# 4 Zero-Moment Point

For the Zero-Moment point the following holds:

> ZMP (zero-moment point) is a point on the floor where the resultant moment of the gravity, the inertial force of the mobile manipulator and the external force is zero. *If the ZMP is within the support polygon, the mobile manipulator is balanced.* [13]

This means that when the pose of the robot is known, the movements of the robot are known (the inertial force of the mobile manipulator) and the external forces on the robot are known, it is possible to calculate the point on the floor where all these moments sum to zero. When this point is within the support polygon of the robot (the base, the foot, or the area spanned by the wheels), the robot is balanced in the sense described in section 2.1. Because this zero-moment point involves the inertial forces (acceleration, centripetal forces), it accounts for dynamic effects. This means that the ZMP-method is a suitable method to calculate the dynamic balance of a mobile robot.

## 4.1 Definition

Vukobratović [15] coined the term ZMP:

> In Figure 4.1 an example of force distribution across the foot is given. As the load has the same sign all over the surface, it can be reduced to the resultant force $\boldsymbol{F}_P$, the point of attack of which will be in the boundaries of the foot. Let the point on the surface of the foot, where the resultant $\boldsymbol{F}_P$ passed, be denoted as the zero-moment point, or ZMP in short.

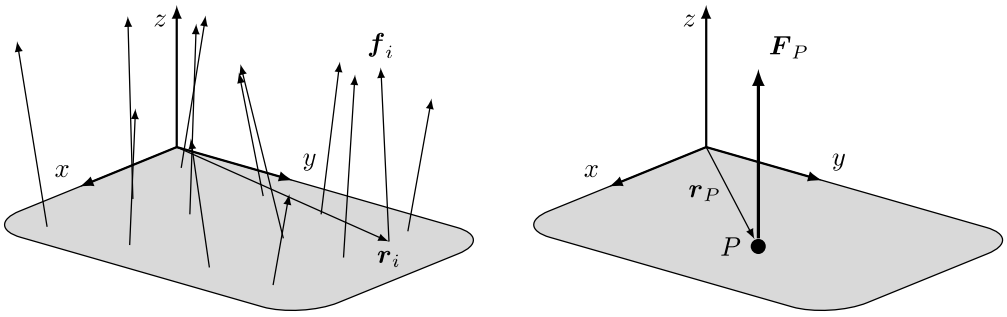



**Figure 4.1:** Force distribution with individual forces $\boldsymbol{f}_i$ (left), resultant force $\boldsymbol{F}_P$ and point of attack $P$ (right).

In the situation shown in Figure 4.1, all reaction forces $\boldsymbol{f}_i$ and contact points $\boldsymbol{r}_i$ for the reaction forces are known, and can thus be reduced to one resultant force $\boldsymbol{F}_P$ at the contact point $P$ with position vector $\boldsymbol{r}_P$.

The force vectors $\boldsymbol{f}_i$ have the form:

$$\boldsymbol{f}_i = [f_{i,x}, f_{i,y}, f_{i,z}]^T \tag{4.1}$$

and the position vectors $\boldsymbol{r}_i$:

$$\boldsymbol{r}_i = [r_{i,x}, r_{i,y}, r_{i,z}]^T \tag{4.2}$$

where the components are defined along the axes of the coordinate frame x-y-z depicted in Figure 4.1.

The center of pressure (CoP) can be calculated using the following formula [11]:

$$\boldsymbol{r}_P = \frac{\sum_{i=1}^{n} \boldsymbol{r}_i f_{i,z}}{\sum_{i=1}^{n} f_{i,z}} \tag{4.3}$$

According to this definition, the CoP always exists within the support polygon (shaded area in Figure 4.1), because it uses the reaction forces on this support polygon.

When the CoP is known, the torque around it can be calculated:

$$\boldsymbol{\tau} = \sum_{i=1}^{n} (\boldsymbol{r}_i - \boldsymbol{r}_P) \times \boldsymbol{f_i} \tag{4.4}$$

where $\boldsymbol{r}_P$ is the location of the CoP as calculated in (4.3) and $\boldsymbol{r}_i$ and $\boldsymbol{f}_i$ are the individual position and force of the reaction forces.

When substituting components of the vectors in equation 4.4, the following equations are obtained:

$$\tau_x = \sum_{i=1}^{n} (r_{i,y} - r_{P,y}) \times f_{i,z} - \sum_{i=1}^{n} (r_{i,z} - r_{P,z}) \times f_{i,y} \tag{4.5}$$

$$\tau_y = \sum_{i=1}^{n} (r_{i,z} - r_{P,z}) \times f_{i,x} - \sum_{i=1}^{n} (r_{i,x} - r_{P,x}) \times f_{i,z} \tag{4.6}$$

$$\tau_z = \sum_{i=1}^{n} (r_{i,x} - r_{P,x}) \times f_{i,y} - \sum_{i=1}^{n} (r_{i,y} - r_{P,y}) \times f_{i,x} \tag{4.7}$$

When the support polygon is parallel to the $xy$-plane the following holds: $r_{i,z} = r_{P,z}$. Moreover, when the support polygon lies in the $xy$-plane, these components become $r_{i,z} = r_{P,z} = 0$. In either case, the second term in (4.5) and the first term in (4.6) become zero.

The definition of the CoP (4.3) can be used to further reduce (4.5) and (4.6):

$$\tau_x = \tau_y = 0 \tag{4.8}$$

The remaining torque around the $z$-axis ($\tau_z$) is nonzero. The fact that $\tau_z$ is nonzero has no practical consequences when it is assumed that there is no slip between support polygon and ground. In that case, the vertical torque $\tau_z$ is canceled by friction forces. The fact that the torques in $x$ and $y$ direction are zero is the reason why this point is also called the Zero-Moment Point.

## 4.2 Computed ZMP

The definition in section 4.1 assumes that there is a finite number of reaction forces that can be summed to one resultant force. If all these are known, the CoP can be calculated. By using pressure sensors on the robot base, this is a convenient way to practically measure the CoP location. This information can be used to control the robot balance online. However, it is not possible to make a prediction about the robot balance resulting from a motion profile using this method. It can only act on already measured forces, not on a planned motion.

The *Computed ZMP* is a way to overcome this problem. It works by using known information about the robot (mass of links, position and type of joints, velocities and accelerations) to calculate the resulting reaction force and torque on the base of the robot, and from there on find a point where the reaction torques become zero. This point is the computed ZMP.

In Figure 4.2 the support polygon of the robot is the shaded area. This could be determined by the actual position of a mobile base of a service robot, a foot of a bipedal robot or an ordinary base of a stationary robot. The moving robot exerts forces and moments onto this support polygon, due to accelerating and rotating parts of its body. These forces and moments can be represented by one resultant force $\boldsymbol{F}_R$ and one resultant moment $\boldsymbol{M}_R$ acting on point $R$. A reaction force $\boldsymbol{F}_P$ and moment $\boldsymbol{M}_P$ can be calculated for any point $P$ to keep the robot in balance, i.e. such that all forces and moments sum to zero. There is one location for point $P$ where the reaction moment $\boldsymbol{M}_P$ is zero, this is the Zero-Moment Point.

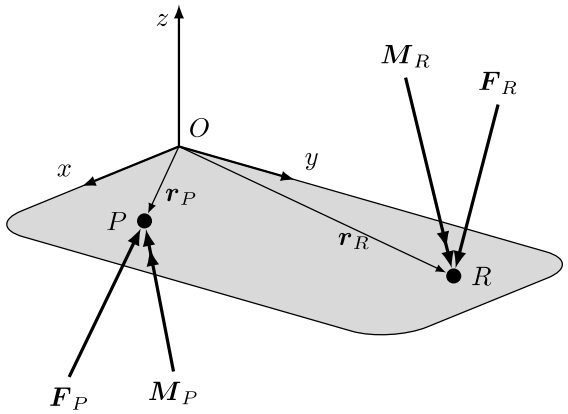



Figure 4.2: Resulting force $\boldsymbol{F}_R$ and moment $\boldsymbol{M}_R$ from the robot on its support polygon (shaded area), and a reaction force $\boldsymbol{F}_P$ and moment $\boldsymbol{M}_P$, to keep the robot in balance.

To sum up, it is necessary for the forces and moments to sum up to zero for the robot to be in balance. This gives the equations for equilibrium:

$$\boldsymbol{F}_P + \boldsymbol{F}_R = 0 \tag{4.9}$$

$$\boldsymbol{M}_P + \boldsymbol{M}_R + \boldsymbol{r}_P \times \boldsymbol{F}_P + \boldsymbol{r}_R \times \boldsymbol{F}_R = 0 \tag{4.10}$$

Written out in vector components equation (4.10) becomes:

$$M_{P,x} + M_{R,x} + r_{P,y}F_{P,z} - r_{P,z}F_{P,y} + r_{R,y}F_{R,z} - r_{R,z}F_{R,y} = 0 \tag{4.11}$$

$$M_{P,y} + M_{R,y} + r_{P,z}F_{P,x} - r_{P,x}F_{P,z} + r_{R,z}F_{R,x} - r_{R,x}F_{R,z} = 0 \tag{4.12}$$

$$M_{P,z} + M_{R,z} + r_{P,x}F_{P,y} - r_{P,y}F_{P,x} + r_{R,x}F_{R,y} - r_{R,y}F_{R,x} = 0 \tag{4.13}$$

When the base and the points $R$ and $P$ are in the $xy$-plane, the $r_z$ components become zero. Moreover, when it is assumed that there is no slip between base and ground, the forces in the $xy$-plane and the moments around the $z$-axis are canceled due to friction. This results in:

$$M_{P,x} + M_{R,x} + r_{P,y}F_{P,z} + r_{R,y}F_{R,z} = 0 \tag{4.14}$$

$$M_{P,y} + M_{R,y} - r_{P,x}F_{P,z} - r_{R,x}F_{R,z} = 0 \tag{4.15}$$

The ZMP is the point where the moment $\boldsymbol{M}_P$ becomes zero (by definition), this reduces the equilibrium equations to:

$$M_{R,x} + r_{P,y}F_{P,z} + r_{R,y}F_{R,z} = 0 \tag{4.16}$$

$$M_{R,y} - r_{P,x}F_{P,z} - r_{R,x}F_{R,z} = 0 \tag{4.17}$$

Together with (4.9) these equations can be used to calculate the ZMP ($\boldsymbol{r}_P$).

## 4.3 Cart-Table Model

A well known approximation to the computed ZMP is the so called cart-table model [11], shown in Figure 4.3.

In this figure a mass $M$ is on a massless table, and has an acceleration of $\ddot{x}$. The position of mass $M$ with respect to origin $O$ is given by $r_{M,x}$ and $r_{M,z}$ in the $x$ and $z$ direction respectively. The ZMP (point $P$) can be calculated with (4.9) and (4.17). In this case the resulting forces and moments are taken around point $R$. Equilibrium around $R$ results in:

$$F_{R,z} = -mg \tag{4.18}$$

$$F_{R,x} = -m\ddot{x} \tag{4.19}$$

$$M_{R,y} = (r_{M,x} - r_{R,x})mg - r_{M,z}m\ddot{x} \tag{4.20}$$

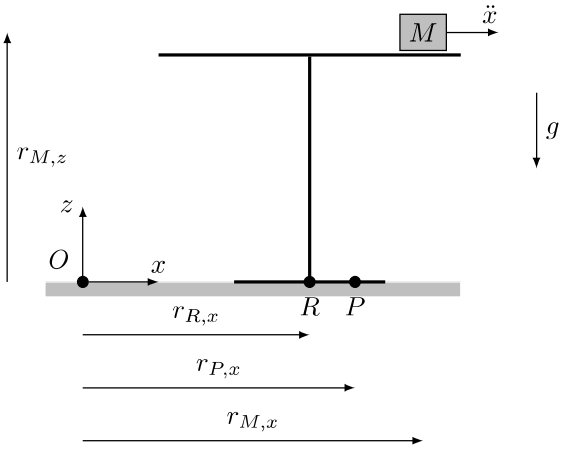



Figure 4.3: Cart-table model

Assuming no-slip conditions, the force in $x$ direction vanishes. Using (4.9) and (4.17) gives:

$$F_{P,z} = mg \tag{4.21}$$

$$r_{P,x}mg = (r_{M,x} - r_{R,x})mg - r_{M,z}m\ddot{x} + r_{R,x}mg \tag{4.22}$$

$$r_{P,x} = r_{M,x} - \frac{r_{M,z}}{g}\ddot{x} \tag{4.23}$$

This shows that the ZMP can be anywhere on the $x$ axis, depending on the acceleration $\ddot{x}$ of the mass. Also, from (4.22) it can be seen that the (arbitrarily chosen) location of point $R$ has no consequence in the resulting ZMP position.

As long as the computed ZMP is within the foot of the table, this system is in balance. When the ZMP leaves the foot of the table there is no equilibrium anymore, which will result in the table tipping over.

This cart-table model can be used to represent an entire robot, the mass $M$ represents the mass of the entire robot while $x$ and $\ddot{x}$ denote motion and acceleration of the actual center of mass of the robot in the x-direction

## 4.4 Computed ZMP for Multibody Systems

In the case of a service robot, the computed ZMP method needs to be adapted to handle multibody systems. In essence this means that a method is needed to compute the resultant force $\boldsymbol{F}_R$ and resultant moment $\boldsymbol{M}_R$ indicated in Figure 4.2 from the known kinematic information (position, velocity, acceleration, mass and inertia) of each robot link.

Figure 4.4 shows a mobile robot consisting of $n$ rigid parts (links), each with its own mass $m_i$ and body fixed inertia tensor $I_i$. Furthermore, $\boldsymbol{r}_{ci}$, $\boldsymbol{R}_i$ and $\boldsymbol{\omega}_i$ are the position vector, the rotation matrix and angular velocity of the center of mass of the $i$-th link respectively. The total mass and the center of mass of the robot can be calculated with:

$$m = \sum_{i=1}^{n} m_i \tag{4.24}$$

$$\boldsymbol{r}_c = \sum_{i=1}^{n} \frac{m_i \boldsymbol{r}_{ci}}{m} \tag{4.25}$$

Assuming that all kinematic information is available (measured or calculated by robot kinematics) and that the no-slip condition holds, the linear and angular momenta of this system of rigid bodies can be used to calculate the resultant force and moment with respect to the origin $O$. For these

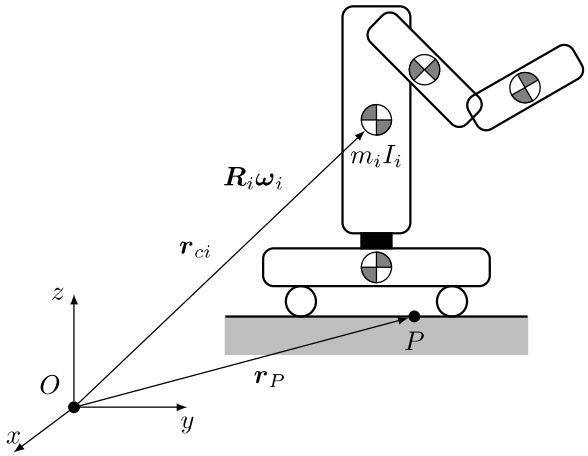



**Figure 4.4:** Multibody model of a mobile robot and reference frame $O$.

calculations, *Newton-Euler equations* can be used that state that the sum of external forces and the sum of external moments result in change of linear and angular momenta respectively [14]:

$$\sum \boldsymbol{F} = \dot{\boldsymbol{P}} \tag{4.26}$$

$$\sum \boldsymbol{M} = \dot{\boldsymbol{H}} \tag{4.27}$$

where $\dot{\boldsymbol{P}}$ and $\dot{\boldsymbol{H}}$ are change in Linear momentum and angular momentum respectively. These quantities can be computed using forward kinematics relationships for the robot. For now it is just assumed that $\dot{\boldsymbol{P}}$ and $\dot{\boldsymbol{H}}$ are known.

When the mobile robot is in balanced equilibrium, the external forces and moments ($\boldsymbol{F}_E$ and $\boldsymbol{M}_E$) that cause the movement of the robot and consequently the change in $\boldsymbol{P}$ and $\boldsymbol{H}$, are in equilibrium with a resulting force ($\boldsymbol{F}_R$) and moment ($\boldsymbol{M}_R$). These are related to each other via: $\boldsymbol{F}_E = -\boldsymbol{F}_R$ and $\boldsymbol{M}_E = -\boldsymbol{M}_R$. Together with gravitational effects, this result in:

$$\sum \boldsymbol{F} = \boldsymbol{F}_E + m\boldsymbol{g} = -\boldsymbol{F}_R + m\boldsymbol{g} = \dot{\boldsymbol{P}} \tag{4.28}$$

$$\sum \boldsymbol{M} = \boldsymbol{M}_E + \boldsymbol{r}_c \times m\boldsymbol{g} = -\boldsymbol{M}_R + \boldsymbol{r}_c \times m\boldsymbol{g} = \dot{\boldsymbol{H}} \tag{4.29}$$

Here $\boldsymbol{g}$ is a 3D gravitational acceleration vector: $\boldsymbol{g} = [g_x, g_y, g_z]^T$. The resulting force and moment are given by:

$$\boldsymbol{F}_R = -\dot{\boldsymbol{P}} + m\boldsymbol{g} \tag{4.30}$$

$$\boldsymbol{M}_R = -\dot{\boldsymbol{H}} + \boldsymbol{r}_c \times m\boldsymbol{g} \tag{4.31}$$

All these forces and moments are given with respect to the origin $O$, which means that point $R$ in Figure 4.2 coincides with the origin $O$, resulting in $\boldsymbol{r}_R = 0$. Combining this with (4.9) and (4.10) gives:

$$\boldsymbol{F}_P = -\boldsymbol{F}_R \tag{4.32}$$

$$\boldsymbol{M}_P = \dot{\boldsymbol{H}} - \boldsymbol{r}_c \times m\boldsymbol{g} - \boldsymbol{r}_P \times (m\boldsymbol{g} - \dot{\boldsymbol{P}}) \tag{4.33}$$

Now, by definition, the first and second component of $\boldsymbol{M}_P$ are zero. When written in vector components this results in:

$$M_{P,x} = 0 = \dot{H}_x - r_{c,y} m g_z + r_{c,z} m g_y - r_{P,y}(\dot{P}_z - m g_z) + r_{P,z}(\dot{P}_y - m g_y) \tag{4.34}$$

$$M_{P,y} = 0 = \dot{H}_y - r_{c,z} m g_x + r_{c,x} m g_z - r_{P,z}(\dot{P}_x - m g_x) + r_{P,x}(\dot{P}_z - m g_z) \tag{4.35}$$

With the assumption that the floor lies in the $xy$-plane ($\boldsymbol{r}_{P,z} = 0$), the ZMP can be calculated:

$$r_{P,x} = \frac{-\dot{H}_y + r_{c,z} m g_x - r_{c,x} m g_z}{\dot{P}_z - m g_z} \tag{4.36}$$

$$r_{P,y} = \frac{\dot{H}_x - r_{c,y} m g_z + r_{c,z} m g_y}{\dot{P}_z - m g_z} \tag{4.37}$$

From this, it can be seen that when the robot is stationary the ZMP coincides with the projection of the center of mass. Important to notice here is that it is assumed that the $x$ and $y$ component of the gravity vector are not always zero. This is a result of the $xy$ plane of the reference frame being defined such that the floor is in this plane. In case of an inclined floor, the gravity vector is not aligned with the $z$-axis of the reference frame anymore.

