

# **Simulation of a 2-link Brachiating Robot with Open-Loop Controllers**

David Ufford  
Northwestern University  
June 2009

# 1. Project Overview

The goal of this project was to write a complete simulation of a 2-link swinging robot (the “MonkeyBot”). The simulation models swinging, free flight, and impacts with user-inputted parameters.

A secondary goal of this project was to attempt to identify the stability of the MonkeyBot with an open-loop controller. Gaits for horizontal swinging and their dependence on initial conditions were examined.

# 2. MonkeyBot Overview

The MonkeyBot is a two-link robot, currently in the prototype stage, which is designed to climb along vertical walls. It attaches to the wall surface using electromagnets at the end of each link. These magnets are configured to allow the device to pivot about the attachment point. The electromagnets can be simultaneously or individually activated as needed.

The two links are connected by an actuated rotating joint. With proper control, the motor at this joint can leverage the 2-link swinging dynamics to create desired motion. A combination of swinging the links and changing the activated electromagnet will allow the robot to move about the wall.

The MonkeyBot is modeled as two massive links connected by a central joint. The links of length  $L_1, L_2$  each have mass  $m_1, m_2$  and rotational inertia ( $I_1, I_2$ ). The center of gravity (CG) is located axially a distance  $r_1, r_2$  along the link.

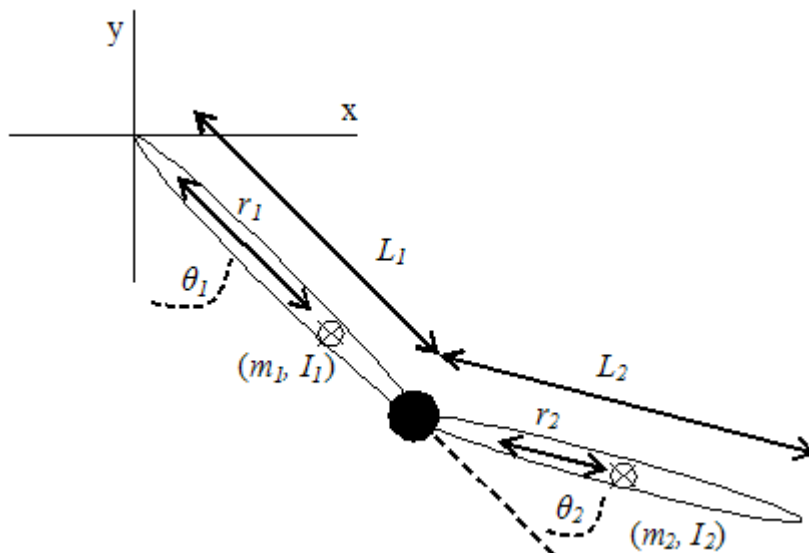


Figure 1: Diagram of 2-link MonkeyBot model with parameters.

If we contain the device to planar movement, the system can be fully expressed in terms of four generalized coordinates, contained in vector  $q$ .

$$q = \begin{bmatrix} x \\ y \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$(x, y)$  is the coordinate location of the “top” link.  $\theta_1$  is the angle of the top link relative to vertical, and  $\theta_2$  is the relative angle between the links.

The MonkeyBot is capable of engage in three general types of motion phases that are defined by the configuration of active magnets

1. Free flight - no magnets are active, system has four degrees of freedom.
2. Swinging – one magnet is activated, system has two degrees of freedom.
3. Fixed – both magnets activated so that the robot cannot move. Dynamics during this state are trivially zero, and are not analyzed.

Transitioning to the swinging or fixed phase involves activating the magnet on one or both links. The resultant clamping of the magnet to the wall surface is a collision that must be accounted for in dynamics and motion planning.

### 3. MonkeyBot Dynamics

#### 3.1 Motion Dynamics

The simplest way to obtain the mechanical dynamics of our system is via Lagrangian dynamics. The Lagrangian  $L$  of the system is defined as the difference between the kinetic ( $K$ ) and potential ( $V$ ) energies of the system.

