16-665

HW4

Boxiang Fu

boxiangf

11/26/2024

Part 1.

Q1. Referring to Fig 1 in the question sheet, the net force in the x- and y-directions are:

$$f_x = f_{\ell,x} + f_{\tau,x}$$

$$f_y = f_g + f_{\ell,y} + f_{\tau,y}$$

$$= -Mg + f_{\ell,y} + f_{\tau,y}$$ (note that $\vec{g}$ is in the $-y$ direction)

(vector decomposed notation)

Using trigonometry and denoting $f_\ell = \|\vec{f}_\ell\|$ and $f_\tau = \|\vec{f}_\tau\|$, we have

$$f_x = f_{\ell,x} + f_{\tau,x}$$

$$= f_\ell \cos\theta + f_\tau \cos\left(\frac{\pi}{2}+\theta\right)$$ (note $\vec{f}_\ell \perp \vec{f}_\tau$)

$$= f_\ell \cos\theta - f_\tau \sin\theta \quad (2.1)$$ (using identity $\cos\left(\frac{\pi}{2}+x\right) = -\sin x$)

$$f_y = -Mg + f_{\ell,y} + f_{\tau,y}$$

$$= -Mg + f_\ell \sin\theta + f_\tau \sin\left(\frac{\pi}{2}+\theta\right)$$

$$= -Mg + f_\ell \sin\theta + f_\tau \cos\theta \quad (2.2)$$ (using identity $\sin\left(\frac{\pi}{2}+x\right) = \cos x$)

(angle decomposed notation)

Q2. From SOHCAHTOA and the definition of torque,

$\sin\theta = \frac{y_0}{\ell}$ and $\cos\theta = \frac{x}{\ell}$, and $\ell f_\tau = \tau \Rightarrow f_\tau = \tau/\ell$

So we have:

(2.1): $f_x = f_\ell \cdot \frac{x}{\ell} - \frac{\tau}{\ell} \cdot \frac{y_0}{\ell}$

$\therefore f_x = \frac{f_\ell x}{\ell} - \frac{\tau y_0}{\ell^2}$ (2.3)

(2.2): $f_y = -Mg + f_\ell \cdot \frac{y_0}{\ell} + \frac{\tau}{\ell} \cdot \frac{x}{\ell}$

$\therefore f_y = -Mg + \frac{f_\ell y_0}{\ell} + \frac{\tau x}{\ell^2}$ (2.4)

Q3. Let $f_y = 0$, then (2.4) gives

$0 = -Mg + \frac{f_\ell y_0}{\ell} + \frac{\tau x}{\ell^2}$

$\Rightarrow Mg - \frac{\tau x}{\ell^2} = f_\ell \frac{y_0}{\ell}$

$\therefore f_\ell = \frac{\ell Mg}{y_0} - \frac{\tau x}{\ell y_0}$ (2.5)

Q4. (2.5) into (2.3) gives

$f_x = \left(\frac{\ell Mg}{y_0} - \frac{\tau x}{\ell y_0}\right)\frac{x}{\ell} - \frac{\tau y_0}{\ell^2}$

$= \frac{Mgx}{y_0} - \frac{\tau x^2}{\ell^2 y_0} - \frac{\tau y_0}{\ell^2}$ (2.6)

Noting that by Pythagorean theorem,

$\ell^2 = x^2 + y_0^2$

So (2.6):

$$f_x = \frac{Mgx}{y_0} - \frac{T}{y_0}\left(\frac{x^2}{\ell^2} + \frac{y_0^2}{\ell^2}\right)$$

$$= \frac{Mgx}{y_0} - \frac{T}{y_0}\left(\frac{\ell^2}{\ell^2}\right)$$

$$\therefore f_x = \frac{Mgx}{y_0} - \frac{T}{y_0} \qquad (2.7)$$

Q5. From page 6 of the lectures, we have (approx.)

$$p = \frac{T}{Mg}$$

$$\Rightarrow T = pMg \qquad (2.8)$$

(2.8) into (2.7) gives

$$f_x = \frac{Mgx}{y_0} - \frac{pMg}{y_0}$$

$$= \frac{Mg}{y_0}(x-p) \qquad (2.9)$$

Noting that $f_x = M\ddot{x}$, we obtain

$$M\ddot{x} = \frac{Mg}{y_0}(x-p) \qquad (2.10)$$

as required.

Q6. Given $v_0 > 0$ and $v_e = 0$ with p constant, we have

$$\Delta KE = \frac{1}{2}Mv_e^2 - \frac{1}{2}Mv_0^2$$
$$= -\frac{1}{2}Mv_0^2$$

$$W = \int_{-x_T}^{0} f_x \, dx$$
$$= \int_{-x_T}^{0} \frac{Mg}{y_0}x - \frac{Mg}{y_0}p \, dx$$
$$= \left[\frac{1}{2}\frac{Mg}{y_0}x^2 - \frac{Mg}{y_0}px\right]_{-x_T}^{0}$$
$$= -\left(\frac{1}{2}\frac{Mg}{y_0}x_T^2 + \frac{Mg}{y_0}px_T\right)$$

By the conservation of energy,

$$\Delta KE = W$$
$$\Rightarrow -\frac{1}{2}Mv_0^2 = -\left(\frac{1}{2}\frac{Mg}{y_0}x_T^2 + \frac{Mg}{y_0}px_T\right)$$
$$v_0^2 = \frac{g}{y_0}x_T^2 + 2\frac{g}{y_0}px_T$$
$$\frac{y_0}{g}v_0^2 = x_T^2 + 2px_T$$
$$x_T^2 + 2px_T - \frac{y_0}{g}v_0^2 = 0$$

