Control for Humanoid Locomotion: From Pendulum Models to MPC with Centroidal Dynamics*

Joshua Johannson

***Abstract*— Performing versatile robotic tasks in arbitrary environments often requires traversing difficult terrain. Humanoid-legged robots offer great potential for those applications. However, achieving stable locomotion with two legs remains challenging, since many aspects like robustness, collision avoidance and real-time performance need to be considered which makes the generation of feasible motion trajectories complex. In this work, we will provide an overview of various classical, non-learning approaches for the control of humanoid walking. Thereby we will move from simpler pendulum-based methods over single rigid body dynamics with MPC to full dynamics utilizing DDP. Finally, we compare the presented approaches in terms of their optimization capabilities and performance.**

I. INTRODUCTION

Humanoid locomotion remains a challenging problem because i.a. the dynamics are complex and computations need to be fast to enable reacting to external disturbances. Reactions that are too slow may lead to catastrophic failures such as falling down. Either computation is fast enough to fulfill this requirement or reference trajectories are produced that are then tracked by a separate feedback controller. To cope with these challenges, we will look at three classes of approaches.

1) The first class of methods utilizes strongly simplified models to generate walking motions. Here the robot is abstracted as an Inverted Pendulum (IP), which reduces computational costs. Due to this simplicity, predictive planning may not be required [12], since the walking motions are directly derivable from the model. Multiple approaches in this category utilize the IP model to infer the trajectory for the body of the robot from predefined step locations [11, 12]. An extension of the IP incorporating an additional spring and damper allows for robust predictive planning of reference trajectories via optimal control [10]. We will discuss these approaches in Section III.

2) Another class of methods considers a more complex model that represents the robot as a single rigid body, which is also called centroidal dynamics model. Generally, this model is used to plan optimal walking trajectories over a predictive future horizon. Depending on the formulation, this calculation may be slow and just allows replanning with low update rates when used repetitively as Model Predictive Control. Therefore, resulting trajectories may be executed by employing a separate fast-tracking controller, that often utilizes a non-simplified dynamics model and only considers the next time step for optimization [7]. Linearisation around the current state may be used to reduce the computational time of the single rigid body dynamics-based optimization [7]. This dynamics model can also be combined with full-body kinematics to incorporate collision constraints [3]. These methods will be discussed in Section IV.

3) Also, the full dynamics model, which considers all joints and links in the dynamics and suffers from less model errors, can be used for planning optimal trajectories. However, this yields increased computational costs, which can be addressed by using Differential Dynamic Programming to optimize over trajectories [5]. We look at this class of methods in Section V-A.

In the final Section VI we will compare the discussed methods in terms of their optimization flexibility. We examine which properties, such as step positions and step timings, a method can adapt and which properties have to be predefined.

A. Related Work

Several other relevant works are using the centroidal dynamics model for legged locomotion which we don't consider in detail in this paper. An approach by Chignoli et al. [2], which is similar to the work presented in Section IV-B, aims to create and execute acrobatic jumping motions for humanoid robots. Chignoli et al. also combine centroidal dynamics with full-body kinematics during trajectory optimization via a Kino-Dynamic Planner. This allows them to consider joint constraints and the effect of the joint velocities on the angular momentum on the body. Note that the contact sequence is fixed here.

Garcia et al. [9] also use the centroidal dynamics model to implement MPC for humanoid walking. They linearize around a given reference trajectory and thereby also consider variable inertia and angular momentum of the body. The resulting optimal control problem with linear dynamics is solved via Quadratic Programming.

II. BACKGROUND

In this Section, we will discuss fundamentals for optimal control, which is relevant for approaches presented later that do planning. Regarding notation, we denote vectors and matrices with $\mathbf{bold}$ letters and scalars with $normal$ letters.

A. Optimal Control

A common approach to find trajectories, e.g. for a walking gait, that are optimal w.r.t. some cost function is to formulate an Optimal Control (OC) problem [6] which itself is a form of a constraint optimization problem. Often the dynamics

*This work is part of the seminar "Humanoid Robotics" at TU Darmstadt.

of a system are only available in the form of an Ordinary Differential Equation (ODE) which defines the change of the current system state $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$ depending on the current state $\mathbf{x}(t)$ and control $\mathbf{u}(t)$ input. An OC problem offers the advantage that an ODE is enough and no closed-form solution for the dynamics is required. A general simplified form of the OC problem looks like this [6]:

$$\min_{\mathbf{x}(t),\mathbf{u}(t),T} \quad E(\mathbf{x}(t)) + \int_0^T L(\mathbf{x}(t), \mathbf{u}(t))dt \tag{1}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2}$$

$$\mathbf{x}_0 = \mathbf{x}(0)$$

$$0 = r(\mathbf{x}(T)) \qquad (\text{final constraints}) \tag{3}$$

$$0 = h(\mathbf{x}(t), \mathbf{u}(t)) \quad (\text{path constraints})$$

The solution of the OC problem consists of the optimal state and control trajectories $\mathbf{x}(t), \mathbf{u}(t)$ that minimize the objective (1). Thereby $E(\mathbf{x}(t))$ stands for the cost in the final state, e.g. distance to a desired state, and $L(\mathbf{x}(t), \mathbf{u}(t))$ represents the cost of intermediate states that will be accumulated over the time horizon T. The time horizon is the length of the run period of the system that is considered during optimization and may be also an optimization variable, e.g. when a solution with minimal runtime is required. From the constraints, (2) ensures that the system dynamics is respected over the whole horizon. Additional constraints may be added for initial or final conditions (3), e.g. for fixing the start state. Also, intermediate conditions, i.e. path constraints, can be encoded by constraints (3).

B. Solving Optimal Control Problems

The aforementioned OC problems are commonly solved using direct methods that transform the original formulation into a Non-Linear Programming (NLP) [6]. The NLP problem can then be solved with standard numerical solvers.

1) Direct Single Shooting: A direct method to solve the OC problem which uses time-wise discretized of the controls is Direct Single Shooting [6]. Here, only the control actions $\mathbf{u}_t$ are optimized, whereas the state trajectory is obtained from the initial state $\mathbf{x}_0$ and the control trajectory by numerical integration. The time domain is discretized, such that the optimization is performed over a finite number of actions $\mathbf{u}_t$, and the integral in the objective (1) turns into a sum.