The final step in these calculations is the determination of the change in linear and angular momenta. With respect to the origin, the linear momentum is given by [11]:

$$\boldsymbol{P} = \sum_{i=1}^{n} m_i \dot{\boldsymbol{r}}_{ci} \tag{4.38}$$

where $\dot{\boldsymbol{r}}_{ci}$ is the linear velocity of the center of mass of the $i$-th link.

Furthermore, the angular momentum is given by [11]:

$$\boldsymbol{H} = \sum_{i=1}^{n} \boldsymbol{r}_{ci} \times m_i \dot{\boldsymbol{r}}_{ci} + \boldsymbol{I}_i \boldsymbol{\omega}_i \tag{4.39}$$

where $\boldsymbol{I}_i$ is the inertia tensor of link $i$ with respect to the ground fixed reference frame with origin $O$. This is calculated from the body fixed inertia tensor $I_i$ using:

$$\boldsymbol{I}_i = \boldsymbol{R}_i I_i \boldsymbol{R}_i^T \tag{4.40}$$

Here, $\boldsymbol{R}_i$ is the rotation matrix that describes the rotation of link $i$ with respect to the reference frame, see appendix B.

The rates of change in the linear and angular momenta are the time derivatives of (4.38) and (4.39) respectively. The inertia matrix $\boldsymbol{I}_i$ in equation (4.39) has a time dependency (due to the rotation matrix $\boldsymbol{R}_i$), so the time derivative of this matrix is also needed [14]:

$$\dot{\boldsymbol{I}}\boldsymbol{\omega} = \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \tag{4.41}$$

which results in a change of linear momentum:

$$\dot{\boldsymbol{P}} = \sum_{i=1}^{n} m_i \ddot{\boldsymbol{r}}_{ci} \tag{4.42}$$

and for the change in angular momentum:

$$\dot{\boldsymbol{H}} = \sum_{i=1}^{n} \dot{\boldsymbol{r}}_{ci} \times m_i \dot{\boldsymbol{r}}_{ci} + \boldsymbol{r}_{ci} \times m_i \ddot{\boldsymbol{r}}_{ci} + \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{I}_i \boldsymbol{\omega}_i) \tag{4.43}$$

$$= \sum_{i=1}^{n} \boldsymbol{r}_{ci} \times m_i \ddot{\boldsymbol{r}}_{ci} + \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{I}_i \boldsymbol{\omega}_i) \tag{4.44}$$

where the cross product $\dot{\boldsymbol{r}}_{ci} \times m_i \dot{\boldsymbol{r}}_{ci}$ is always zero.

To sum up, the ZMP can be calculated with (4.36), (4.37), (4.42) and (4.44) when the following quantities are known:

- $m_i$: Mass of link $i$

- $I_i$: Body fixed inertia matrix for link $i$
- $\boldsymbol{r}_{ci}$: Position vector of center of mass of link $i$, with respect to frame $O$
- $\boldsymbol{R}_i$: Rotation matrix for link $i$, with respect to frame $O$
- $\boldsymbol{\omega}_i$: Angular velocity of link $i$, with respect to frame $O$
- $\ddot{\boldsymbol{r}}_{ci}$: Acceleration of center of mass of link $i$, with respect to frame $O$
- $\dot{\boldsymbol{\omega}}_i$: Time derivative of angular velocity of link $i$, with respect to frame $O$

Mass $m_i$ and inertia $I_i$ are known for each robot link, the other quantities need to be calculated with the forward kinematics relationships.

## 4.5 Summary

In this chapter the Zero-Moment Point is defined and a method of calculating this ZMP from reaction forces is described. Next the computed ZMP method is explained and adapted to multibody systems. This computed ZMP for multibody systems uses the knowledge about the velocities and accelerations of all robot links to compute the change in linear and angular momenta. This in turn enables the computation of the ZMP.

The next step is to calculate the link velocities and accelerations from known joint coordinates. This will be the topic of chapter 5.

# 5 Robot Kinematics

In the previous chapter it is shown that the ZMP can be computed via the change in linear ($\dot{\boldsymbol{P}}$) and angular ($\dot{\boldsymbol{H}}$) momenta when the kinematics of the robot are known. The quantities that are needed for the ZMP computation are $\boldsymbol{r}_{ci}$, $\ddot{\boldsymbol{r}}_{ci}$, $\boldsymbol{\omega}_i$ and $\dot{\boldsymbol{\omega}}_i$. Also, the rotation matrix $\boldsymbol{R}_i$ is needed.

A suitable method for the calculation of these quantities from known robot joint coordinates is by using the *Denavit-Hartenberg parameters* [12, 3] to get *homogeneous transformation* matrices. These matrices can then be used to compute the dynamics of the robot.

Homogeneous transformations are treated in detail in [12, 3], the Denavit-Hartenberg convention can also be found in these sources. In appendices A and B an overview of homogeneous transformations and Denavit-Hartenberg parameters respectively is given.

## 5.1 Forward Dynamics

By the knowledge of the robot forward dynamics, the joint coordinates ($d_i$ or $\theta_i$) and their time-derivatives can be related to the link velocities ($\dot{\boldsymbol{r}}_i$, $\boldsymbol{\omega}_i$) and accelerations ($\ddot{\boldsymbol{r}}_i$, $\dot{\boldsymbol{\omega}}_i$) needed to compute the change in linear (4.42) and angular (4.44) momenta.

The relations between link dynamics and joint coordinates for revolute and prismatic joints are treated in [12] and are also derived for both cases in appendix C. The results are summarized below using the frames and position vectors defined in Figure 5.1.

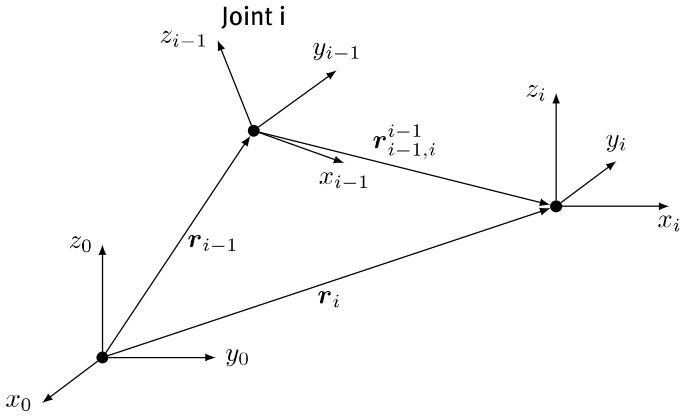



Figure 5.1: Linked frames and their position vectors.

In general, independent of joint type, the location of the origin of frame i is given by $\boldsymbol{r}_i$:

$$\boldsymbol{r}_i = \boldsymbol{r}_{i-1} + \boldsymbol{R}_{i-1}^0 \boldsymbol{r}_{i-1,i}^{i-1} \tag{5.1}$$

$$= \boldsymbol{r}_{i-1} + \boldsymbol{r}_{i-1,i} \tag{5.2}$$

where $\boldsymbol{R}_{i-1}^0$ is the homogeneous transformation between reference frame 0 and link frame $i-1$, computed with Denavit-Hartenberg parameters as in equation B.1.

For a revolute joint the following relations are found:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \boldsymbol{z}_{i-1} \tag{5.3}$$

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i-1,i} + \ddot{\theta}_i \boldsymbol{z}_{i-1} \tag{5.4}$$

$$\ddot{\boldsymbol{r}}_i = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) \tag{5.5}$$

and for a prismatic joint:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} \tag{5.6}$$

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} \tag{5.7}$$

$$\ddot{\boldsymbol{r}}_i = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) + 2\boldsymbol{\omega}_i \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} \tag{5.8}$$

## 5.2 Summary

In chapter 4 the computations for the Zero-Moment Point are laid out. These computations demand the knowledge of all link positions, velocities and accelerations.The forward kinematics of a kinematic chain are determined with the use of the Denavit-Hartenberg convention and the knowledge of the joint coordinates. The forward kinematics are used to calculate the forward dynamics of a robot link. For each joint type, prismatic and revolute, equations are derived and used in the ZMP algorithm in chapter 6.

# 6 ZMP Algorithm

The basics for the predictive algorithm to determine robot balance stated in section 2.2, are laid out in sections 4 and 5. The mathematical background in these chapters is used to design an algorithm that calculates the Zero-Moment Point from known joint coordinates. In this chapter the algorithm itself is presented.

First the input for the algorithm is discussed, then the algorithm itself is presented and finally the output of the algorithm is treated.

## 6.1 Input

From section 5.1 it is clear that the calculation of the link velocities and accelerations are dependent on a description of the robot. Next to these kinematic parameters, the mass, inertia and also the center of mass of each link need to be known. These quantities are the first set of inputs to the algorithm.

| link | angle | offset | length | twist | type | CoM | mass | inertia |
|---|---|---|---|---|---|---|---|---|
| 1 | $\theta_1$ | $d_1$ | $a_1$ | $\alpha_1$ | $\sigma_1$ | $\boldsymbol{r}^1_{1,c1}$ | $m_1$ | $I_1$ |
| 2 | $\theta_2$ | $d_2$ | $a_2$ | $\alpha_2$ | $\sigma_2$ | $\boldsymbol{r}^2_{2,c2}$ | $m_2$ | $I_2$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n$ | $\theta_n$ | $d_n$ | $a_n$ | $\alpha_n$ | $\sigma_n$ | $\boldsymbol{r}^n_{n,cn}$ | $m_n$ | $I_n$ |

**Table 6.1:** table with link properties

In table 6.1 a kinematic description of a mobile robot is shown. For every link there are parameters for its geometric properties: angle, offset, length and twist, defined in the DH-convention [12] or [3]. The joint type $\sigma$ denotes the type of joint, revolute or prismatic, and thus tells the algorithm what the actuated parameter is ($\theta$ or $d$). The center of mass $\boldsymbol{r}^i_{i,ci}$ is defined with respect to frame $i$, also the inertia is defined around the center of mass with respect to this frame as shown in Figure 6.1.

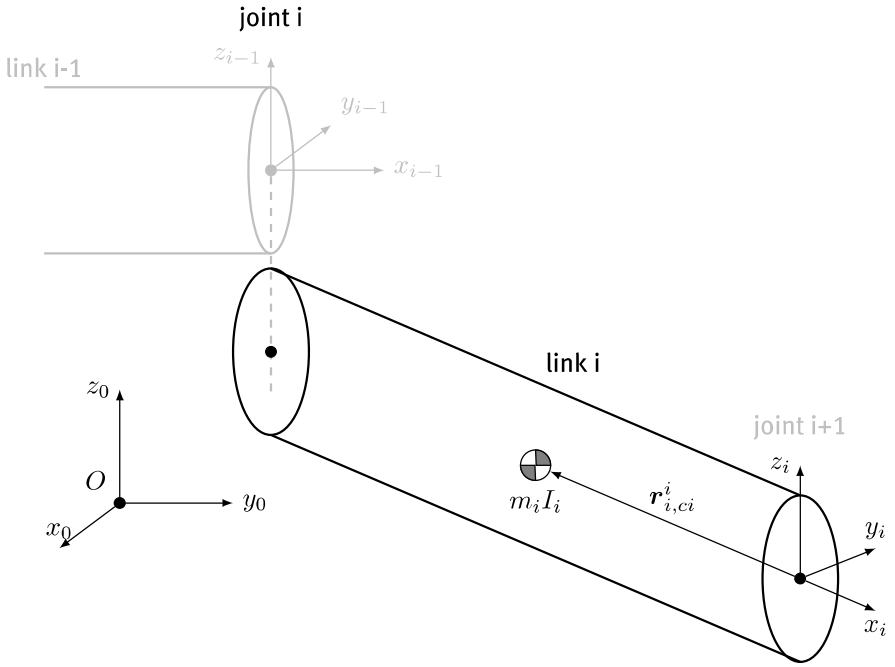



**Figure 6.1:** Link of a robotic arm (link i), reference frame $O$, link frame $i$ and center of mass $i$.

The second part of the input to the algorithm are the joint coordinates. These are: $\theta_i$, $\dot{\theta}_i$ and $\ddot{\theta}_i$ for a revolute joint and $d_i$, $\dot{d}_i$ and $\ddot{d}_i$ for a prismatic joint.

## 6.2 Calculations

The calculations needed to compute the actual Zero-Moment Point are described below, see Figure 6.2 for details about the used position vectors. All calculations are performed for each subsequent link in the robot.

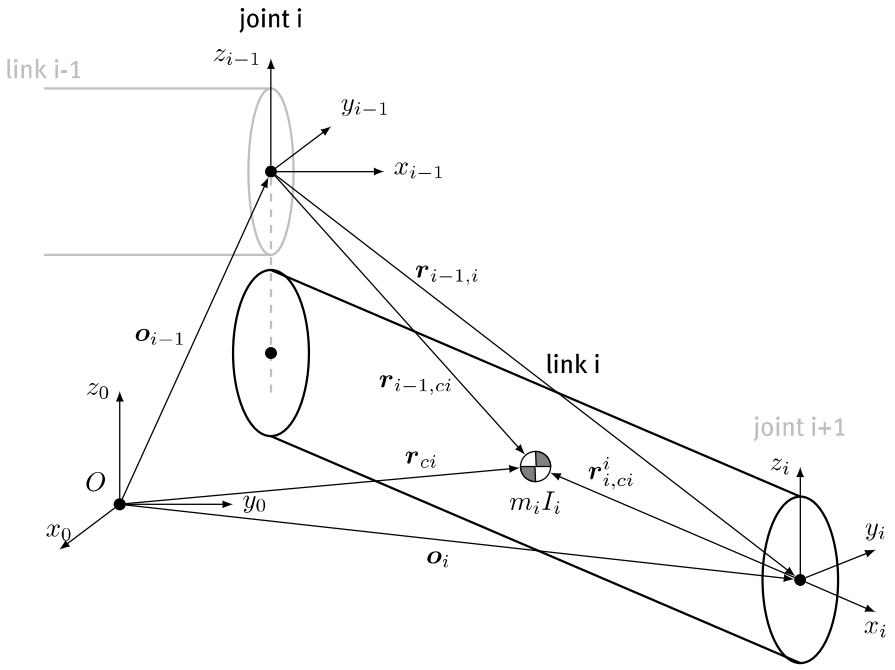



**Figure 6.2:** Position vectors for link $i$.

First, the homogeneous transformation matrix $\boldsymbol{T}_i^0$ between the reference frame $O$ and frame $i$ is constructed:

$$\boldsymbol{T}_i^0 = \boldsymbol{T}_{i-1}^0 \boldsymbol{A}_i \tag{6.1}$$

where $\boldsymbol{A}_i$ is given by (B.1) and depends on the joint coordinate of joint $i$. $\boldsymbol{T}_{i-1}^0$ is the transformation between the reference frame and frame $i-1$, which is the body attached frame for the previous link.

Next, the $\boldsymbol{z}_{i-1}$-axis is determined, this axis is given by the first three elements in the third column of $\boldsymbol{T}_{i-1}^0$:

$$\boldsymbol{z}_{i-1} = \boldsymbol{T}_{i-1}^0(1:3,3) \tag{6.2}$$

The vector $\boldsymbol{r}_{ci}$ from the reference frame to the center of mass of link $i$ is given by:

$$\begin{bmatrix} \boldsymbol{r}_{ci} \\ 1 \end{bmatrix} = \boldsymbol{T}_i^0 \begin{bmatrix} \boldsymbol{r}^i_{i,ci} \\ 1 \end{bmatrix} \tag{6.3}$$

like in (A.3), where $\boldsymbol{r}^i_{i,ci}$ is given as input in table 6.1.

