2MAE404 MIMO control
Homework report 3
(with later improvements)

Maciej Michałów Marco Wijaya

December 2, 2022

1 Linearisation

The goal of this excercise is to provide a linearised model of the dynamics of the Flying Chardonnay drone. Just like in homework no.1, the state is $x = [v_n, v_d, \theta, \dot{\theta}, \gamma, \dot{\gamma}]^T$, and the input $u = [T_1, T_2]^T$ is the thrust on both propellers. Additionally, the drone is affected by wind with a velocity $w = [w_n, w_d]^T$.

In homework no. 1, the full nonlinear dynamics model was implemented. It can be denoted as:

$$\dot{x}(t) = f(x, u, w) \tag{1}$$

The parameters of the problem are:

```
%% Problem parameters

% drone physical parameters
drone.mass = 1; % mass of drone md = 1 kg
drone.cupmass = 1; % mass of cupp mc = 1 kg
drone.l = 1; % length of rod l = 1 m
drone.l_d = 1; % distance to propeller ld = 1 m
drone.J = 1; % drone moment of inertia J = 1 kg*m^2
drone.CD = 0.01; % drone drag coefficients CD = 0.01
drone.g = 10; % gravity [m/s^]

% wind vector
wind = [0;0];

% parameters defining the control objectives
target.rd = 0;
target.rn = 4;

T_hover = (drone.mass+drone.cupmass)*drone.g;
```

The simplified model is to be linearised with respect to the trim point associated with static hovering. The parameters of this point are: a zero state $x_0 = [0, 0, 0, 0, 0, 0]^T$, zero

wind $w_0 = [0, 0]^T$ and thrust that balances the drone's weight: $u_0 = g(m_d+m_c)\cdot[0.5, 0.5]^T$. This is coded as:

```
n = 6; % dimension of state vector

u_trim = [T_hover/2; T_hover/2];
x_trim = zeros(n,1);
w_trim = [0; 0];
```

The linearised model should be of the form:

$$\dot{x}(t) = A\Delta x + B\Delta u + E\Delta w = Ax + B(u - u_0) + Ew \tag{2}$$

The matrices are computed numerically, using the complex step method, which can be expressed using the equation:

$$\frac{dg}{dx}(x) \approx \text{Im}\left(\frac{g(x + j\Delta x)}{\Delta x}\right) \tag{3}$$

The matrices can be defined as:

$$A = J(f(x, u_0, w_0) = \left[\frac{\partial f}{\partial x_1}(x_0, u_0, w_0); \cdots ; \frac{\partial f}{\partial x_6}x_0, u_0, w_0)\right] \tag{4}$$

$$B = J(f(x_0, u, w_0) = \left[\frac{\partial f}{\partial u_1}x_0, u_0, w_0); \frac{\partial f}{\partial u_2}x_0, u_0, w_0)\right] \tag{5}$$

$$E = J(f(x_0, u_0, w) = \left[\frac{\partial f}{\partial w_n}x_0, u_0, w_0); \frac{\partial f}{\partial w_d}x_0, u_0, w_0)\right] \tag{6}$$

Each of the columns can be computed by a single application of the complex step method:

```
A = zeros(n,n);
B = zeros(n,2);
E = zeros(n,2);

epsilon = 1e-6;

% A matrix
for i = 1:n
    dx = zeros(n,1);
    dx(i) = epsilon*1j;
    A(:,i) = imag(drone_dynamics(x_trim+dx,u_trim,drone, w_trim))/epsilon;
end

% B matrix
for i = 1:2
    du = [0; 0];
    du(i) = epsilon*1j;
    B(:,i) = imag(drone_dynamics(x_trim,u_trim+du,drone, w_trim))/epsilon;
end
```

```
% E matrix
for i = 1:2
    dw = [0; 0];
    dw(i) = epsilon*1j;
    E(:,i) = imag(drone_dynamics(x_trim,u_trim,drone, w_trim+dw))/epsilon;
end
```