2) Direct Multiple Shooting: The Multiple Shooting method [6] can be derived by adapting Single Shooting so that also the discretized state values $\mathbf{x}_t$ are considered as optimization variables. Thereby additional continuity constraints are introduced that ensure that the value of a next state $\mathbf{x}_{t+1}$ is equal to the integrated dynamics from the previous state and control $\mathbf{x}_t, \mathbf{u}_t$. Typically, optimization variables only involve state values at certain time points, specifically every N time steps, and are called knot points. The continuity constraints then are expreseed as $\mathbf{x}_{t+N} = f_{int}(\mathbf{x}_t, \mathbf{u}_t)$, where f_{int} is the dynamics integration function for N time steps. An example formulation very similar to Direct Multiple Shooting is shown in Equation 14. The resulting NLP problem may then be solved with a numerical solver. Note that multiple shooting offers the advantage that the solver can be easily warm-started by reusing a previous solution trajectory [4, 6], e.g. in an MPC setting. With single shooting, warm-starting is not directly possible.

3) Differential Dynamic Programming (DDP): DDP is based on the single or multiple shooting formulation of an OC problem [4]. The solving process consists of two main steps [4]: 1) The dynamics constraints and cost function are approximated around an initial guessed trajectory, $\mathbf{x}_t, \mathbf{u}_t$. This is done via a first and second-order Taylor series approximation. The resulting problem is linear in the dynamics and quadratic in the cost function. Thus it can be solved via the Linear Quadratic Regulator[1], which yields a linear feedback controller not just a new control trajectory. 2) The feedback controller is then used in the forward pass to simulate the system and create a new trajectory to approximate around. Now the first step is repeated with the new trajectory as an initial guess. Note that the result of this algorithm is a feedback controller that can be employed to follow the optimal trajectory calculated in the forward pass.

C. Model Predictive Control

To incorporate feedback, an OC problem can be solved repeatedly with varying starting conditions. Thereby the starting state will be set to the current real observed state. From the solution of the OC problem, just the first control action is executed. These two steps will be repeated to react to perturbations in the real system and to do the corresponding replanning. This procedure is called Model Predictive Control (MPC) [6].

III. PENDULUM BASED APPROACHES

A simple model used for bipedal locomotion is the Linear Inverted Pendulum (LIP) [13, 11], where we consider the whole robot as a point mass m concentrated at some point $\mathbf{r} = [r_x, r_y, r_z]^T$, see Figure 1. Thereby $\mathbf{r}$ is called the center of mass (CoM). The pendulum connects the CoM with the pendulum base $\mathbf{p}$ which is in ground contact ($p_x = 0$). Note that the distance between CoM and $\mathbf{p}$, i.e. the leg length, is variational as for a prismatic joint. The LIP can be derived from the general inverted pendulum [11]. Here we are looking at the 3D case where CoM is constrained to move parallel to the ground. The dynamics of the general inverted pendulum depends on the pendulum angles θ_x, θ_y and actuation torques τ_x, τ_y acting on the pendulum base. These actuation torques are directly associated with the respective pendulum angles. Additionally, the gravity and the ground reaction force F_p act on the point mass of the pendulum. To make dynamics linear, τ_x, τ_y can be remapped to new virtual actuation torques u_x, u_y which eliminates the dependency of the dynamics on θ_x and θ_y. By further assuming that the ground is flat and horizontal we obtain the dynamics for the LIP[2]:

[1]This is just the case when the dynamics are linearly approximated.

[2]For the previous assumptions.

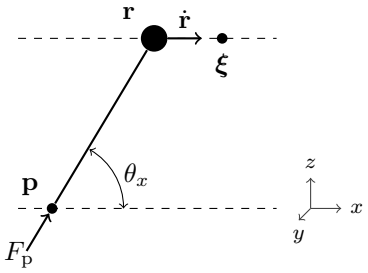

Fig. 1: LIP Model with DCM
The pendulum connects the point mass at $\mathbf{r}$ with the pendulum base at $\mathbf{p}$. θ_x denotes the angle between the pendulum and the flat ground in the x-z plane. Note that here we just show a 2D side view of the 3D inverted pendulum, thus the angle θ_y in the y-z plane is not visible. The DCM position ξ is in the direction of the point mass velocity $\dot{\mathbf{r}}$. F_p is the force of the pendulum base on the ground.

$$\begin{aligned} \ddot{r}_x &= \frac{g}{r_z}(r_x - p_x) + \frac{u_x}{m r_z} \\ \ddot{r}_y &= \frac{g}{r_z}(r_y - p_y) - \frac{u_y}{m r_z} \end{aligned} \tag{4}$$

Note that the CoM is locked in the z-direction, thus $\ddot{r}_z = 0$. We can observe that movements in the x and y directions are fully decoupled. Often the actuation torques are assumed to be zero ($u_x = u_y = 0$), which further simplifies (4). Since these differential equations are linear, they can directly be solved, e.g. via the Laplace Transform. This yields a closed-form trajectory equation $\mathbf{r}(t)$ of the CoM for given initial boundary conditions (e.g. the initial CoM position and velocity) as well as a predefined base location $\mathbf{p}$ and CoM height. $\mathbf{r}(t)$ results in a hyperbolic curve in the x-y plane.

A. LIP Walking Pattern Generation

Kajita et al. [11] utilize the LIP to generate walking trajectories for humanoid robots. They assume a fixed gait schedule where a single support phase for the first foot is followed by a double support phase. Subsequently, both phases are repeated but with the other foot in contact during the single support phase. Both phases have fixed durations. By varying the predefined foot contact locations the walking direction and speed can be adapted. The LIP model now allows inferring CoM reference trajectories from the foot step locations. For the single support phases equations (4) (with $u_x = u_y = 0$) are solved to derive the CoM trajectory. During the double support phases, the CoM moves with a constant velocity which is equal to the last CoM velocity from the previous single support phase. Kajita et al. validate the resulting walking pattern on a humanoid robot in simulation. Therefore, they map the reference trajectories to joint angles via inverse kinematics and track them with a PD controller. This implies that the pendulum base $\mathbf{p}$ corresponds to the robot's foot currently in ground contact.

B. LIP and Divergent Component of Motion

The dynamics of the LIP model can be split into a stable and unstable part [14, 15, 8, 12]. The unstable component is called the Divergent Component of Motion (DCM). For conciseness, we only consider the x component of the LIP dynamics (4) and DCM. The y part of the DCM can be derived completely analogously. Here we also assume zero actuation torques ($u_x = u_y = 0$). For simplicity, we introduce $\omega^2 = g/r_z$, now we get (5) for the x component of the LIP dynamics from (4).

$$\ddot{r}_x = \omega^2 r_x - \omega^2 p_x \tag{5}$$

We have to consider that the foot of a real robot is not a point contact, as assumed by the LIP, but makes contact over a whole surface. Thus, a foot can induce torques which would violate our previous assumptions that $u_x = u_y = 0$. Instead of assuming that the pendulum base $\mathbf{p}$ matches the point where the foot is attached to the robot's leg, we can consider $\mathbf{p}$ to be the Zero Moment Point (ZMP). The ZMP is a virtual point where applied torques of the feet in contact become zero and just forces remain. Note that for stability, the ZMP has to be inside the support polygon which is the convex hull of the stance feet.