The vector $\boldsymbol{r}_{i-1,ci}$ from frame $i-1$ to the center of mass of link $i$ is calculated with:

$$\boldsymbol{r}_{i-1,ci} = \boldsymbol{r}_{ci} - \boldsymbol{o}_{i-1} \tag{6.4}$$

$$= \boldsymbol{r}_{ci} - \boldsymbol{T}_{i-1}^0(1:3,4) \tag{6.5}$$

In similar fashion, the vector $\boldsymbol{r}_{i-1,i}$ from frame $i-1$ to frame $i$ is calculated using:

$$\boldsymbol{r}_{i-1,i} = \boldsymbol{o}_i - \boldsymbol{o}_{i-1} \tag{6.6}$$
$$= \boldsymbol{T}_i^0(1:3,4) - \boldsymbol{T}_{i-1}^0(1:3,4) \tag{6.7}$$

When all these position vectors are calculated, it is possible to calculate the rotational and linear velocities and accelerations for the center of mass and end of each link. For the calculation of the change in linear and angular momenta ($\dot{\boldsymbol{P}}$ and $\dot{\boldsymbol{H}}$), the velocity and acceleration of the center of mass are of interest, but from section 5.1 it is clear that the velocity and acceleration of the (end of) previous link are also needed. That is, the velocity and acceleration of frame $i-1$ with position vector $\boldsymbol{r}_{i-1} = \boldsymbol{o}_{i-1}$ in Figure 6.2.

For a revolute joint this results in (see also appendix C.2):

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \boldsymbol{z}_{i-1} \tag{6.8}$$
$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_i \times \dot{\theta}_i \boldsymbol{z}_{i-1} + \ddot{\theta}_i \boldsymbol{z}_{i-1} \tag{6.9}$$
$$\ddot{\boldsymbol{r}}_{ci} = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,ci} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,ci}) \tag{6.10}$$
$$\ddot{\boldsymbol{r}}_i = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) \tag{6.11}$$

For a prismatic joint this gives (see also appendix C.3):

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} \tag{6.12}$$
$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} \tag{6.13}$$
$$\ddot{\boldsymbol{r}}_{ci} = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,ci} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,ci}) + 2\boldsymbol{\omega}_i \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} \tag{6.14}$$
$$\ddot{\boldsymbol{r}}_i = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) + 2\boldsymbol{\omega}_i \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} \tag{6.15}$$

With these velocities and accelerations, the changes in linear and angular momenta can be calculated. The changes in the linear and angular momenta are summations over all links, see (4.42) and (4.44). This gives for link $i$:

$$\dot{\boldsymbol{P}}_i = \dot{\boldsymbol{P}}_{i-1} + m_i \ddot{\boldsymbol{r}}_{ci} \tag{6.16}$$
$$\dot{\boldsymbol{H}}_i = \dot{\boldsymbol{H}}_{i-1} + \boldsymbol{r}_{ci} \times m_i \ddot{\boldsymbol{r}}_{ci} + \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{I}_i \boldsymbol{\omega}_i) \tag{6.17}$$

where $\boldsymbol{I}_i$ is given by (4.40) as $\boldsymbol{I}_i = \boldsymbol{R}_i I_i \boldsymbol{R}_i^T$, and $I_i$ is an input from table 6.1.

Now the position of the Zero-Moment Point can be calculated using (4.36) and (4.37):

$$r_{P,x} = \frac{-\dot{H}_y + r_{c,z} m g_x - r_{c,x} m g_z}{\dot{P}_z - m g_z}$$
$$r_{P,y} = \frac{\dot{H}_x - r_{c,y} m g_z + r_{c,z} m g_y}{\dot{P}_z - m g_z}$$

Here, $m$ and $\boldsymbol{r}_c$ are calculated using (4.24) and (4.25).

## 6.3 Summary

The ZMP method described in chapter 4 together with the forward kinematics and dynamics described in chapter 5 are used to design an algorithm that computes the location of the ZMP. This algorithm needs a description of the robot joint using the DH-convention, mass and inertia parameters and joint coordinates to calculate link dynamics and the location of the ZMP. This algorithm is implemented in Matlab and used to simulate some robot movements, see chapter 8.

The developed ZMP algorithm can be integrated in ROS, which is a requirement from the Fontys mechatronics lab.

# 7 SimMechanics Test Framework

By means of the developed ZMP algorithm in section 6 one of the goals of this project has been fulfilled as this algorithm can be used on a light weight PC on a mobile robot itself. However, the ZMP algorithm does not take into account flexibilities in the robot mechanism. At present the effect of these flexibilities, such as robot link stiffness, control stiffness and suspension stiffness, are unknown. Therefor, to validate the ZMP algorithm and to be able to simulate flexibilities in the robot mechanism, a SimMechanics test framework has been designed. This framework uses a different method to calculate the ZMP (see section 7.2) and uses standard SimMechanics blocks to calculate link dynamics, but can use exactly the same input. Results from the ZMP algorithm and this framework should be exactly the same. Furthermore, in SimMechanics flexibility can easily be added to a robot model by adding springs, dampers and controllers. Hereby, the effects of neglecting these robot flexibilities in the ZMP algorithm can be studied.

The disadvantage of a Simulink based method is that it can only run on a PC with Matlab installed. Therefor, this method is not suitable for *online* use on mobile robot systems in the Fontys mechatronics lab.

This test framework consists of three main components, the base transform, the ZMP calculation and the robot links, see Figure 7.1.

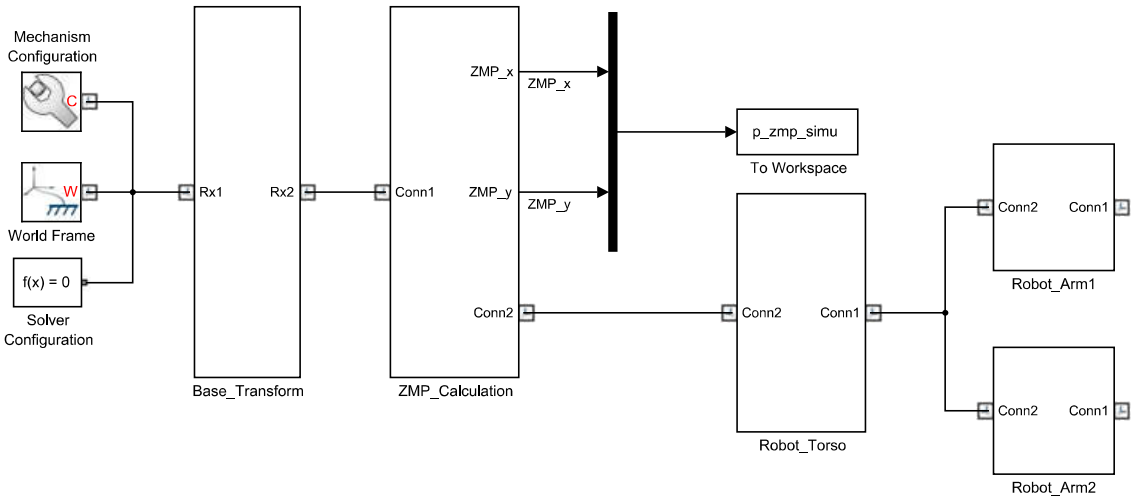



Figure 7.1: SimMechanics test framework overview

The first part (the base transform) is the part of the framework that rotates the ground contact when the robot is on an inclined floor and it accelerates this ground contact when the robot accelerates. So it facilitates all movements of the mobile robot base.

The next part is the ZMP calculation, this part uses a SimMechanics joint to calculate the center of pressure between the robot and the moving ground contact from the first part.

The last part models the dynamics of the robotic links. A torso with arms for instance. These are modeled using the SimMechanics joints and inertias, to be able to use the Denavit-Hartenberg convention as the input. In Figure 7.1 the `Robot_Torso` block for instance consists of 2 links, the `Robot_Arm` blocks consist of four links each. These blocks can be stacked onto each other to create more complicated robots.

## 7.1 Base Transform

The base transform is simply a set of rotations and translations to describe the motion of the robot base (driving), see Figure 7.2. The first three consecutive rotations are performed to enable the simulation of driving on an inclined floor. The next are three consecutive translations that enable the robot to

drive with a prescribed velocity and acceleration. These rotations and translations are generated with lookup tables from the prescribed position, velocity and acceleration for each rotation or translation.

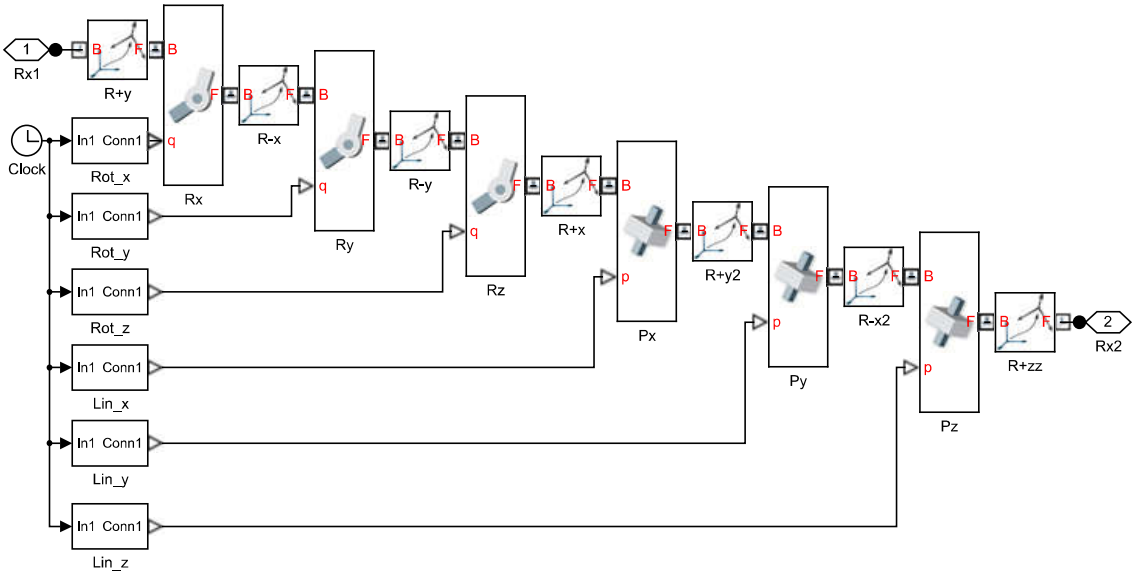



Figure 7.2: SimMechanics base transform

## 7.2 ZMP Calculation

Computation of the ZMP is done by calculating the center of pressure on the moving support. As long as the ZMP is within the support polygon it coincides with the center of pressure [4].

In SimMechanics it is possible to use a 6 DOF bushing joint that enables rotation around the x, y and z-axes and translation along these axes. This joint type has also the ability to compute the needed torque and force to perform a certain motion. In this case the joint is kept still (zero rotation and translation for all DOF), and it calculates the forces and torques needed to keep it still. This results in a reaction force and reaction moment on this joint of:

$$\boldsymbol{F}_R = [F_x\, F_y\, F_z]^T \tag{7.1}$$

$$\boldsymbol{M}_R = [M_x\, M_y\, M_z]^T \tag{7.2}$$

These forces and moments can be used to calculate the ZMP. Recall equations 4.9 and 4.10:

$$\boldsymbol{F}_P + \boldsymbol{F}_R = 0$$

$$\boldsymbol{M}_P + \boldsymbol{M}_R + \boldsymbol{r}_P \times \boldsymbol{F}_P + \boldsymbol{r}_R \times \boldsymbol{F}_R = 0$$

where $\boldsymbol{F}_P$ and $\boldsymbol{M}_P$ are a force and moment in point $P$ at $\boldsymbol{r}_P$ (this is the Zero-Moment Point) to keep the robot in balance. In this case the point $R$ at $\boldsymbol{r}_R$ coincides with the joint location by definition, so $\boldsymbol{r}_R = [0\,0\,0]^T$.

The moment $\boldsymbol{M}_P$ in point $P$ is zero by definition, which leads to (see section 4.2):

$$r_{P,y} F_{R,z} = M_{R,x} \tag{7.3}$$

$$r_{P,x} F_{R,z} = -M_{R,y} \tag{7.4}$$

This is used in SimMechanics to calculate the ZMP ($\boldsymbol{r}_P$), as it can be seen in Figure 7.3. Here $f_z$ is the force in the z-direction of the joint, and $t_x$ and $t_y$ are torques around the x- and y-axis respectively.

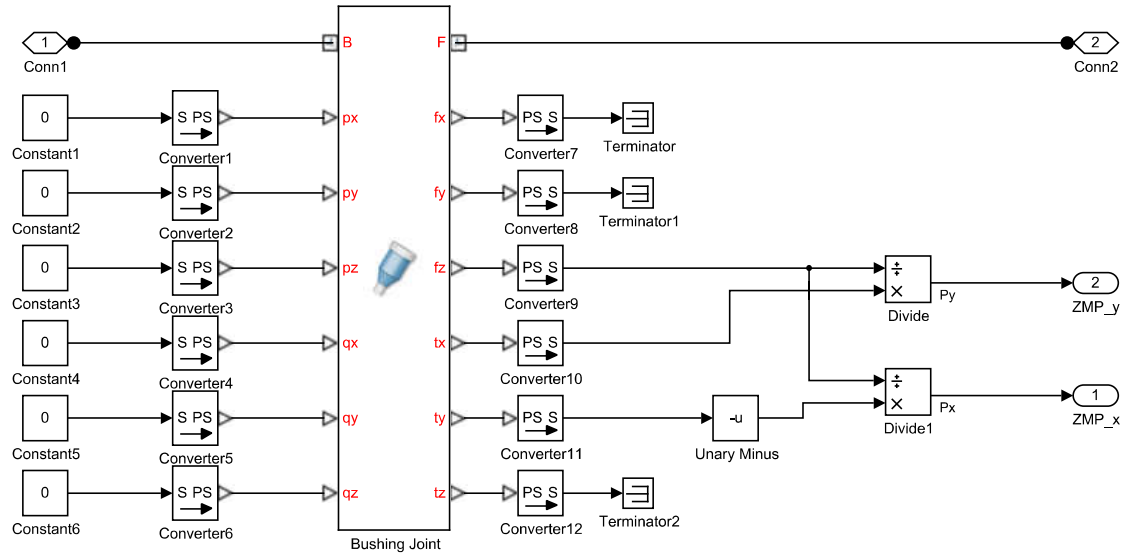



Figure 7.3: SimMechanics ZMP calculation

## 7.3 Robot Links

The last part in the SimMechanics test framework is a robot link. This link is build with two joints and two transforms and has a mass and inertia, as shown in Figure 7.4.

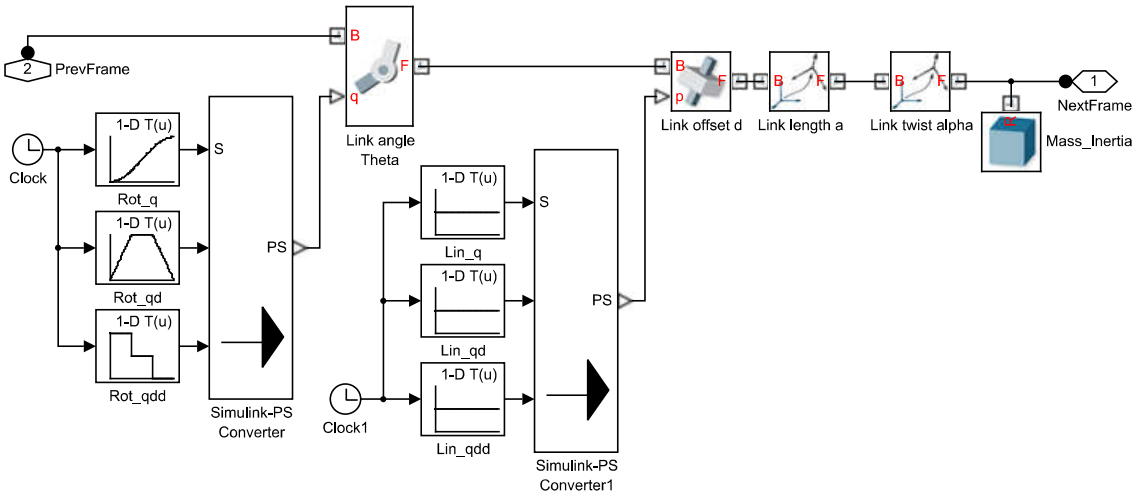



Figure 7.4: SimMechanics robot link

The first joint is a revolute joint that represents the rotation around the z-axis with joint angle $\theta$ as defined by the Denavit-Hartenberg convention in appendix B. The next joint implements the link offset $d$. Only one of these joints is actuated, the other has a static value. The joint actuation is implemented with lookup tables for position, velocity and acceleration.

The last two transformations are the link length $a$ and the link twist $\alpha$. These two are always static.

As many links as necessary can be added to each other to create kinematic chains. As it can be seen in Figure 7.1 these chains can also be added together and allow for diverging chains.

When joint flexibility is investigated, it is possible to add an extra joint at the beginning of the

chain. This joint can be given an internal spring stiffness and a damping ratio. SimMechanics can now compute the effects of this flexibility in the robot.

## 7.4 Summary

A test framework is designed in SimMechanics that allows quick modeling and simulation of a mobile robot using the same inputs as for the ZMP algorithm. This SimMechanics framework also facilitates the possibility to include various forms of stiffness into the robot model, which is not possible with the ZMP algorithm.

# 8 Simulations

The ZMP algorithm as described in section 6 is investigated in simulations in this chapter. Also, simulation results from a SimMechanics model are compared with the results obtained using the analytical formulas for the ZMP.

First a simple robot is constructed to compare analytic results with results of the ZMP algorithm, to exemplify the difference between the center of mass and the ZMP, and to compare results obtained using the SimMechanics model with the results determined using the ZMP algorithm.

Next a more complicated robot is analyzed, and also joint flexibility is considered.

## 8.1 Pole on Cart Simulation

First a simple robot is constructed and simulated. With this robot it is relatively easy to calculate the Zero-Moment Point by hand, so the algorithm can be checked. The robot is a simple moving base with an upright pole (torso) on top. The pole is attached to the moving base slightly to the front of the base, see Figure 8.1.

