$y(t) = Ce^{At}x(0) + C\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau + Du(t)$$

- Octave Simulations
- Question: How to make the trajectory to decay?
- Choose an $u(t)$ to make system trajectory to decay.
- Convert the above system to autonomous system.

Making a System into a Stable System

- System with input

$$\dot{x}(t) = 2x(t) + 3u(t)$$
$$y(t) = 3x(t) - u(t)$$

- Convert the above system to autonomous system with $A < 0$
- Suppose $u(t) = -x(t)$ then

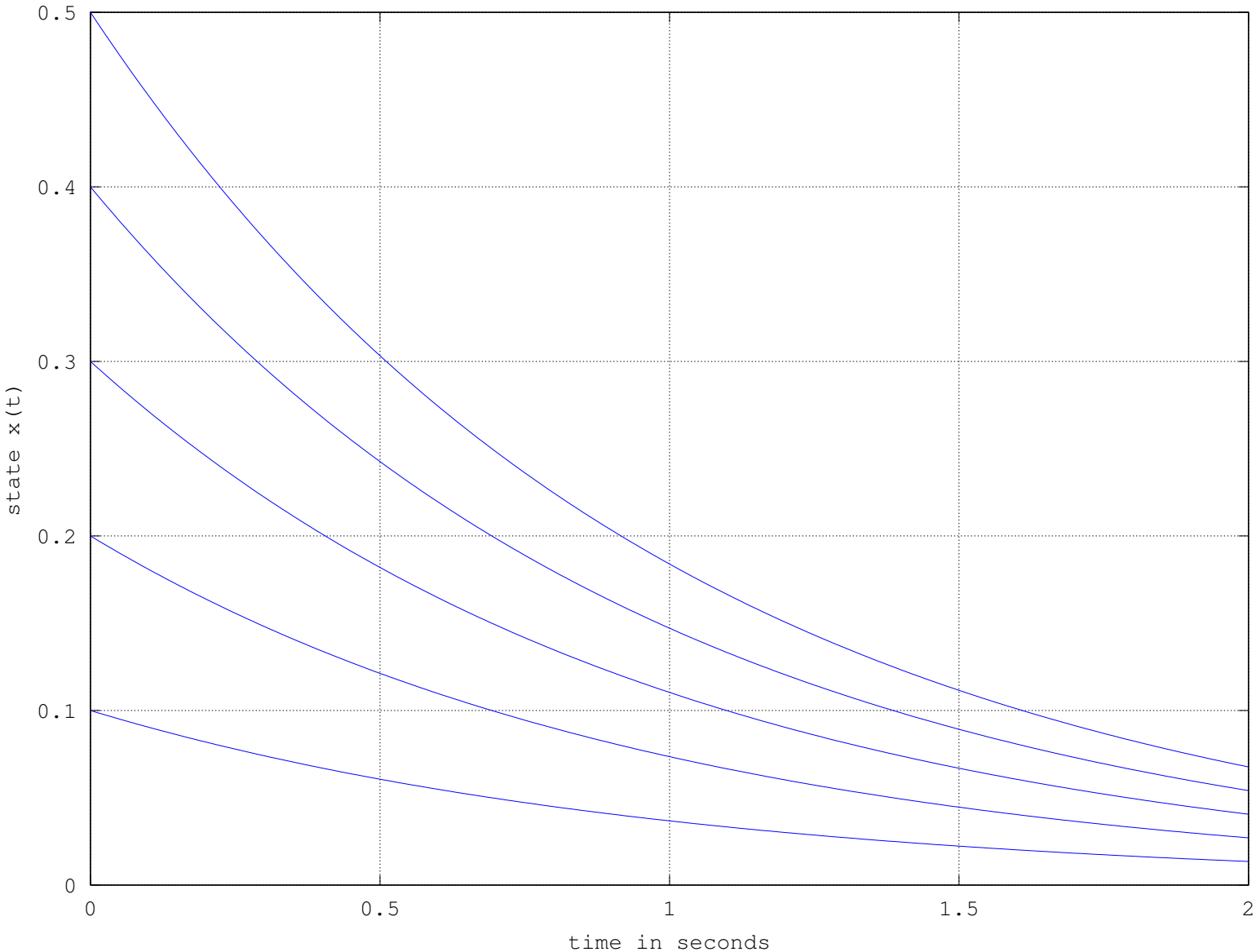
$$\dot{x}(t) = 2x(t) + 3(-x(t))$$
$$y(t) = 3x(t) - (-x(t))$$

- Autonomous system

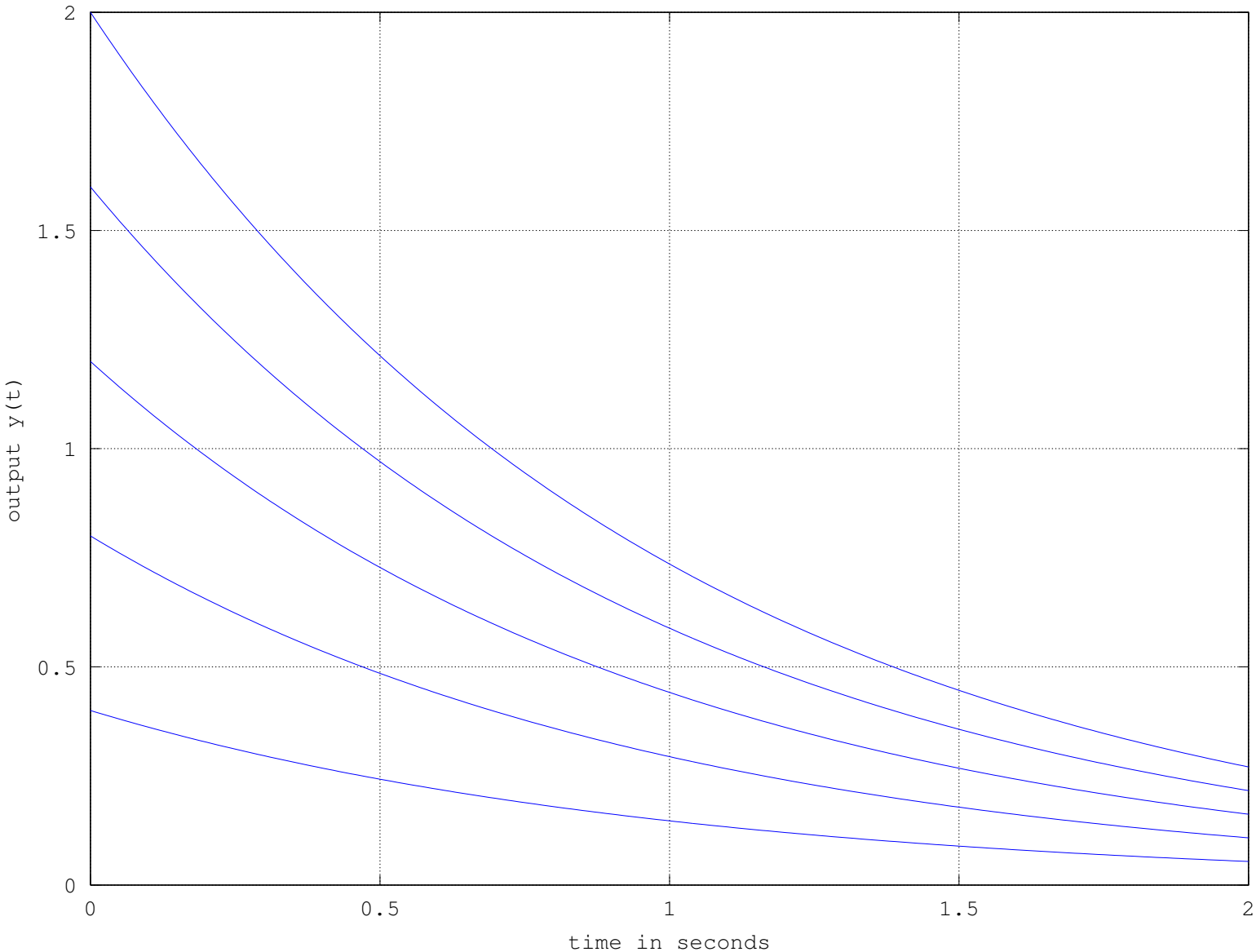
$$\dot{x}(t) = -x(t)$$
$$y(t) = 4x(t)$$

- Octave simulations

A = [-1];B = [0];C = [4];D = [0]; t = 0:0.01:2;



A = [-1];B = [0];C = [4];D = [0]; t = 0:0.01:2;



Non-scalar systems Examples

- System with inputs

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

- Without Inputs

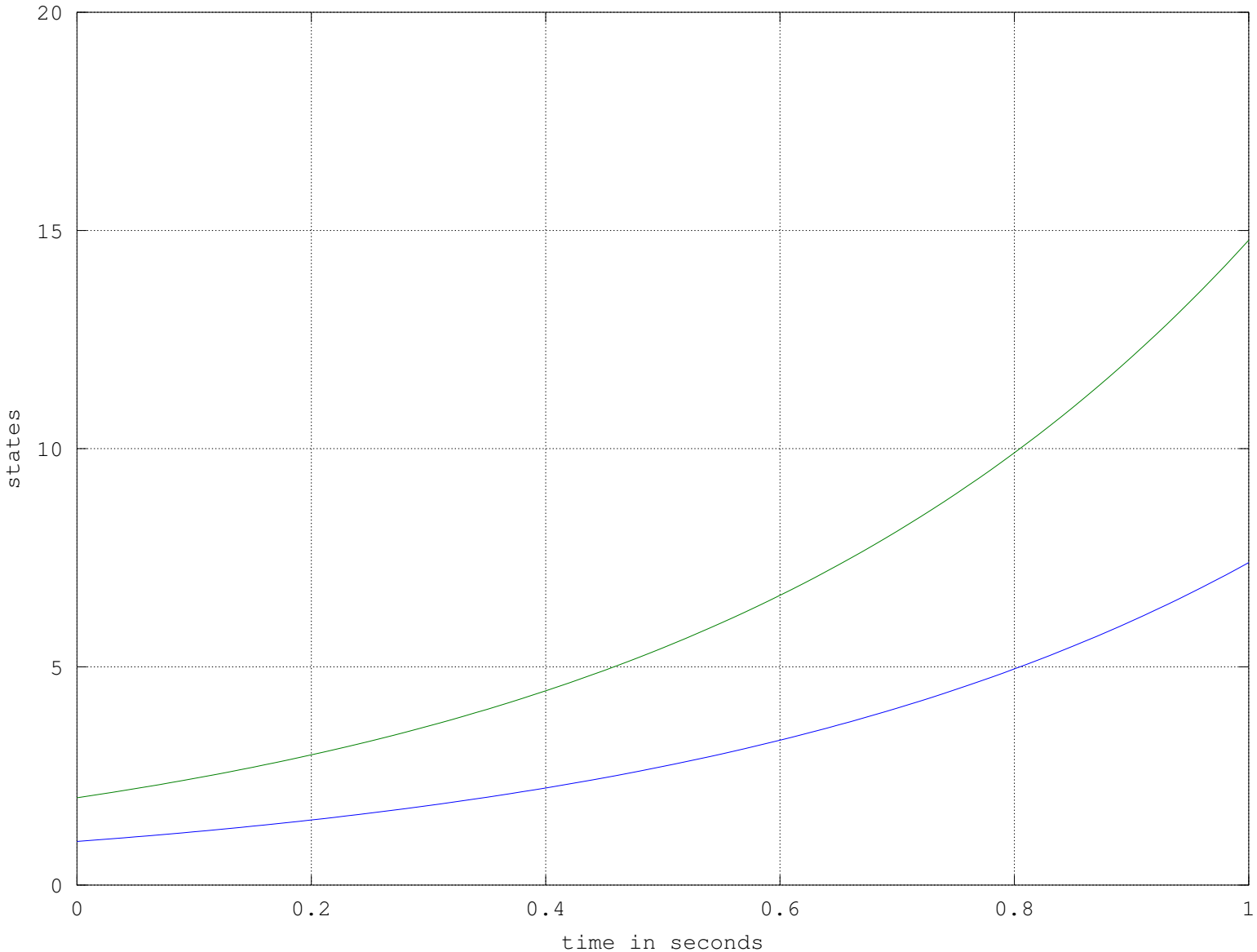
$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) \qquad \dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \mathbf{x}(t)$$