We can observe that (5) is unstable since the poles of the corresponding transfer function (18) are on the right half plane, for the derivation see Appendix VIII-A. For splitting the dynamics into stable and unstable parts, we have to apply a coordinate transformation (6) [14, 15].

$$\begin{pmatrix} \xi_x \\ s_x \end{pmatrix} = \begin{pmatrix} r_x + \frac{1}{\omega}\dot{r}_x \\ r_x - \frac{1}{\omega}\dot{r}_x \end{pmatrix} \tag{6}$$

To express the LIP dynamics with our transformation, we take the derivative of (6) and replace the appeared $\ddot{r}_x$ terms with (5). After also replacing r_x and $\dot{r}_x$ terms with (6) we get the new transformed LIP dynamics (7).

$$\begin{pmatrix} \dot{\xi}_x \\ \dot{s}_x \end{pmatrix} = \begin{pmatrix} \dot{r}_x + \omega r_x - \omega p_x \\ \dot{r}_x - \omega r_x + \omega p_x \end{pmatrix} = \begin{pmatrix} +\omega \xi_x - \omega p_x \\ -\omega s_x + \omega p_x \end{pmatrix} \tag{7}$$

Note that (7) still fully captures the LIP dynamics (5). Since in (7) $\dot{\xi}_x$ and $\dot{s}_x$ are independent of each other, we can look at their individual transfer functions (8).

$$F_{\xi_x}(p) = \frac{\omega}{\omega - p} \qquad F_{s_x}(p) = \frac{\omega}{\omega + p} \tag{8}$$

We observe that s_x is always stable since the poles of its transfer function $F_{s_x}(p)$ are on the left half-plane. ξ_x on the other hand is unstable, because its transfer function $F_{\xi_x}(p)$ has poles on the right half-plane. For the derivation of the transfer functions see Appendix VIII-B. Consequently, just the unstable ξ_x, which is the DCM, needs to be controlled [12]. When also considering the y component of the DCM, we get the general vector form of $\boldsymbol{\xi}$.

$$\boldsymbol{\xi} = \mathbf{r} + \frac{1}{\omega}\dot{\mathbf{r}} \tag{9}$$

As shown in Figure 1 the DCM $\boldsymbol{\xi}$ is a point in the movement direction of the CoM.

C. Walking via LIP and DCM

One application of LIP with the DCM formulation is the generation of reference trajectories from predefined step locations. Kasaei et al. [12] present an approach where the step locations come from a simple step planner, that generates these locations based on given parameters (e.g. step length, step duration). The ZMP trajectories $\mathbf{p}_t$ are calculated from the footstep positions via a stepping function, where $\mathbf{p}_t$ instantly changes when a step is taken and the contact location changes. Solving Equations (4), (7) and (9), directly allows to infer the DCM and CoM trajectories between two changing ZMP locations. This is similar to the work presented in Section III-A, except that the unstable DCM component of the LIP is considered. Swing leg motions for the time period when a leg is not in contact with the ground because it moves towards the next step location, are generated by interpolating the step locations with spline polynomials. All these stages work offline and create reference trajectories.

Online execution for walking is done by tracking the reference trajectories via a Linear-Quadratic-Gaussian controller (LQG). To cope with external pushes on the robot, an additional online step adjustment process is introduced. This ensures that the robot remains stable even when the current real ZMP moves on the edge of the feet support polygon. By again utilizing Equations (7) and (9) the DCM at the end of the current step can be predicted by using the current real DCM as an initial value condition. With this new next DCM, the next step location can be adjusted to avoid falling. Note that this can be calculated effectively in closed form.

D. Variable Height Inverted Pendulum

The LIP model can be extended to allow height variations of the CoM which yields the Variable Height Inverted Pendulum (VHIP). In his work [1] Caron utilizes the VHIP to improve bipedal stabilization. Because of the conservation of angular momentum, increasing the height of the CoM can capture some of the robot's motions when it is pushed. This allows to avoid falling over without the requirement of doing an additional capture step, even when with the LIP model the ZMP would move to the edge of the feet support region. Changing the CoM in the vertical z direction allows shifting the ZMP to a certain extent back within the support region of the feet, which may be required for keeping balance.

Similar to the approach in Section III-C, Caron expresses the VHIP dynamics via the DCM $\boldsymbol{\xi}$. Note, that here the CoM velocity in the z direction may not be zero, so $\boldsymbol{\xi}$ is the 3D DCM as defined in [8]. The 3D DCM extends our previous definition (7) where ω is now changing over time. Thus, the DCM may not be the same height as the CoM, see Figure 1 for comparison.

To derive a controller that stabilizes around a given reference trajectory, which may come from a previous optimization step, Caron formulates the DCM error dynamics $\Delta\hat{\boldsymbol{\xi}}$, which is the difference between the current and the desired DCM. $\Delta\hat{\boldsymbol{\xi}}$ is then linearized around the reference trajectory. In the end $\Delta\hat{\boldsymbol{\xi}}$ should behave like a given desired error dynamics $\Delta\hat{\boldsymbol{\xi}}^*$, which is a P-controller with an error converging to zero. To make $\Delta\hat{\boldsymbol{\xi}}$ behave like $\Delta\hat{\boldsymbol{\xi}}^*$, a Quadratic programming problem is formulated that minimizes the distance between these desired and actual error dynamics. It is solved in realtime via a QP solver and optimizes over $\Delta\hat{\boldsymbol{\xi}}$ and the ZMP $\mathbf{p}$. Also, constraints like the CoM height limits are considered. The ZMP $\mathbf{p}$, resulting from the optimization, is tracked with a whole-body admittance controller for execution on the robot.

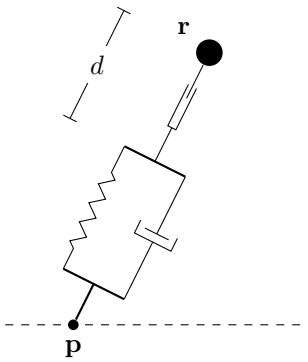

Fig. 2: ASLIP Model
Parallel spring-damper in series with a linear actuator that can vary the length d. Here the leg is the stance phase. The pendulum connects the CoM $\mathbf{r}$ with the pendulum base $\mathbf{p}$.

E. Actuated Spring Loaded Inverted Pendulum

To further extend the VHIP model, an additional spring and damper can be added, which results in the Actuated Spring Loaded Inverted Pendulum (ASLIP) [10]. Figure 2 illustrates the composition of the ASLIP model: The single ground contact $\mathbf{p}$ is first connected to a parallel spring-damper system. The spring-damper itself is connected to the CoM $\mathbf{r}$ via a linear actuator that can vary the length d. By incorporating the spring-damper the system gains more compliance.

