Direct collocation tutorial

Friedl De Groote, Maarten Afschrift, Jente Willaert

December 17, 2019

Contents

Chapter 1

Dynamics of the pendulum test

1.1 Theory

In this tutorial, we use the pendulum test for spasticity [1] as an example to illustrate numerical simulations of movement, and the use of direct collocation to solve optimal parameter estimation and optimal control problems.

The pendulum test has been shown to be sensitive to the presence and severity of spasticity. While the patient is seated, the examiner drops the lower leg from the horizontal position, with the knee joint extended; the lower leg is then allowed to swing freely under the influence of gravity and joint kinematics are recorded.

We describe the dynamics of the lower leg during the pendulum test based on a torque-driven biomechanical model (Figure 1.1). The model consists of a planar lower leg segment with passive stiffness and damping to simulate non-contractile musculotendon properties. Active joint torques representing muscle contractile behavior consist of a constant baseline torque to represent muscle tone, and a delayed sensory feedback torque to simulate reflex activity (Figure 1.1).

Let us first consider a healthy subject in which reflex activity is absent. In that case, only gravity, damping and a constant baseline tone act on the lower leg. Knee joint motion is described by the equation of motion of the planar lower leg segment under the assumption that the knee joint center is stationary during the test:

$$I\frac{d^2\theta(t)}{dt^2} = mgl_c cos\theta(t) - T_b - B\frac{d\theta(t)}{dt}, \tag{1.1}$$

where t is time, θ is the knee joint angle ($\theta = 0$ corresponds to full extension and negative values correspond to knee flexion), I is the moment of inertia of the lower leg with respect to the knee, m is the mass of the lower leg, g is gravitational acceleration, and l_c is the distance between the knee joint center and the center of mass of the lower leg (Figure xx).

The movement of the lower leg is described by a second order differential equation

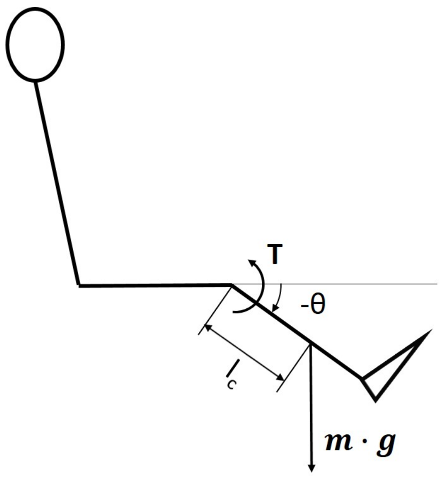

Figure 1.1 Pendulum test model.

(1.1), i.e., the second (but not the third or any higher) derivative of the knee angle with respect to time appears in the equation. Hence, the movement of the lower leg can be described by two states: knee joint angle θ and knee joint angular velocity $\dot{\theta}$, where $\dot{\theta}$ is short notation for $\frac{d\theta}{dt}$. And based on (1.1) we can write explicit equations for the state derivatives:

$$\frac{d\theta(t)}{dt} = \dot{\theta}(t), \tag{1.2}$$

$$\frac{d\dot{\theta}(t)}{dt} = \frac{1}{I}\left(mgl_c cos\theta(t) - T_b - B\dot{\theta}(t)\right). \tag{1.3}$$

1.2 matlab example

Chapter 2

Forward simulations

2.1 Theory

Based on the equations of motion (equations 1.2-1.3) we can numerically simulate the lower leg movement if we know the initial state, i.e., the knee joint angle θ_0 and angular velocity $\dot{\theta}_0$ at $t = 0$.

To this aim, we discretize the states. Or in other words, we represent the continuous states, $\theta(t)$ and $\dot{\theta}(t)$, over a time interval $[t_0\, t_f]$ by a finite number of values representing the state at discrete time instants i: $[\theta_0\, \theta_1\, \ldots\, \theta_i\, \ldots\, \theta_N]$ and $[\dot{\theta}_0\, \dot{\theta}_1\, \ldots\, \dot{\theta}_i\, \dot{\ldots}\, \dot{\theta}_N]$. For simplicity, let us consider $N+1$ equidistant time instants. In that case, the time between consecutive time instants Δt is constant and equals $\frac{t_f - t_0}{N}$.