- Octave Simulations – Trajectories are exploding or decaying

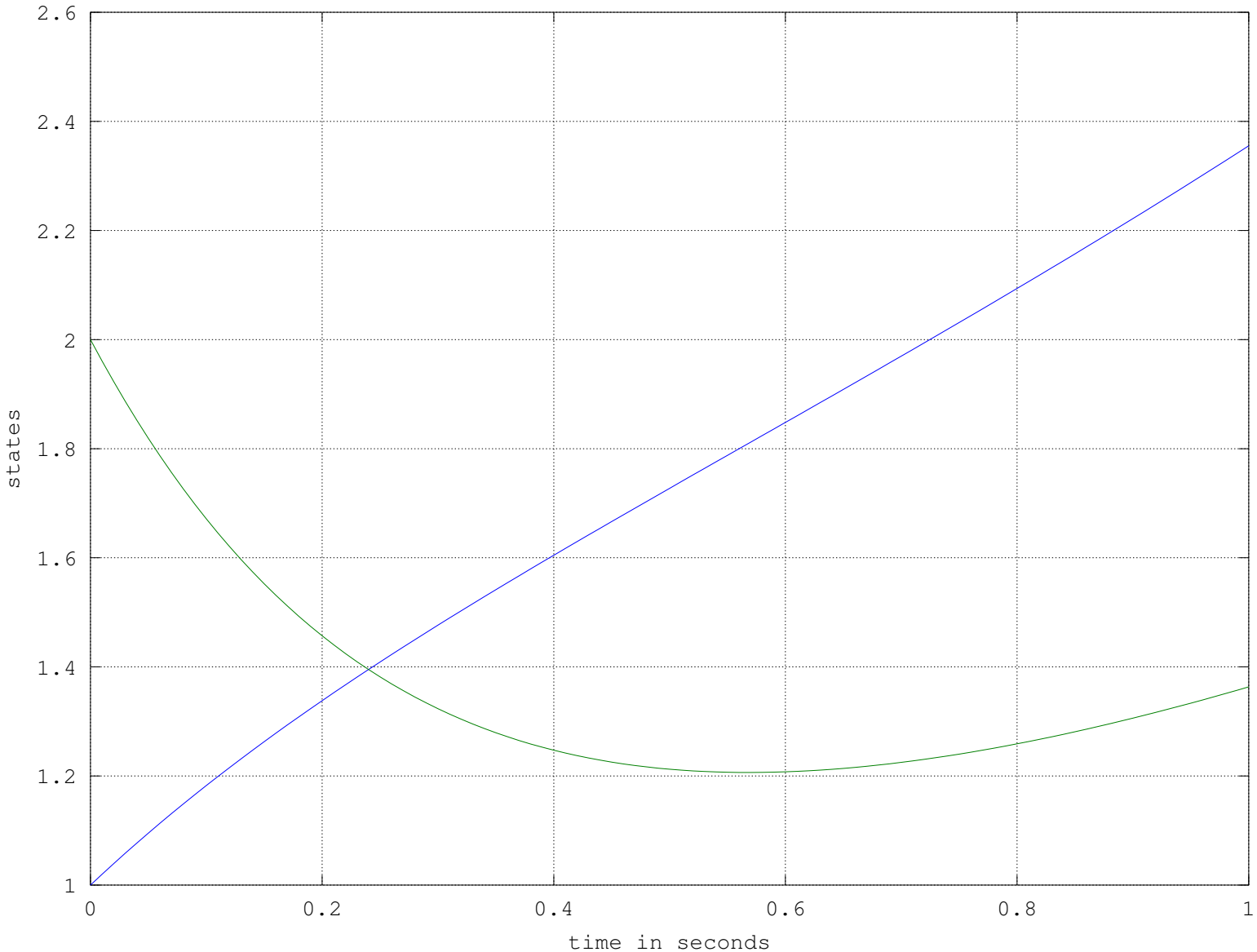
Matlab Program

```
A = [0 1; 2 -3 ]; B = [0;0]; C = [0 0]; D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u = 0;
[Y,T,X]=lsim(sys1,u,t,[1 2]);
plot(t,X);
xlabel("time in seconds");
ylabel("states and input");
grid on; hold on;
plot(t,u,'r');
```

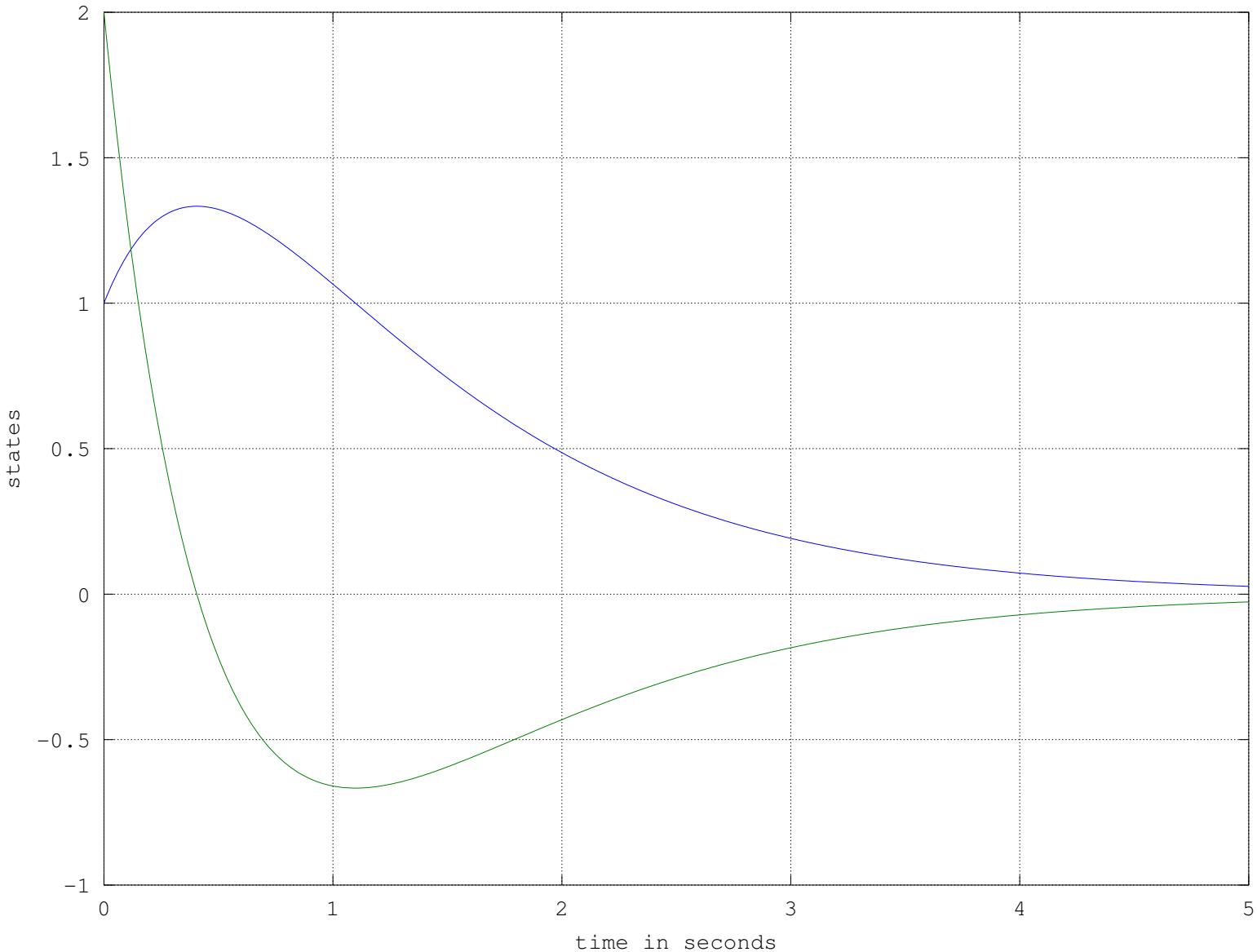
Autonomous: A = [0 1; 2 -3];Eigen values =1,2



Autonomous: A = [0 1; 2 -3];Eigen values = 0.56,-3.56



Autonomous: A = [0 1; -2 -3];Eigen values = -1,-2



Eigenvalues and Eigenvectors of a matrix

- If $A\mathbf{x} = \lambda\mathbf{x}$ then
- λ is called eigenvalue
- $\mathbf{x}$ is called eigenvector
- Then $[A - \lambda I]\mathbf{x} = 0$ and $\mathbf{x} \neq \mathbf{0}$
- Roots of determinant of $[A - \lambda I] = 0$ gives the eigenvalues
- Corresponding Eigenvectors are obtained by solving $[A - \lambda I]\mathbf{x} = 0$

Eigenvalues and Eigenvectors of a matrix: Examples

- Consider

$$A = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \qquad A - \lambda I = \begin{bmatrix} -\lambda & 1 \\ -2 & 3-\lambda \end{bmatrix}$$

- Determinant of $A - \lambda I = \lambda^2 - 3\lambda + 2$
- Roots of $(\lambda^2 - 3\lambda + 2) = 0$ are $\lambda = 1, 2$
- Similarly for

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \qquad A - \lambda I = \begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix}$$

- Determinant of $A - \lambda I = \lambda^2 + 3\lambda + 2$
- Roots of $(\lambda^2 - 3\lambda + 2) = 0$ are $\lambda = -1, -2$
- Determinant of $A - \lambda I$ is called characteristic polynomial of A

Characteristic polynomial

- Consider

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ a_1 & a_2 & \dots & a_{n-1} & a_n \end{bmatrix}$$

- Characteristic polynomial of A is

$$\lambda^n - a_n\lambda^{n-1} - \dots - \lambda a_2 - a_1$$

Non-scalar Systems Summary

- System with inputs

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

- Without Inputs

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) \qquad \dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \mathbf{x}(t)$$

- Octave Simulations – Trajectories are exploding or decaying
- Explosion: Eigenvalues of A are 1,2
- Decaying: Eigenvalues of A are -1,-2
- Stable systems have Decaying trajectories
- How do we convert Unstable system to Stable system?