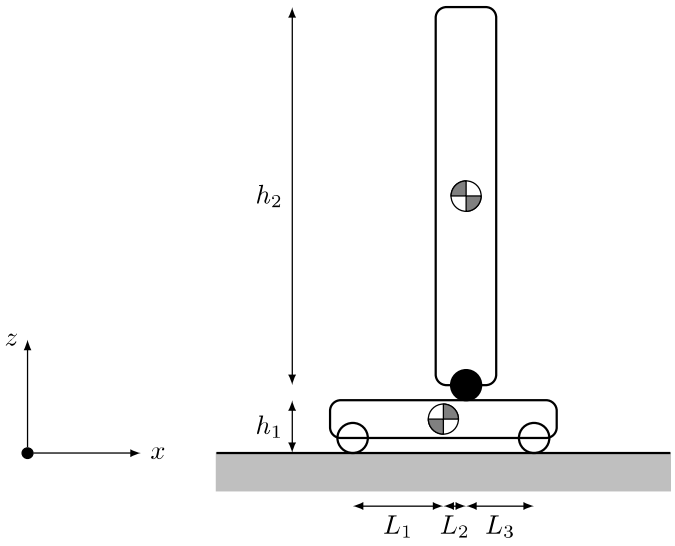



Figure 8.1: Pole on Cart

The location of the Zero-Moment Point with respect to the center of the base of the robot can be calculated with equations (4.24) for the total mass ($m$), (4.25) for the center of mass of the total robot ($\boldsymbol{r}_c$), (4.42) and (4.44) for the change in linear ($\dot{\boldsymbol{P}}$) and angular ($\dot{\boldsymbol{H}}$) momenta, respectively, and (4.36) for the x-position of the Zero-Moment Point ($r_{P,x}$).

When the acceleration of the robot base is denoted by $\ddot{q}$, while $\boldsymbol{r}_{ci}$ and $\ddot{\boldsymbol{r}}_{ci}$ are the center of mass and acceleration of the center of mass per link, respectively, the following relations for the base of the

robot can be given:

$$\boldsymbol{r}_{c1} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}h_1 \end{bmatrix} \tag{8.1}$$

$$\ddot{\boldsymbol{r}}_{c1} = \begin{bmatrix} \ddot{q} \\ 0 \\ 0 \end{bmatrix} \tag{8.2}$$

$$\dot{\boldsymbol{P}}_1 = m_1 \ddot{\boldsymbol{r}}_{c1} = \begin{bmatrix} m_1\ddot{q} \\ 0 \\ 0 \end{bmatrix} \tag{8.3}$$

$$\dot{\boldsymbol{H}}_1 = \boldsymbol{r}_{c1} \times m_1 \ddot{\boldsymbol{r}}_{c1} = \begin{bmatrix} 0 \\ \frac{1}{2}h_1 m_1 \ddot{q} \\ 0 \end{bmatrix} \tag{8.4}$$

Similar holds for the torso:

$$\boldsymbol{r}_{c2} = \begin{bmatrix} L_2 \\ 0 \\ h_1 + \frac{1}{2}h_2 \end{bmatrix} \tag{8.5}$$

$$\ddot{\boldsymbol{r}}_{c2} = \begin{bmatrix} \ddot{q} \\ 0 \\ 0 \end{bmatrix} \tag{8.6}$$

$$\dot{\boldsymbol{P}}_2 = m_2 \ddot{\boldsymbol{r}}_{c2} = \begin{bmatrix} m_2\ddot{q} \\ 0 \\ 0 \end{bmatrix} \tag{8.7}$$

$$\dot{\boldsymbol{H}}_2 = \boldsymbol{r}_{c2} \times m_2 \ddot{\boldsymbol{r}}_{c2} = \begin{bmatrix} 0 \\ (h_1 + \frac{1}{2}h_2) m_2 \ddot{q} \\ 0 \end{bmatrix} \tag{8.8}$$

Using (4.36) and $g_x = 0,\ g_z = -g$ leads to:

$$r_{P,x} = \frac{-\dot{H}_y + r_{c,z} m g_x - r_{c,x} m g_z}{\dot{P}_z - m g_z} = \frac{-\dot{H}_y + r_{c,x} m g}{m g} \tag{8.9}$$

where the change in the angular momentum around the y-axis is:

$$\dot{H}_y = (\tfrac{1}{2}h_1 m_1 + (h_1 + \tfrac{1}{2}h_2) m_2) \ddot{q} \tag{8.10}$$

and the location of the center of mass in the x-direction is:

$$r_{c,x} = \frac{m_2 L_2}{m1 + m2} \tag{8.11}$$

This indicates that for a positive acceleration of the robot (positive $\ddot{q}$) the angular momentum part is shifting the Zero-Moment Point backwards, while the center of mass part is shifting the ZMP towards the front (to $L_2$). With no acceleration of the robot the ZMP will coincide with the projection of the center of mass.

A simulation is made with the following parameters:

| | | |
|---|---|---|
| $L_1 = 0.15$ | $h_1 = 0.05$ | $m_1 = 2$ |
| $L_2 = 0.05$ | $h_2 = 0.5$ | $m_2 = 6$ |
| $L_3 = 0.1$ | | |

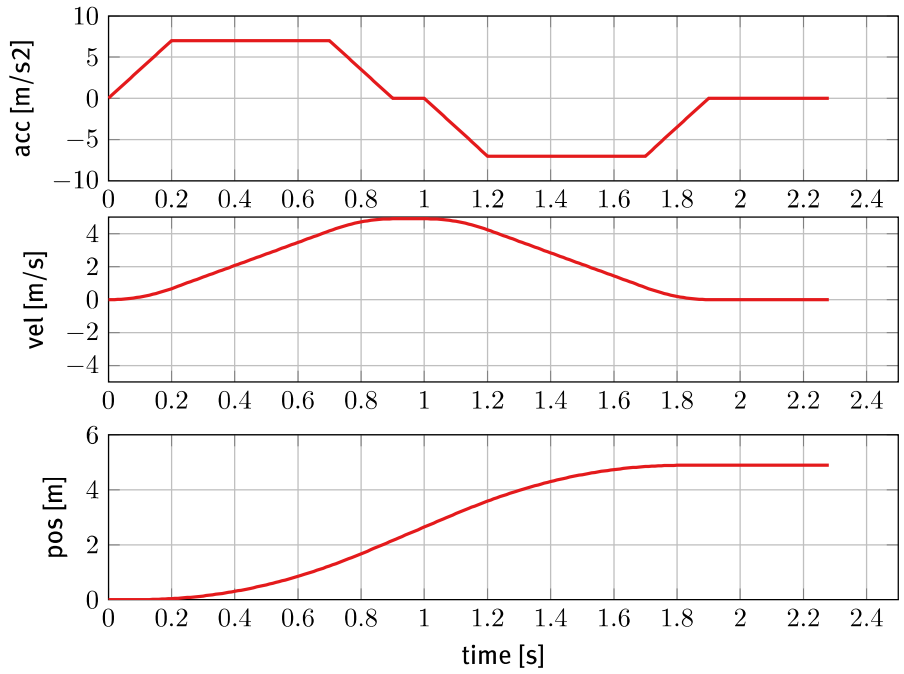



Figure 8.2: Motion profile pole on cart

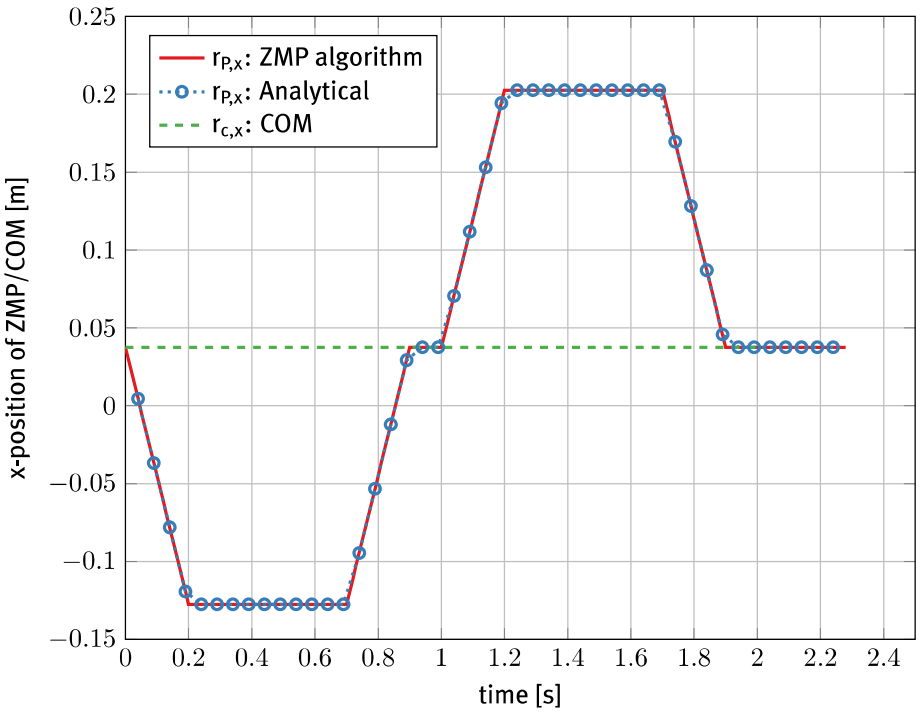



Figure 8.3: Zero-Moment Point and Center of Mass for pole on cart model

The mobile robot moves with the 3rd-order motion profile shown in Figure 8.2. The resulting location of the Zero-Moment Point and the location of the center of mass are shown in Figure 8.3.

In this figure, results of simulation using the ZMP algorithm and with the analytic solution from equations (8.9) and (8.10) are shown. Both results match perfectly, which indicates that the ZMP algorithm is usable in this case.

From Figure 8.3 it is clear that the acceleration of the robot has a great influence on the location of the ZMP. The center of mass itself would indicate that the robot would be balanced along the entire movement, because it is always in between the rear wheels ($-L_1$) and the front wheels ($L2 + L3$). The support polygon is the area between $x = -0.15$ and $x = 0.15$. The Zero-Moment Point however gets ahead of the front wheels of the robot when decelerating at around 1.1 seconds. This indicates that the robot would start tipping over at that moment. It is important to note that while the ZMP criterion guarantees a balanced robot when the ZMP stays inside the support polygon, the opposite is not necessarily true; when the ZMP leaves the support polygon it is still possible that the robot stays balanced (not tipping over completely).

### 8.1.1 Balance by Torso Leaning

Figure 8.3 shows that the ZMP leaves the support polygon while decelerating, and thus starts to tip over. When a joint is added between the base of the robot and the torso, such that the torso can rotate about the y-axis (lean forward and backward), it becomes possible to balance the robot by rotating the torso. By shifting the center of gravity to the front while accelerating and to the back while decelerating, the ZMP shifts to the center of the robot compared to the stationary torso situation. This can be understood from equation (8.9) and (8.10) where the effects of $\dot{H}_y$ and $r_{c,x}$ can counteract each other.

A simulation is carried out where the torso on the cart rotates as described above, this gives the results shown in Figure 8.4. This figure shows that the ZMP indeed shifts towards the center of the robot, and the ZMP never leaves the support polygon. The center of mass of the robot gets in front of the front wheels of the robot while accelerating, which in a static situation would lead to an unbalanced robot.

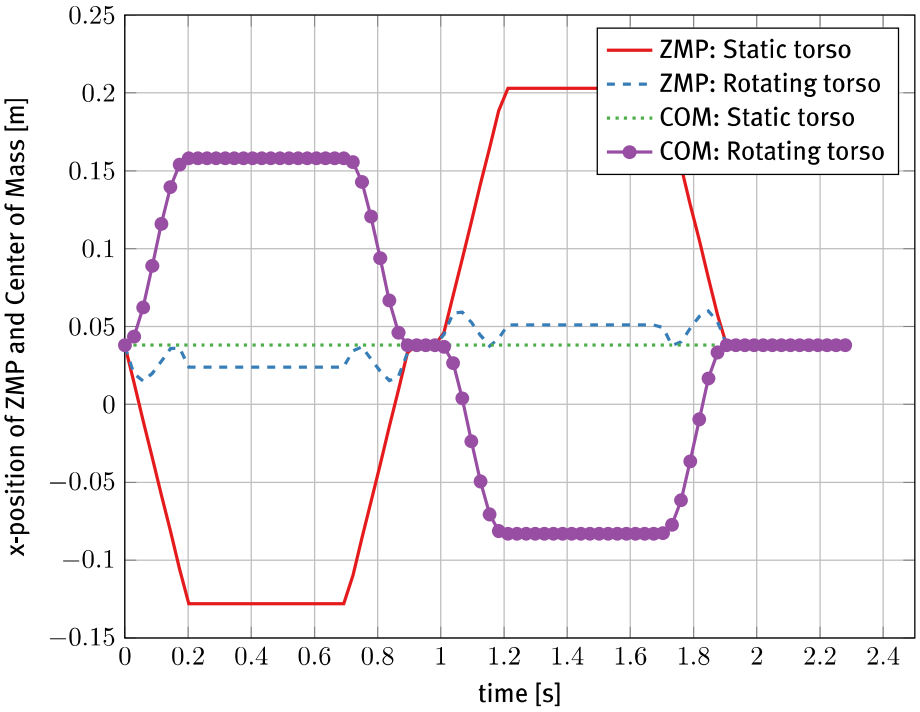



**Figure 8.4:** Zero-Moment Point and Center of Mass for pole on cart model, with static torso and rotating torso

This simulation shows that both acceleration of the robot as well as its center of mass are influencing the location of the ZMP. A statically balanced robot could become unbalanced in a dynamical situation and vice versa.

### 8.1.2 SimMechanics Simulation

As the last check, a SimMechanics simulation is executed. The difference between the SimMechanics model and the ZMP algorithm is that the SimMechanics model can simulate the robot tipping over. To achieve this, a hinge is added at the edge of the support polygon of the robot (at $x = 0.15$) which enables the robot base to rotate around its y-axis (direction into the paper in Figure 8.1). The resulting rotation around this y-axis of the robot base can be seen in Figure 8.5. This kind of rotation indicates that the entire robot is tipping over.

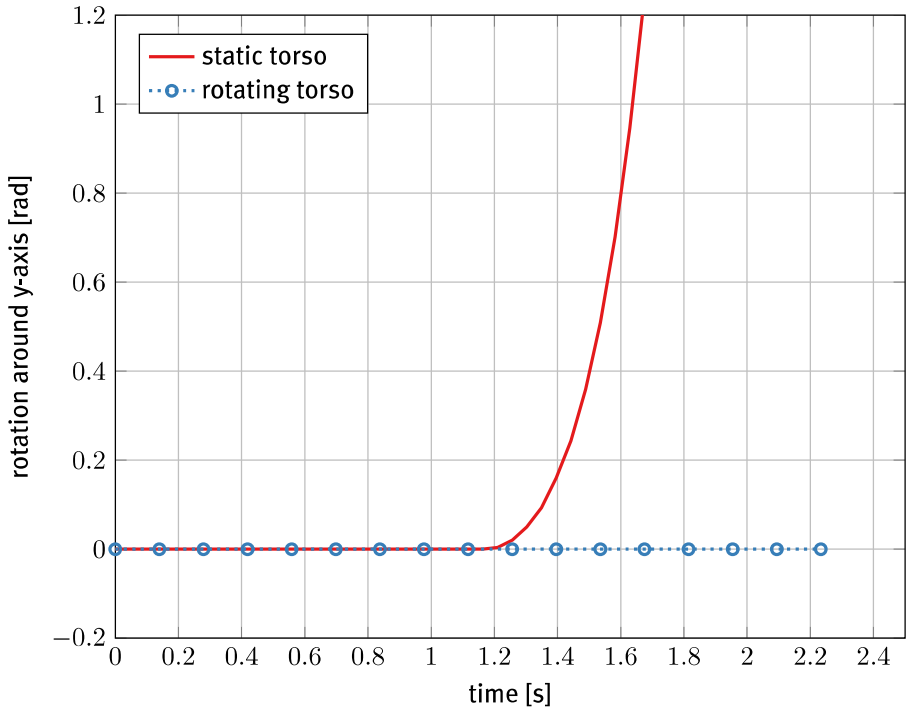


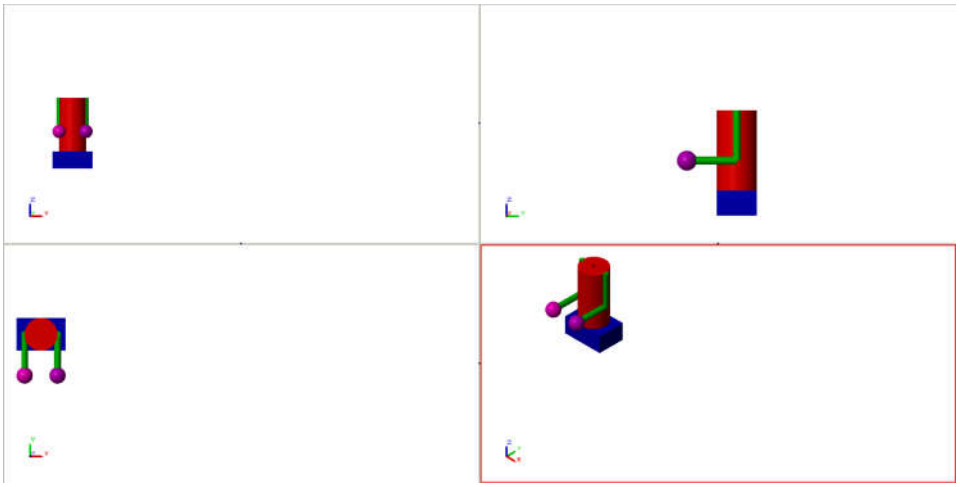

**Figure 8.5:** rotation of the robot base about its y-axis for static torso and leaning torso

By comparing Figures 8.4 and 8.5 it can be noticed that the robot with the static torso indeed starts to tip over when the ZMP is outside the support polygon around 1.1 seconds. The robot with the rotating torso stays perfectly horizontal, and thus, stays balanced.