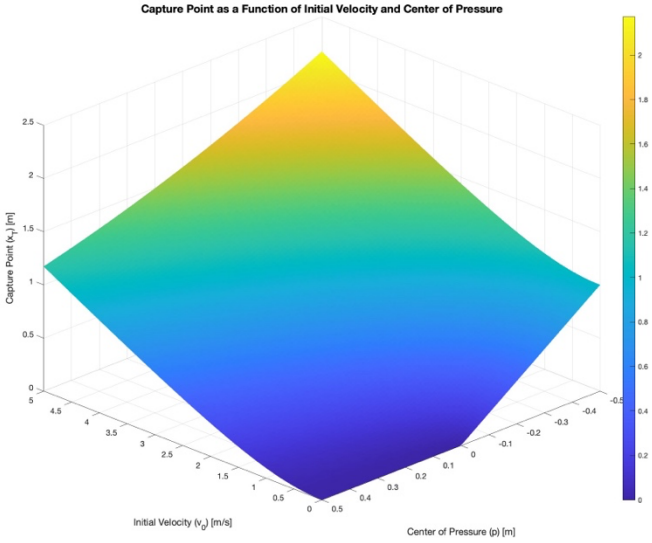
By the quadratic equation,

$$x_T = \frac{-2p \pm \sqrt{(2p)^2 - 4\left(-\frac{y_0}{g}v_0^2\right)}}{2}$$
$$= \frac{-2p \pm 2\sqrt{p^2 + \frac{y_0}{g}v_0^2}}{2}$$
$$x_T = -p \pm \sqrt{p^2 + \frac{y_0}{g}v_0^2}$$

Since the model is valid only if $x_T \in [-p, p]$, only the positive solution is valid. So

$$\therefore x_T = -p + \sqrt{p^2 + \frac{y_0}{g} v_0^2}$$

The plot of $x_T = f(p, v_0)$ is as follows (using typical $y_0 = 1$):



The capture point is where the humanoid's CoM should be placed next. Therefore, if the capture point is within the polygon of support, the robot is able to resume standing balance using the ankle and does not require a step. From the plot, this happens for small initial velocities and positive CoPs (which is reasonable since smaller v_0 requires less torque to stop so the ankle can generate, and a larger CoP means less correction to the CoM is needed so the ankle can also correct). Conversely, if the capture point is outside the polygon of support, the torque required from the ankle exceeds the maximum torque that can be generated by the ankle. This is when the robot should make a step to regain balance. This occurs when the CoP is negative and the initial velocity is large. With a typical humanoid foot of about ~ 0.26m, the humanoid should use ankle strategy if $x_T < 0.26$ and step if $x_T \geq 0.26$.

Part 2.

Q1. The motor selected is the Maxon EC90 Flat 260W with part number 500267

From the spec sheets, we have

$J_M = 5060\ g \cdot cm^2$

$= 5060 \times 100^{-2} \times 10^{-3}\ kg \cdot M^2$

$= 5060 \times 10^{-7}\ kg \cdot M^2$

$= 0.000506\ kg \cdot M^2$

$\tau_{max} = 964\ mNM$

$= 964 \times 10^{-3}\ NM$

$= 0.964\ NM$

Q2. The relationship between maximum (overload) current and duty cycle is given by Maxon (pg. 44) as

$$\frac{I'_{max}}{I_{max}} = \sqrt{\frac{1}{t_{DC\%}}}$$ where $t_{DC\%}$ is duty cycle percentage

I_{max} is max. continous current

I'_{max} is max. overload current

Torque and current are related by

$\tau = k_m I$, where k_m is the torque constant.

So $\frac{\tau^*_{max/km}}{\tau_{max/km}} = \sqrt{\frac{1}{t_{DC\%}}}$

$\Rightarrow \tau^*_{max} = \tau_{max}\sqrt{\frac{1}{t_{DC\%}}}$

With $t_{DC\%} = 0.5$, we obtain

$\tau^*_{max} = 0.964\sqrt{\frac{1}{0.5}}$

$= 0.964\sqrt{2}$

$\approx 0.964 \times 1.41$

$\therefore \tau^*_{max} = 1.36\ Nm$

The gear ratio required is

$N = \frac{\tau_{req}}{\tau^*_{max}}$

$= \frac{Freq \cdot r}{\tau^*_{max}}$ (r is gearbox radius)

$= \frac{1.37Mgr}{\tau^*_{max}}$

$= \frac{1.37 \cdot 80 \cdot 9.81 \cdot 0.05}{1.36}$

$= 39.53$

≈ 40

$\therefore$ The gear ratio necessary is 40.

Q3.

By the definition of gear ratio N, we have

$$N = \frac{\tau_s}{\tau_{ext}} \quad (\tau_s \text{ is the torque generated by the spring})$$

$$\Rightarrow \tau_{ext} = \frac{F_s \cdot r}{N}$$

From Q2, we have $N = 40$, $r = 0.05\,m$,

so $\tau_{ext} = \frac{0.05}{40} F_s$

$$= 0.00125\, F_s$$

From rotational dynamics equation,

$$J_M \ddot{\theta} = \tau_m - \tau_{ext}$$

and $J_m = 0.000506\ kg \cdot m^2$, we obtain

$$0.000506\,\ddot{\theta} = \tau_m - 0.00125\, F_s$$

$$\therefore \ddot{\theta} = 1976.3\,\tau_m - 0.247\, F_s$$

as the equation of motion for motor position θ_m

Q4. The cascaded controller is implemented as follows:

Outer loop:

We use a PID controller with a low-pass filter on the I term.

Let $e_y \equiv y_{des} - y$, we obtain

$$F_s^{des} = k_p^{outer} e_y + k_d^{outer} \dot{e}_y + k_i^{outer} (1-\lambda) \int e_y + Mg \qquad (PID_{outer})$$

where Mg accounts for inherent gravity compensation and $(1-\lambda)$ is the low pass filter.