Next, we approximate the exact time derivatives by numerical derivatives. Using a simple backward backward Euler scheme results in:

$$\frac{d\theta_i}{dt} \approx \frac{\theta_{i+1} - \theta_i}{\Delta t}, \tag{2.1}$$

$$\frac{d\dot{\theta}_i}{dt} \approx \frac{\dot{\theta}_{i+1} - \dot{\theta}_i}{\Delta t}. \tag{2.2}$$

Using these approximations for the derivatives, we can now discretize the dynamic equations 1.2-1.3:

$$\frac{\theta_{i+1} - \theta_i}{\Delta t} = \dot{\theta}_i, \tag{2.3}$$

$$\frac{\dot{\theta}_{i+1} - \dot{\theta}_i}{\Delta t} = \frac{1}{I}\left(mgl_c cos\theta_i - T_b - B\dot{\theta}_i\right). \tag{2.4}$$

We can solve these equations for θ_{i+1} and $\dot{\theta}_{i+1}$:

$$\theta_{i+1} = \theta_i + \dot{\theta}_i \Delta t, \tag{2.5}$$

$$\dot{\theta}_{i+1} = \dot{\theta}_i + \frac{\Delta t}{I}\left(mgl_c cos\theta_i - T_b - B\dot{\theta}_i\right). \tag{2.6}$$

Starting from the given initial states θ_0 and $\dot{\theta}_0$, we can iteratively solve equations 2.5-2.6 to find the states at times $1 \dots N$. Such procedure for numerical integration is often referred to as time marching, since the discrete states at consecutive time instants are iteratively computed. For an implementation, see Part2_Discretization.m (section Part 2A).

The use of variable instead of fixed time steps and more accurate discretization schemes than backward Euler can improve numerical accuracy of the integration. Matlab provides different solvers for numerical integration of ordinary differential equations (e.g., equations 1.2-1.3). For an example implementation using ode23, see Part2_Discretization.m (section Part 2B).

Alternatively, we can insert the approximations for the derivatives directly in equation 1.1 after bringing all the terms to the left hand side of the equation:

$$I\frac{\dot{\theta}_{i+1} - \dot{\theta}_i}{\Delta t} - mgl_c cos\theta(t) + T_b + B\frac{\theta_{i+1} - \theta_i}{\Delta t} = 0. \tag{2.7}$$

Equations 2.5 and 2.7 for $i = 1 \dots N$ form a set of equations that can be solve numerically, e.g. by using fsolve in matlab, for θ_i and $\dot{\theta}_i$ if the initial states θ_0 and $\dot{\theta}_0$ are known. For an implementation, see Part2_Discretization.m (section Part 2C). As opposed to time marching forward integration, here all discrete states are simultaneously computed.

2.2 Matlab example

Chapter 3

Controller

To do.

Chapter 4

Optimal parameter estimation and optimal control

Here, we discuss different numerical approaches to find the values for the baseline torque T_b and damping B that result in simulated knee angle trajectories that best match experimental knee angle trajectories.

Finding the model parameters T_b and B that result in the best agreement between measured and simulated knee angle trajectories requires solving a dynamic optimization problem. The agreement between measured and simulated knee angle trajectories can be described by

$$J = \sum_{i=1}^{N} (\theta_i - \hat{\theta}_i)^2, \tag{4.1}$$

where θ_i and $\hat{\theta}_i$ are respectively simulated and experimental joint angle at time instant i. Hence, the corresponding optimization problem is to minimize the cost function J with respect to optimization variables $p = [T_b\ B]$.

4.1 Direct shooting

Shooting methods evaluate the cost function J by using time marching forward integration to solve for θ_i. Gradient based optimization methods typically require many function evaluations to obtain derivative information through finite differences. Hence, in every iteration of the optimization algorithm the dynamics are evaluated multiple times through forward integration.

To obtain accurate approximations of the derivatives through finite differences, it is important that the same numerical operations are performed when evaluating $J(p)$ and $J(p + \Delta p)$. In this case, this means that the discretization scheme used by the integrator should be the same. For an example implementation using Matlab's fmincon see Part4_Shooting_Fmincon.m.

Shooting methods are not very well suited for systems with stiff dynamics, where the simulated state trajectories have a high sensitivity to the optimization variables. Typically, this sensitivity increases with simulation time.

4.2 Direct collocation

In contrast to shooting methods, direct collocation methods do not rely on time-marching to evaluate the dynamics. Instead, the discretized states are considered as additional optimization variables and the discretized state equations 2.5 and 2.7 are imposed as constraints. This results in a non-linear programming problem with a large number of optimization variables and constraints as compared to direct shooting. However, the sparsity of the resulting optimization problems makes them tractable. By reducing the sensitivity of the cost function to the optimization variables, direct collocation methods are often numerically more efficient than shooting methods. Note that the dynamic equations do not need to be satisfied in every iteration of the optimization algorithm when using direct collocation.

For an example implementation using Matlab's fmincon see Part4a_DirectCollocation _Fmincon.m. By exploiting sparsity and by using algorithmic differentiation, numerical efficiency can be improved. For example implementations using CasADi (www.casadi.org) see Part4b_DirectCollocation_Casadi.m and Part4a_DirectCollocation_Casadi_Opti.m.

Bibliography

[1] De Groote, F., Blum, K.B., Horslen, B.C., Ting, L.H. Interaction between muscle tone, short-range stiffness and increased sensory feedback gains explains key kinematic features of the pendulum test in spastic cerebral palsy: A simulation study. *PLoS ONE*, 13(10):e0205763, 2018.