Optimization of a point-mass walking model using direct collocation and sequential quadratic programming

Chris Dembia

June 5, 2015
1 Introduction

It is possible to uncover some fundamental principles of human locomotion with a model that consists of only a point mass and a massless telescoping (compressible) leg. Srinivasan and Ruina found that such a model chooses to walk if asked to move at slow speeds, and to run if asked to move at high speeds [SR06]. The model they used is shown in Figure 1. The model interacts with its environment via a telescoping linear actuator (its leg). The motion consists of two phases: a stance phase, during which the stance leg is in contact with the ground; and a swing phase during which the point mass is in free flight. Specifically, their model chose between three distinct gaits, depending on the speed and step length of the gait: an inverted pendulum walk, an impulsive run, and a pendular run; these are shown in Figure 2.

Srinivasan and Ruina found these gaits by solving an optimal control problem in which the objective was to maximize the energy efficiency of the gait. The parameters in the optimization consisted of the initial state of the model and the force applied by the actuator throughout the stance phase. To evaluate the objective function, they performed a forward simulation (integration) of the model’s equations of motion, starting from the initial state.

Their method is called single shooting; the optimizer chooses an initial state and “shoots” to the final state by integrating. This method is typically challenging for optimizers, since the motion of a nonlinear dynamical system can be very sensitive to the initial state. Another
approach is to discretize the state of the model through time, and to enforce the dynamics at these time points through explicit constraints in the optimization problem. This method is called \textit{direct collocation} \cite{Bet10}. When formulated via direct collocation, optimal control problems are far less sensitive to variations in the parameters. Direct collocation problems have many more parameters than single shooting problems, but are extremely sparse.

In this work, we replicate the work of Srinivasan and Ruina using direct collocation instead of single shooting. We also solve the problem with a suite of different cost functions, and explore the effect of these different costs on the resulting motion.

\section{The problem}

Here, we describe the optimal control problem that Srinivasan and Ruina \cite{SR06} solved. The variables consist of the model’s state \( s(t) = (x(t), y(t), \dot{x}(t), \dot{y}(t)) \), where \((x(t), y(t))\) is the position of the point mass; the force applied to the ground by the leg \( F(t) \); and the duration of the stance phase \( t_s \). The step length \( d \), speed \( v \), and step duration \( t_{\text{step}} = d/v \) are specified. The leg length \( l(t) = \sqrt{x^2(t) + y^2(t)} \) has a maximum value \( l_{\text{max}} \) during the stance phase, and the leg also has a finite strength \( F_{\text{max}} \). The motion must also be periodic.

Formally, the optimal control problem is

\[
\begin{align*}
\text{minimize} \quad & C = \frac{1}{mgd} \int_{0}^{t_{\text{step}}} E(s(t), F(t)) \, dt \\
\text{subject to} \quad & m\ddot{x} = Fx/l, \quad 0 \leq t \leq t_s \quad \text{stance dynamics} \\
& m\ddot{y} = Fy/l - mg, \quad 0 \leq t \leq t_s \quad \text{stance dynamics} \\
& m\ddot{x} = 0, \quad t_s < t \leq t_{\text{step}} \quad \text{swing dynamics} \\
& m\ddot{y} = -mg, \quad t_s < t \leq t_{\text{step}} \quad \text{swing dynamics} \\
& 0 \leq l(t) \leq l_{\text{max}}, \quad 0 \leq t \leq t_s \quad \text{leg length} \\
& 0 \leq F(t) \leq F_{\text{max}}, \quad 0 \leq t \leq t_s \quad \text{leg strength} \\
& 0 < t_s \leq t_{\text{step}} \quad \text{stance duration} \\
& x(t_{\text{step}}) - x(0) = d \quad \text{distance traveled} \\
& \begin{bmatrix} y(0) \\ \dot{x}(0) \\ \dot{y}(0) \end{bmatrix} = \begin{bmatrix} y(t_{\text{step}}) \\ \dot{x}(t_{\text{step}}) \\ \dot{y}(t_{\text{step}}) \end{bmatrix} \quad \text{periodicity}
\end{align*}
\]

(1)

The point mass has mass \( m \), and \( g \) is the acceleration due to Earth’s gravity on Earth’s surface. For the integrand cost \( E \), Srinivasan and Ruina used the positive work performed by the leg, \( E = [F(t)l(t)]^+ \); with this \( E \), then \( C \) is the \textit{cost of transport}, which is a dimensionless measure of the energetic cost of traveling a certain distance. In their work, the dynamics constraints were enforced via forward integration. In this work, we use \( m = 50 \text{kg} \), \( g = 9.81 \text{m/s}^2 \), \( l_{\text{max}} = 0.5 \text{m} \), \( F_{\text{max}} = 1000 \text{N} \). Different gaits are found by varying the nondimensional step length \( D = d/l_{\text{max}} \) and speed \( V = v/\sqrt{gl_{\text{max}}} \).
2.1 Methods