The rationale for the I term is to converge to the steady state y_{des} faster. However, we use a low-pass filter so the term adapts over time and "forgets" accumulated errors many timesteps ago so that we do not carry forward these exogenous disturbances. This enables a smoother convergence when the timesteps are discretized.

Inner loop:

After obtaining F_s^{des}, we obtain using Hooke's law

$$\Delta l^{des} = \frac{F_s^{des}}{k}$$

Since $\Delta l = \Delta l_y + \Delta l_M$, we obtain

$$\Delta l_M^{des} = \Delta l^{des} - \Delta l_y$$
$$= \frac{F_s^{des}}{k} - (y_0 - y)$$

From the equation in Figure 2 of the question prompt, we obtain

$$\theta_M^{des} = \frac{N}{r}\Delta l_m^{des}$$
$$= \frac{N}{r}\left[\frac{F_s^{des}}{K} - (y_0 - y)\right] \qquad \text{(Remap)}$$

Define $e_\theta \equiv \theta_M^{des} - \theta_M$, we use a PD controller on τ_M:

$$\tau_m = k_p^{inner} e_\theta + k_d^{inner} \dot{e}_\theta \qquad \text{(PID inner)}$$

By the dynamics equation given:

$$M\ddot{y} = F_s - Mg$$
$$J_M \ddot{\theta} = \tau_M - \tau_{ext}$$

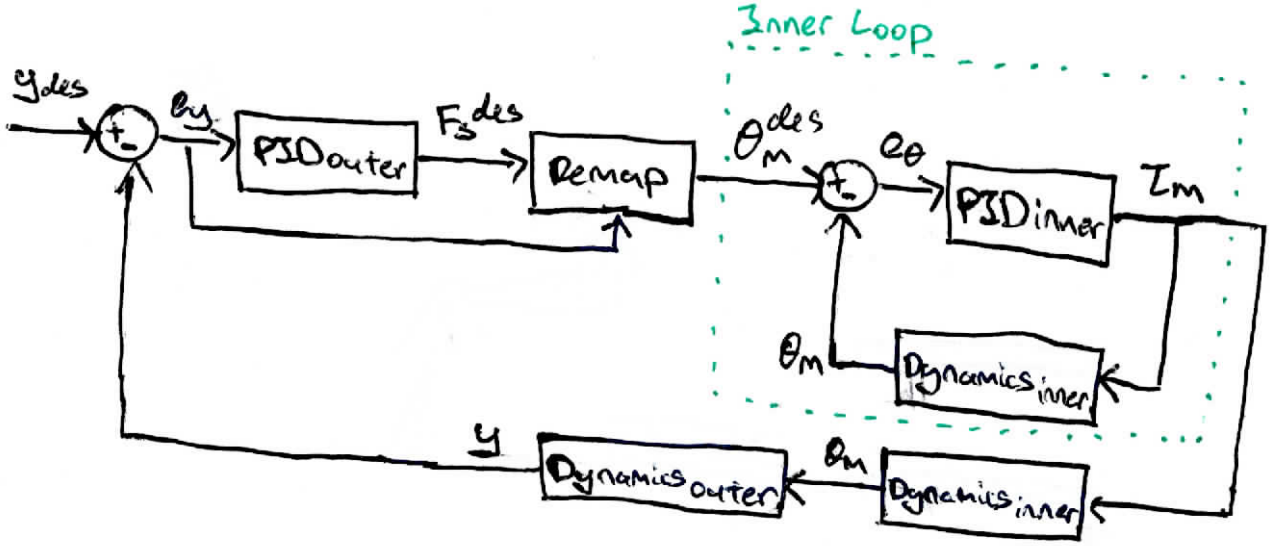
and $\tau_{ext} = \frac{F_s \cdot r}{N}$ from Q3

and $F_s = k\left[(y_0 - y) + \frac{r}{N}\theta_m\right]$ by Hooke's law, the dynamics of the system is given as

$$M\ddot{y} = k\left[(y_0 - y) + \frac{r}{N}\theta_m\right] - Mg \qquad \text{(Dynamics}_{outer}\text{)}$$
$$J_M\ddot{\theta}_m = \tau_M - \frac{r}{N}k\left[(y_0 - y) + \frac{r}{N}\theta_M\right] \qquad \text{(Dynamics}_{inner}\text{)}$$

which is a 2nd order coupled ODE and is solved numerically in the MATLAB implementation.

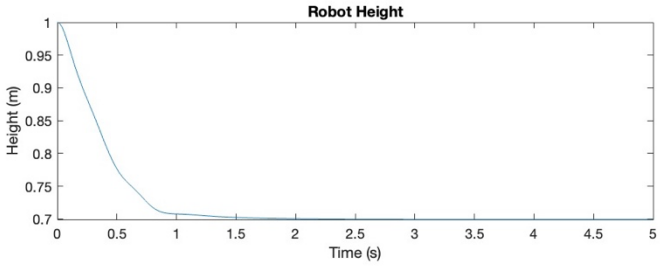
The flow diagram is as follows:

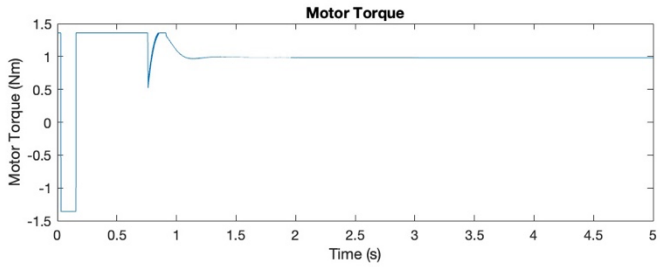


Q5.