In their work, Green et al. [10] propose a method utilizing the ASLIP model to generate walking trajectories for bipedal robots. Their approach is robust to unknown variations of the ground height. They model the ASLIP dynamics by considering two distinct cases. 1) During the flight phase there is no ground contact and the CoM just moves due to its constant velocity and the gravity. The leg moves according to the current control input $\ddot{d}$ and the spring-damper dynamics. 2) During the stance phase the foot position $\mathbf{p}$ is fixed and the linear actuator can apply force in leg direction through the spring-damper on the CoM. The control input consist of the linear actuator acceleration $\ddot{r}$.

Here it is assumed that the real robot has only one foot on the ground at maximum. Therefore, to map the ASLIP model to a real robot with two legs the following walking circle is considered: 1) First, the robot is in the flight phase where no leg has ground contact. 2) Then one leg makes ground contact during the stance phase. Afterward, these two steps repeat but with the other leg in contact during the stance

phase. As a simplification just one-half of this circle is taken into account. This is sufficient since the stance phases of both feet are equal when the robot is symmetrical.

To generate an open loop trajectory that is robust to variable ground height, Green et al. optimize over multiple disturbance cases simultaneously. Thereby each disturbance case has a different initial CoM height, which corresponds to varying the ground height. The key point here is to link the control trajectory of all these cases so that one control trajectory leads to the same final goal CoM height and velocity for all cases. To now solve for a control trajectory, an optimal control problem, see Section II-A, is formulated and solved via direct collocation.

Experimental evaluation in simulation shows that the applied simultaneous optimization of multiple disturbance cases leads to the desired robustness. Even though the optimized control trajectory is just played back in an open control loop, variations in ground height can be successfully rejected so that the CoM always reaches the final goal position.

Note, that the ASLIP model is simulated for evaluation, instead of using the real robot. Thus, for applying the optimized control trajectory to a real robot, additional effort is necessary for mapping control actions.

IV. Single Rigid Body Based Approaches

A robot model often used in trajectory planning for walking robots is the Single Rigid Body Model (SRBM), also called centroidal dynamics [7, 3]. Here the robot is simplified as a single rigid body on which forces and torques from contact points $\mathbf{c}_i$, as well as the gravity, can act. The mass m of the rigid body has an expansion and is distributed over the body. This differs from pendulum models where the robot is just treated as a point mass. Considering the mass distribution allows to take the effects of moments on the angular velocity into account. The legs and feet are massless in this model.

For describing the dynamics we separate the effects of the forces of the contact points and the gravity on the CoM $\mathbf{r}$ from the effect of the contact points on the angular velocity ω:

$$\ddot{\mathbf{r}} = \sum_i \frac{\mathbf{F}_i}{m} + \mathbf{a}_g \tag{10}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \sum_i (\mathbf{d}_i \times \mathbf{F}_i + \mathbf{m}_i) \tag{11}$$

Here we consider the 3D case, so all vectors are three-dimensional. Figure 3 shows a two-dimensional illustration of the SRBM. Equation (10) indicates that the sum of all contact forces $\mathbf{F}_i$ and the gravity $\mathbf{a}_g$ lead to an acceleration of the CoM position $\mathbf{r}$. In Equation (11) the inertia matrix $\mathbf{I}$ describes the distribution of the mass. Note that $\mathbf{I}$ is expressed in the CoM frame and has to be fixed w.r.t. the CoM, to that $\mathbf{I}$ is constant. We can observe that torques acting on the body, i.e. the right side of (11), cause a change in angular velocity $\boldsymbol{\omega}$. The whole torque acting on the body consists of the individual torques $\mathbf{m}_i$ applied by each contact, and the torques induced by the displacement $\mathbf{d}_i$ of the contact forces from the CoM. The vector $\mathbf{d}_i$ is expressed in the CoM frame and points from the CoM to the contact point $\mathbf{c}_i$. Note, that when $\boldsymbol{\omega} \neq 0$ then the direction of $\mathbf{F}_i$ and $\mathbf{d}_i$ will change over time even when the global contact locations and forces don't change. This is because both vectors are expressed in the CoM frame and the body rotates.

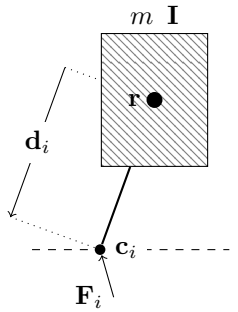

Fig. 3: Single Rigid Body Model
The rigid body with CoM location $\mathbf{r}$ has the mass m and its mass distribution is captured by the inertia matrix $\mathbf{I}$. The vector $\mathbf{d}_i$ describes the displacement from the CoM to the contact location $\mathbf{c}_i$. From the contact location the force $\mathbf{F}_i$ and torque $\mathbf{m}_i$ (not shown in the Figure) act on the body. Note that there are generally multiple contacts, although just one is shown here.

A. MPC with Linearized Centroidal Dynamics

A common way to utilize the SRBM is its integration into an MPC optimal control scheme. Here a common challenge is to minimize the optimization time of the optimal control problem to increase reactivity. In their work, Ding et al. [7] cope with this by linearizing the centroidal dynamics around the current state.

Ding et al. consider a fixed walking gait where single support phases of the two feet are alternating without an intermediate flight phase. Therefore, always one of the feet is on the ground. All timings, e.g. when a touchdown occurs, are fixed. These restrictions allow to model the dynamics of changing foot contact locations $\mathbf{c}$ by simply moving the foot instantaneously directly before it makes contact with the ground during touchdown. Note, that this immediate movement of a foot does not violate the dynamics, since legs and feet are considered massless. The location change of the ground contact from the previous to the next location is denoted by $\delta\mathbf{c}$. The foot contact location is incorporated into the SRBM dynamics (11) via $\mathbf{d}_i = \mathbf{c} - \mathbf{r}$.