Direct collocation

We discretize Problem 1 into \( T_{\text{step}} \) mesh points \( t_1, t_2, \ldots, t_{T_{\text{step}}} = t_{\text{step}} \). The transition from stance to swing occurs at time index \( i = T_s \); that is, \( t_s = t_{T_s} \leq t_{\text{step}} \). In order to express the second order dynamics of the system using first order differential equations, we introduce the variables \( w = \dot{x} \) and \( z = \dot{y} \) for the horizontal and vertical speed of the point mass. The discretization yields state variables \( s_1, s_2, \ldots, s_{T_{\text{step}}} \) (with \( s_i = (x_i, y_i, w_i, z_i) \)) and leg forces \( F_1, F_2, \ldots, F_{T_s} \).

We enforce the differential constraints using a numerical method for solving ordinary differential equations, namely the implicit Euler method: for a differential equation \( \dot{y} = f(y, t) \), we obtain \( y \) via

$$ y_{i+1} = y_i + h_i f(y_{i+1}, t_{i+1}) $$

We use an implicit integrator because they are unconditionally stable (the solution won’t “blow up”). The resulting finite-dimensional optimization problem is

$$ \begin{align*}
\text{minimize} & \quad C_d = \frac{1}{2} \sum_{i=1}^{T-1} h_i [E(s_{i+1}, F_{i+1}) + E(s_i, F_i)] \\
\text{subject to} & \quad x_{i+1} - x_i = h_i w_{i+1} \quad i = 1, 2, \ldots, T_{\text{step}} - 1 \\
& \quad y_{i+1} - y_i = h_i z_{i+1} \quad i = 1, 2, \ldots, T_{\text{step}} - 1 \\
& \quad w_{i+1} - w_i = h_i \frac{F_{i+1} x_{i+1}}{m l_{i+1}} \quad i = 1, 2, \ldots, T_s - 1 \\
& \quad z_{i+1} - z_i = h_i \left( \frac{F_{i+1} y_{i+1}}{m l_{i+1}} - g \right) \quad i = 1, 2, \ldots, T_s - 1 \\
& \quad w_{i+1} - w_i = 0 \quad i = T_s, \ldots, T_{\text{step}} - 1 \\
& \quad z_{i+1} - z_i = -h_i g \quad i = T_s, \ldots, T_{\text{step}} - 1 \\
& \quad l_i^2 \leq l_{\text{max}}^2 \quad i = 1, 2, \ldots, T_s \\
& \quad 0 \leq F_i \leq F_{\text{max}} \quad i = 1, 2, \ldots, T_s \\
& \quad 0 < t_s \leq t_{\text{step}} \\
& \quad x_{T_{\text{step}}} - x_1 = d \\
& \quad \begin{bmatrix} y_1 \\ w_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} y_{T_{\text{step}}} \\ w_{T_{\text{step}}} \\ z_{T_{\text{step}}} \end{bmatrix}.
\end{align*} $$

(2)

The scalars \( h_i = t_{i+1} - t_i \) are the integration step sizes. The performance cost is now approximated by the discretized \( C_d \), which is computed using the trapezoidal rule. This problem has a total \( n = 4T_{\text{step}} + T_s + 1 \) variables (4 states, the leg force during stance, and \( t_s \)) and 4(\( T_{\text{step}} - 1 \)) dynamics constraints. For all results shown in this report, we use \( T_s = 100, T_{\text{step}} = 200 \). The value of \( T_{\text{step}} \) is chosen to be sufficiently large so that the error in the ODE solution is below some tolerance, typically \( 10^{-3} \).

For all of the constraints, the parameters at only one or two of the mesh points is present; therefore, this problem is very sparse. Since \( E \) may be non-convex, and since the third and fourth constraints in Problem 2 are non-affine, this problem is not convex.
Sequential quadratic programming

We solve Problem 2 using sequential quadratic programming [GMS94]. At each iteration \( k \) of the method, we approximate the problem as a quadratic program about our current iterate \( q^{(k)} \). The solution of this QP, \( q^{(k)*} \), is used to compute a search direction \( \Delta q^{(k)} = q^{(k)*} - q^{(k)} \) along which to perform a line search. The result of this line search is the next iterate \( q^{(k+1)} \).

Within the QP, the quadratic objective is

\[
\nabla_q C_d^{(k)} (q - q^{(k)}) + \frac{1}{2} (q - q^{(k)}) \nabla^2_q C_d^{(k)} (q - q^{(k)}),
\]

where \( \nabla_q C_d^{(k)} \) and \( \nabla^2_q C_d^{(k)} \) are the gradient and Hessian evaluated at \( q^{(k)} \), respectively.