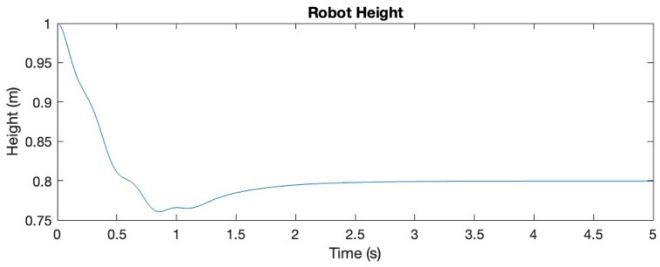
The humanoid is stabilized as follows:

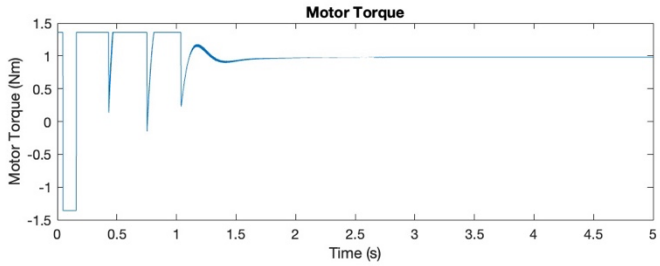
$y_{des} = 0.7m$:



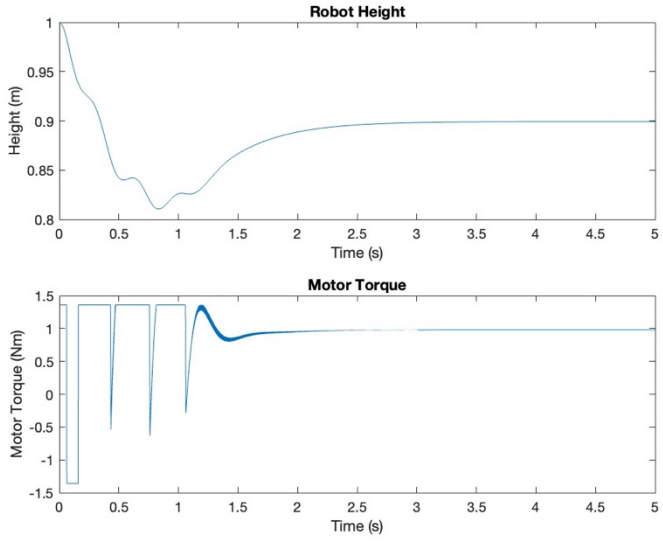


$y_{des} = 0.8m$:





$y_{des} = 0.9m$:



Q6. The temperature change can be modelled according to the Maxon catalog as follows:

$$\Delta T_{max} = \frac{(R_{th1} + R_{th2}) \cdot R \cdot I_{mot}^2}{1 - \alpha_{cu}(R_{th1} + R_{th2}) \cdot R \cdot I_{mot}^2}$$

where $R_{th1} = 1.82\ K/W$ is the winding-housing thermal resistance

$R_{th2} = 1.78\ K/W$ is the housing-environment thermal resistance

$\alpha_{cu} = 0.0039$ is the thermal resistance coefficient of copper

$R = R_{25}(1 + \alpha_{cu}(T_A - 25))$ is the electrical resistance

$R_{25} = 0.844\ \Omega$ is the electrical resistance at room temp.

T_A is the ambient temperature (°C)

$I_{mot} = \frac{\tau}{k_m}$ is the electrical current

$k_m = 0.231\ Nm/A$ is the torque constant

τ is the torque

ΔT_{max} is the maximum temperature difference from ambient temperature (°C)

Starting at room temperature T=25, the thermal motor dynamics are as follows:

$$\Delta T(t) = \Delta T_{max}\left(1 - e^{-\frac{t}{\tau_{th}}}\right)$$

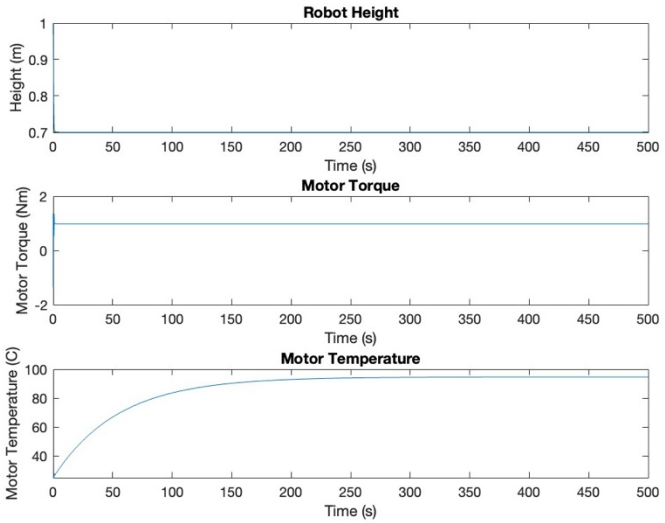
where $\tau_{th} = 54.3\,s$ is the thermal time constant (winding) (s)

$\Delta T(t)$ is the temperature difference from ambient temperature at time t (°C)

The dynamics are as follows:

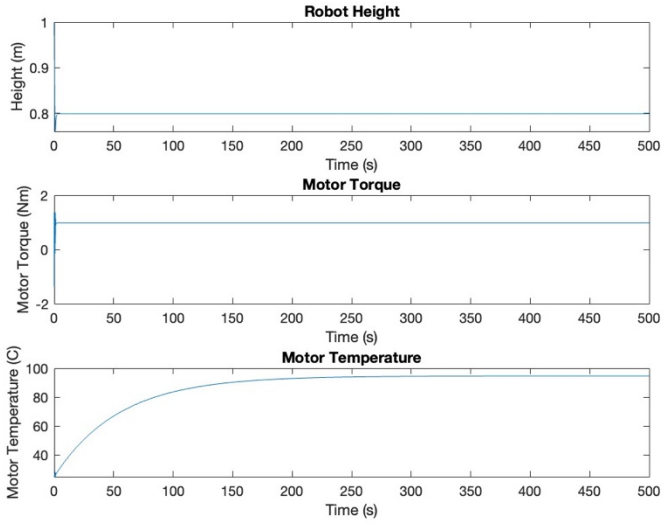
$y_{des} = 0.7\,m$:

Steady state temp.: 94.68 °C, which is below the maximum winding temp. bound of 125°C



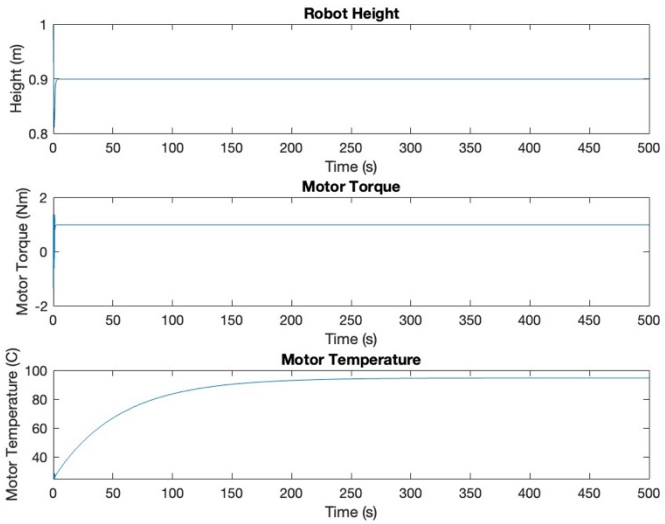
$y_{des} = 0.8\,m$:

Steady state temp.: 94.68°C, which is below the maximum winding temp. bound of 125°C



$y_{des} = 0.9_m$:

Steady state temp.: 94.68°C, which is below the maximum winding temp. bound of 125°C



The steady state temp. does not differ between heights. This is because at steady state, the required torque is to counter the gravitational force, which is Mg regardless of the target height.

For a knee actuator on a real humanoid, the outcome is unrealistic for the following reasons:

- Knee actuator will experience dynamic loads with different torques. The steady-state ignores torque variations and underestimate transient thermal spikes.
- The steady state does not account for duty cycles, which may exceed max. temp. bound during peak activity phases.
- The actuator is enclosed, which increases the housing-environment thermal resistance (R_{th2}).
- The ambient temperature may change, resulting in varying electrical resistance R.
- Hist temperatures from continuous operation can quickly degrade the motor.

Part 3

Q1. The other 2 elements planned are:

FP: The stance plan for the foot point (FP) of the swing leg.

This is done through a quadratic spline (i.e. order 2 polynomial) given the following conditions:

Initial:

$x_0, \dot{x}_0$

$y_0, \dot{y}_0$

Final:

x_f = desired step length + nominal capture point of opposite leg

$y_f = 0$ (foot ends on ground)

TR: Nominal plan for the trunk

The desired TR acceleration, velocity, and position are all initialized to 0, and only the desired trunk acceleration is updated at each timestep depending on the humanoid joint angles and velocities.

Q2. The plans are modified as follows:

FP: Only the acceleration plan of the foot point (FP) is modified. This is done with a PD controller that takes into account the FP position error (with gain $k_p = 100$) and the FP velocity error (with gain $k_d = 10$) and adds it to the original FP acceleration (i.e. it acts as a feedforward term).

TR: Only the acceleration plan of the trunk (TR) is modified. This is done with a PD controller that takes into account the TR joint angle error (with gain $k_p = 0.5$) and the TR joint velocity error (with gain $k_d = 0.05$) and adds it to the original TR acceleration (which was initialized to 0).

Q3.

The inequality $GRFx \leq \mu GRFy$ can be reformulated as

$$\begin{bmatrix} 1 & -\mu \end{bmatrix} \begin{bmatrix} GRFx \\ GRFy \end{bmatrix} \leq 0$$

The inequality $-\mu GRFy \leq GRFx$ can be reformulated as

$$\begin{bmatrix} -1 & -\mu \end{bmatrix} \begin{bmatrix} GRFx \\ GRFy \end{bmatrix} \leq 0$$

So we have

$$Aineq = \begin{bmatrix} zeros(2,10) & \begin{matrix} 1 & -\mu \\ -1 & -\mu \end{matrix} \end{bmatrix}$$

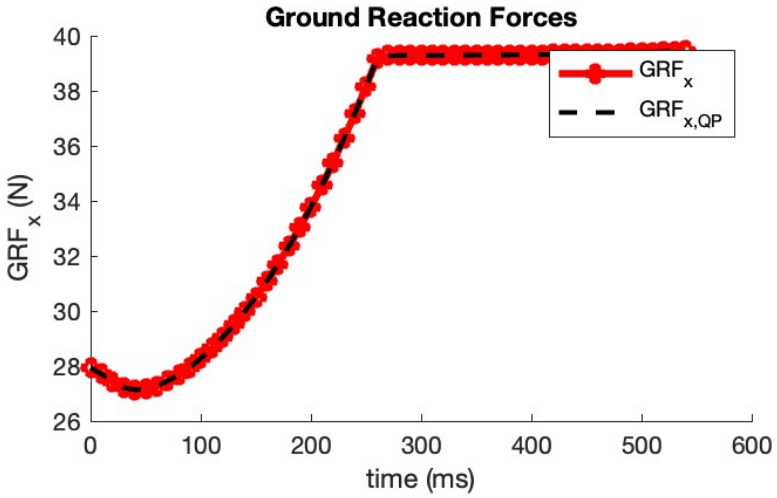
$$bineq = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