Unstable to stable systems

- System with inputs

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

- Systems without inputs

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) \qquad \dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \mathbf{x}(t)$$

- If $u(t) = 0$ we obtain the first system (unstable)
- What $u(t)$ to choose to make it into stable second system?
- Verify if we let $u(t) = -6x_2(t)$ in the first system then we get the second system.

Changing the Characteristic polynomial

- Consider

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ a_1 & a_2 & \dots & a_{n-1} & a_n \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} u(t)$$

- System polynomial $p(\lambda) = \lambda^n - a_n\lambda^{n-1} - \ldots - a_2\lambda - a_1$
- When $u(t) = 0$ system is stable if $p(\lambda)$ has all negative roots
- If desired system polynomial is $\lambda^n - b_n\lambda^{n-1} - \ldots - b_2\lambda - b_1$
- How to choose $u(t)$?
- $u(t) = (b_1 - a_1)x_1 + (b_2 - a_2)x_2 + \ldots + (b_n - a_n)x_n$
- $u(t) = K\mathbf{x}(t)$ where $K = [(b_1 - a_1)\ (b_2 - a_2)\ \ \ldots\ \ (b_n - a_n)]$

Unstable to stable system Arbitrary

- Special Systems (Canonical System)

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ a_1 & a_2 & \dots & a_{n-1} & a_n \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} u(t)$$

- Arbitrary System

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} u(t)$$

- How to stabilize arbitrary system?
- Choose K such that $u(t) = K\mathbf{x}(t)$ will stabilize the system

Transformation of systems and stabilization

- Consider $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$
- Suppose $\mathbf{z}(t) = P\mathbf{x}(t)$ and P invertible then

$$\begin{aligned}\dot{\mathbf{z}}(t) &= P\dot{\mathbf{x}}(t) \\ &= PA\mathbf{x}(t) + PB\mathbf{u}(t) \\ &= PAP^{-1}\mathbf{z}(t) + PB\mathbf{u}(t)\end{aligned}$$

- Choose P is such that $\mathbf{z}(t)$ has the special structure (How?)
- Then it is easy to choose $u(t) = K\mathbf{z}(t) = KP\mathbf{x}(t)$
- K is made of difference of the desired coefficients with existing coefficients
- $P^{-1} = [A^{n-1}b - a_nA^{n-1}b - \ldots - a_2b \quad \ldots \quad Ab - a_nb \quad b]$

Transformation Example

- Consider the system

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t)$$

- Eigen values of A 0.2679, 3.7321 – unstable system
- Characteristic Polynomial of A $p(\lambda) = \lambda^2 - 4\lambda + 1$
- Transformation matrix

$$\begin{aligned} P^{-1} &= [Ab - 4b \quad b] \\ &= \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix} \qquad \Longrightarrow \quad P = \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \end{aligned}$$

- Transformed system $\mathbf{z}(t) = P\mathbf{x}(t)$

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 4 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

Example contd.

- Current Polynomial is $p(\lambda) = \lambda^2 - 4\lambda + 1$
- Let the desired polynomial be $p(\lambda) = \lambda^2 + 3\lambda + 2$
- Gain matrix for transformed system
  $K = [1 - 2 \quad -4 - 3] = [-1 \quad -7]$
- For the original system

$$\widehat{K} = KP \qquad = [-1 \quad -7] \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \qquad = [-2 \quad -5]$$

- After Feedback $u(t) = \widehat{K}\mathbf{x}(t)$

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} [2 \quad -3]\mathbf{x}(t) \qquad = \begin{bmatrix} -1 & -4 \\ 0 & -2 \end{bmatrix} \mathbf{x}(t)$$

- Verify eigenvalues of new A are -1 and -2

Controllable Systems

- Definition 1: A system in which the coefficients of the system polynomial can be varied arbitrarily by state feedback
- Roots of the system polynomial are called poles of the system
- Roots of the system polynomial are dependent on the coefficients of the system polynomial
- Definition 2: A system in which arbitrary pole placement is possible by state feedback
- Definition 3: A system is said to be controllable if one can compute an input $u(t)$ such that one can reach any arbitrary final state from any initial state in a finite time.

State Transfer Problem

- Dynamical System:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

- initial state $= \mathbf{x_0}$, final state $= \mathbf{x_f}$
- Find $\mathbf{u}^*(t)$ for $t \in [t_0, t_f]$ such that when $\mathbf{x}(t_0) = \mathbf{x_0}$, the solution of $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}^*(t)$ is such that $\mathbf{x}(t_f) = \mathbf{x_f}$
- Recall the solution

$$\mathbf{x}(t_f) = e^{At_f}\mathbf{x_0} + \int_{t_0}^{t_f} e^{A(t_f-\tau)}B\mathbf{u}^*(\tau)d\tau$$

- Question: How to find such inputs?

Standard method for computing u for state transfer

- Consider $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$
- Then $\mathbf{x_f} = \mathbf{x}(t_f) = e^{At_f}\mathbf{x_0} + \int_{t_0}^{t_f} e^{A(t_f-\tau)}B\mathbf{u}(\tau)d\tau$
- Consider $W_C(t_f) = \int_{t_0}^{t_f} e^{A(t_f-\tau)}BB^T e^{A^T(t_f-\tau)}d\tau$
- Let $\mathbf{u}^*(t) = B^T e^{A^T(t_f-\tau)}W_C^{-1}(t_f)(\mathbf{x_f} - e^{At_f}\mathbf{x_0})$
- Then $\mathbf{x}(t_f) = e^{At_f}\mathbf{x_0} + W_C(t_f)W_C^{-1}(t_f)(\mathbf{x_f} - e^{At_f}\mathbf{x_0}) = \mathbf{x_f}$
- Controllability condition: Gramian matrix $W_C(t_f)$ should be invertible
- Arbitrary state transfer is possible if and only if $W_C(t_f)$ is invertible

Example - state transfer using Gramian

- System Dynamics

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \implies e^{At} B = \begin{bmatrix} t \\ 1 \end{bmatrix}$$

- Gramian

$$W_C(t_f) = \int_0^{t_f} \begin{bmatrix} (t_f - t)^2 & t_f - t \\ t_f - t & 1 \end{bmatrix} dt = \begin{bmatrix} \frac{1}{3} t_f^3 & \frac{1}{2} t_f^2 \\ \frac{1}{2} t_f^2 & t_f \end{bmatrix}$$

- Let $\mathbf{x}(0) = [0\ 0]^T$ and $\mathbf{x}(t_f) = [1\ 0]^T$
- Input

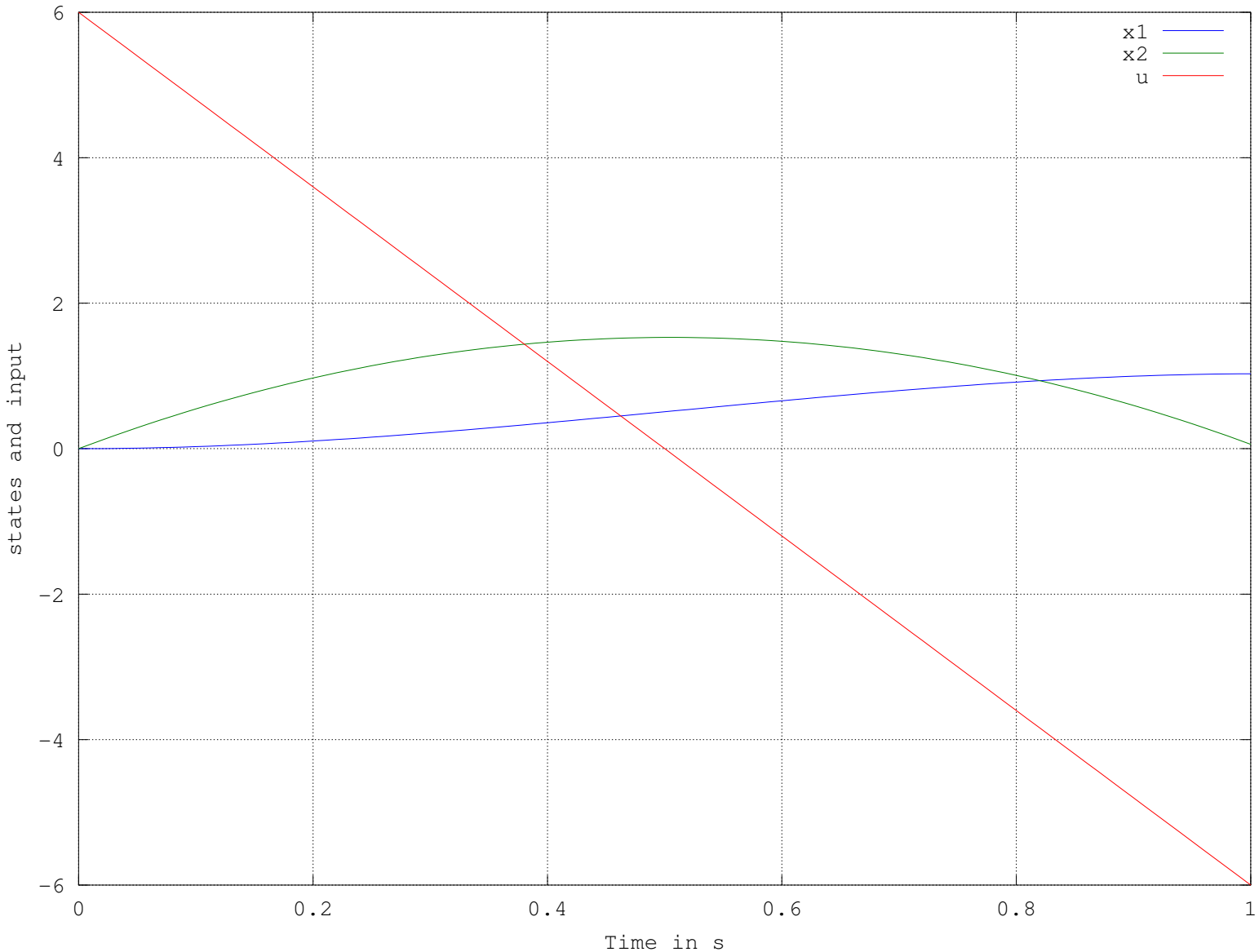
$$u(t) = [t_f - t\ 1] \begin{bmatrix} 12/t_f^3 \\ -6/t_f^2 \end{bmatrix} = \frac{6}{t_f^3}(t_f - 2t)$$