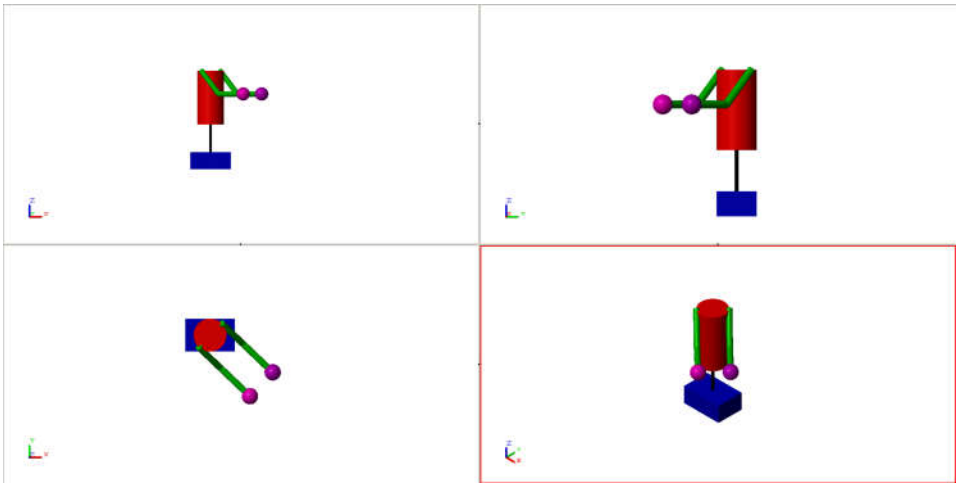
## 8.2 Mobile Service Robot Simulation

A more complex robot is designed with a more challenging motion profile, to compare the ZMP algorithm results with the SimMechanics results. The robot is a wheeled mobile robot, with a heavy base, a torso that can translate in the vertical direction and can rotate about this axis. It also has two arms consisting each of an upper and lower arm. In the starting position the robot has lowered its torso and turned it sideways. It has its arms in a ninety degree angle, with a load on each 'hand'. The robot moves along a straight line, while elevating its torso and rotating it towards the moving direction. It stretches its arms forward while driving, see Figure 8.6.

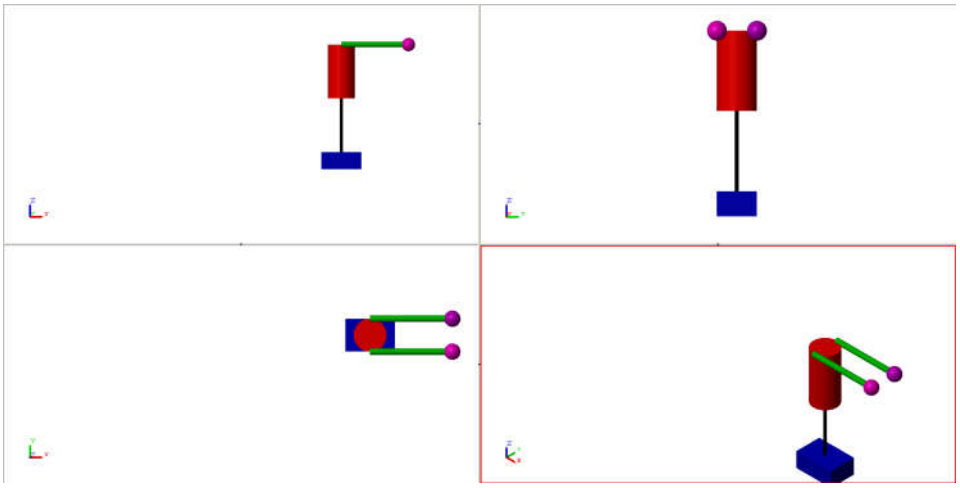
All joints of the robot and also the mobile base of the robot are actuated such as to follow a second order motion profile like the one shown in Figure 8.7. Because of the step-like profile of the joint



(a) Starting pose



(b) Halfway pose



(c) Final pose

**Figure 8.6:** Mobile robot in the starting, halfway and final pose

acceleration, there will be steps in the joint torques that causes sudden changes in Zero-Moment Point position.

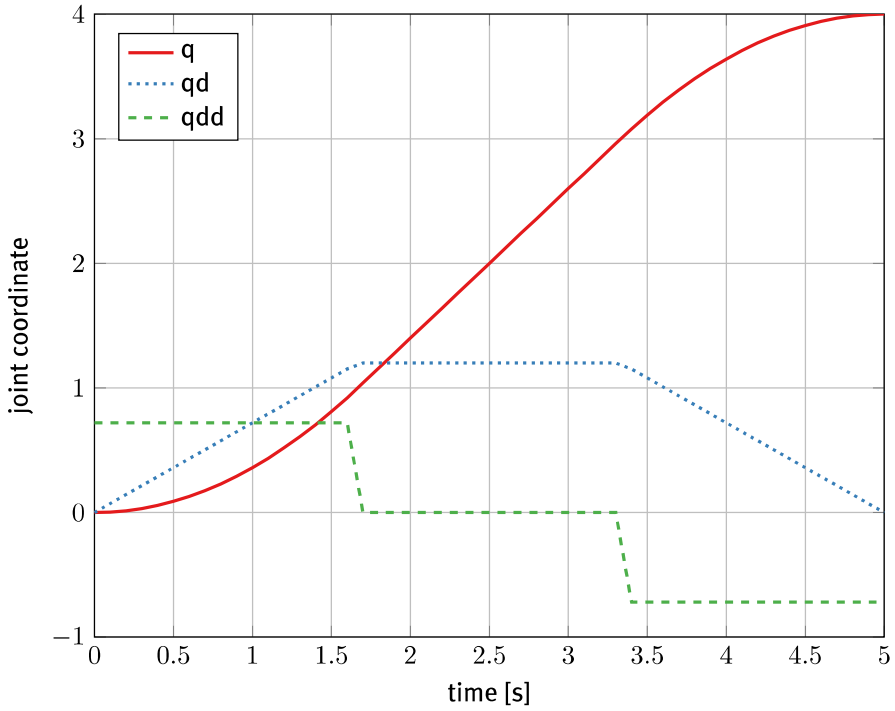



Figure 8.7: Motion profile for the moving platform

In the simulation case-studies considered in this section, it is assumed that the robot joints follow the motion profile exactly. There is no actuator-, control- or link-flexibility. This means that the ZMP trajectories computed using the ZMP algorithm and determined from the SimMechanics model should *exactly* be the same as long as the ZMP is within the support polygon of the robot.

The definition of the ZMP given by equation (4.3) assumes that there exists an equilibrium between all forces on the robot and a reaction force within the support polygon of the robot. When the calculated ZMP leaves the support polygon of the robot, this assumption is not true and thus this definition for the Zero-Moment Point can not be used anymore.

### 8.2.1 Simulation for an Intrinsically Balanced Mobile Robot

In this simulation the SimMechanics model does not have the ability to tip-over, the robot base has only one degree of freedom: translation along the global x-axis. Without the ability to tip over (rotation around the x- or y-axes) the robot behaves as if the base of the robot is glued to the ground or as if the support polygon is always big enough for the ZMP to fit into. The ZMP algorithm and SimMechanics model should give exactly the same results in this simulation.

The resulting Zero-Moment Point is calculated and shown in Figure 8.8. The start of the constant velocity phase is marked with black dots in this figure, the end of the constant velocity phase is marked with crosses. It can be seen that the ZMP algorithm and the SimMechanics model give the same results. It can also be seen that the x-position of the Zero-Moment Point has two sudden steps around $x = 0\,[m]$ and $x = 0.2\,[m]$ due to the steps in joint acceleration.

In this figure the center of mass of the robot is also shown. The robot is turning its torso and stretching its arms, so it is expected that the center of mass travels in a circle with increasing radius. The figure confirms that.

When the robot starts its movement, it is accelerating (left part of Figure 8.8, before the constant velocity phase), which shifts its ZMP backwards (-x direction) with respect to the center of mass.

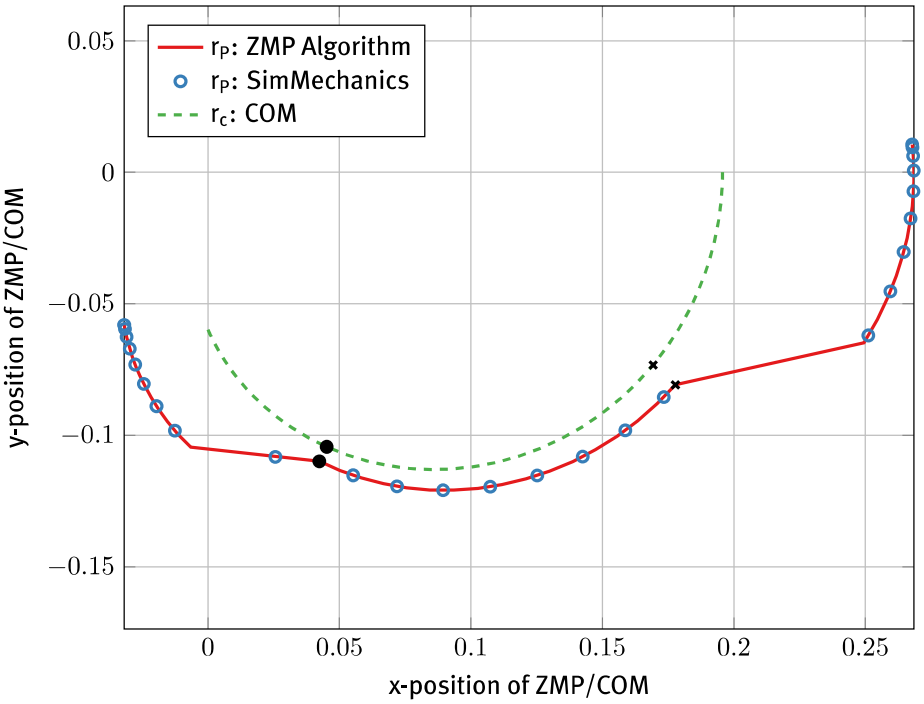



**Figure 8.8:** Zero-Moment Point calculated with ZMP algorithm and SimMechanics. The center of mass is calculated with the ZMP algorithm. The black dots mark the start of the constant velocity phase, the crosses mark the end of the constant velocity phase.

After this acceleration phase, the robot enters a constant velocity phase, all joints and the mobile base move with constant velocity. In the simple pole on cart model of Figure 8.4, the ZMP and center of mass coincided for the constant velocity phase due to the absence of accelerations. In the case of this service robot however, there are rotations involved with constant rotational velocity. An example shows what happens in that case:

Consider a point mass $m$ at position $\boldsymbol{r} = [0,\, r_y,\, r_z]^T$ which rotates around the z-axis with a constant rotational velocity $\boldsymbol{\omega} = [0,\, 0,\, \omega_z]^T$. This point mass undergoes an acceleration $\ddot{\boldsymbol{r}}$, given by equation (6.10):

$$\ddot{\boldsymbol{r}}_{ci} = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,ci} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,ci})$$

For $\ddot{\boldsymbol{r}}_{i-1} = 0$ and $\dot{\boldsymbol{\omega}} = 0$ the following is obtained:

$$\ddot{\boldsymbol{r}} = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}) = \begin{bmatrix} 0 \\ -\omega_z^2 r_y \\ 0 \end{bmatrix} \tag{8.12}$$

So this point mass undergoes an acceleration in the y-direction. For the linear and angular momenta this results in (using equations (4.42) and (4.44)):

$$\dot{\boldsymbol{P}} = m\ddot{\boldsymbol{r}} = \begin{bmatrix} 0 \\ -m\omega_z^2 r_y \\ 0 \end{bmatrix} \tag{8.13}$$

$$\dot{\boldsymbol{H}} = \boldsymbol{r} \times m\ddot{\boldsymbol{r}} = \begin{bmatrix} m\omega_z^2 r_y r_z \\ 0 \\ 0 \end{bmatrix} \tag{8.14}$$

The constant rotational velocity of this point mass results in a change of the linear momentum in the y-direction, and a change of the angular momentum in the x-direction. Comparing this with the ZMP computed using equations (4.36) and (4.37) gives a shift in the y-location of the ZMP:

$$r_{P,y} = \frac{\dot{H}_x - r_{c,y} m g_z + r_{c,z} m g_y}{\dot{P}_z - m g_z}$$

This effect can be seen in the middle part (constant velocity phase) of Figure 8.8. The shift is negative in the y-direction because the robot has its arms in the negative y part (a negative value of $r_y$ in the point mass example).

When the robot starts decelerating (the right hand side of the figure), the ZMP shifts forwards (+x direction) with respect to the center of mass. The shift is bigger than during accelerating due to the different pose the robot has. The robot has increased the height of its torso and raised its arms.

This simulation is a nice demonstration of the advantages of the use of the Zero-Moment Point over the center of mass. The center of mass is closer to the center of the robot than the ZMP during the entire simulation. For example, if an edge of the support polygon of the robot would be located at $x = 0.2$, the center of mass would stay within this support polygon while the ZMP would leave it. Thus the center of mass method gives a too optimistic evaluation of balance.

### 8.2.2 Simulation of a Mobile Robot with a Small Support Polygon

The Zero-Moment Point as computed and shown in Figure 8.8 should indicate the balance of the robot according to theory, see section 4 or [11, 13, 15, 17]. This means that when the ZMP is within the support polygon it guarantees that the robot is in balance, it does not tip-over. When the ZMP leaves the support polygon the robot starts tipping over, but it could still be balanced and not fall over completely. To check this, another simulation is performed in which the robot has a support polygon and can tip-over in SimMechanics. A hinge is added to the base of the robot in the SimMechanics model. The hinge enables the robot to rotate around the side of its base (tipping over its x-axis), this hinge defines the support polygon of the robot. So when the ZMP crosses the axis of this hinge, the robot starts tipping over. The hinge is placed at $0.1m$ from the center of the robot in the -y direction.

The results of this simulation are shown in Figure 8.9. The first thing to notice is the difference between the ZMP algorithm and the SimMechanics results at the start of the simulation. This is an effect of the added hinge, this hinge is modeled as a very stiff rotational spring-damper and as such generates some parasitic oscillation. It can be seen that the ZMP crosses the support polygon at $y = -0.1$ between 1.5 and 3 sec. The ZMP criterion does not guarantee balance anymore in this period, and indicates that the robot would start tipping over. The SimMechanics model calculates the center of pressure instead of the ZMP (see section 7.2), which coincides with the ZMP as long as the ZMP is within the support polygon. When the ZMP goes outside the support polygon, it does not have any meaning anymore, whereas the center of pressure is still correct. The fact that the center of pressure stays at the edge of the support polygon after 1.5 seconds and never returns within the support polygon, means that the robot is tipping over all that time. This still does not guarantee that the robot is out of balance and will fall over completely.

In Figure 8.10 the rotation of the robot about its x-axis is shown. It can be seen that the robot starts tipping over when the ZMP crosses the axis of the hinge. When looking at the angle of the z-axis it becomes clear that the robot has fallen over completely. This figure, in combination with Figure 8.9, indicates that the Zero-Moment point in this case is an adequate qualifier for balance, as the robot starts to tip-over when the ZMP leaves the support polygon.

### 8.2.3 Simulation of a Mobile Robot with a Slightly Bigger Support Polygon

A next test is conducted where the hinge in the SimMechanics model is moved slightly and placed at $y = -0.117\,[m]$, thus enlarging the support polygon.