with $\mu = 0.8$

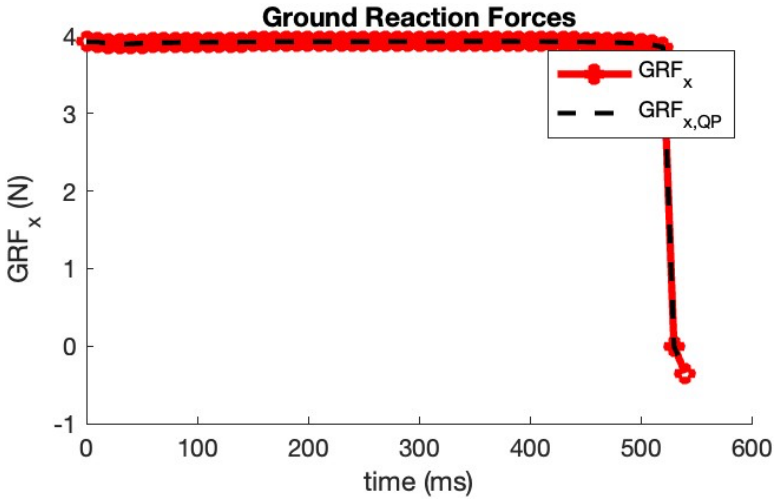
Refer to appendix for code.

Q4.

With $\mu = 0.1$, the horizontal GRF max out at about 40N



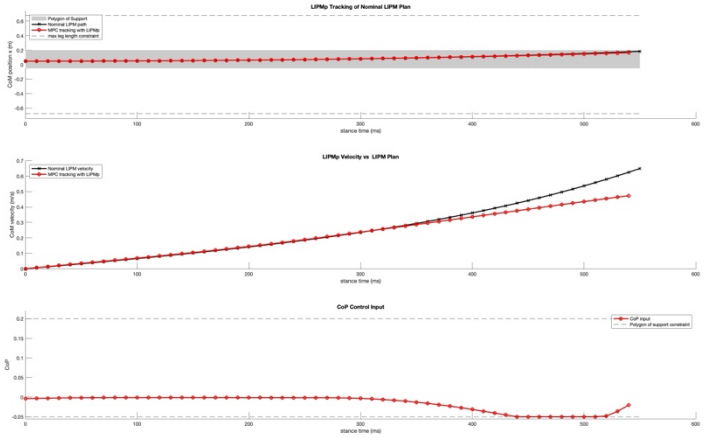
With $\mu = 0.01$, the horizontal GRF max out at about 4N



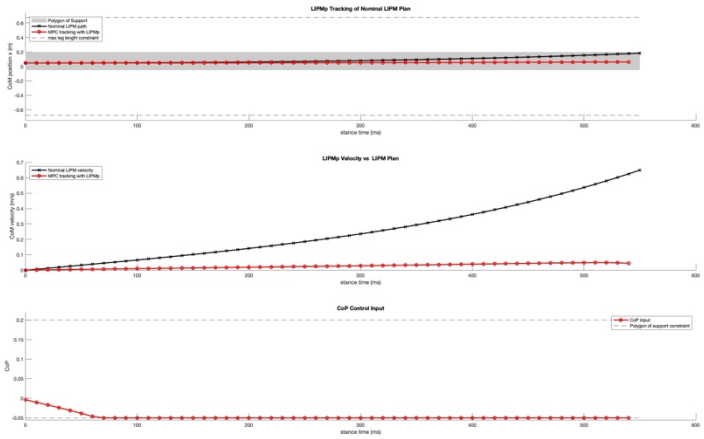
The limitation on the horizontal GRF would cause deviations from the planned CoM. This deviation is significantly more noticable when $\mu = 0.01$.

For the $\mu = 0.1$ case, the horizontal GRF maxs out at approx. 250 ms. This is when we see the desired and achieved CoMs starting to deviate, thereby causing velocity and position to also deviate from the planned path as there is not enough horizontal force for the CoM to keep up with its planned path.

Additionally, the CoP exceeds the polygon of support at approx. 430 ms, which could signal that the humanoid becomes unstable at this point and could potentially fall.



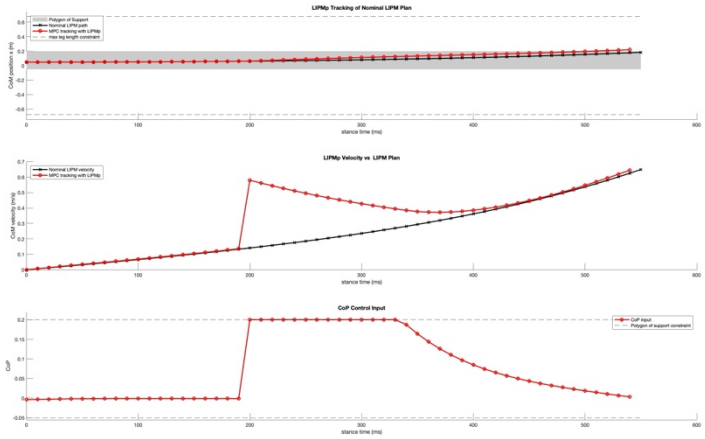
For the μ=0.01 case, the deviation is more pronounced. This is because the horizontal GRF maxs out at almost 0 ms, thereby causing there to not be enough force to keep up with the planned CoM trajectory. We see that acceleration, velocity, and position all fail to keep up with their desired trajectories. Finally, the CoP exceeds the polygon of support at approx. 70ms, after which the humanoid would most surely fall over.



Q5. When a disturbance impulse of 20Ns is applied, we see a spike in CoM acceleration. This causes the CoM velocity to also spike and the CoM to deviate from its intended path.