The resulting matrices are:

$$A = \begin{bmatrix} -0.01 & 0 & -10 & 0 & 10 & 0 \\ 0 & -0.005 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -0.01 & 0 & 0 & 0 & 20 & 0 \end{bmatrix} \tag{7}$$

$$B = \begin{bmatrix} 0 & 0 \\ -0.5 & -0.5 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \\ 1 & -1 \end{bmatrix} \tag{8}$$

$$E = \begin{bmatrix} 0.01 & 0 \\ 0 & -0.005 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.01 & 0 \end{bmatrix} \tag{9}$$

2 Phase planes and linearisation

Consider a pendulum with the following equations of motion (with numerical values already substituted):

$$\frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ -10\sin\theta - \dot{\theta} \end{bmatrix} \tag{10}$$

2.1 Phase portraits around the $(\theta, \dot{\theta}) = (0, 0)$ trim point

The following MATLAB code generates the phase portrait:

```
%% nonlinear phase portrait at (0,0)

theta = linspace(-pi, pi, 25);
theta_dot = linspace(-2*pi, 2*pi, 25);
[x,y]=meshgrid(theta, theta_dot); % grid of points around (0,0)
u = y; % theta'
v = -10.*sin(x) - y; % theta"
```

```
E = -10*cos(x) + y.^2; % potential + kinetic energy

figure;
quiver(x,y,u,v,1.5,'k'); % phase plot
hold on;
contour(x, y, E % energy contours
axis tight;
title('Phase portrait of nonlinear dynamics around trim point (0,0)');
xlabel('\theta [rad]');
ylabel('d\theta/dt [rad/s]');
xticks([-pi, -pi/2, 0, pi/2, pi]);
xticklabels({'-\pi', '-\pi/2', '0', '\pi/2', '\pi'});
yticks([-2*pi, -pi, 0, pi, 2*pi]);
yticklabels({'-pi', '-\pi/2', '0', '\pi/2', '\pi'});
hold off;
```

The resulting phase portrait is shown in Figure 1.

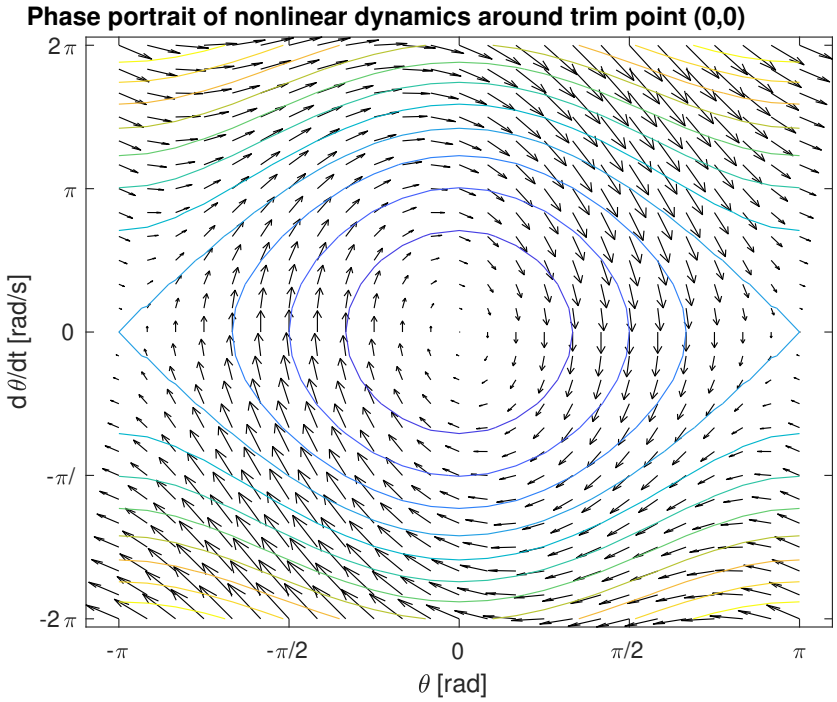

Figure 1: Phase portrait of nonlinear dynamics around the trim point $(\dot{\theta}, \theta) = (0, 0)$. The colored lines correspond to constant energy values.