The states of SRBM dynamics, e.g. CoM position $\mathbf{r}$ and CoM rotation, are summarized together with the current foot contact location in the state vector $\mathbf{x}$. The contact force $\mathbf{F}_i$ and contact torque $\mathbf{m}_i$ are combined into the control vector $\mathbf{u}$. This results in the dynamics function $f(\mathbf{x}, \mathbf{u}) = \dot{\mathbf{x}}$. Now f is linearized around the current state $\mathbf{x}^*$ and control $\mathbf{u}^*$ values. The dynamics are also time-discretized by doing Euler integration of the linearized dynamics. This leads to

the linear and time discrete form, where t is the time step:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + T_s[\underbrace{f(\mathbf{x}^*, \mathbf{u}^*) + \mathbf{A}\Delta\mathbf{x}_t + \mathbf{B}\Delta\mathbf{u}_t}_{\text{linearized dynamics}}] + g_t(\delta\mathbf{c}) \tag{12}$$

$$\text{where} \quad \Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}^* \quad \Delta\mathbf{u}_t = \mathbf{u}_t - \mathbf{u}^*$$

For the Euler integration, the linearized dynamics is multiplied with the timestep size T_s. The matrices $\mathbf{A}$ and $\mathbf{B}$ are calculated by taking the derivates of the dynamics f at the current observed state $\mathbf{x}^*$ and control $\mathbf{u}^*$ w.r.t. the state and control, see Equation 13.

$$\mathbf{A} = \left.\frac{\delta f}{\delta \mathbf{x}}\right|_{\mathbf{x}^*,\mathbf{u}^*} \qquad \mathbf{B} = \left.\frac{\delta f}{\delta \mathbf{u}}\right|_{\mathbf{x}^*,\mathbf{u}^*} \tag{13}$$

The function $g_t(\delta\mathbf{c})$ is just active (non-zero) directly before the next touchdown and moves the contact location by $\delta\mathbf{c}$. $g_t(\delta\mathbf{c})$ is required since moving the contact locations is not part of the dynamics f.

Finally, Ding et al. formulate an optimal control problem for planning walking trajectories using (12):

$$\begin{aligned}
\min_{\mathbf{x}_t,\mathbf{u}_t,\delta\mathbf{c}} \quad & \sum_{t=0}^{N} \gamma^t l_t(\mathbf{x}_t, \mathbf{u}_t) \\
\text{s.t.} \quad & \mathbf{x}_{t+1} = (12) && \text{(dynamics)} \\
& \mathbf{x}_0 = \mathbf{x}_{\text{start}} && \text{(start state)} \\
& 0 \geq g_s(\mathbf{x}_t, \delta\mathbf{c}) && \text{(kin.-, step- constraints)} \\
& 0 \geq g_f(\mathbf{u}_t) && \text{(friction constraints)}
\end{aligned} \tag{14}$$

Here control and state trajectories $\mathbf{x}_t, \mathbf{u}_t$, as well as the next two step-location changes $\delta\mathbf{c}$, are optimized over a fixed time horizon of N steps. This optimization over discretized state and control trajectories is very similar to a direct multiple shooting formulation, see Section II-B.2. Also, just like in direct multiple shooting, the state trajectory variables are linked together via constraints containing the time discrete dynamics (12). Additional kinematic constraints limit the maximum leg extension and step size based on the robot's capabilities. The height value of the contact also has to match the given terrain height at the specific contact location (step constraints). To avoid slipping, friction constraints ensure that the contact forces of the feet, which are modeled as line constants, stay within a defined friction cone.

The objective to minimize consists of the accumulated cost function $l_t(\mathbf{x}_t, \mathbf{u}_t)$ over all the time steps. Thereby the cost function is discounted with $\gamma \leq 0$ to decrease the weight of the time steps further in the future. This is beneficial since the errors of the linearization will increase towards these later time steps. The cost function $l_t(\mathbf{x}_t, \mathbf{u}_t)$ itself penalizes deviations from the commanded CoM velocity, deviations from a reference control value and deviations from a reference step location. The reference control value is just the required force to hold the body horizontally over the foot. The reference step location can be computed via a heuristic, e.g. the capture point.

Since all the constraints are linear and the objective is quadratic the OC problem can be formulated as a quadratic programming (QP) problem and thus solved efficiently. This allows repetitive execution as MPC with a solving frequency of 100Hz.

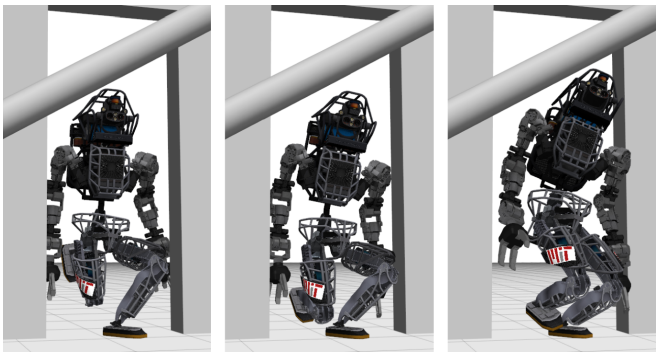

Fig. 4: Atals walks Through an Obstructed Door Using the Approach Presented in Section IV-B. Figure from [3].

To verify their approach, Ding et al. track the first value of solved trajectories from the MPC with a Task-Space controller in simulation. The Task-Space controller outputs the necessary joint torques to track the trajectories. They demonstrate that terrain with varying heights can be traversed. But the maximum slope of the terrain has to be limited, otherwise, their method fails. Note, that linearization around the current state and not around a reference trajectory offers the advantage of not becoming unstable when diverging too much from the reference. However, it induces increasing errors for later time steps within the predictive horizon. Therefore, the maximum length of the prediction horizon of the MPC is limited and the discounting by γ has to be used in the cost function.

B. Centroidal Motion Planning with Whole Body Kinematics

For planning motions of humanoids, the centroidal dynamics model can be combined with the full-body kinematics during optimization. Dai et al. [3] explore this approach, offering the advantage that potential collisions of all body links can be considered during planning. Moreover, they allow for the optimization of the non-fixed contact sequence of the feet and hands. Ultimately, everything is formulated as an optimization problem.

To incorporate the dynamics, Dai et al. add the two equations of the centroidal dynamics, (10) and (11) as constraints. Since also the kinematic joint configurations are considered, the inertia matrix $\mathbf{I}(\mathbf{q})$ here also depends on the current joint configuration $\mathbf{q}$. This is because different joint configurations change the mass distribution, e.g. when the legs are strongly bent the mass is more concentrated than when the legs are fully extended. This way, the centroidal dynamics model is extended by taking the changes of the mass distribution into account. The relationship between contact locations $\mathbf{c}_i$ and (11) is expressed via $\mathbf{d}_i = \mathbf{c}_i - \mathbf{r}$. This is similar to the work shown in Section IV-A. Note, that here arbitrary contact schedules are allowed, e.g. zero, one or two legs are on the ground. The set of contacts does not just include the feet but also the hands of the robot.

The joint configuration has to be linked to the centroidal dynamics model. To do this, Dai et al. formulate kinematic constraints:

$$\mathbf{r} = kin_{CoM}(\mathbf{q})$$
$$\mathbf{c}_i = kin_{c_i}(\mathbf{q})$$

Here the CoM location $\mathbf{r}$ is restricted to the value resulting from the forward kinematics $kin_{CoM}(\mathbf{q})$ of the current joint configuration. The contact locations have to match the forward kinematics for computing the corresponding contact location $kin_{c_i}(\mathbf{q})$. This way, the inverse kinematics will indirectly emerge when solving the later formulated optimization problem that has these constraints. Both constraints are summarized in the constraint g_{kin}, which is later used in the optimization.