The resulting y position of the ZMP is shown in Figure 8.11, the rotation of the robot about its x-axis is shown in 8.12. From these figures it can be seen that the robot starts tipping over but recovers

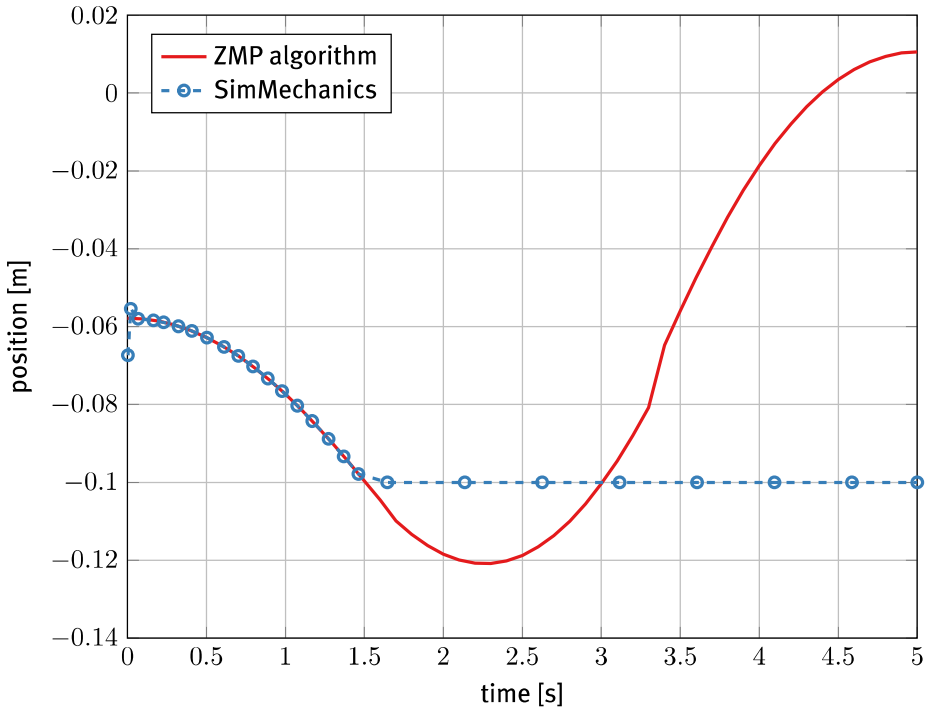



**Figure 8.9:** y-position of Zero-Moment Point, hinge at y = -0.1 [m]

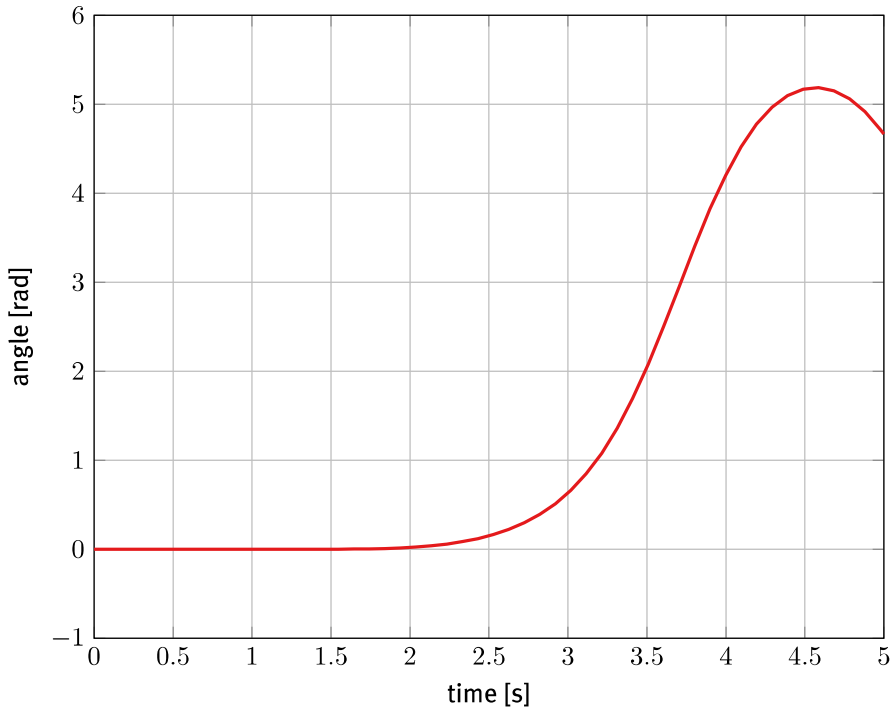



**Figure 8.10:** Rotation of robot base about x-axis, hinge at y = -0.1 [m]

again. The small hiccup in the rotation of the robot about its x-axis around 4.5 seconds is a result of a spring-damper construction used in SimMechanics to model the hinge rotational stops.

The ZMP only leaves the support polygon for less than a second and is a maximum of $3\,mm$ away from the support polygon, and the robot is at the brink of falling over completely. A slightly smaller support polygon will result in a total crash. This shows that the margin for error in these calculations is small. It is advisable to always use some kind of safety margin around the calculated ZMP such that small errors in the robot model do not have a large impact on the actual balance of the robot.

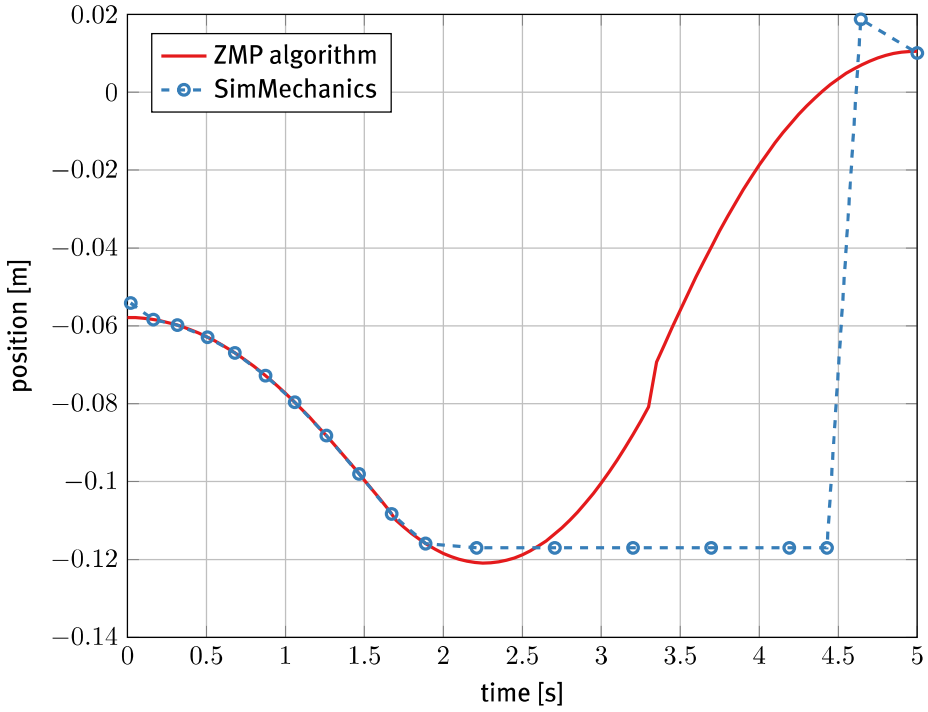



**Figure 8.11:** y-position of the Zero-Moment Point, hinge at y = -0.117 [m]

### 8.2.4 Concluding Remarks on Comparison Between the ZMP Algorithm and SimMechanics Model

The ZMP algorithm and the SimMechanics model give the same results for all above tests, they only differ when the ZMP is outside the support polygon, i.e. when the robot start to tip-over. This means that it is possible to only use the ZMP algorithm for online balance prediction, and only use the SimMechanics model for offline verifications. It has to be stated again that in all simulations so far, joint flexibility and link flexibility are not considered.

## 8.3 Joint Flexibility

The assumption that there is no joint flexibility is only acceptable when the stiffness of the joint actuation is so high that joint coordinate deviations with respect to the motion profile are relatively small. Next to joint flexibility, link flexibility may also influence the robot balance. In the ZMP algorithm it is assumed that all links are rigid and that the joint coordinates are perfectly known. In reality the links have limited stiffness and also the accuracy of the actual joint coordinates is limited by resolution of position sensors.

Since the ZMP algorithm considered in this report does not include flexible dynamics of the robot links and joints, it has to be investigated how these dynamics influence the Zero-Moment Point in SimMechanics. Such an investigation may imply that additional safety margin around the computed ZMP has to be taken to account for the robot flexibilities.

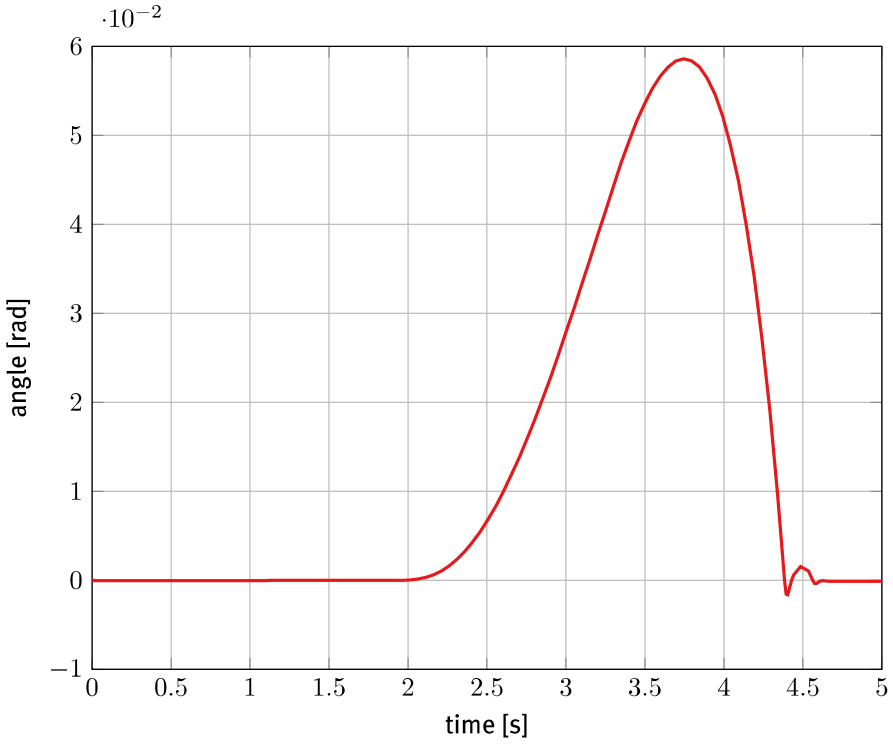



**Figure 8.12:** Rotation of the robot around its x-axis, hinge at y = -0.117 [m]

To mimic flexibility in the robot, some joints with joint-stiffness have been added to the SimMechanics model. A universal joint with two rotational degrees of freedom (around the x-axis and y-axis) is added at the base of the robot to simulate the suspension of the mobile robot base. The stiffness parameters of this joint are chosen such that the resulting natural frequency due to this joint is at 3 Hz, which corresponds with the findings of Bouten [2]. The damping ratio is chosen to be 0.1, which results in a not well damped response.

All the robot joints (translation and rotation of the torso, rotation of the arms) are given joint flexibility with a stiffness resulting in a natural frequency of 10 Hz with a damping ratio of 0.7. All these parameters are chosen such as to accentuate the contribution of the flexibilities to the total robot dynamics. In reality these parameters depend on natural stiffness of the robot joints and actuators, forced stiffness of the motion controllers, and link stiffness; these parameters can experimentally be determined.

### 8.3.1 Normal Maneuver with Joint Flexibility

The mobile robot with joint flexibility performs the maneuver as described in section 8.2 with again all joints as well as the moving robot base following the second order motion profile shown in Figure 8.7. However, this time the SimMechanics model incorporates some flexibility in the robot mechanism. The resulting trajectory for the x-component of the ZMP is shown in Figure 8.13.

The x-component of the ZMP motion starts with a small oscillation, this is due to the gravity in SimMechanics. The gravity forces kick in at the start of the simulation and thus excites oscillations around the static equilibria of the various springs in the robot model. In reality, the robot would already be in static equilibrium, so these effects can be ignored.

The sudden jumps in x-component of the ZMP coincide with the jumps in acceleration in the motion profile (see Figure 8.7). These jumps in acceleration coincide with the jumps in forces and moments on the robot, and thus result in oscillations. At around 3.5 seconds the jump is higher due to the higher overall center of mass of the robot. The induced oscillations are also stronger.

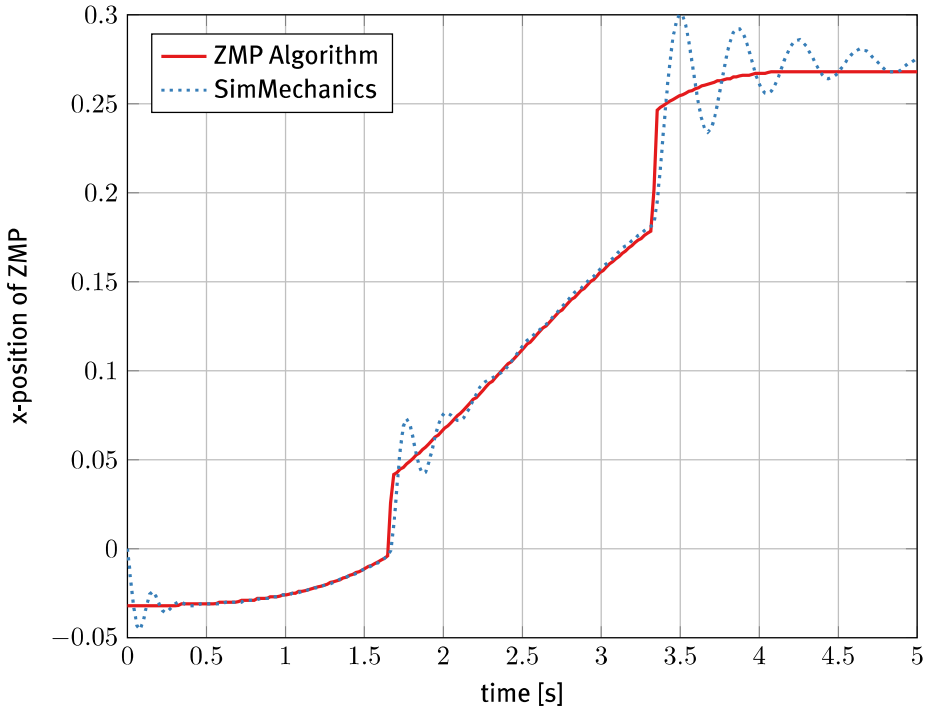



Figure 8.13: x-position of ZMP, with flexibility

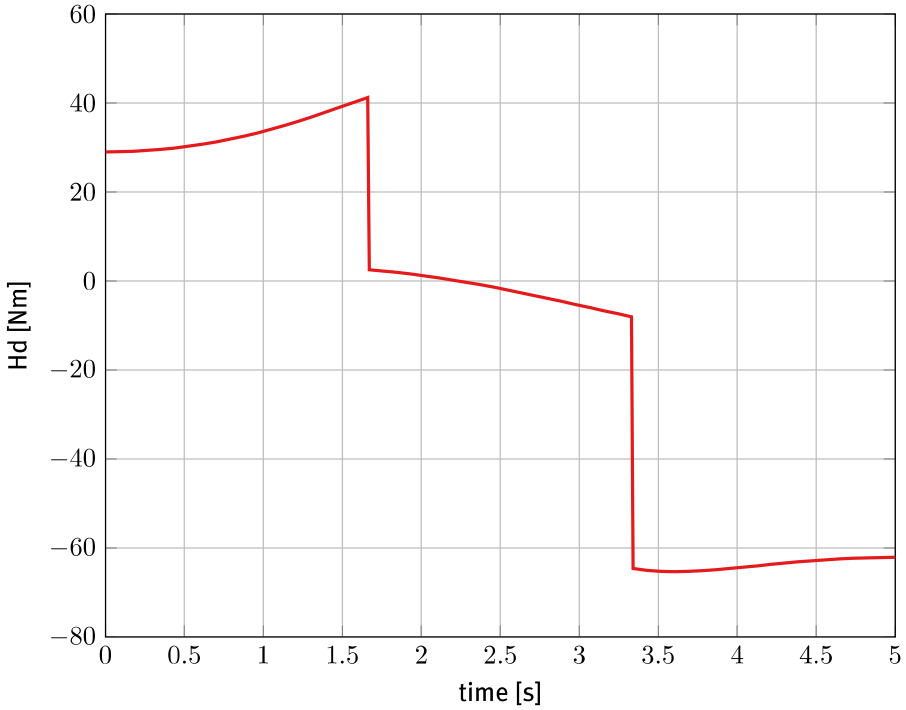



Figure 8.14: Change in angular momentum around the y-axis $\dot{H}_y$

When looking at the change in the angular momentum $\dot{H}_y$ computed with the ZMP algorithm in Figure 8.14 and recalling equation (4.36):

$$r_{P,x} = \frac{-\dot{H}_y + r_{c,z} m g_x - r_{c,x} m g_z}{\dot{P}_z - m g_z}$$

it can be concluded that the jumps in the ZMP are due to large moments on the robot at the time instants when the jumps in acceleration occur.

For the y-component of the ZMP, shown in Figure 8.15, the same initial oscillations occur due to the gravitational forces in SimMechanics. After these initial deviations from the predicted location according to the ZMP algorithm, only some minor deviations occur at around 3.5 seconds. This is also due to the sudden change in acceleration.

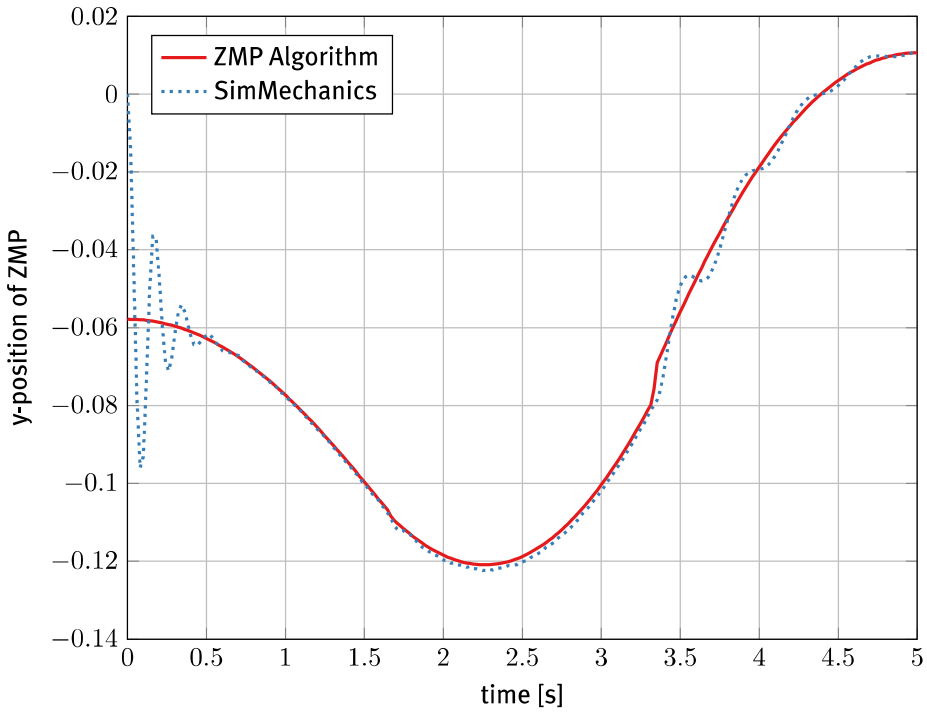



Figure 8.15: y-position of ZMP, with flexibility

The resulting x-y plot of the ZMP is shown in Figure 8.16. From this figure it becomes clear that the deviations in the x-direction are more severe that the deviations in the y- direction. This is a result of the acceleration of the entire robot in the x-direction. The spring stiffness of the suspension of the robot in combination with the mass of the robot (concentrated at the center of mass), result in behavior somewhat similar to a second order mass-spring-damper system.