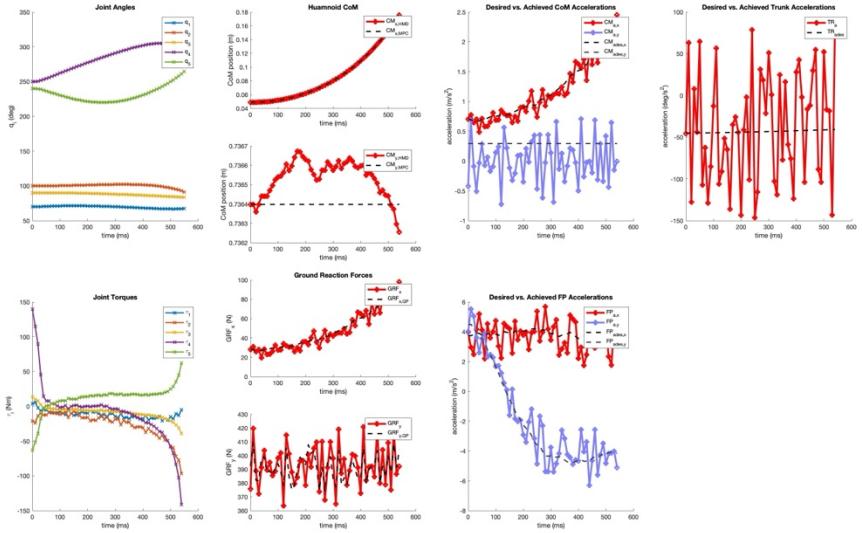
However, the MPC controller reacts very quickly and adjusts the CoM desired acceleration in the opposite direction of the impulse. Effectively decelerating the CoM and return the CoM to the planned trajectory and velocity. We noticably see this in the velocity plot whereas there is a slight offset in the position plot.

How this is accomplished is by adjusting (via the MPC controller) the CoM desired acceleration to the opposite direction of the impulse. This causes the achieved CoM accelerations to follow this new desired plan, thereby returning the humanoid to its intended path.



When random noise of up to +/- 5 Nm is added to the actuator, we immediately see that the tracking of the behavior goals becomes worse. The actual behavior is mean centered at the desired behaviour but there is random fluctations around it.

However, presumably due to the usage of MPC control, the humanoid's CoM is relatively well tracked. The same cannot be said for the trunk and foot point, which uses a PD controller. Nevertheless, we see that increasing actuator noise decreases the tracking accuracy of the behaviour goals relative to no noise and cause the behaviour to become jagged rather than smooth. The reactive updating of the plans (see Q2) also causes the goals to become non-smooth in order to compensate and make the humanoid follow its intended walking plan.



Appendices:

A.1. Code for Q1

```
% Legged Mobility
% Part 1 Q6
% Author: Boxiang Fu
clear;

g = 9.81;
y0 = 1.0;

v0 = linspace(0, 5, 100);
p = linspace(-0.5, 0.5, 100);

% Create meshgrid
[v0_mesh, p_mesh] = meshgrid(v0, p);

% Capture point calculation
xT = -p_mesh + sqrt(p_mesh.^2 + (y0 * v0_mesh.^2) / g);

% Plot
figure;
surf(v0_mesh, p_mesh, xT, 'EdgeColor', 'none');

xlabel('Initial Velocity (v_0) [m/s]', 'FontSize', 12);
ylabel('Center of Pressure (p) [m]', 'FontSize', 12);
zlabel('Capture Point (x_T) [m]', 'FontSize', 12);
title('Capture Point as a Function of Initial Velocity and Center of Pressure',
'FontSize', 14);

colorbar;
view(-135, 30);
grid on;
```