A major reason for incorporating the joint configurations lies in the ability to reason about the locations of the robot's links. This can be used to check for collisions of the links and add additional constraints to avoid these collisions. For this purpose, Dai et al. attach a convex collision geometry to each link and each global object, e.g. to walls or doors. Now minimum distances between each pair i, j of these collision geometries can be formulated by the function d_{ij}. To avoid collisions, the value of d_{ij} has to stay above a minimum distance for each potential collision pair. Since this would result in a large number of constraints, these restrictions on d_{ij} are considered by putting the value of constraint violations into a hinge-loss-like function. These loss values of all collision pairs are then summed up. The sum value becomes zero when all constraints are not violated. Therefore, the number of required constraints for collisions can be reduced to just one, i.e. g_{coll}.

Finally, the planning of trajectories is formulated via an Optimal Control problem. First, the dynamics are time discretized via numerical integration $f_{int}(\mathbf{x}_t)$. Thereby, next state $\mathbf{x}_{t+1}$ depends on the previous state $\mathbf{x}_t$ values. Note that in contrast to the work presented in Section IV-A, here the discretized dynamics are not linear. Control and state values are not separated here, so there is just a state trajectory $\mathbf{x}_t$. Dai et al. solve the Optimal Control problem with direct multiple shooting, see Section II-B.2:

$$\begin{aligned}
\min_{\mathbf{x}_t, \delta o_t} \quad & \sum_{t=0}^{N} l_t(\mathbf{x}_t)\delta o_t & \\
\text{s.t.} \quad & \mathbf{x}_{t+1} = f_{int}(\mathbf{x}_t) & \text{(dynamics)} \\
& \mathbf{x}_0 = \mathbf{x}_{\text{start}} & \text{(start state)} \\
& 0 = g_{kin}(\mathbf{x}_t) & \text{(kinematic constraints)} \\
& 0 = g_{coll}(\mathbf{x}_t) & \text{(collision constraints)} \\
& 0 \geq g_{contact}(\mathbf{x}_t) & \text{(contact regions)} \\
& 0 \geq g_{fric}(\mathbf{x}_t) & \text{(friction constraints)} \\
& 0 \geq g_{joint}(\mathbf{x}_t) & \text{(joint limits)} \\
& \text{Eq. (16)} & \text{(comp. contact constraints)}
\end{aligned} \tag{15}$$

Here the state $\mathbf{x}$ consists of the CoM position, CoM rotation, joint configuration $\mathbf{q}$ and the first derivatives of these quantities. The dynamics for $\mathbf{q}$ just results from integrating the joint velocities $\dot{\mathbf{q}}$. Contact locations $\mathbf{c}_i$, contact torques and forces are also part of the state. The state values of all time steps are optimization variables. Also, the durations δo_t of the individual time steps are optimized. This allows to adapt the contact timings. Additionally, there are the previously discussed kinematic and collision constraints. Since for every time step the joint configuration is known during optimization, supplementary constraints for respecting the joint limits, which are part of g_{joint}, can be added. The contact location can also be restricted to allowed contact areas, e.g. stepping stones or handles for the hands to hold on. These contact restrictions are formulated in the constraint $g_{contact}$.

The cost function $l_t(\mathbf{x}_t)$ which is minimized over the whole trajectory, contains the norms of the deviation from the nominal joint configuration, the joint velocities and CoM accelerations. So all these entities will be minimized.

One set of constraints is still missing from the optimal control problem. It has to be ensured that the contact forces and torques are only non-zero when the corresponding contact touches the ground or another object in the scene. Dai et al. offer two different approaches to this: 1) When the contact sequence is fixed in advance, the condition that the contact forces and moments must be zero is added for time steps in which no contact takes place. 2) When the contact sequence should be optimized, complementary constraints are added to each time step. Therefore, we consider the distance $\phi_j(\mathbf{q})$ for each contact j to its contact surface. The complementary constraints ensure that for a timestep either the distance to the contact surface is zero or the normal contact force F_j^n is zero, see (16a).

$$\phi_j(\mathbf{q})F_j^n = 0 \tag{16a}$$
$$\phi_j(\mathbf{q})[\tau_j]^2 = 0 \tag{16b}$$
$$\begin{aligned} F_j^n(\mathbf{c}_{j,t} - \mathbf{c}_{j,t-1})^T\mathbf{t}_x = 0 \\ F_j^n(\mathbf{c}_{j,t} - \mathbf{c}_{j,t-1})^T\mathbf{t}_y = 0 \end{aligned} \tag{16c}$$

The same holds for the corresponding complementary constraints for the contact torques τ_j, see (16b). Dai et al. also ensure that the contacts do not slide when in contact with the ground with constraint (16c). The unit vectors $\mathbf{t}_x, \mathbf{t}_y$ are tangent vectors of the contact surface and are orthogonal to the normal force F_j^n. When in ground contact, i.e. F_j^n is non-zero, (16c) states that the position change of a contact location between two timesteps projected onto the contact surface tangent directions must be zero, i.e. no sliding along the surface. Additionally, there are constraints for the contacts to obey a defined friction cone which are defined as g_{fric}.

The optimization problem with all these constraints is solved with a numerical solver. Note that solving one trajectory takes minutes to hours. Dai et al. show that their approach works with the Atlas humanoid robot for multiple scenarios in simulation where a feedback controller is used for executing calculated trajectories. In simulation the Atlas robot successfully walks over flat terrain, does jumps and

walks through an obstructed door, see Figure 4. Dai et al. also demonstrate working optimization of the contact schedule.

V. FULL DYNAMICS PLANNING

Instead of using reduced models as shown in Sections III and IV, the full dynamics model, which includes the effects of the links, may be utilized. With the full dynamics, the joints and their respective velocities and acceleration as well as the actual link masses are considered. This offers the advantage that planned optimal joint trajectories more closely obey the dynamics of the real system. However, managing the increased computational effort is a challenge.

A. Whole Body MPC with Full Dynamics

An interesting approach for planning optimal trajectories for humanoid walking using the full dynamics is implemented by Dantec et al. [5]. In their work, they solve the constructed OC problem with Differential Dynamic Programming (DDP). The approach is then deployed on a real robot via MPC. Here a fixed contact schedule with predefined timings is assumed.

For the dynamics model, the effects of the joints on the moments of the links are considered by utilizing the inertia matrices of all the links. Also, the effects of gravitation, Coriolis and centripetal forces are part of the dynamics. To reduce computational load, just the joints of the legs and two for the torso are taken into account, the arms are locked in a fixed position during walking.