The dynamics of θ are always linear; however, this is not true for $\dot{\theta}$. The dynamics of $\dot{\theta}$ around the trim point $(\dot{\theta}, \theta) = (0, 0)$ are given by the equation:

$$\ddot{\theta} = -10\theta - \dot{\theta} \tag{11}$$

The phase portrait is generated by using the same MATLAB code as previously, only with a modified equation for $\ddot{\theta}$:

```
v = -10.*x - y; % theta"
```

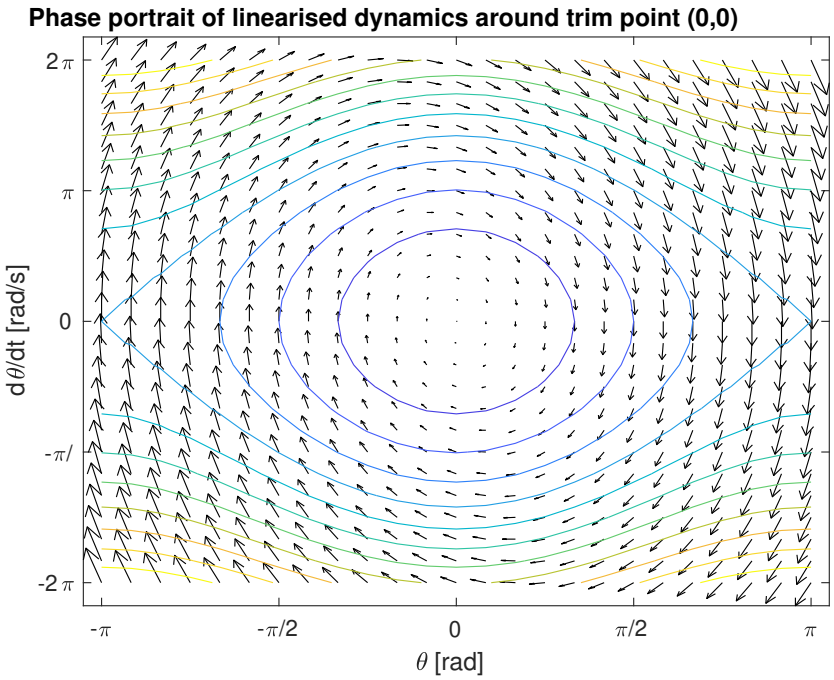

Figure 2: Phase portrait of linear dynamics around the trim point $(\dot{\theta}, \theta) = (0, 0)$. The colored lines correspond to constant energy values.

The resulting phase portrait is shown in Figure 2. As expected, the behaviour of both systems is almost identical close to the trim point. However, further away the behaviour of the linear system becomes less accurate. Most notably, the unstable stationary points, corresponding to a vertical upwards position of the pendulum, are not modelled at all. Furthermore, the system can spontaneously gain mechanical energy – some phase flow vectors cross the constant energy lines towards higher energy levels, which is not physical. Note that the nonlinear system in Figure 1 does not exhibit neither of the behaviours.

2.2 Phase portraits around the $(\theta, \dot{\theta}) = (\pi, 0)$ trim point

The pendulum has also an unstable trim point at $(\dot{\theta},\theta) = (\pi,0)$. The phase portraits around this point can be generated the same way as previously:

```
theta = linspace(0, 2*pi, 25);
theta_dot = linspace(-2*pi, 2*pi, 25);
[x,y]=meshgrid(theta, theta_dot); % grid of points around (pi,0)
u = y; % theta'
v = -10.*sin(x) - y; % theta"
E = -10*cos(x) + y.^2; % potential + kinetic energy

figure;
quiver(x,y,u,v,1.5,'k'); % phase portrait
hold on;
contour(x, y, E); % energy contours
axis tight;
title('Phase portrait of nonlinear dynamics around trim point (\pi,0)');
```

```
xlabel('\theta [rad]');
ylabel('d\theta/dt [rad/s]');
xticks([0, pi/2, pi, 3*pi/2, 2*pi]);
xticklabels({'0', '\pi/2', '\pi', '3\pi/2', '2\pi'});
yticks([-2*pi, -pi, 0, pi, 2*pi]);
yticklabels({'-2\pi', '-\pi/', '0', '\pi', '2\pi'});
hold off;
```

The nonlinear phase portrait is shown in Figure 3.