The only constraints that must be approximated for the QP are the third and fourth from Problem 2. We define the following:

\[
\begin{align*}
a_i &= w_{i+1} - w_i - h_i \frac{F_{i+1} x_{i+1}}{m l_{i+1}}, \\
b_i &= z_{i+1} - z_i - h_i \left( \frac{F_{i+1} y_{i+1}}{m l_{i+1}} - g \right), \\
u_i &= (w_{i+1}, w_i, x_{i+1}, y_{i+1}, F_{i+1}), \\
v_i &= (z_{i+1}, z_i, x_{i+1}, y_{i+1}, F_{i+1}).
\end{align*}
\]

The quantities \( a_i \) and \( b_i \) are the residuals in third and fourth constraints, and \( u_i \) and \( v_i \) are the parameters contained in the third and fourth constraints. The affine approximations to these constraints are

\[
\begin{align*}
a_i^{(k)} + (\nabla_u a_i^{(k)})^T (u_i - u_i^{(k)}) &= 0, \\
b_i^{(k)} + (\nabla_v b_i^{(k)})^T (v_i - v_i^{(k)}) &= 0.
\end{align*}
\]

The gradients in these equations are

\[
\begin{align*}
(\nabla_u a_i)^T &= \begin{bmatrix}
1 & -1 & h_i & -1 & h_i & -1 & \frac{F_{i+1} x_{i+1}}{m l_{i+1}} & \frac{F_{i+1} y_{i+1}}{m l_{i+1}} & -h_i & 1 & \frac{x_{i+1}}{m l_{i+1}}
\end{bmatrix}, \\
(\nabla_v b_i)^T &= \begin{bmatrix}
1 & -1 & h_i & -1 & h_i & -1 & \frac{F_{i+1} x_{i+1}}{m l_{i+1}} & \frac{F_{i+1} y_{i+1}}{m l_{i+1}} & -h_i & 1 & \frac{y_{i+1}}{m l_{i+1}}
\end{bmatrix}.
\end{align*}
\]

To solve the SQP, we use the SNOPT package [GMS05]. SNOPT is a package for solving large sparse non-convex problems, and approximates the Hessian using a BFGS update. For the problems we explore, SNOPT requires between 5 to 35 iterations to converge; on a modern personal computer, this takes about 1 minute. During the first few outer iterations, the QP solver takes potentially hundreds of steps. After about 5 outer iterations, when the iterate is nearly feasible, the QP usually only requires two steps.

Transcribing the problem

The procedure of converting Problem 1 into Problem 2 is called transcription. We use the PSOPT optimal control package to perform this transcription for us [Bec10]. PSOPT also iteratively solves the problem on larger meshes until an ODE error tolerance is satisfied.
Figure 3: Leg length and leg force for the three gaits produced by [SR06].

3 Results

Reproducing the results of Srinivasan and Ruina, 2006

By varying the values of the non-dimensional step length $D$ and non-dimensional speed $V$, Srinivasan and Ruina discovered three distinct gaits (Figure 2). Figure 3 shows the leg force $F(t)$ and leg length $l(t)$ for a specific instance of each of these three gaits. The pendular walk emerges from small $D$ and small $V$, the impulsive run emerges for all $D > 0$ with $V$ above 1, and the hybrid pendular run emerges from $D$ above 1 for $V < 1$.

We reproduced these results qualitatively using the methods presented in section 2.1; the three gaits are shown in Figure 4. The force $F$ is not smooth as a result of the non-smooth cost $E$. Notably, the pendular walk does not contain a swing phase, and the force for both pendular gaits is U-shaped.

Exploring different cost functions

We explore three different cost functions and analyze their effect on the resulting walking solution, similar to work done by [Sri11]. For all three, we use $D = 0.5$ and $V = 0.5$.

Minimize force over the gait cycle. With $E = F(t)$, the solution minimizes the amount of force used throughout the whole motion. The solution shows that this causes the introduction of a swing phase that lasts as long as the stance phase.

Minimize squared leg force. The cost $E = F^2(t)$ penalizes large forces. This objective might capture the desire to reduce joint loading for those with joint pain. With this gait, the model does not change its height at all, and the leg force remains close to only what is necessary to counteract gravity; $F(t) \approx mg$.

Minimize gross work. The cost $E = F(t)l(t)$ essentially allows the model to propel itself using the energy absorbed from breaking. Indeed, with this cost function, the net work over the gait is 0. The resulting motion is spring-like.
Figure 4: Three distinct gaits (left to right): pendular walk, impulsive run, pendular run.

Figure 5: Walking with different costs (left to right): leg force, squared leg force, gross work.
References