- Octave results

Matlab Program

```
A = [0 1; 0 0 ];
B = [0;1];
C = [0 0];
D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u = 6*(1 - t.*2);
[Y,T,X] = lsim(sys1,u,t,[0 0]);
plot(T,X);
hold on;
plot(T,u,'r');
grid on;
xlabel("Time in s");
ylabel("states and input");
```

State Transfer: Grammian Input

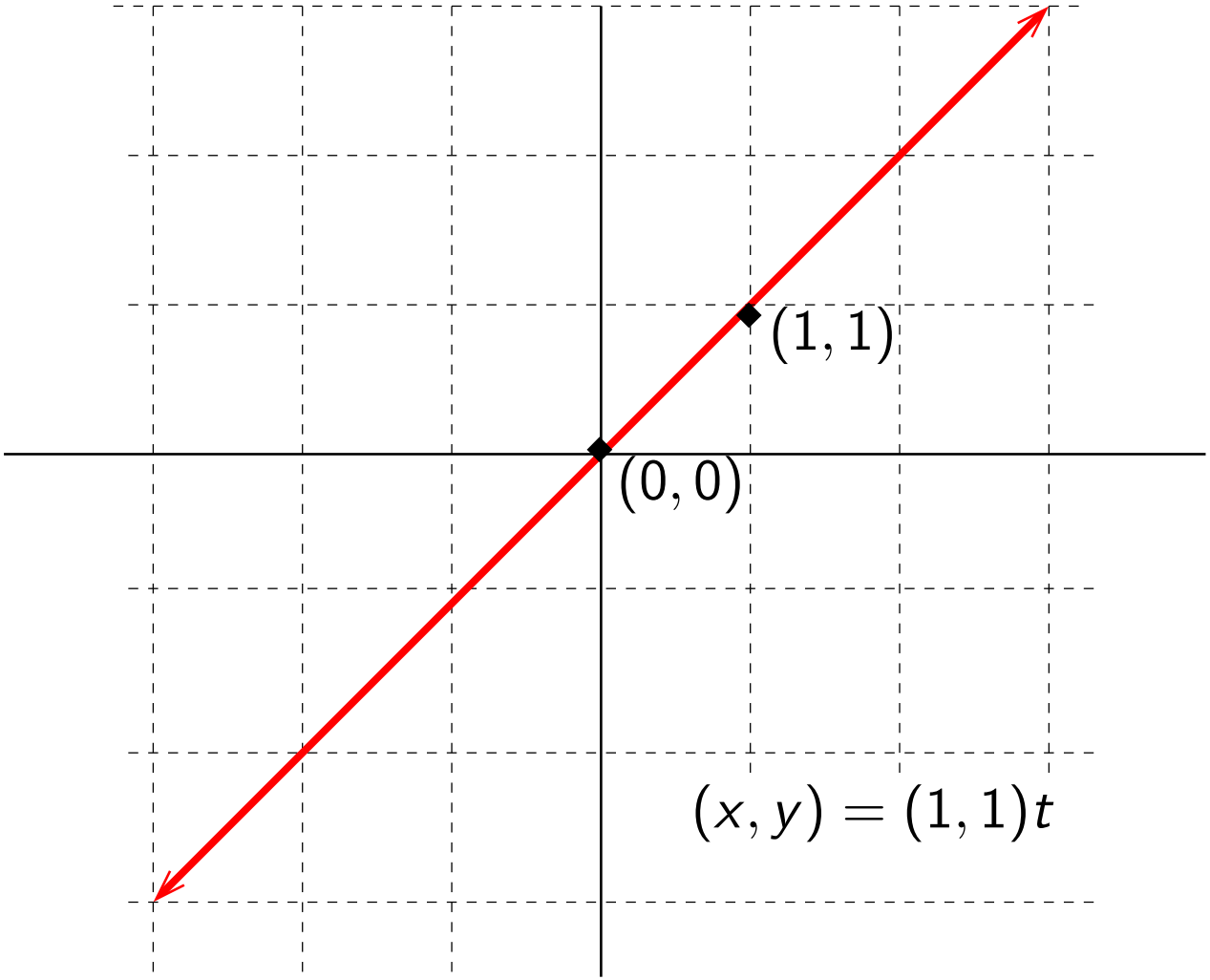


Inputs for State Transfer

- Question Are there more than one inputs for state transfer?
- Answer Yes.
- Question Then how to compute other inputs?
- Answer Think of optimal control
- Question Are there simpler techniques?
- Answer Yes! Use parameterisation
- Question But what is parameterisation?

Parameterisation of a straight line through (1,1) and (0,0)

What does parameterisation mean?



State Transfer: Example Scalar Case

- Consider a scalar system

$$\dot{x}(t) = 2x(t) + 3u(t)$$

- Suppose it is required that $x(0) = 0.5$ and $x(1) = 2$
- Question: How to compute $u(t)^*$?
- Rewrite the above equation

$$u(t) = \frac{\dot{x}(t) - 2x(t)}{3}$$

- If $x(t)$ is any trajectory of the original system the corresponding input $u(t)$ is given by above equation.
- $u(t)$ has been parameterised in terms of the states of the system!

State Transfer: Example contd.

- Given system $\dot{x}(t) = 2x(t) + 3u(t)$
- Parameterised system $u(t) = \frac{\dot{x}(t) - 2x(t)}{3}$
- Both the system are equivalent
- Requirement $x(0) = 0.5$ and $x(1) = 2$
- Choose any once differentiable funtion $x(t) \in [0, 1]$ which has above property
- One such choice: $x(t) = 0.5 + 1.5t$
- Input for state transfer is given by

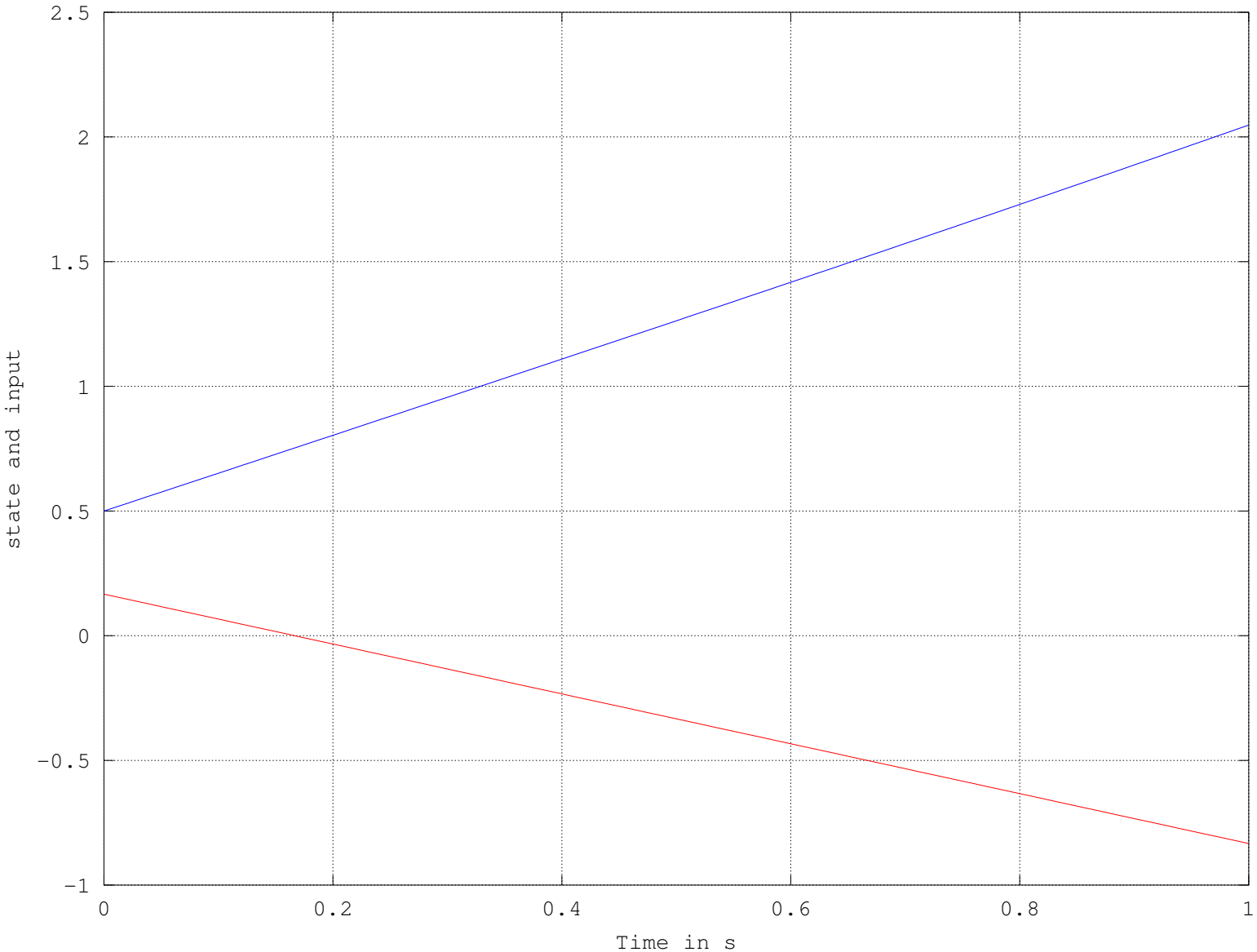
$$u^*(t) = \frac{0.5 - 3t}{3}$$

- Octave results

Matlab Program

```
A = [2]; B = [3]; C = [0]; D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u = 0.5/3-3*t./3;
[Y,T,X] = lsim(sys1,u,t,[0.5 ]);
plot(T,X);
hold on; grid on;
plot(T,u,'r');
ylabel("state and input");
xlabel("Time in s");
```

State Transfer: Parameterised Input: Scalar System



State Transfer: Example Multistate Case

- System Dynamics

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \implies u = \dot{x}_2$$

- Let $x_1(t) = z(t)$, then $x_2(t) = \dot{x}_1(t) = \dot{z}(t)$ and $u(t) = \ddot{z}(t)$
- Above two systems are equivalent.
- Let $\mathbf{x}(0) = [0\ 0]^T$ and $\mathbf{x}(1) = [1\ 0]^T$
- Then $z(0) = x_1(0) = 0$, $\dot{z}(0) = x_2(0) = 0$, $z(1) = x_1(1) = 1$, $\dot{z}(1) = x_2(1) = 0$
- Fit any curve for $z(t)$ which satisfies above boundary conditions.