These figures clearly show that there should be an additional safety margin around the computed ZMP to account for eventual deviations due to flexibility in the robot. This also indicates that it is wise to run SimMechanics simulations of a robot instead of just trusting the ZMP algorithm. In section 9.2.1, some ways to cope with these issues are explored. Moreover, in the next section a third order motion profile is used to minimize parasitic dynamics effects due to robot flexibility.

### 8.3.2 Third Order Motion Profile for Robot with Flexible Dynamics

Figure 8.14 shows jumps in the change of angular momentum which result in oscillations of the robot. These jumps are due to sudden steps in acceleration as shown in Figure 8.7. If a third order motion profile was used, there would not be steps in acceleration and thus not sudden changes of angular momentum.

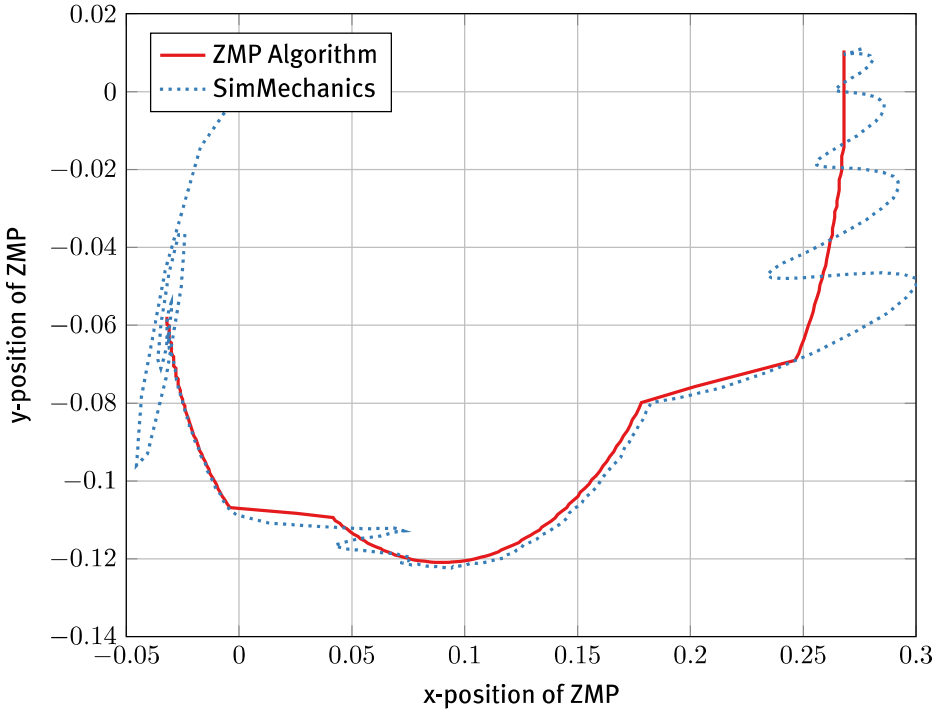



Figure 8.16: xy-position of ZMP, with flexibility

A new simulation is performed with a different motion profile for the movement of the robot base. This motion profile is shown in Figure 8.17.

This motion profile has a higher maximum acceleration and velocity than the second order profile to be able to finish its move in the same time. The resulting motion of the Zero-Moment Point is shown in Figure 8.18. Ignoring the parasitic oscillations at the start of the simulation, this figure shows that the computed ZMP is closer to the SimMechanics results with flexible dynamics.

The ZMP paths computed with the SimMechanics model that correspond to the second and third order motion profiles are compared in Figure 8.19. The ZMP path achieved with the third order motion profile has a slightly smaller enveloping area, and could thus be seen as the better motion profile with respect to robot balance.

## 8.4 Summary

Simulations are performed to compare the results of the ZMP algorithm to analytic results and to results from the SimMechanics model. Simulations with SimMechanics show that the ZMP algorithm is able to predict the (un)balance of the robot in the absence of disturbances and flexibilities. When flexibilities are incorporated in the SimMechanics model, results from the ZMP algorithm and the SimMechanics model are not similar anymore which accentuate a need to incorporate influence of flexibilities into the ZMP calculations.

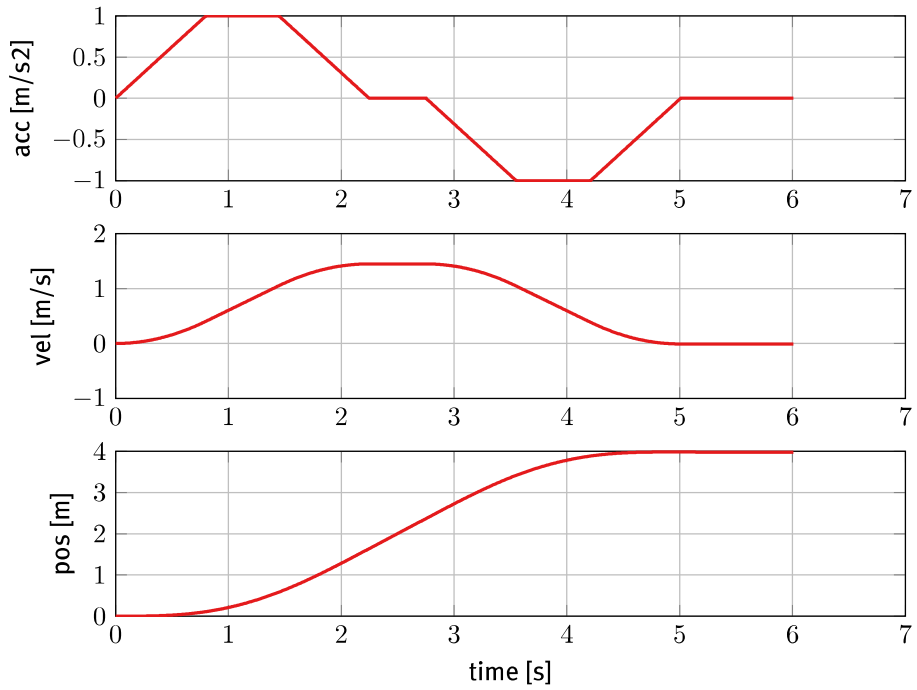



Figure 8.17: Third order motion profile for the moving base

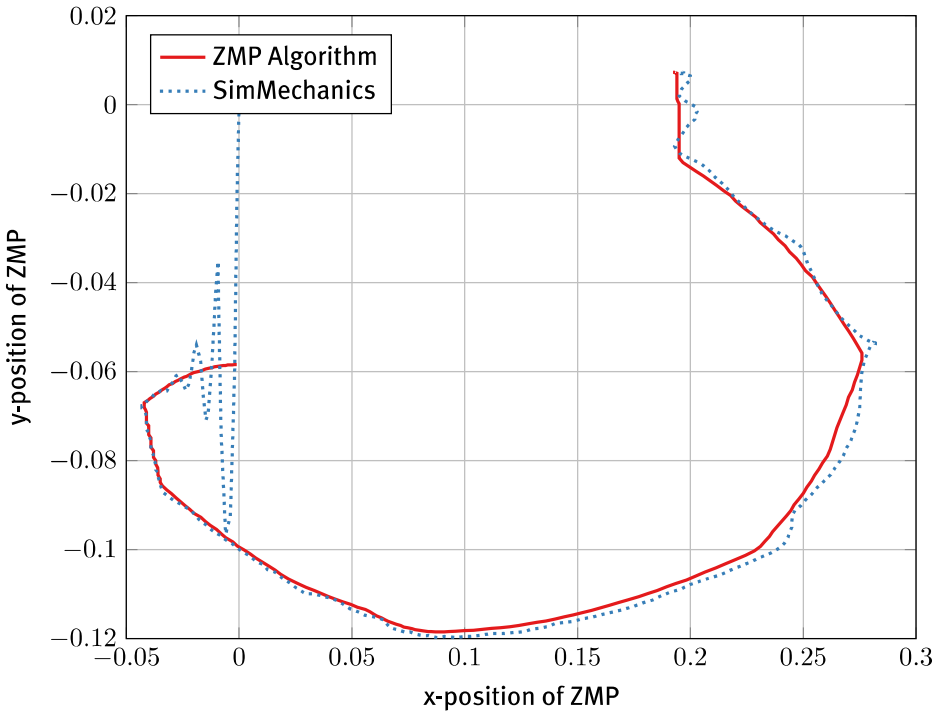



Figure 8.18: xy-location of ZMP, with third order motion profile

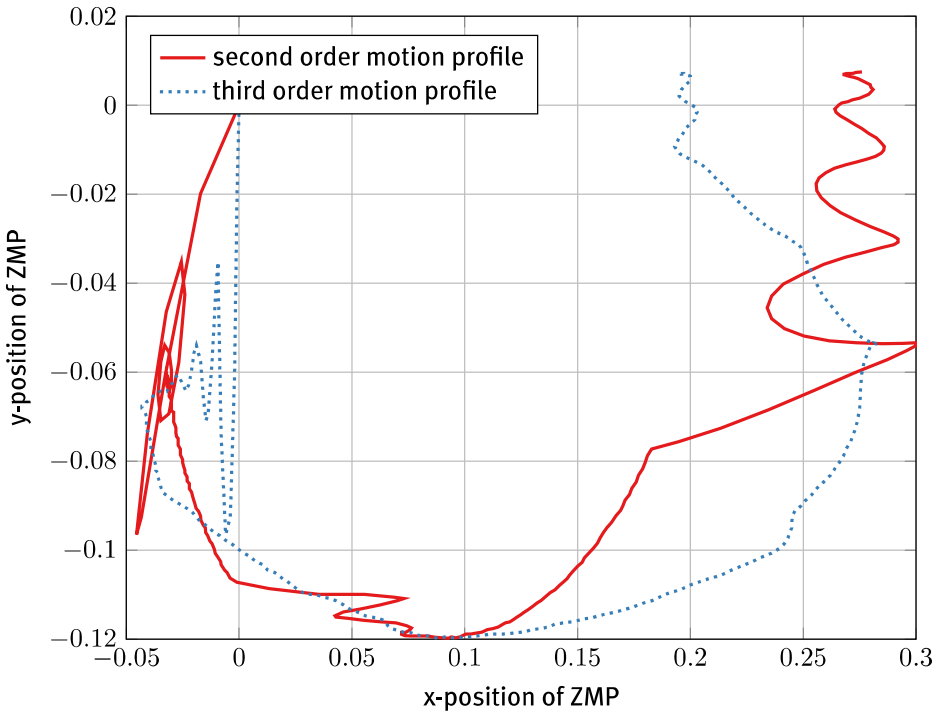



Figure 8.19: xy-location of ZMP computed with SimMechanics, comparison between second and third order motion profile

# 9 Discussion

## 9.1 Conclusion

An algorithm to predict the balance of a mobile service robot is created, based on the *Computed Zero-Moment Point* method. With this method, knowledge about joint coordinates of the robot is used to compute the rigid robot dynamics. These robot dynamics are subsequently used to compute the trajectory of the ZMP. As long as the ZMP stays within the support polygon of the robot, the robot is in balance. Thus, the ZMP algorithm can be used to make predictions about the robot balance for a given motion trajectory of the robot. This ZMP algorithm can be integrated into the ROS environment to run online on a mobile robot. The created Matlab code is directly usable for students in the Fontys mechatronics lab.

A Simulink/SimMechanics test framework is designed for validation of the ZMP algorithm. Simulation results for the ZMP algorithm and the SimMechanics model are similar for a robot of rigid dynamics. This indicates online usability of the ZMP algorithm on mobile service robots.

The SimMechanics test framework is used to investigate the effects of robot flexibilities on the robot balance. These results are compared with the results obtained with the ZMP algorithm, which does not take into account these robot flexibilities. From these simulations it becomes clear that robot flexibilities can have a major effect on the ZMP trajectory and consequently on the robot balance. This is particularly important in the presence of high accelerations, originating from a given motion profile or due to unmodeled disturbances. In section 9.2 some suggestions are presented on how to deal with robot flexibilities.

## 9.2 Recommendations

The ZMP algorithm is currently implemented in Matlab so it can be used by students in the Fontys mechatronics lab. It is recommended that the algorithm is implemented in the ROS environment such that it can be used and tested in practice on a real mobile service robot.

### 9.2.1 Coping with Flexibility

Figure 8.11 shows an example in which the robot starts tipping over and just barely returns to static balance. From this example it can be concluded that even a small or short crossing of the support polygon by the ZMP can lead to unbalance. Figure 8.16 shows that flexibility in the robot can cause relatively big deviations from the ZMP path computed using analytical formulas which only take rigid robot dynamics into account. These two cases combined show that it is important to know how to accommodate effects of flexibilities in ZMP calculations.

There are three general distinct ways to do this; the first is to design a safety margin around the computed ZMP such that the actual ZMP never leaves the support polygon. The second way is to incorporate flexible dynamics into an improved ZMP computation. The last way is to try to minimize contribution of flexible dynamics to the robot balance.

#### Safety Margin on ZMP

To design a margin around the computed ZMP some knowledge of the flexible dynamics is needed, it is assumed that stiffness parameters of the robot flexibilities are known, either by measurements or by design.

The SimMechanics framework can be used to create a model of the robot and incorporate flexibility at desired locations. The robot can then be simulated in various poses and with various motion profiles. This will show deviations between the computed and actual ZMP paths like shown in Figure 8.16. These deviations can then be used to design the safety margin. Furthermore, the *Stability Degree* and the *Valid Stable Region*, introduced in [13], could be used to quantify the degree of balance.

### Improved ZMP Computation

The current algorithm to compute the ZMP can only handle rigid dynamics, so an obvious way to improve it is to incorporate robot flexibilities. This involves deriving and implementing a totally new algorithm and could be an area for further research.

Another way to incorporate robot flexibilities into the ZMP computation, is to use the SimMechanics model (or an equivalent dynamics toolkit). The SimMechanics model is readily available so this would seem an obvious solution. A disadvantage of the SimMechanics model is the greater complexity of the model and subsequent computations which could lead to increased computation time compared to the current ZMP algorithm. Another problem is implementing this model on an actual robot, because of the need of a Matlab license. The Mechatronics lab at Fontys has the requirement that a Matlab license is not needed to implement the algorithm.

A different strategy would be to use the normal computed ZMP but superimpose a second order mass-spring-damper system that resembles the lowest eigenmode of the robot. In Figure 8.13 it could be seen that the parasitic dynamics resemble a second order system, so it may be possible to use that knowledge to approximate these deviations from the computed ZMP. The lowest eigenmode of the robot occurs from a combination of a dominant spring and mass. If these could be identified, the knowledge of forces and moments on the robot could be used to compute maximum deflections of these mass-spring-damper systems due to these loads. There exist analytic solutions for second order systems, and also simple formulas for the computation of overshoot in such systems are known, see for instance [6]. If, for instance, the suspension of the robot would cause the lowest eigenmode, the stiffness of this suspension and the mass of the robot could be used to create an equivalent second order model. Then the ZMP algorithm computes the sum of all forces and moments by means of the change in the linear and angular momenta (see equations (4.26) and (4.27)). In Figure 8.14 the change in the angular momentum around the y-axis is shown, here the jumps could be used as input for a 'step response' on the equivalent mass-spring-damper system. The result could then be used to compute a margin around the computed ZMP. This method is only slightly more complex than the current ZMP algorithm at the cost of ignoring the more complicated dynamics of the robot.

### Minimize Parasitic Flexible Dynamics

Figures 8.18 and 8.19 show that using a third order motion profile reduces the parasitic dynamics in flexible robots. Knowing this, it is recommended to always use third order motion profiles. However, this has only positive effect on 'modeled behavior' like changes in acceleration but no effect on unmodeled behavior like disturbances. Collisions with the environment, emergency stops or sudden changes in loading of the robot, are disturbances that would still excite these parasitic dynamics. This is another reason to design a safety margin around the computed ZMP.

# Appendix A Homogeneous Transformations

Homogeneous transformation is a method to represent rigid motion, that is, motion without deformation. So only the translation and rotation of an object can be represented with a homogeneous transformation. More about homogeneous transformations can be found in [12] or [3].

A homogeneous transformation is a matrix which represents rotation and translation of the following form:

$$\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{R}_i^{i-1} & \boldsymbol{o}_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{A.1}$$

where $\boldsymbol{R}_i^{i-1}$ represents the rotation between frame $i$ and frame $i-1$, and $\boldsymbol{o}_i^{i-1}$ represents the translation between the origins of the frames.

In Figure A.1 a homogeneous transformation between a frame $i-1$ and a frame $i$ is shown. Here, point $p$ is rigidly connected to frame $i$ as noted by vector $\boldsymbol{r}_p^i$. From this figure it is clear that the following relation between the position vectors $\boldsymbol{r}_p^i$ and $\boldsymbol{r}_p^{i-1}$ holds:

$$\boldsymbol{r}_p^{i-1} = \boldsymbol{o}_i^{i-1} + \boldsymbol{R}_i^{i-1}\boldsymbol{r}_p^i \tag{A.2}$$

This equals:

$$\begin{bmatrix} \boldsymbol{r}_p^{i-1} \\ 1 \end{bmatrix} = \boldsymbol{A}_i \begin{bmatrix} \boldsymbol{r}_p^i \\ 1 \end{bmatrix} \tag{A.3}$$

So, when a homogeneous transformation matrix $\boldsymbol{A}_i$ between two frames is known, the position and orientation of all points rigidly connected to frame $i$ can be calculated, relative to frame $i-1$.