A.2. Code for Q2

```
% Legged Mobility
% Part 2 Q4
% Author: Boxiang Fu
clear;

% Parameters
M = 80; % mass of robot (kg)
g = 9.81; % gravitational acceleration (m/s^2)
k = 20000; % spring stiffness (N/m)
r = 0.05; % rack and pinion radius (m)
J_m = 0.000506; % motor inertia (kg m^2)
N = 40; % gear ratio

% PID control parameters
kp_outer = 2000; % proportional gain for outer loop
kd_outer = 50; % derivative gain for outer loop
ki_outer = 5; % integral gain for outer loop

kp_inner = 20; % proportional gain for inner loop
kd_inner = 1; % derivative gain for inner loop

% Initial conditions
y0 = 1; % nominal height (m)
```

```
y = y0; % initial height (m)
ydot = 0; % initial velocity (m/s)
theta_m = 0; % initial motor angle (rad)
thetadot_m = 0; % initial motor angular velocity (rad/s)
int_error_y = 0; % integral of height error

tau_m_max = 1.36; % motor torque limit (N m)
lambda = 0.05; % low pass filter on integral term

% Desired conditions
y_des = 0.9; % desired height (m)

% Simulation parameters
dt = 0.0001; % time step (s)
outer_loop_steps = 5; % refresh rate between inner and outer loop
t_final = 500; % simulation duration (s)
time = 0:dt:t_final;

% Initialize variables
y_values = zeros(size(time));
tau_m_values = zeros(size(time));
theta_m_values = zeros(size(time));
F_s_des_values = zeros(size(time));

% Thermal motor dynamics
% Parameters
R_th1 = 1.82; % winding-housing thermal resistance (K/W)
R_th2 = 1.78; % housing-environment thermal resistance (K/W)
alpha_cu = 0.0039; % thermal resistance of copper
R_25 = 0.844; % electrical resistance at room temperature (ohm)
k_m = 0.231; % torque constant (Nm/A)
T_amb = 25; % ambient temperature (C);
tau_th = 54.3; % winding thermal time constant (s);

% Variables
T = 25; % initial temperature
T_values = zeros(size(time));

for i = 1:length(time)

    % Outer loop
    if mod(i-1, outer_loop_steps) == 0
        e_y = y_des - y;
        int_error_y = (1 - lambda) * int_error_y + e_y * (outer_loop_steps *
dt);
        F_s_des = kp_outer * e_y - kd_outer * ydot + ki_outer * int_error_y + M
* g;
    end

    % Inner loop
    delta_l_des = F_s_des / k;
    delta_l_m_des = delta_l_des - (y0 - y);
    theta_m_des = (N / r) * delta_l_m_des;

    e_theta = theta_m_des - theta_m;
    tau_m = kp_inner * e_theta - kd_inner * thetadot_m;
    % Clamp motor torque so it stays within operating limits of Maxon EC90
    tau_m = min(max(tau_m, -tau_m_max), tau_m_max);

    % Motor dynamics
    F_s = k * ((y0 - y) + (r / N) * theta_m);
    thetaddot_m = (tau_m - (r/N) * F_s)/ J_m;
```

```
    thetadot_m = thetadot_m + thetaddot_m * dt;
    theta_m = theta_m + thetadot_m * dt;

    % Robot dynamics
    yddot = (F_s - M * g) / M;
    ydot = ydot + yddot * dt;
    y = y + ydot * dt;

    % Save results
    y_values(i) = y;
    tau_m_values(i) = tau_m;
    theta_m_values(i) = theta_m;
    F_s_des_values(i) = F_s_des;

    % Thermal dynamics
    I_mot = tau_m / k_m;
    R = R_25 * (1 + alpha_cu * (T_amb - 25));

    deltaT_max = ((R_th1 + R_th2) * R * I_mot^2) / (1 - alpha_cu * (R_th1 +
R_th2) * R * I_mot^2);
    deltaT = deltaT_max * (1 - exp(-time(i)/tau_th));
    T = T_amb + deltaT;
    T_values(i) = T;
end

% Plot results
figure;
subplot(3, 1, 1);
plot(time, y_values);
xlabel('Time (s)');
ylabel('Height (m)');
title('Robot Height');

subplot(3, 1, 2);
plot(time, tau_m_values);
xlabel('Time (s)');
ylabel('Motor Torque (Nm)');
title('Motor Torque');

subplot(3, 1, 3);
plot(time, T_values);
xlabel('Time (s)');
ylabel('Motor Temperature (C)');
title('Motor Temperature');

% subplot(4, 1, 3);
% plot(time, theta_m_values);
% xlabel('Time (s)');
% ylabel('Motor Angle (rad)');
% title('Motor Angle');
%
% subplot(4, 1, 4);
% plot(time, F_s_des_values);
% xlabel('Time (s)');
% ylabel('Desired Spring Force (N)');
% title('Desired Spring Force');
```

A.3. Code for Q3

```
function QP = QP_BuildConstraints(QP)

%
% QP_BuildConstraints.m  -  Build constraint terms for instantaneous QP of
%                             the humanoid model
%
% Inputs:
% QP: QP object (custom)
%
% Output:
% QP: QP object with constraint equation terms Aeq and beq created or updated


% Theory:
% (1) The equations of motion of a kinematic chain are given by
%     M*ddq + C*dq + N = tau, where M is the mass matrix, C is
%     the Coriolis matrix and N is the gravitational vector.
%
% (2) The equations can be realigned as M*ddq -tau = -h with h=C*dq+N,
%     which can be used to define a constraint on the joint accelerations
%     and torques:
%     [M -eye(5)] * [ddq tau]' = -h
%
% (3) A second set of equations of motion is used to constrain the
%     leg forces F not covered in the first equation set.
%
%     The equation of motion for the center of mass is given by
%     m*CM_a = F +m*gVec with gVec = [0 -g]'. The CoM acceleration is related to
%     the joint accelerations by CM_a = d/dt(CM_v) = d/dt(Jcm*q) = Jcm*ddq +
dJcm*dq,
%     where Jcm is the Jacobian mapping the CoM to the joint angles. Combining
the two equations
%     yields:
%     F + m*gVec = m*(Jcm*ddq + dJcm*dq)
%
% (4) This equation can be realigned to a second constraint on the optimization
variable:
%     [m*Jcm  -eye(2)] * [ddq F]' = m*(gVec-dJcm*dq)
%
% (5) Combining the two constraint equations yields
%     [  M    |  -eye(5)   | zeros(5,2)] * [ddq] = [   -h       ]
%     [ m*Jcm | zeros(2,5) |  -eye(2)  ]   [tau]   [m*(gVec-dJcm*dq)]
%                                         [ F ]
%
%     This equation fits the standard equality constraint Aeq * x = beq with
%     x   = [ddq tau F]',
%     Aeq = [M -eye(5) zeros(5,2); m*Jcm zeros(2,5) -eye(2)], and
%     beq = [ -h m*(gVec-dJcm*dq)]'
%

% assign equality constraint terms
QP.Aeq = [       QP.Dyn.M      -eye(5)    zeros(5,2); ...
          QP.Dyn.m*QP.Kin.Jcm zeros(2,5)   -eye(2)  ];

QP.beq = [           -QP.Dyn.h; ...
          QP.Dyn.m*(QP.Dyn.gVec-QP.Kin.dJcmxdq)];


% Theory:
% (1) The horizontal friction needs to stay within the friction cone,
```

```
%     Fx <= mu*Fy, where mu is the friction coefficient.
%
% (2) The equation can be reformulated into
%     [1 -mu]*[Fx Fy]' <= 0
%
% (2) The corresponding  constraint is given by
%     Aineq*x <= bineq, with
%     x   = [ddq tau F]',
%     Aineq = [zeros(1,10) 1 -mu], and
%     bineq = 0
%
% (3) Similarly, Fx >= -mu*Fy, which can be formulated as
%     [-1 -mu]*[fX Fy]' <= 0, is implemented as inequality
%     constraint on [ddq tau F]'
 mu = 0.8;
 QP.Aineq = [zeros(1,10) 1 -mu; zeros(1,10) -1 -mu];
 QP.bineq = [0; 0];
```