Example continued

- Let $z(t) = at^3 + bt^2 + ct + d$
- $\dot{z}(t) = 3at^2 + 2bt + c$
- $z(0) = d = 0$ and $\dot{z}(0) = c = 0$
- $z(1) = a + b = 1$ and $\dot{z}(1) = 3a + 2b = 0$
- On solving we get, $a = -2$, $b = 3$
- $z(t) = -2t^3 + 3t^2$
- $u(t) = \ddot{z}(t) = -12t + 6$ will transfer the state $\mathbf{x}(0) = [0\ 0]^T$ to $\mathbf{x}(1) = [1\ 0]^T$ for the system

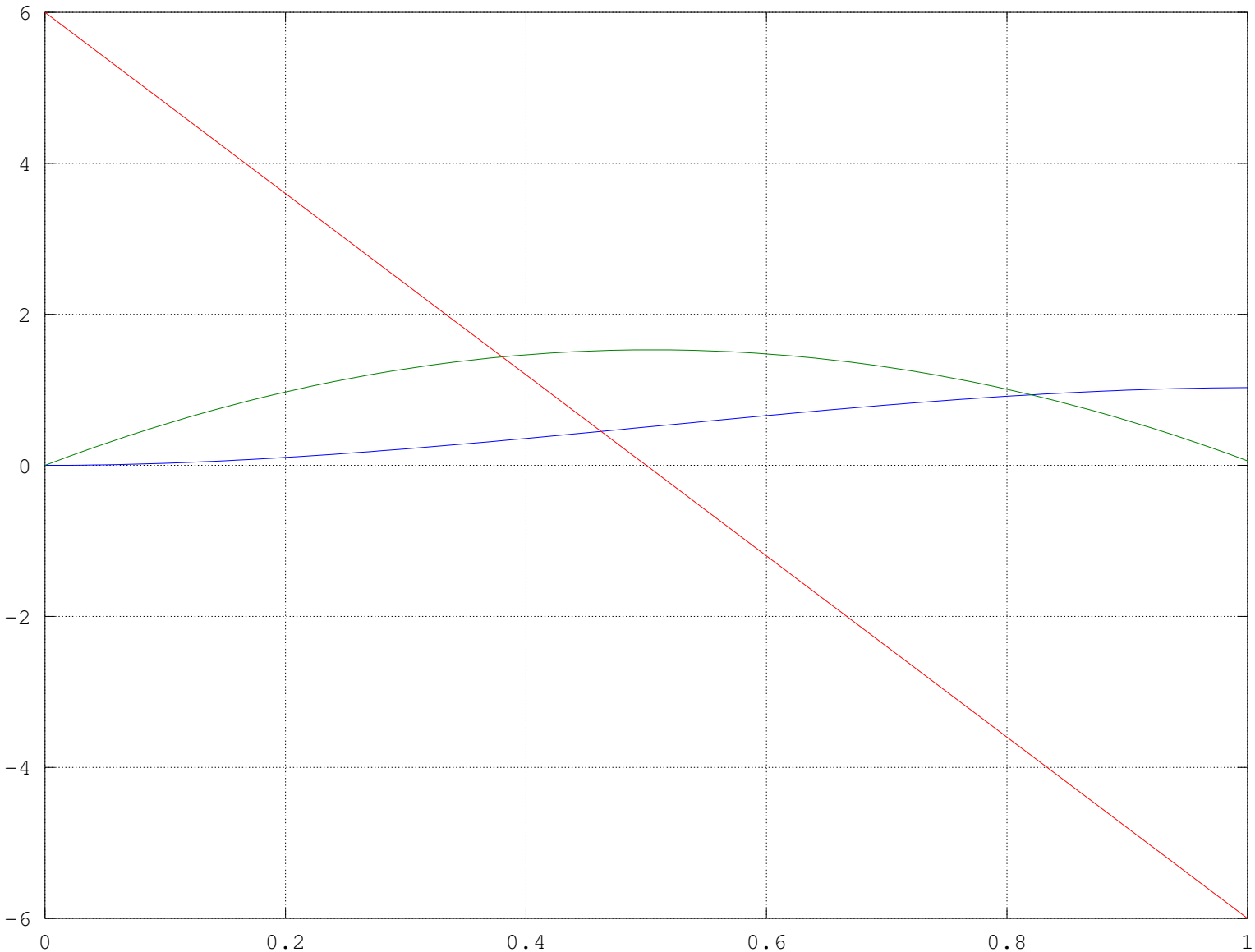
$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

- octave results

Matlab Program

```
A = [0 1; 0 0 ];
B = [0;1];
C = [0 0];
D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u = -12*t + 6;
[Y,T,X]=lsim(sys1,u,t,[0 0]);
plot(T,X);
hold on;
grid on;
plot(T,u,'r');
```

State Transfer: Parameterised Input: Polynomial Input



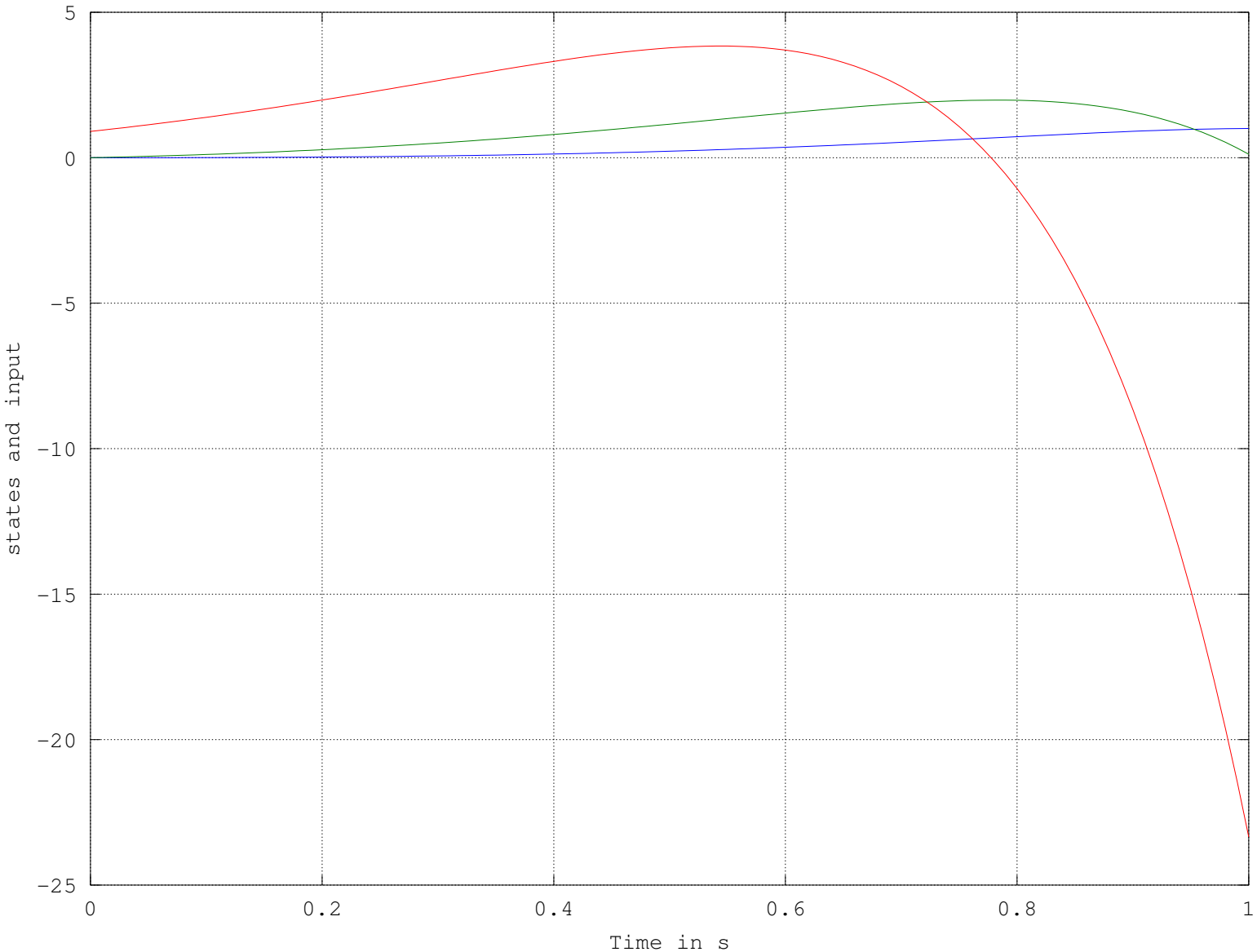
Example continued: Different curve

- Suppose: $z(t) = ae^t + be^{2t} + ce^{3t} + de^{4t}$
- $\dot{z}(t) = ae^t + 2be^{2t} + 3ce^{3t} + 4de^{4t}$
- $z(0) = a + b + c + d = 0$
- $\dot{z}(0) = a + 2b + 3c + 4d = 0$
- $z(1) = ae^1 + be^2 + ce^3 + de^4 = 1$
- $\dot{z}(1) = ae^1 + 2be^2 + 3ce^3 + 4de^4 = 0$
- On solving,
  $a = 0.6434, b = -1.4777, c = 1.0252, d = -0.1909$
- Input for state transfer:
  $u^*(t) = \ddot{z}(t) = 0.6434e^t - 5.9109e^{2t} + 9.2668e^{3t} - 3.0539e^{4t}$
- octave results