The formulated optimal control problem for generating walking trajectories uses the joint values $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ as the state $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$, the control vector $\mathbf{u}$ consists of the joint torques τ. Similar to direct shooting, the dynamics are time discretized and added as constraints for the corresponding next state value $\mathbf{x}_{t+1} = f_{int}(\mathbf{x}_t, \mathbf{u}_t)$, so that the next state is equal to the integrated dynamics $f_{int}(\mathbf{x}_t, \mathbf{u}_t)$ with the previous control and state values. The resulting OC problem is shown in (17).

$$\begin{aligned} \min_{\mathbf{x}_t, \mathbf{u}_t} \quad & \sum_{t=0}^{N} l(\mathbf{x}_t, \mathbf{u}_t, t) \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = f_{int}(\mathbf{x}_t) \qquad \text{(dynamics)} \\ & \mathbf{x}_0 = \mathbf{x}_{\text{start}} \qquad \text{(start state)} \end{aligned} \tag{17}$$

This formulation is similar to (15) in Section IV-B, except that the step times are not optimized but just the state and control values, and that the constraints just contain the dynamics. The objective of the OC problem consists of the accumulated costs $l(\mathbf{x}_t, \mathbf{u}_t, t)$ over all time steps. Note, that to be able to solve the OC problem in real time, additional constraints are included implicitly as penalty terms in the cost function. The cost function for each time step consists of the following five terms:

1) A cost for deviating from the nominal state and control values. These nominal values describe the initial robot joint configuration (half-sitting position) and the required joint torques to compensate the gravity.
2) To make a foot that is currently in the swing phase move to the next defined contact location on the ground, a deviation cost for the feet to follow desired swing trajectories is included. These desired swing trajectories are predefined and interpolate between two subsequent contact locations.
3) A penalty term for ensuring that the kinematic constraints and control bounds are obeyed.
4) To avoid slipping, the feet have to stay in a defined friction cone. Since adding this constraint directly to the cost as a penalty function leads to long solving times, Dantec et al. instead include a cost for following a reference wrench contact trajectory $\boldsymbol{\tau}_d(t)$ for each foot. Thereby $\boldsymbol{\tau}_d(t)$ consists of the contact forces and torques that directly emerge from the predefined contact schedule. $\boldsymbol{\tau}_d(t)$ is always zero, except when the corresponding foot is in ground contact the z-axis component of $\boldsymbol{\tau}_d(t)$ is equal to the required force to support the robot's weight.
5) Additional to ensure obeying the friction cone constraint, a term for regulating the Center of Pressure (CoP) is added.

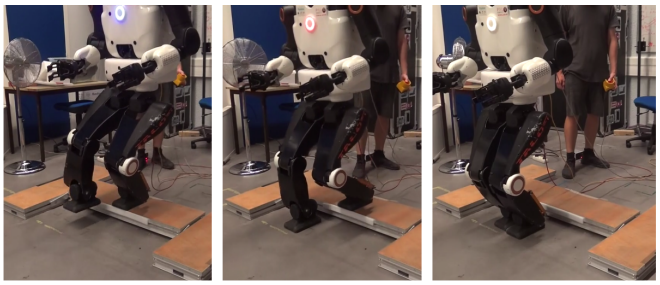

Fig. 5: Talos Climbs Down a 10cm Step Using the Approach Described in Section V-A. Figure from [5].

The OC problem with the described cost function is solved via DDP, see Section II-B.3. A nice consequence of using DDP is that it yields not just optimal trajectories as a solution, but also optimal linear feedback gains. These feedback gains can be used to implement a simple controller that executes the optimized trajectories.

For the OC problem a predictive horizon of 1.5s is considered. Thereby the horizon is split into 150 discretization points. Due to the efficient constraint formulation, DDP can solve the problem in one iteration, which takes on average 15ms. Note, that sometimes the solving time increases up to 50ms. After DDP finds a solution, it is executed using the resulting optimal feedback gains. The OC is then solved again using the last real observed state as a new starting condition. This way, the optimal linear feedback controller bridges the time gap resulting from the solving time of the DDP.

Dantec et al. verify their approach using the Talos humanoid robot. They show that Talos can walk over flat terrain and climb up and down a step, see Figure 5. Note that for each of these scenarios the contact schedule and interpolated feet swing trajectories have to be adapted. Additionally,

	Method	Objective/Solving	Reactiveness			Can Optimize			
			Open Loop OC	Reactive MPC	Reactive, Without OC	Step Pos	Gait Timing	Contact Pattern	CoM Trajectory
Pendulum	LIP Walking Pattern [11], III-A	- / closed-form				×	×	×	✓
	LIP DCM Walking [12], III-C	- / closed-form, LQG			•	✓✓	×	×	✓✓
	VHIP DCM Stabilization [1], III-D	min tracking error / QP formulation			•⋆	×	×	×	✓✓
	ASLIP Opti [10], III-E	none / direct collocation	•			×	✓	×	✓
SRBM	MPC Lin. Centroidal [7], IV-A	track command vel. / QP formulation		•		✓✓	×	×	✓✓
	Centriodal with full Kin. [3], IV-B	min accelerations … / multiple shooting	•			✓	✓	✓	✓
Full Dyn.	Whole Body MPC [5], V-A	track swing ref …/ DDP		•		×	×	×	✓✓

⋆ Requires a given reference trajectory.
✓ Can be optimized just offline.
✓✓ Can be optimized online.

TABLE I: Comparison of Presented Methods

The column "Objective/Solution" shows how a method calculates a solution and, in the case of using an optimization problem, how its objective is formulated. "Reactiveness" describes if a method can react online to external disturbances. Thereby, we differentiate whether an Optimal Control formulation is used. When no sub-column in "Reactiveness" is selected, a method is not using Optimal Control and can not react online to disturbances. The column "Can Optimize" lists the properties a method can adapt/optimize. Note that "Gait Timing" relates to the duration of the gait phase (e.g. double/single support phases) while "Contact Pattern" describes the sequence in which the feet make contact. Generally, the values a method can not adapt/optimize are also required input values.

their approach requires careful tuning of the weights of the individual terms in the cost function to produce stable motions.

VI. COMPARISON OF METHODS

To compare the presented methods we can look at the different properties in Table I. First, we have to differentiate between methods that just produce reference trajectories while not supporting online reactions to disturbances, and methods that include this reactive behavior. Furthermore, some methods employ Optimal Control for predictive planning, while others do not utilize this approach. The combination of Optimal Control and reactiveness results in doing MPC.

Most of the pendulum-based approaches are not based on Optimal Control, but either utilize closed-form solutions or employ another optimization problem. The methods presented in Sections III-A and III-C both focus on deriving CoM trajectories from predefined step locations. Note, that in both approaches the CoM height is fixed. The work of Kajita et al., III-A, just allows for the creation of reference trajectories from the step locations. In contrast, in III-C this idea is extended to allow for reactive adaptation of the CoM trajectory and the step locations to avoid falling. Here the predefined reference step positions come from a step planner with fixed parameters.