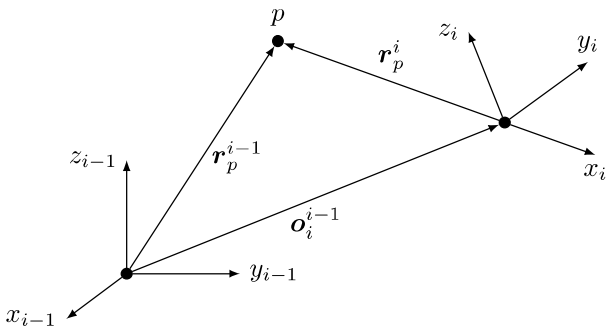



**Figure A.1:** Homogeneous transformation between frame $i-1$ and frame $i$

For a set of subsequent homogeneous transformations the total transformation from the first frame $i$ to the last frame $j$ is given by:

$$\boldsymbol{T}_j^i = \boldsymbol{A}_i\boldsymbol{A}_{i+1}\cdots\boldsymbol{A}_j = \begin{bmatrix} \boldsymbol{R}_j^i & \boldsymbol{o}_j^i \\ 0 & 1 \end{bmatrix} \tag{A.4}$$

So for a chain of rigid links, like a robotic arm, when every subsequent transformation from one link to the next is known, all points on each link can be referenced back to an arbitrary frame.

# Appendix B Denavit-Hartenberg Convention

When a robotic arm consists of rigid links, connected to each other with actuated joints, there is a systematic way to calculate the homogeneous transformation for each link. This is called the *Denavit-Hartenberg convention*. For an in dept look at this convention and a description how to apply it to a robotic arm, see [12] or [3]. One assumption in this convention is that the linking joint between two links is actuated only along the $z$-axis for a prismatic joint, and only around the $z$-axis for a revolute joint. This has some consequences for the assignment of link frames that are not discussed here.

In the Denavit-Hartenberg (DH) convention a link and its joint are described with four parameters, one of which is the actuated joint variable. These four parameters are:

- $\theta_i$ joint angle, actuated for revolute joint
- $d_i$ link offset, actuated for prismatic joint
- $a_i$ link length
- $\alpha_i$ link twist

When the links of a robot are correctly described in the DH convention the result is a homogeneous transformation matrix for each link:

$$\boldsymbol{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

Here the upper left $[3 \times 3]$ part gives the rotation matrix $\boldsymbol{R}_i^{i-1}$, the upper right $[3 \times 1]$ part gives the translation $\boldsymbol{o}_i^{i-1}$. The bottom row is the way it is so a transformation like (A.3) which results in (A.2) is possible.

To sum up, when a link is parametrized with the Denavit-Hartenberg convention and the joint coordinate is known ($d_i$ for a prismatic joint, and $\theta_i$ for a revolute joint), the rotation $\boldsymbol{R}_i^{i-1}$ and the translation $\boldsymbol{o}_i^{i-1}$ between frames are also known.

# Appendix C Forward Dynamics

## C.1 General Case

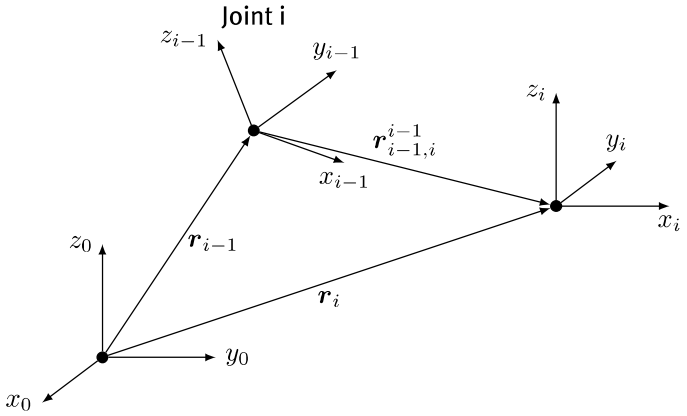



**Figure C.1:** Position vectors between linked frames.

In Figure C.1 a reference frame (0) is shown, together with two frames that are linked together with a joint. In this case frame $i$ is rigidly connected to a joint that enables translation or rotation along or about the $z_{i-1}$ axis. The position of the origin of frame $i$ is given by position vector $\boldsymbol{r}_i$ with respect to the reference frame:

$$\boldsymbol{r}_i = \boldsymbol{r}_{i-1} + \boldsymbol{R}_{i-1}^0 \boldsymbol{r}_{i-1,i}^{i-1} \tag{C.1}$$

$$= \boldsymbol{r}_{i-1} + \boldsymbol{r}_{i-1,i} \tag{C.2}$$

It is important to notice that this case is different from that in (A.2) because here the vector $\boldsymbol{r}_{i-1,i}^{i-1}$ is not rigidly connected to frame $i-1$, it depicts the position of frame $i$ relative to frame $i-1$. Frame $i$ can rotate about or translate along the $z_{i-1}$-axis, so the vector $\boldsymbol{r}_{i-1,i}^{i-1}$ changes with joint $i$. The Rotation matrix and position vector are known from the homogeneous transformation matrices $\boldsymbol{T}$ as shown in (A.4), where the position vector $\boldsymbol{r}_{i-1,i}$ can be calculated with:

$$\boldsymbol{r}_{i-1,i} = \boldsymbol{o}_i - \boldsymbol{o}_{i-1} \tag{C.3}$$

Due to the additive property of angular velocities, when expressed in the same coordinate frame, the angular velocity of frame $i$ is given by:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \boldsymbol{R}_{i-1}^0 \boldsymbol{\omega}_{i-1,i}^{i-1} \tag{C.4}$$

$$= \boldsymbol{\omega}_{i-1} + \boldsymbol{\omega}_{i-1,i} \tag{C.5}$$

where $\boldsymbol{\omega}_{i-1}$ is the angular velocity of frame $i-1$ with respect to frame 0, and $\boldsymbol{\omega}_{i-1,i}$ is the relative angular velocity between frame $i-1$ and $i$ with respect to frame 0.

The linear velocity of frame $i$ can be determined by differentiating (C.1):

$$\dot{\boldsymbol{r}}_i = \dot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{R}}_{i-1}^0 \boldsymbol{r}_{i-1,i}^{i-1} + \boldsymbol{R}_{i-1}^0 \dot{\boldsymbol{r}}_{i-1,i}^{i-1} \tag{C.6}$$

$$= \dot{\boldsymbol{r}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{v}_{i-1,i} \tag{C.7}$$

where $\boldsymbol{v}_{i-1,i}$ denotes the linear velocity of frame $i$ relative to frame $i-1$, with respect to frame 0. It can be seen that the velocity of frame $i$ has three components, the first is the velocity of the previous frame $(i-1)$, the second is the effect of a rotating $i-1$ frame and the last is the relative velocity of frame $i$.

In similar fashion, the angular acceleration of frame $i$ results from differentiating (C.4):

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} + \dot{\boldsymbol{R}}^0_{i-1}\boldsymbol{\omega}^{i-1}_{i-1,i} + \boldsymbol{R}^0_{i-1}\dot{\boldsymbol{\omega}}^{i-1}_{i-1,i} \tag{C.8}$$

$$= \dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{\omega}_{i-1,i} + \dot{\boldsymbol{\omega}}_{i-1,i} \tag{C.9}$$

Here $\dot{\boldsymbol{\omega}}_{i-1,i}$ is the angular acceleration of frame $i$ relative to frame $i-1$.

Furthermore, the time derivative of (C.7) gives the linear acceleration of frame $i$:

$$\ddot{\boldsymbol{r}}_i = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1} \times \dot{\boldsymbol{r}}_{i-1,i} + \dot{\boldsymbol{v}}_{i-1,i} \tag{C.10}$$

The relative velocities $\boldsymbol{\omega}_{i-1,i}$ and $\boldsymbol{v}_{i-1,i}$ and accelerations $\dot{\boldsymbol{\omega}}_{i-1,i}$ and $\dot{\boldsymbol{v}}_{i-1,i}$ are dependent of joint type and are discussed below.

## C.2 Revolute Joint

A revolute joint enables the connected link to rotate about the $\boldsymbol{z}_{i-1}$-axis by an amount of $\theta_i$. For the rotational velocity of frame $i$ relative to frame $i$, $\boldsymbol{\omega}^{i-1}_{i-1,i}$ in (C.4), this means that it can be written as:

$$\boldsymbol{R}^0_{i-1}\boldsymbol{\omega}^{i-1}_{i-1,i} = \boldsymbol{R}^0_{i-1}\dot{\theta}_i\boldsymbol{z}^{i-1}_{i-1} \tag{C.11}$$

$$\boldsymbol{\omega}_{i-1,i} = \dot{\theta}_i\boldsymbol{z}_{i-1} \tag{C.12}$$

which results in:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{\theta}_i\boldsymbol{z}_{i-1} \tag{C.13}$$

The linear velocity of frame $i$ relative to frame $i-1$ is a result of the distance between these two frames and the rotational velocity between them:

$$\boldsymbol{v}_{i-1,i} = \boldsymbol{\omega}_{i-1,i} \times \boldsymbol{r}_{i-1,i} \tag{C.14}$$

$$= \dot{\theta}_i\boldsymbol{z}_{i-1} \times \boldsymbol{r}_{i-1,i} \tag{C.15}$$

The total linear velocity of frame $i$ is then:

$$\dot{\boldsymbol{r}}_i = \dot{\boldsymbol{r}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1,i} \times \boldsymbol{r}_{i-1,i} \tag{C.16}$$

$$= \dot{\boldsymbol{r}}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i} \tag{C.17}$$

$$= \dot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{r}}_{i-1,i} \tag{C.18}$$

Furthermore, the angular acceleration of frame $i$ relative to frame $i-1$ is given by:

$$\boldsymbol{R}^0_{i-1}\dot{\boldsymbol{\omega}}^{i-1}_{i-1,i} = \boldsymbol{R}^0_{i-1}\ddot{\theta}_i\boldsymbol{z}^{i-1}_{i-1} \tag{C.19}$$

$$\dot{\boldsymbol{\omega}}_{i-1,i} = \ddot{\theta}_i\boldsymbol{z}_{i-1} \tag{C.20}$$

this results in

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{\omega}_{i-1,i} + \ddot{\theta}_i\boldsymbol{z}_{i-1} \tag{C.21}$$

$$= \dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i-1,i} + \ddot{\theta}_i\boldsymbol{z}_{i-1} \tag{C.22}$$

The $\boldsymbol{\omega}_{i-1}$ term in (C.21) can be changed to $\boldsymbol{\omega}_i$ because the cross product $\dot{\theta}_i\boldsymbol{z}_{i-1} \times \dot{\theta}_i\boldsymbol{z}_{i-1} = 0$ in $\boldsymbol{\omega}_i \times \boldsymbol{\omega}_{i-1,i}$.

Lastly, the linear acceleration of frame $i$ relative to frame $i-1$ becomes:

$$\dot{\boldsymbol{v}}_{i-1,i} = \dot{\boldsymbol{\omega}}_{i-1,i} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1,i} \times \dot{\boldsymbol{r}}_{i-1,i} \tag{C.23}$$

Combining (C.10) with (C.14) and (C.23) gives:

$$\ddot{\boldsymbol{r}}_i = \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1} \times \dot{\boldsymbol{r}}_{i-1,i} + \dot{\boldsymbol{\omega}}_{i-1,i} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1,i} \times \dot{\boldsymbol{r}}_{i-1,i} \tag{C.24}$$

$$= \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times \dot{\boldsymbol{r}}_{i-1,i} \tag{C.25}$$

$$= \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) \tag{C.26}$$

This shows that when $\theta_i$, $\dot{\theta}_i$ and $\ddot{\theta}_i$ are known for a revolute joint, $\boldsymbol{\omega}_i$, $\dot{\boldsymbol{r}}_i$, $\dot{\boldsymbol{\omega}}_i$ and $\ddot{\boldsymbol{r}}_i$ can be calculated.

## C.3 Prismatic Joint

A prismatic joint enables the connected link to translate along the $\boldsymbol{z}_{i-1}$-axis by an amount of $d_i$. This means that the rotational velocity of frame $i$ relative to frame $i-1$ is zero:

$$\boldsymbol{\omega}_{i-1,i} = 0 \tag{C.27}$$

resulting in:

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} \tag{C.28}$$

The linear velocity of frame $i$ relative to frame $i-1$ is a result of the velocity that the actuator generates:

$$\boldsymbol{v}_{i-1,i} = \dot{d}_i \boldsymbol{z}_{i-1} \tag{C.29}$$

This gives:

$$\begin{aligned} \dot{\boldsymbol{r}}_i &= \dot{\boldsymbol{r}}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i} + \dot{d}_i \boldsymbol{z}_{i-1} && \text{(C.30)} \\ &= \dot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{r}}_{i-1,i} && \text{(C.31)} \end{aligned}$$

The rotational acceleration is only dependent on the rotational acceleration of the previous frame:

$$\dot{\boldsymbol{\omega}}_{i-1,i} = 0 \tag{C.32}$$

which results in:

$$\dot{\boldsymbol{\omega}}_i = \dot{\boldsymbol{\omega}}_{i-1} \tag{C.33}$$

The linear acceleration of frame $i$ relative to frame $i-1$ is:

$$\begin{aligned} \dot{\boldsymbol{v}}_{i-1,i} &= \frac{d}{dt}(\dot{d}_i \boldsymbol{z}_{i-1}) && \text{(C.34)} \\ &= \frac{d}{dt}(\boldsymbol{R}^0_{i-1} \dot{d}_i \boldsymbol{z}^{i-1}_{i-1}) && \text{(C.35)} \\ &= \dot{\boldsymbol{R}}^0_{i-1} \dot{d}_i \boldsymbol{z}^{i-1}_{i-1} + \boldsymbol{R}^0_{i-1} \ddot{d}_i \boldsymbol{z}^{i-1}_{i-1} && \text{(C.36)} \\ &= \boldsymbol{\omega}_{i-1} \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} && \text{(C.37)} \end{aligned}$$

The linear acceleration of frame $i$ can now be stated as:

$$\begin{aligned} \ddot{\boldsymbol{r}}_i &= \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i} + \dot{d}_i \boldsymbol{z}_{i-1}) + \boldsymbol{\omega}_{i-1} \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} && \text{(C.38)} \\ &= \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) + 2\boldsymbol{\omega}_{i-1} \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} && \text{(C.39)} \\ &= \ddot{\boldsymbol{r}}_{i-1} + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i}) + 2\boldsymbol{\omega}_i \times \dot{d}_i \boldsymbol{z}_{i-1} + \ddot{d}_i \boldsymbol{z}_{i-1} && \text{(C.40)} \end{aligned}$$

# References


[1] V. Antoska, K. Jovanović, V. M. Petrović, N. Baščarević, and M. Stankovski, *Balance analysis of the mobile anthropomimetic robot under disturbances - ZMP approach,* International Journal of Advanced Robotic Systems, (2013).

[2] J. T. Bouten, *Stability and dynamic analysis of a service robot,* traineeship report, University of Technology, Eindhoven, 2011.

[3] P. I. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB,* no. v. 73 in Springer tracts in advanced robotics, Springer, Berlin, 2011.

[4] M. H. P. Dekker, *Zero-moment point method for stable biped walking,* traineeship report, University of Technology, Eindhoven, 2009.

[5] G. J. Dohmen, *Mechanical design of a service robot torso for AMIGO v2.0,* internship report, eindhoven, University of Technology, Eindhoven, Eindhoven, Apr. 2012.

[6] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback control of dynamic systems,* Prentice Hall, Upper Saddle River, NJ, 4th ed., 2002.

[7] L. Hoyet, F. Multon, K. Mombaur, and E. Yoshida, *Balance in dynamic situations: role of the underlying model,* Computer Methods in Biomechanics and Biomedical Engineering, 12 (2009), pp. 147 – 149.

[8] Q. Huang, S. Sugano, and I. Kato, *Stability control for a mobile manipulator using a potential method,* in Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94, vol. 2, 1994, pp. 839–846 vol.2.

[9] Q. Huang, S. Sugano, and K. Tanie, *Stability compensation of a mobile manipulator by manipulator motion: feasibility and planning,* in , Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1997. IROS '97, vol. 3, 1997, pp. 1285–1292 vol.3.

[10] E. G. Papadopoulos and D. A. Rey, *A new measure of tipover stability margin for mobile manipulators,* in Proceedings of the IEEE International Conference on Robotics and Automation, 1996, pp. 3111 – 3116.

[11] B. Siciliano and O. Khatib, *Springer handbook of robotics,* Springer, Berlin, 2008.

[12] M. W. Spong, *Robot modeling and control,* John Wiley & Sons, Hoboken, NJ, 2006.

[13] S. Sugano, Q. Huang, and I. Kato, *Stability criteria in controlling mobile robotic systems,* in Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93, vol. 2, 1993, pp. 832–838 vol.2.

[14] N. van de Wouw, *Multibody Dynamics, lecture notes,* University of Technology, Eindhoven, 2010.

[15] M. Vukobratović and J. Stepanenko, *On the stability of anthropomorphic systems,* Mathemathical Biosciences, 15 (1972), pp. 1–37.

[16] L. Yanjie, W. Zhenwei, and Z. Hua, *The dynamic stability criterion of the wheel-based humanoid robot based on ZMP modeling,* in Control and Decision Conference, 2009. CCDC '09. Chinese, 2009, pp. 2349–2352.

[17] P. v. Zutven, D. Kostić, and H. Nijmeijer, *On the stability of bipedal walking,* in Simulation, Modeling, and Programming for Autonomous Robots, N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. v. Stryk, eds., no. 6472 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, Jan. 2010, pp. 521–532.