Matlab Program

```
M = [1 1 1 1; 1 2 3 4; exp(1) exp(2) exp(3) exp(4);
     exp(1) 2*exp(2) 3*exp(3) 4*exp(4) ];
coef = [1 0 0 0 ; 0 4 0 0 ; 0 0 9 0 ; 0 0 0 16]
        * (inv(M)*[0;0;1;0]);
A = [0 1; 0 0 ]; B = [0;1]; C = [0 0]; D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u = coef'*[exp(t.*1) ; exp(t.*2) ;exp(t.*3) ;exp(t.*4)]
[Y,T,X]=lsim(sys1,u,t,[0 0]);
plot(T,X);
hold on; grid on;
plot(T,u,'r');
xlabel("Time in s");
ylabel("states and input");
```

State Transfer: Parameterised Input: Exponential Input



Example continued: multi-input case

- Consider

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ -2 & -3 & -2 \\ 0 & 0 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- $\dot{x}_1 = x_2$, $\dot{x}_2 = -2x_1 - 3x_2 - 2x_3 + u_1$ and $\dot{x}_3 = -3x_3 + u_2$
- Let $z_1 = x_1$ and $z_2 = x_3$
- Therefore, $x_2 = \dot{x}_1 = \dot{z}_1$ and $\dot{x}_2 = \ddot{z}_1 = -2z_1 - 3\dot{z}_1 - 2z_2 + u_1$.
- Also $\dot{z}_2 = -3z_2 + u_2$.
- Input parameterisation: $u_1 = \ddot{z}_1 + 2z_1 + 3\dot{z}_1 + 2z_2$ and $u_2 = \dot{z}_2 + 3z_2$
- State parameterisation: $x_1 = z_1$, $x_2 = \dot{z}_1$, $x_3 = z_2$
- Parameters are z_1 and z_2

Example continued: multi-input case

- Parameterisation with operators

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} D^2 + 3D + 2 & 2 \\ 0 & D + 3 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ D & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

- Let $\mathbf{x}(0) = [0\ 1\ 0]^T$ and $\mathbf{x}(1) = [1\ 0\ 1]^T$
- Boundary conditions on $\mathbf{z}(t)$

$$z_1(0) = x_1(0) = 0 \qquad \dot{z}_1(0) = x_2(0) = 1 \qquad z_2(0) = x_3(0) = 0$$
$$z_1(1) = x_1(1) = 1 \qquad \dot{z}_1(1) = x_2(1) = 0 \qquad z_2(1) = x_3(1) = 1$$

- Any function $\mathbf{z}(t)$ which satisfies the above boundary conditions can be used to compute the control inputs for state state transfer.

Example continued: multi-input case

- Assume polynomial forms for parameters $z_1(t)$ and $z_2(t)$

$$z_1(t) = at^3 + bt^2 + ct + d$$
$$z_2(t) = et + f$$

- So $\dot{z}_1(t) = 3at^2 + 2bt + c$
- Applying boundary conditions

$$z_1(0) = 0 \implies d = 0$$
$$\dot{z}_1(0) = 1 \implies c = 1$$
$$z_2(0) = 0 \implies f = 0$$
$$z_1(1) = 1 \implies a + b + 1 = 1 \implies a + b = 0$$
$$\dot{z}_1(1) = 0 \implies 3a + 2b + 1 = 0$$
$$z_2(1) = 1 \implies e = 1$$

- Finally, $a = -1$, $b = 1$, $c = 1$, $d = 0$, $e = 1$, $f = 0$

Example continued: multi-input case

- Parameter functions for state transfer

$$z_1(t) = -t^3 + t^2 + t$$
$$z_2(t) = t$$

- Control inputs

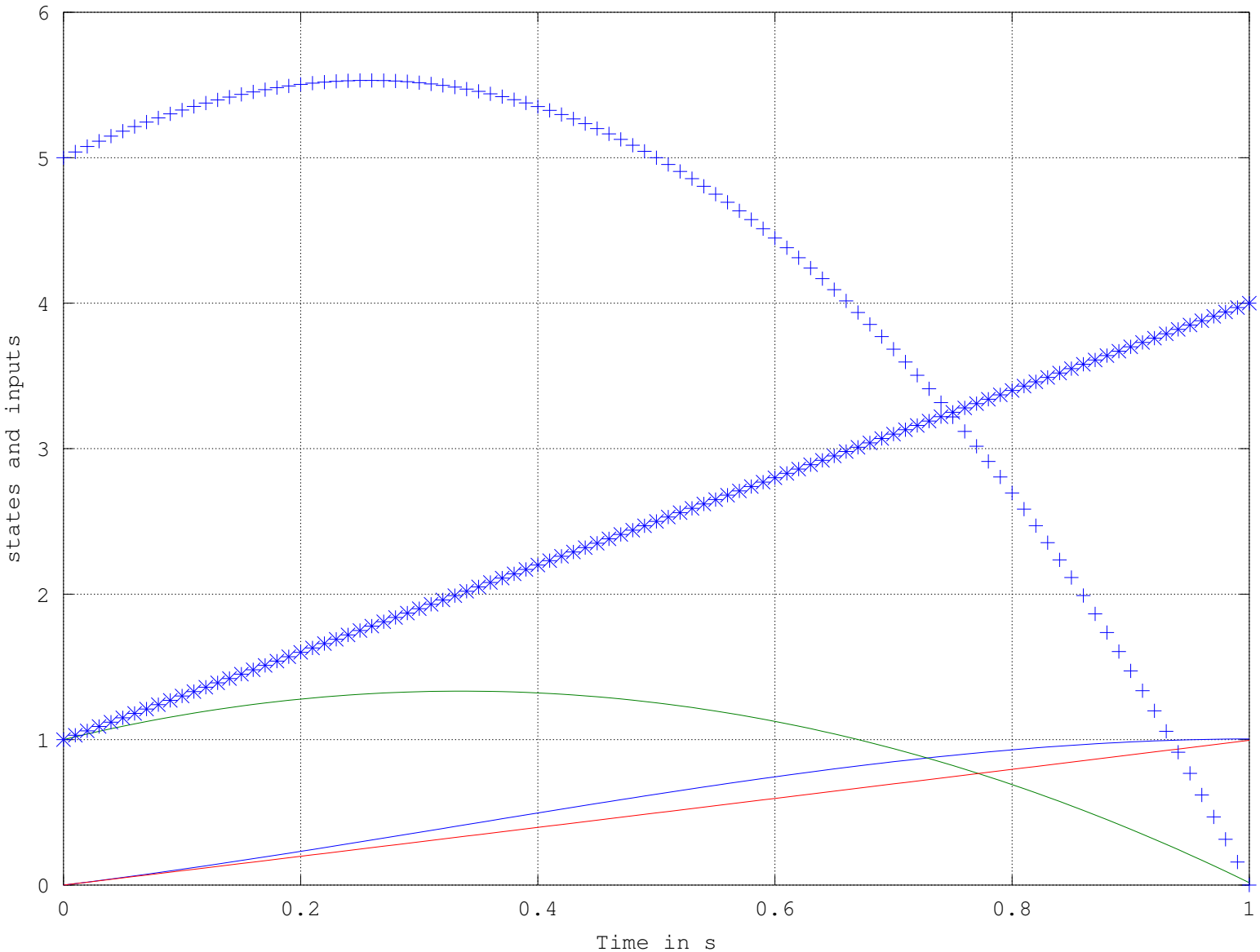
$$\begin{aligned} u_1 &= \ddot{z}_1 + 3\dot{z}_1 + 2z_1 + 2z_2 \\ &= (-6t + 2) + 3(-3t^2 + 2t + 1) + 2(-t^3 + t^2 + t) + 2(t) \\ &= -2t^3 - 7t^2 + 4t + 5 \\ u_2 &= \dot{z}_2 + 3z_2 = 1 + 3t \end{aligned}$$

- octave results

Matlab Program

```
  A = [0 1 0 ; -2 -3 -2; 0 0 -3 ];B = [0 0;1 0;0 1];
  C = [ 0 0 0];D = [0 0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u1 = -2*t.^3 -7*t.^2 + 4*t + 5;
u2 = 1 + 3*t;
[Y,T,X]= lsim(sys1,[u1' u2'],t,[0 1 0]);
plot(T,X);
hold on;
grid on;
plot(T,u1,'+');
plot(T,u2,'*');
xlabel("Time in s");
ylabel("states and inputs");
```

State Transfer: Parameterised Input: Multi Inputs



Example: Uncontrollable System

- Consider

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & -2 & -4 \\ 1 & -3 & -5 \\ 0 & 0 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u$$

- Is this system controllable?
- Expanding: $\dot{x}_1 = -2x_2 - 4x_3 + u$, $\dot{x}_2 = x_1 - 3x_2 - 5x_3$, and $\dot{x}_3 = -3x_3$
- Let $z_1 = x_2$ and $z_2 = x_3$
- From second expression: $x_1 = \dot{z}_1 + 3z_1 + 5z_2$
- From first expression: $u = \dot{x}_1 + 2z_1 + 4z_2$
- Finally: $u = \ddot{z}_1 + 3\dot{z}_1 + 2z_1 - 11z_2$
- State parameterisation: $x_1 = \dot{z}_1 + 3z_1 + 5z_2$, $x_2 = z_1$ and $x_3 = z_2$
- z_1 and z_2 are parameters.

Example: Uncontrollable System: contd.