In comparison, the work by Caron, see Section III-D, focuses on stabilization during standing and therefore does not include adaptation of step positions. However, it improves stability since varying the CoM height can be used to avoid falling. Note, that this approach can be adapted for walking motions. In this case, the steps and CoM path would be predefined to provide a reference trajectory. An advantage of these simpler pendulum-based methods is the fast solving time, e.g. 0.1ms [1], which yields high feedback control rates.

The only pendulum-based method that does predictive planning with OC is presented in Section III-E. A feature of this method is that it allows the optimization of gait phase timings, but the gait sequence is fixed (alternating flight phase and one-foot contact phase). Note that here just the gait cycle is optimized and the actual terrain structure is not considered. However, planning is done over multiple scenarios with varying flat ground heights, so the gait incorporates some implicit robustness on the terrain structure. This comes at the cost of increased solving times (4.3s on average), therefore this method can't be used for real-time feedback.

A combination of the more complex single rigid body dynamics model and linearization around the current state bridges the gap between Optimal Control and being reactive in the work presented in Section IV-A. Here the linearization enables real-time MPC that also incorporates changing foot locations for stabilization. Nevertheless, the gait timings and contact schedule, as well as reference contact locations have to be predefined. Additionally, the linearization leads to

errors in the predictive horizon which, inter alia, restricts the types of terrains this method works with.

In contrast, the approach described in IV-B offers the most flexibility, since here the whole gait schedule with timings, positions and pattern is part of the optimization. The incorporation of joint values in the planning, similar to [2] (see Section I-A), allows for additional consideration of joint and link collision constraints. But the gained flexibility comes at the cost of slow solving times, which makes this method non-real-time and unusable as MPC.

The last method V-A is similar to IV-A [7], when looking at Table I. Both methods offer fast (re)planning which is deployed as MPC. But V-A uses the full dynamics model without linearization which avoids large errors in the planning horizon. It also directly yields optimal low-level feedback gains. However, the contact locations can not be adapted during replanning and feet reference trajectories have to be predefined. But the authors plan to include online step adaptation in future work [5].

VII. CONCLUSION

In this report, we looked at various approaches for controlling humanoid-legged robots. We divided these into the three classes "pendulum based", "SRBM dynamics based" and "full dynamics". Finally, we compared the presented methods in terms of reactiveness and optimization capabilities. None of the considered methods allow optimizing over all walking parameters, shown in Table I while providing reactiveness. The work presented in Section IV-B is most flexible but not real-time capable.

VIII. APPENDIX

A. LIP Transfer Function Derivation

$$\begin{aligned} \ddot{r}_x &= \omega^2 r_x - \omega^2 p_x \\ &\downarrow \text{Laplace transform} \\ p^2 R_x(p) &= \omega_0^2 R_x(p) - \omega_0^2 P_x(p) \\ R_x(p)(p^2 - \omega^2) &= -\omega^2 P_x(p) \\ F(p) = \frac{R_x(p)}{P_x(p)} &= \frac{\omega^2}{-p^2 + \omega^2} = \frac{1}{1 - p^2/\omega^2} \end{aligned} \tag{18}$$

B. DCM Transfer Functions Derivations

$$\begin{aligned} \dot{\xi}_x &= +\omega \xi_x - \omega p_x \\ &\downarrow \text{Laplace transform} \\ p\Xi_x(p) &= +\omega \Xi_x(p) - \omega P_x(p) \\ \Xi_x(p)(p - \omega) &= -\omega P_x(p) \\ F_{\xi_x}(p) = \frac{\Xi_x(p)}{P_x(p)} &= \frac{\omega}{\omega - p} \end{aligned} \tag{19}$$

$$\begin{aligned} \dot{s}_x &= -\omega s_x + \omega p_x \\ &\downarrow \text{Laplace transform} \\ pS_x(p) &= -\omega S_x(p) + \omega P_x(p) \\ S_x(p)(p + \omega) &= \omega P_x(p) \\ F_{s_x}(p) = \frac{S_x(p)}{P_x(p)} &= \frac{\omega}{\omega + p} \end{aligned} \tag{20}$$

REFERENCES

[1] Stéphane Caron. "Biped Stabilization by Linear Feedback of the Variable-Height Inverted Pendulum Model". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9782–9788. DOI: 10.1109/ICRA40945.2020.9196715.

[2] Matthew Chignoli et al. *The MIT Humanoid Robot: Design, Motion Planning, and Control For Acrobatic Behaviors*. 2021. arXiv: 2104.09025 [cs.RO].

[3] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. "Whole-body motion planning with centroidal dynamics and full kinematics". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. 2014, pp. 295–302. DOI: 10.1109/HUMANOIDS.2014.7041375.

[4] Ewen Dantec, Michel Taix, and Nicolas Mansard. "First order approximation of model predictive control solutions for high frequency feedback". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4448–4455.

[5] Ewen Dantec et al. "Whole-Body Model Predictive Control for Biped Locomotion on a Torque-Controlled Humanoid Robot". In: *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE. 2022, pp. 638–644.

[6] Moritz Diehl et al. "Fast direct multiple shooting algorithms for optimal robot control". In: *Fast motions in biomechanics and robotics: optimization and feedback control* (2006), pp. 65–93.

[7] Yanran Ding et al. "Orientation-Aware Model Predictive Control with Footstep Adaptation for Dynamic Humanoid Walking". In: *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. 2022, pp. 299–305. DOI: 10.1109/Humanoids53995.2022.10000244.

[8] Johannes Englsberger, Christian Ott, and Alin Albu-Schäffer. "Three-dimensional bipedal walking control using divergent component of motion". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 2600–2607.

[9] Gabriel García, Robert Griffin, and Jerry Pratt. "MPC-based Locomotion Control of Bipedal Robots with Line-Feet Contact using Centroidal Dynamics". In: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. 2021, pp. 276–282. DOI: 10.1109/HUMANOIDS47582.2021.9555775.

[10] Kevin Green, Ross L Hatton, and Jonathan Hurst. "Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1445–1451.

[11] Shuuji Kajita et al. "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 1. IEEE. 2001, pp. 239–246.

[12] Mohammadreza Kasaei, Nuno Lau, and Artur Pereira. "A Robust Biped Locomotion Based on Linear-Quadratic-Gaussian Controller and Divergent Component of Motion". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1429–1434. DOI: 10.1109/IROS40897.2019.8967778.

[13] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. "Springer handbook of robotics". In: 2nd ed. Vol. 200. Springer, 2008. Chap. 17.4, pp. 430–432.

[14] Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. "Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 1084–1091. DOI: 10.1109/IROS.2009.5354662.

[15] Jifei Xu. "Biped walking trajectory design and stabilization". https://www.ideals.illinois.edu/items/102787/bitstreams/327204/data.pdf. MA thesis. 2017.