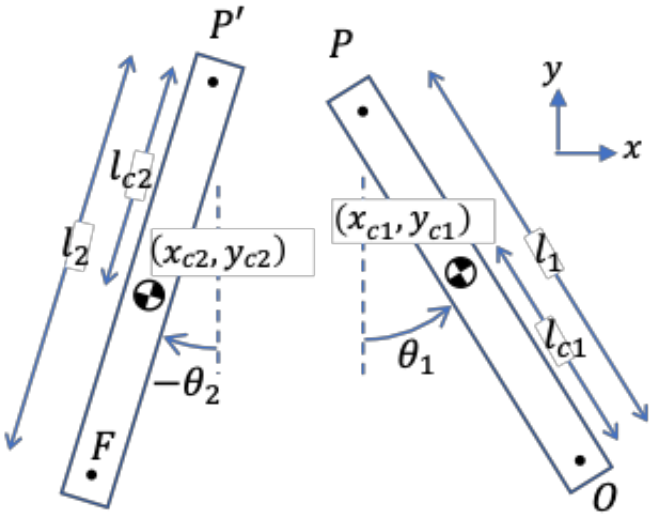
Homework 2

1. A two-link system has a forward kinematics Jacobian yielding $\dot{\mathbf{x}}_e \triangleq \left[\dot{x}_P, \dot{y}_P, \dot{\theta}_2\right]^T$, the translational velocity of the endpoint and angular velocity of that link (arranged in a single column). The Jacobian is multiplied by $\left[\dot{q}_1, \dot{q}_2\right]^T$.

$$J = \begin{bmatrix} L\cos q_1 + \frac{L}{2}\cos(q_1 - q_2) & -L\cos(q_1 - q_2) \\ -L\sin q_1 + \ldots & L\sin(q1 - q2) \\ -1 & 1 \end{bmatrix}$$

   A. What is a possible system diagram for this Jacobian? What is the length of the second segment?
   B. Fill in the missing entries of the Jacobian.
   C. What are the definitions of q_1 and q_2 ? Show on system diagram.
   D. Find the Jacobian yielding the translational velocity and angular velocity of the first link.

2. Find the constraint Jacobian C for the two segments shown, which are to be connected to form a planar biped model. The pose $\mathbf{X}$ is defined as $[x_{c1}, y_{c1}, \theta_1, x_{c2}, y_{c2}, \theta_2]^T$. Point O is to be attached to the ground origin, and the two pelvis points to each other, both with hinge joints. The constraint function should be the Jacobian of the position of point O relative to origin, and of the position error $\left(x_{P'} - x_P,\ y_{P'} - y_P\right)$, such that $C\dot{\mathbf{X}} = \epsilon$, with ϵ kept near zero.



   A. Find the missing elements:

$$C = \begin{bmatrix} 1 & 0 & l_{c1}\cos\theta_1 & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & -1 & \ldots & 0 & 1 & -l_{c2}\sin\theta_2 \end{bmatrix}$$

   B. For the same system, an event function would be needed to halt the simulation when the foot F strikes ground. What would be an appropriate event function (using $\mathbf{X}$)? And what would be an appropriate state vector for the ordinary differential equation?

$$z = \left[x_{c1}, y_{c1}, \theta_1, x_{c2}, y_{c2}, \theta_2, \dot{x}_{c1}, \dot{y}_{c1}, \dot{\theta}_1, \dot{x}_{c2}, \dot{y}_{c2}, \dot{\theta}_2\right]^T$$

   C. The pose Jacobian (forward kinematics) is

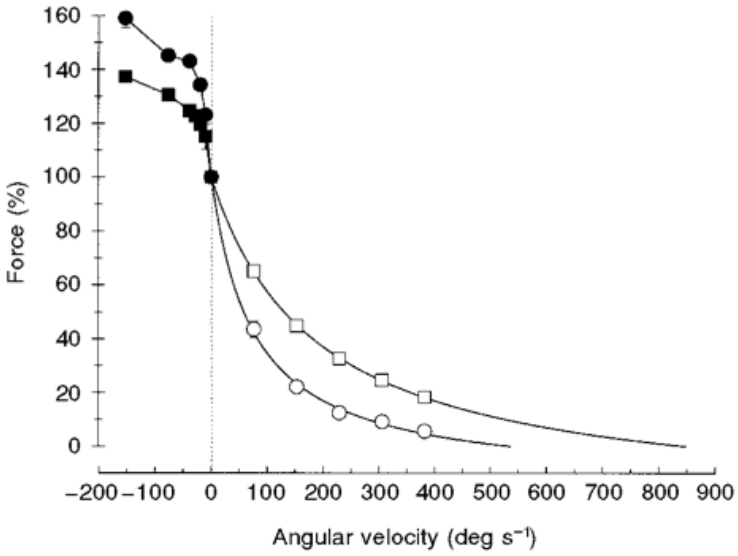
$$J = \begin{bmatrix} -l_{c1}\cos q_1 & 0 \\ -l_{c1}\sin q_1 & 0 \\ 1 & 0 \\ -l_1\cos q_1 & l_{c2}\cos q_2 \\ -l_1\sin q_1 & l_{c2}\sin q_2 \\ 0 & 1 \end{bmatrix}$$

Test whether $CJ = 0$, and in your own words, explain why it should be.

