Homework 06

1. The Matlab file `runjump.m` runs a simulation of Alexander's jumping model. The simulation includes integration of the equations of motion for ground contact, along with computation of the aerial phase to determine the eventual maximum height and the distance of the jump. The program then produces contour plots of the maximum height and distance as a function of run-up speed and leg plant angle. The last portion of the program demonstrates how the same computations may be repeated for the system with a tendon with zero compliance (i.e., no tendon at all). Use `runjump` and modify if necessary, to answer the following questions.
   A. Notice that the states are defined in Cartesian coordinates $[x,\ y,\ \dot{x},\ \dot{y}]^T$, but the state-derivative depends on non-Cartesian coordinates such as leg angle θ, knee angle ϕ, and knee torque T. From the state-derivative function, briefly summarize the steps to convert from and back to Cartesian coordinates for the state derivative.
   B. The contour plot shows that the optimum long jump distance is achieved for a fast run-up and a relatively steep plant angle. If a long jumper is particularly slow, is the optimum plant angle reduced or increased?
   C. The high jump simulation uses an initial knee angle of 170°. What is the effect of varying the knee angle on the maximal height? Is it necessary to change the run-up speed and plant angle to achieve this height?
   D. The high jump simulation uses a maximum torque of 2.5 mga (body mass m, half leg length a). What is the effect of varying on the maximal height? Is it necessary to change the run-up speed and plant angle to achieve this height?
   E. Vary the force-velocity curve parameter G (say between 1 and 5) to answer the same questions as in problems b and c. Plot the torque-angular velocity relationship (which comes from the force-velocity curve) for a few values of and explain what its effect is on that curve. How is G related to a_f from the Hill muscle model used previously?
   F. The high jump simulation assumes that the initial vertical velocity is 0. In actual jumpers, there is a slight downward velocity of about -0.2 $\sqrt{ag}$. Perform a simulation to determine if this downward velocity affects the optimum jumping strategy, or the maximal height.
   G. The model is simplified when there is no tendon compliance, as is implemented with the state-derivative function `fjumpc0`. Reverse-engineer that simulation: write a set of equations to describe it, and a short description of each equation. Also describe how it differs from the simulation with compliance.
   H. Using the parameter study of G as an example, perform your own parameter study, varying compliance C about the nominal value. Is it advantageous to have more or less tendon compliance, keeping other factors the same, to achieve the highest possible jump, and how does the optimal strategy change? How about the longest possible jump?

2. `runanthrowalk2` demonstrates the "anthropomorphic walking model," which employs human-like inertial parameters. The equations of motion are derived in Mathematica (see source or PDF), and implemented in the state-derivative function.

A. The state-derivative function is missing a few terms. Use the provided equations of motion to fill in the missing terms, and test them by evaluating energy conservation.
B. The total mechanical energy of the system is computed by `energyanthrowalk2`. Use it to test whether the energy change due to heelstrike is consistent with the potential energy change from descending the slope for a step. (See the step length calculated elsewhere in the file.). How would you expect this energy loss to change with increasing step length (keeping step frequency constant) or speed (keeping step length constant)?
C. The file also performs a parameter study varying radius of curvature R of the foot. How does increasing R affect walking speed, step length, and step time? Should it be more advantageous to have point feet or large curved feet?
D. Notice that the state-derivative and step-to-step transition functions both solve for accelerations or post-collision velocities using Matlab backslash "\". Why is it preferable to solve a linear system `Ax=b` with `b=A\x` instead of something like `b=inv(A)*x`? (A web search should yield a brief explanation.)