- Let $x(0) = (0\ 1\ 0)$ and $x(1) = (1\ 0\ 0)$
- We know that $z_1 = x_2$, $z_2 = x_3$ and $x_1 = \dot{z}_1 + 3z_1 + 5z_2$ $\implies \dot{z}_1 = x_1 - 3z_1 - 5z_2$
- $z_1(0) = 1$, $z_2(0) = 0$, $z_1(1) = 0$ and $z_2(1) = 0$
- $\dot{z}_1(0) = -3$ and $\dot{z}_1(1) = 1$
- Assume $z_1(t) = at^3 + bt^2 + ct + d$
- Applying boundary conditions

$$z_1(0) = 1 \implies d = 1$$
$$\dot{z}_1(0) = -3 \implies c = -3$$
$$z_1(1) = 0 \implies a + b - 3 + 1 = 1 \implies a + b = 2$$
$$\dot{z}_1(1) = 1 \implies 3a + 2b + 1(-3) = 1 \implies 3a + 2b = 4$$

- Finally, $a = 0$, $b = 2$, $c = -3$, $d = 1$,

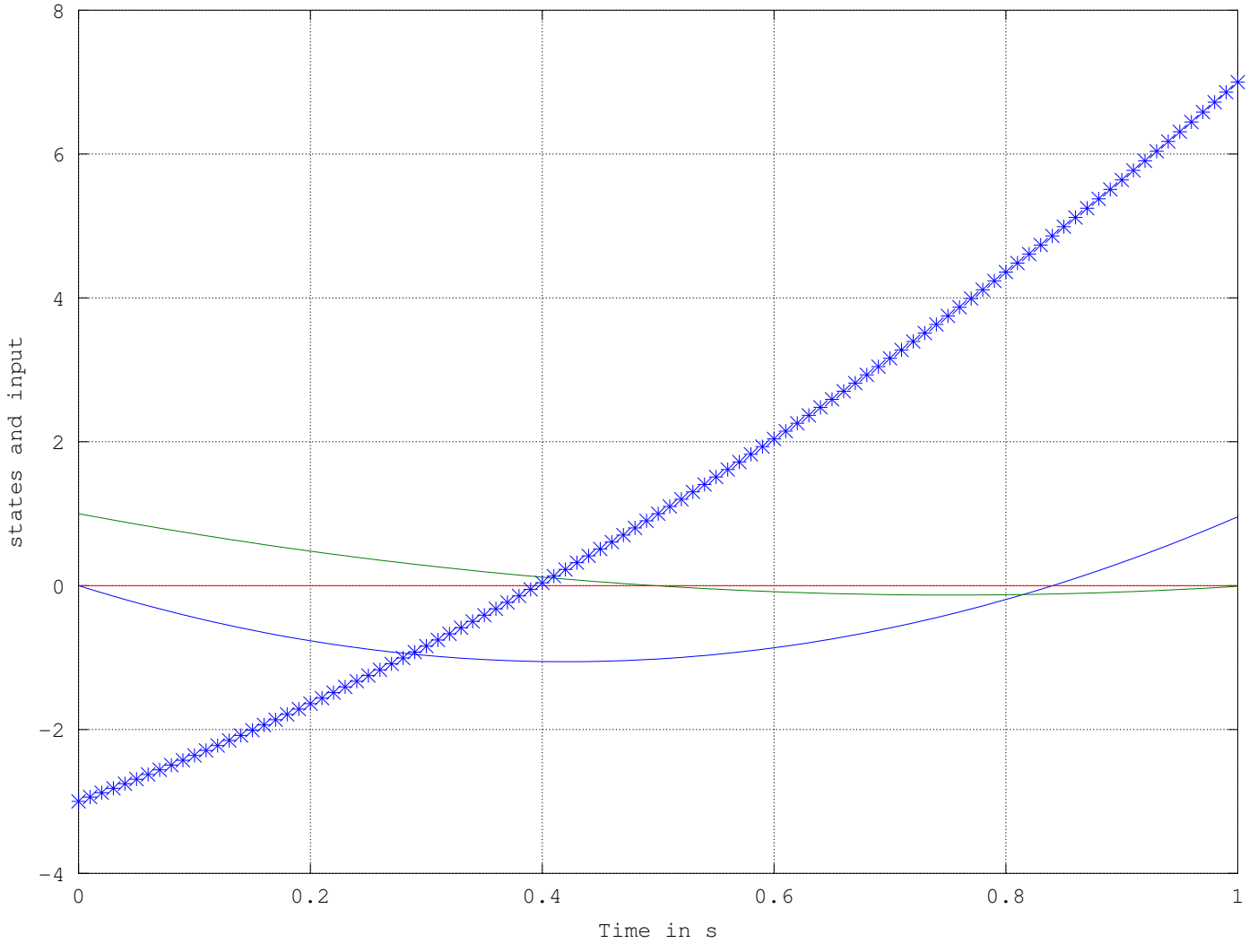
Example Uncontrollable System: contd.

- Parameters $z_1(t) = 2t^2 - 3t + 1$ and $z_2(t) = 0$
- $u(t) = \ddot{z}_1 + 3\dot{z}_1 + 2z_1 - 11z_2$
- $u(t) = 4t^2 + 6t - 3$
- Octave simulations

Matlab Program

```
A = [0 -2 -4; 1 -3 -5; 0 0 -3 ];B = [1;0;0];
C = [0 0 0];D = [0];
sys1 = ss(A,B,C,D);
t = 0:0.01:1;
u = t.^2*4 + t*6 - 3;
[Y,T,X]=lsim(sys1,u,t,[0 1 0]);
plot(T,X);
hold on;
grid on;
plot(T,u,'*');
xlabel("Time in s");
ylabel("states and input");
```

State Transfer: Parameterised Input: Uncontrollable System



Theory and Proofs

Parameterisation Problem

- Consider $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}(t))$
- How to parameterise the solutions of above system?
- That is, when can we express $\mathbf{x}(t)$ and $\mathbf{u}(t)$ in terms of some arbitrary function $\mathbf{z}(t)$?
- Obtain conditions under which

$$\mathbf{x}(t) = \kappa(\mathbf{z}(t))$$
$$\mathbf{u}(t) = \phi(\mathbf{z}(t))$$

where $\mathbf{z}(t)$ is arbitrary so that

- For the initial state $\mathbf{x}(0) = \kappa(\mathbf{z}(0))$
- The function $\mathbf{x}(t) = \kappa(\mathbf{z}(t))$ is the solution of $\dot{\mathbf{x}}(t) = F\big(\mathbf{x}(t), \mathbf{u}(t)\big)$ for the input $\mathbf{u}(t) = \phi(\mathbf{z}(t))$

Applications

Question Where can parameterisation be used?

- State Transfer Problem
  - all inputs to achieve desired state transfer
  - no solution of differential equation is involved
- Optimal Control Problem
  - without invoking Lagrange-multipliers calculus of variations problem is obtained
  - no solution of matrix Riccati differential equations is involved
- Computation of Image Representation in Behavioural Systems Theory

State Transfer Problem

- Dynamical System: $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}(t))$
- initial state $= \mathbf{x_0}$, final state $= \mathbf{x_f}$
- Find $\mathbf{u}^*(t)$ for $t \in [t_0, t_f]$ such that when $\mathbf{x}(t_0) = \mathbf{x_0}$, the solution of $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}^*(t))$ is such that $\mathbf{x}(t_f) = \mathbf{x_f}$
- Claim: If parameterisation exists then it is easy to compute input $\mathbf{u}^*$ which achieves the desired state transfer
- Parameterisation helps to characterise every input $\mathbf{u}^*$ which achieves the desired state transfer

Inputs for State Transfer using parameterisation

- Given System: $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}(t))$
- Assume Parameterisation is available

$$\mathbf{x}(t) = \kappa(\mathbf{z}(t)) \qquad \kappa \text{ is invertible}$$
$$\mathbf{u}(t) = \phi(\mathbf{z}(t))$$

- Requirement: Initial state $\mathbf{x}(t_0) = \mathbf{x_0}$ — Final state $\mathbf{x}(t_f) = \mathbf{x_f}$
- Solution: Set $\mathbf{z}(t_0) = \kappa^{-1}(\mathbf{x_0})$ and $\mathbf{z}(t_f) = \kappa^{-1}(\mathbf{x_f})$
- Choose (by interpolation) any (sufficiently differentiable) function $\mathbf{z}^*(t)$ satisfying the boundary conditions
- Set $\mathbf{u}^*(t) = \phi(\mathbf{z}^*(t))$
- Then with $\mathbf{x}(t_0) = \mathbf{x_0}$, the solution of $\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{u}^*(t))$ satisfies $\mathbf{x}(t_f) = \mathbf{x_f}$

Canonical form and Parameterisation (single input)

- LTI Canonical form

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} 0 & 0 & \dots & 0 & a_1 & b_{k+1,1} & \dots & b_{n,1} \\ 1 & 0 & \dots & 0 & a_2 & b_{k+1,2} & \dots & b_{n,2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & a_k & b_{k+1,k} & \dots & b_{n,k} \\ 0 & 0 & \dots & 0 & 0 & b_{k+1,k+1} & \dots & b_{n,k+1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & 0 & b_{k+1,n} & \dots & b_{n,n} \end{bmatrix} \mathbf{z}(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(t)$$

- Input can be rewritten as

$$u(t) = \frac{d^k}{dt} z_k(t) - a_k \frac{d^{k-1}}{dt^{k-1}} z_k(t) - \dots - a_1 z_k(t) - f(z_{k+1}(t), \dots, z_n(t))$$

- $z_j(t) = f_j(z_k, \dots, \frac{d^{k-1}}{dt^{k-1}} z_k(t), z_{k+1}(t), \dots, z_n(t))$, for $j = 1, \dots, k-1$
- State: $\mathbf{x}(t) = T\mathbf{z}(t)$

Controllable Systems and Parameterisation

- Controllable systems $k = n$

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} 0 & 0 & \dots & 0 & a_1 \\ 1 & 0 & \dots & 0 & a_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & a_n \end{bmatrix} \mathbf{z}(t) + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(t)$$

- Input Parameterisation

$$u(t) = \frac{d^n}{dt^n} z_n(t) - a_n \frac{d^{n-1}}{dt^{n-1}} z_n(t) - \dots - a_1 z_n(t)$$

- State Parameterisation

$$z_j(t) = \frac{d^{n-j}}{dt^{n-j}} z_n(t) - \sum_{i=0}^{n-j-1} a_{(2+(j-1)+i)} \frac{d^i}{dt^i} z_n(t) \qquad j = 1, \dots, n-1$$

- Both input and state variables are parameterised by a single scalar function z_n

Controllable System Parameterisation

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} 0 & 0 & \dots & 0 & a_1 \\ 1 & 0 & \dots & 0 & a_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & a_n \end{bmatrix} \mathbf{z}(t) + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(t)$$

- Characteristic polynomial $p(s) = s^n - a_n s^{n-1} - \dots - a_1$

$$u(t) = (D^n - a_n D^{n-1} - \dots - a_1) z_n(t)$$

$$\mathbf{z}(t) = \begin{bmatrix} D^{n-1} - a_n D^{n-2} - \dots - a_2 \\ D^{n-2} - a_n D^{n-3} - \dots - a_3 \\ \vdots \\ D - a_n \\ 1 \end{bmatrix} z_n(t)$$