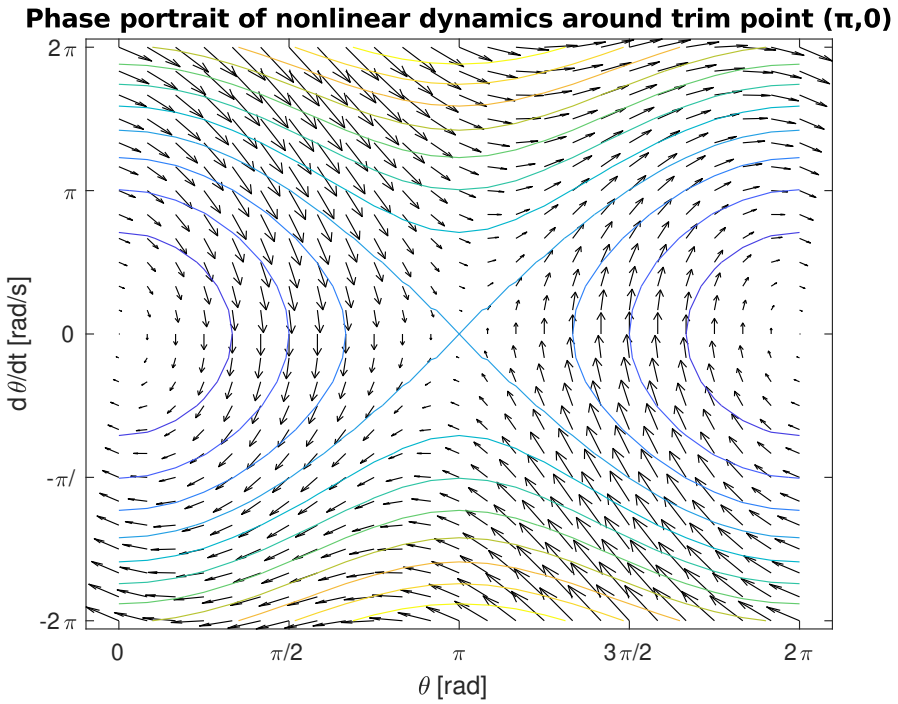

Figure 3: Phase portrait of nonlinear dynamics around the trim point $(\dot{\theta}, \theta) = (\pi, 0)$. The colored lines correspond to constant energy values.

For the linearised case, the only difference is the equation for $\ddot{\theta}$:

```
v = -10.*(pi-x) - y; % linearised theta"
```

The resulting phase portrait is shown in Figure 4.

The differences between both phase portraits are similar to the ones discussed for the previous trim point – the linearised model does not conserve mechanical energy and does not model other stationary points, even if they are in fact attractors of the dynamics (which is the case for the $(0, 0)$ point).