D. Suppose the segment masses are m_1, m_2. How would you find the Jacobian for the whole system center of mass? (Keep this simple, preferably a single matrix multiplication. $\left[\dot{x}_{\text{com}}, \dot{y}_{\text{com}}\right]^T = J_{\text{com}}\dot{q}$)

3. A computational way to minimize a function is to satisfy the necessary condition for a minimum. The necessary condition is that the partial derivatives of the function with respect to its (vector) input x must all be zero. You have already written `fjacobian`, which calculates the partial derivative numerically, and this must be paired with some code to determine the x satisfying the necessary condition. See `findmin_startercode` and `testfindmin` for a partial example.
   A. Write your own `findmin` function, using two functions you have already written.
   B. Notice that the test function uses a known solution $x^* = -Q^{-1}b$ to minimize $\frac{1}{2}x^TQx + b^Tx$. Briefly explain why this is the case.
   C. Even if the necessary conditions are satisfied, the solution x^* might not actually minimize the test function. Briefly explain what why. (It may help to browse a tutorial on numerical optimization.)

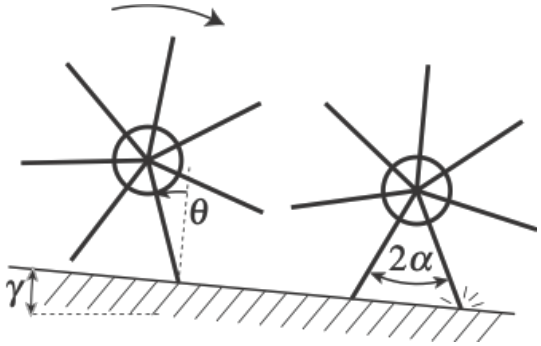
4. The normalized force-velocity curve is described by the formula $\tilde{F_{\text{CE}}} = (1 - \tilde{v_{\text{CE}}})/(1 + \frac{\tilde{v_{\text{CE}}}}{a_f})$, which has a single parameter a_f that describes the shape of the curve. Two other parameters are needed to specify the curve in physical units: the maximum shortening velocity and the peak isometric force. By trial and error, try to roughly estimate parameter values for each of the two curves shown here (except for peak force), and use them to plot the modeled force-velocity curves (during shortening only). Data from Ruiter et al. (2000) J. Physiol. 526.3: 671-681.



5. Devise a full step simulation of the simple rimless wheel shown at right. The system consists of massless legs at a fixed angle 2α apart, with a point mass located at the hip and a radius of gyration of r_{gyr}. It has only one degree of freedom and is simpler than systems you have already simulated. The specifications are as follows: The function should be called `rimlesswheelstep.m`, with similar output to the previous `rimlesswheelstance`, except that the first output should be the initial state of the next step. In the step-to-step transition, the

next step begins with the stance leg resets to α, and its angular velocity to $\dot{\theta}^{+} = \dot{\theta}^{-} \cos 2\alpha$ where plus and minus refer to states just before and after ground contact ($\dot{\theta}^{-}$ is the final velocity of the step that just ended).

A. Use `testrimlesswheel` to test your simulation. Note that there are multiple plots produced by the Matlab cells.
B. Notice that the states vary over time. Briefly explain in your own words the trajectories of θ and $\dot{\theta}$, for example, why $\dot{\theta}$ has an inverted U shape with each step.
C. Briefly explain the energy trajectory, for example why the energy decreases with each step, and why the energy within each step is very nearly, but not exactly, constant.
D. Briefly explain why the initial $\dot{\theta}$ for each consecutive step is changing, and asymptoting toward a steady value over many steps.



E. Suppose you wish to find an initial angular velocity $\dot{\theta}(0)$ such that the system produces a periodic motion, so that multiple steps can be taken one after the other, all repeating the same motion. How can this be implemented using numerical root-finding? Try to find the appropriate $\dot{\theta}(0)$.

Optional: It will be helpful to be able to derive equations of motion analytically. If you have access to Mathematica, try out the following demo to automatically generate equations.

6. We will perform simulations of the ballistic walking model (see the associated journal article on Google Drive).
   A. Download the Dynamics Workbench (Mathematica file `DynamicsWorkbench.m`), and run the Ballistic Walking demo in Mathematica. This demonstrates automated generation of equations of motion, and then simulating the system with from an initial condition, i.e., an initial value problem.
   B. The same equations of motion may be exported into an m-file and added to `ballisticwalk.m`, which demonstrates generally how dynamics simulations will be performed. Perform a simulation and check for energy conservation. Use the same initial conditions, but implement an event function that stops the simulation when the knee reaches full extension.