Arbitrary Controllable System

- Consider the system

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{b}u(t),$$

- Let $\mathbf{z}(t) = T\mathbf{x}(t)$, T non-singular
- Transformed system

$$\dot{\mathbf{z}}(t) = TAT^{-1}\mathbf{z}(t) + T\mathbf{b}u(t).$$

- Complete parameterisation

$$u(t) = p(D)z_n(t)$$

$$\mathbf{x}(t) = T^{-1}\begin{bmatrix} D^{n-1} - a_nD^{n-2} - \ldots - a_2 \\ D^{n-2} - a_nD^{n-3} - \ldots - a_3 \\ \vdots \\ D - a_n \\ 1 \end{bmatrix} z_n(t)$$

Choosing the basis

- How to choose a basis such that a controllable system is transformed into its canonical form?
- For controllable system the following matrix has rank n

$$\begin{bmatrix}\mathbf{b} & A\mathbf{b} & A^2\mathbf{b} & \dots & A^{n-1}\mathbf{b}\end{bmatrix}$$

- We choose the above vectors as basis, that is,

$$\mathbf{x}(t) = \mathbf{b}w_1(t) + A\mathbf{b}w_2(t) + A^2\mathbf{b}w_3(t) + \ldots + A^{n-1}\mathbf{b}w_n(t)$$

- Differentiating

$$\dot{\mathbf{x}}(t) = \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)$$

Choosing the basis

- After differentiating

$$\dot{\mathbf{x}}(t) = \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)$$

- Comparing with original system dynamics

$$A\mathbf{x}(t) + \mathbf{b}u(t) = \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)$$

- Substituting for $\mathbf{x}(t)$

$$A\left(\mathbf{b}w_1(t) + A\mathbf{b}w_2(t) + A^2\mathbf{b}w_3(t) + \ldots + A^{n-1}\mathbf{b}w_n(t)\right) + \mathbf{b}u(t)$$
$$= \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)$$

Choosing the basis

- Continuing the simplification

$$A\mathbf{b}w_1(t) + A^2\mathbf{b}w_2(t) + A^3\mathbf{b}w_3(t) + \ldots + A^n\mathbf{b}w_n(t) + \mathbf{b}u(t)$$
$$= \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)$$

- $\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots, A^{n-1}\mathbf{b}$ are linearly independent implies

$$A^n\mathbf{b} = a_1\mathbf{b} + a_2A\mathbf{b} + a_3A^2\mathbf{b} + \ldots + a_nA^{n-1}\mathbf{b}$$

- On substitution

$$A\mathbf{b}w_1(t) + A^2\mathbf{b}w_2(t) + \ldots$$
$$+ \left(a_1\mathbf{b} + a_2A\mathbf{b} + a_3A^2\mathbf{b} + \ldots + a_nA^{n-1}\mathbf{b}\right) w_n(t) + \mathbf{b}u(t)$$
$$= \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)$$

Choosing the basis

- Collecting terms

$$
\begin{aligned}
&A\mathbf{b}\left(w_1(t) + a_2 w_n(t)\right) + \ldots \\
&+ A^{n-1}\mathbf{b}\left(w_{n-1}t + a_n w_n(t)\right) + \mathbf{b}\left(u(t) + a_1 w_n(t)\right) \\
&= \mathbf{b}\dot{w}_1(t) + A\mathbf{b}\dot{w}_2(t) + A^2\mathbf{b}\dot{w}_3(t) + \ldots + A^{n-1}\mathbf{b}\dot{w}_n(t)
\end{aligned}
$$

- Comparing the co-coefficients $\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots, A^{n-1}\mathbf{b}$

$$
\begin{aligned}
\dot{w}_1(t) &= u(t) + a_1 w_n(t) \\
\dot{w}_2(t) &= w_1(t) + a_2 w_n(t) \\
&\vdots \\
\dot{w}_n(t) &= w_{n-1}(t) + a_n w_n(t)
\end{aligned}
$$

Choosing the basis

- Our initial choice was

$$\mathbf{x}(t) = \mathbf{b}w_1(t) + A\mathbf{b}w_2(t) + A^2\mathbf{b}w_3(t) + \ldots + A^{n-1}\mathbf{b}w_n(t)$$

- If $\mathbf{w}(t) = T\mathbf{x}(t)$ we have

$$\dot{\mathbf{w}}(t) = TAT^{-1}\mathbf{w}(t) + T\mathbf{b}u(t)$$

- Therefore,

$$T^{-1} = \begin{bmatrix}\mathbf{b} & A\mathbf{b} & A^2\mathbf{b} & \ldots & A^{n-1}\mathbf{b}\end{bmatrix}$$

- and the system appears in canonical form

$$\dot{\mathbf{w}}(t) = \begin{bmatrix} 0 & 0 & \ldots & 0 & a_1 \\ 1 & 0 & \ldots & 0 & a_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & a_n \end{bmatrix} \mathbf{w}(t) + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(t)$$

Adjoint method for parameterisation

- Consider the system

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{b}u(t),$$

- Replacing the "dot" operator with D

$$DI\mathbf{x}(t) = A\mathbf{x}(t) + \mathbf{b}u(t)$$

- where DI is

$$\begin{bmatrix} \frac{d}{dt} & 0 & \dots & 0 \\ 0 & \frac{d}{dt} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{d}{dt} \end{bmatrix}$$

Adjoint method for parameterisation

- Collecting $\mathbf{x}(t)$ on LHS

$$DI\mathbf{x}(t) - A\mathbf{x}(t) = \mathbf{b}u(t).$$

- On simplification

$$(DI - A)\mathbf{x}(t) = \mathbf{b}u(t).$$

- Further Simplification

$$\mathbf{x}(t) = (DI - A)^{-1}\mathbf{b}u(t).$$

Adjoint method for parameterisation

- But

$$(DI - A)^{-1} = \frac{\mathrm{adj}(DI - A)}{p(D)},$$

- $p(D)$ is the characteristic polynomial of matrix A in the indeterminate (differential operator) D
- Suppose

$$u(t) = p(D)z(t),$$

- Then

$$\mathbf{x}(t) = (DI - A)^{-1}\mathbf{b}u(t) = \mathrm{adj}(DI - A)bz(t)$$

Computing the Adjoint

- Substituting the transformation T^{-1}

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{b} & A\mathbf{b} & A^2\mathbf{b} & \dots & A^{n-1}\mathbf{b} \end{bmatrix} \begin{bmatrix} p_1(D) \\ p_2(D) \\ \vdots \\ p_{n-1}(D) \\ 1 \end{bmatrix} z_n(t)$$

- Expanding

$$\mathbf{x}(t) = \mathbf{b}p_1(D)z_n(t) + A\mathbf{b}p_2(D)z_n(t) + \ldots + A^{n-2}\mathbf{b}p_{n-1}(D)z_n(t) + A$$

Or

$$\mathbf{x}(t) = \begin{bmatrix} p_1(D)I & p_2(D)A & \dots & p_{n-1}(D)A^{n-2} & A^{n-1} \end{bmatrix} \mathbf{b}z_n(t)$$

Computing the Adjoint

- Therefore

$$\mathrm{adj}(DI - A) = A^{n-1} + p_{n-1}(D)A^{n-2} + \ldots + p_2(D)A + p_1(D)I.$$

- Alternatively,

$$\mathrm{adj}(DI - A) = D^{n-1}I + D^{n-2}p_{n-1}(A) + \ldots + Dp_2(A) + p_1(A)$$

Example: Parameterisation using adjoints

- Consider

$$\dot{\mathbf{x}} = \begin{bmatrix} 5 & 9 \\ -2 & -4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u$$

- Characteristic and associated polynomials

$$\begin{aligned} p(s) &= s^2 - s - 2 \\ p_2(s) &= p(s) \\ p_1(s) &= s - 1 \\ p_0(s) &= 1 \end{aligned}$$

Example: Parameterisation using adjoints

- Adjoint

$$\mathrm{adj}(DI - A) = DI + (A - I) = \begin{bmatrix} D+4 & 9 \\ -2 & D-5 \end{bmatrix}$$

or alternatively,

$$\mathrm{adj}(DI - A) = A + (D-1)I = \begin{bmatrix} D+4 & 9 \\ -2 & D-5 \end{bmatrix}$$

- Parameterisation of states

$$\mathbf{x}(t) = \mathrm{adj}(DI - A)\mathbf{b}z(t) = \begin{bmatrix} 2D-1 \\ -D+1 \end{bmatrix} z(t)$$

- Input parameterisation

$$u(t) = p(D)z(t) = (D^2 - D - 2)z(t)$$

Canonical form and Parameterisation (multi input)

- Consider

$$\Sigma : \qquad \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$

- For every Σ there exists a non-singular state and input transformations T and V, respectively, such that, when $\mathbf{x}(t) = T\mathbf{z}(t)$ and $\mathbf{u}(t) = V\mathbf{v}(t)$,

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} 0 & I_{m1} & 0 & \dots & 0 & 0 \\ * & * & * & \dots & * & * \\ 0 & 0 & I_{m2} & \dots & 0 & 0 \\ * & * & * & \dots & * & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I_{mk} & 0 \\ * & * & * & \dots & * & * \\ 0 & 0 & 0 & \dots & 0 & A_{k+1} \end{bmatrix} \mathbf{z}(t) + \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & 0 \end{bmatrix} \mathbf{v}(t),$$

Canonical form and Parameterisation (multi input)

- Scalar linear differential equations

$$\mathbf{z}_j^{\mathbf{i}}(t) = D^{j-1}\mathbf{z}_1^{\mathbf{i}}(t) \qquad j = 2, \ldots, m_i \qquad i = 1, \ldots, k$$

$$D^{m_i}\mathbf{z}_1^{\mathbf{i}}(t) = \mathbf{v}_i(t) + \sum_{l=1}^{k} \left(\sum_{j=0}^{m_i - 1} \alpha_{ijl} D^j \mathbf{z}_1^l(t) \right) + \sum_{j=1}^{m_{k+1}} \beta_{ij} \mathbf{z}_j^{\mathbf{k+1}}(t)$$

$$i = 1, \ldots, k$$

$$\dot{\mathbf{z}}^{\mathbf{k+1}}(t) = A^{k+1}\mathbf{z}^{\mathbf{k+1}}(t)$$

Canonical form and Parameterisation (multi input)

- Parameterisation of input

$$\mathbf{v}_i(t) = D^{m_i}\mathbf{z}_1^{\mathbf{i}}(t) - \sum_{l=1}^{k}\left(\sum_{j=0}^{m_i-1}\alpha_{ijl}D^j\mathbf{z}_1^l(t)\right) - \sum_{j=1}^{m_{k+1}}\beta_{ij}\mathbf{z}_j^{\mathbf{k+1}}(t)$$

$$i = 1, \ldots, k.$$

Thank You!