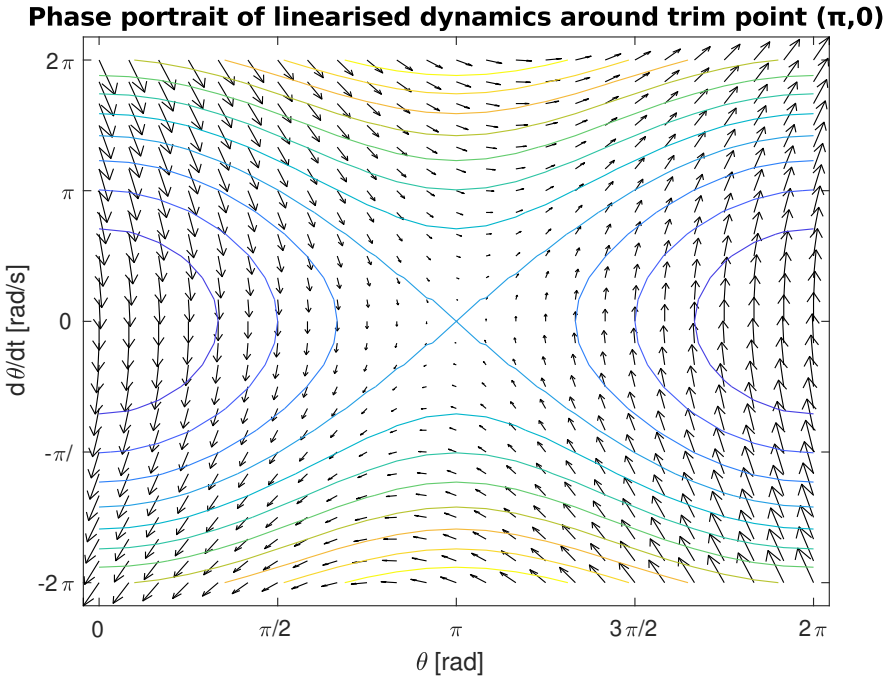

Figure 4: Phase portrait of linear dynamics around the trim point $(\dot{\theta}, \theta) = (\pi, 0)$. The colored lines correspond to constant energy values.

3 Phase portraits and regions of attraction

Consider a nonlinear system described by the equation:

$$\dot{x}(t) = \alpha x^3(t) + u(t) \tag{12}$$

The system is controlled using a linear PD controller (whose goal is to stabilise the system at $x = 0$:

$$u(t) = -k_p x(t) - k_d \dot{x}(t) \tag{13}$$

The closed loop system is therefore:

$$\dot{x}(t) = \alpha x^3(t) - k_p x(t) - k_d \dot{x}(t) \tag{14}$$

This can be easily brought back to canonical form:

$$\dot{x} = \frac{\alpha}{1 + k_d} x^3(t) - \frac{k_p}{1 + k_d} x(t) = f(x) \tag{15}$$

At $x = 0$, $f(x) = 0$, as expected (the controller does not push the state away from the trim point $x = 0$). However, due to the nonlinearity of the dynamics, the stability is only local. The system is locally stable at a given x with respect to the trim point if its dynamics at that x push it back towards $x = 0$, i.e. if the time derivative of the state, $\dot{x}(t) = f(x)$, has an opposite sign to x itself. This is equivalent to the condition:

$$f(x)x < 0 \tag{16}$$

This must be true at least in the vicinity of $x = 0$, since the linearised dynamics in the proximity that point are:

$$x(t) \cdot f(x) \approx x(t) \cdot \left.\frac{\partial f}{\partial x}\right|_{x=0} x(t) = -\frac{k_p}{1+k_d} x^2(t) < 0 \tag{17}$$

$f(x)$ is a polynomial, and therefore is continuous and differentiable. This means that the boundaries of the region of attraction of the closed loop dynamics will be $x \neq 0$ such that $f(x) = 0$ – one negative and one positive one. This leads to the equation:

$$f(x) = \frac{\alpha}{1+k_d} x^3 - \frac{k_p}{1+k_d} x = 0 \tag{18}$$

Since $x \neq 0$, division by x does not eliminate solutions of interest:

$$\frac{\alpha}{1+k_d} x^2 - \frac{k_p}{1+k_d} = 0 \tag{19}$$

This quadratic equation is very straightforward to solve:

$$\frac{\alpha}{1+k_d} x^2 = \frac{k_p}{1+k_d} \tag{20}$$

$$\alpha x^2 = k_p \tag{21}$$

$$x = \pm\sqrt{\frac{k_p}{\alpha}} \tag{22}$$

The closed loop region of attraction of the trim point is therefore $\left(-\sqrt{\frac{k_p}{\alpha}}, \sqrt{\frac{k_p}{\alpha}}\right)$.