$$L(q, \dot{q}) = K(q, \dot{q}) - V(q)$$

For our system, the KE and PE terms can be expressed as:

$$K = \frac{1}{2} \left[ m_1 v_1^2 + m_2 v_2^2 + I_1 \dot{\theta}_1^2 + I_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \right]$$

$$V = g(m_1 y_1 + m_2 y_2)$$

$v_1$  and  $v_2$  are velocities of the CGs expressed in the x-y coordinate system. Likewise  $y_1$  and  $y_2$  are the vertical positions of the CG in the x-y system. They are used in the previous equations for simplicity and conciseness, and are replaced by variables in the  $q$  coordinate system using the following transformations:

$$y_1 = y - r_1 \sin(\theta_1)$$

$$y_2 = y - L_1 \sin(\theta_1) - r_2 \sin(\theta_1 + \theta_2)$$

$$v_1^2 = \left[ \dot{x} + r_1 \cos(\theta_1) \dot{\theta}_1 \right]^2 + \left[ \dot{y} + r_1 \sin(\theta_1) \dot{\theta}_1 \right]^2$$

$$v_2^2 = \left[ \dot{x} + L_1 \cos(\theta_1) \dot{\theta}_1 + r_2 \cos(\theta_1 + \theta_2) (\dot{\theta}_1 + \dot{\theta}_2) \right]^2 + \left[ \dot{y} + L_1 \sin(\theta_1) \dot{\theta}_1 + r_2 \sin(\theta_1 + \theta_2) (\dot{\theta}_1 + \dot{\theta}_2) \right]^2$$

Once the substitutions are made in  $K$  and  $V$  for the generalized  $q$  coordinates, we can generate the Euler-Lagrange equations of motion, written below.  $F$  is the vector of applied external forces in the generalized  $q$  system.

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = F$$

The Euler-Lagrange gives us a system of 4 differential equations, one for each coordinate in  $q$ . The equations can then be manipulated and grouped into the standardized form:

$$F = M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) \quad *(1)$$

Where  $M(q)$  is the mass or inertia matrix,  $C$  is a matrix that contains centrifugal and coriolis terms, and  $g$  is a vector of gravitational forces. For our 2-link system, these matrices are:<sup>i</sup>

$$M(q) = \begin{bmatrix} m_1 + m_2 & 0 & (m_1 r_1 + m_2 L_1)c_1 + m_2 r_2 c_{12} & m_2 r_2 c_{12} \\ 0 & m_1 + m_2 & (m_1 r_1 + m_2 L_1)s_1 + m_2 r_2 s_{12} & m_2 r_2 s_{12} \\ (m_1 r_1 + m_2 L_1)c_1 + m_2 r_2 c_{12} & (m_1 r_1 + m_2 L_1)s_1 + m_2 r_2 s_{12} & I_1 + I_2 + m_1 r_1^2 + m_2 L_1^2 + m_2 r_2^2 + 2m_2 r_2 L_1 c_2 & m_2 r_2^2 + m_2 r_2 L_1 c_2 + I_2 \\ m_2 r_2 c_{12} & m_2 r_2 s_{12} & m_2 r_2^2 + m_2 r_2 L_1 c_2 + I_2 & I_2 + m_2 r_2^2 \end{bmatrix}$$

$$C(q, \dot{q}) =$$

$$\begin{pmatrix} 0 & 0 & -[(m_1 r_1 + m_2 l_1)s_1 \dot{\theta}_1 + (m_2 r_2 s_{12})(\dot{\theta}_1 + \dot{\theta}_2)] & -m_2 r_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 & 0 & (m_1 r_1 + m_2 l_1)c_1 \dot{\theta}_1 + m_2 r_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) & m_2 r_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 & 0 & -m_2 r_2 l_1 s_2 \dot{\theta}_2 & -m_2 r_2 l_1 s_2(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 & 0 & m_2 r_2 l_1 s_2 \dot{\theta}_1 & 0 \end{pmatrix}$$

$$G(q) =$$

$$\begin{pmatrix} 0 \\ g(m_1 + m_2) \\ g(m_1 r_1 + m_2 l_1)s_1 + gm_2 r_2 s_{12} \\ gm_2 r_2 s_{12} \end{pmatrix}$$

where  $s_1 = \sin(\theta_1)$ ,  $c_1 = \cos(\theta_1)$ ,  $c_{12} = \cos(\theta_1 + \theta_2)$ , etc.

The external force vector,  $F$  allows us to apply friction and motor torque to the system. Motor torque, if any, is inserted into the third row of the  $F$  vector, corresponding to a force applied to the relative  $\theta_2$  angle between the joints.

These equations describe the 2-link system with four degrees of freedom. Numerical integration of these general equation \*(1) with the given matrices will result in a simulated MonkeyBot in unrestricted free-flight motion.

### 3.2 Reduced Dynamics for Swinging State

While the free-flight dynamics fully describe the unrestricted system, we need to constrain one of the link ends in order to take advantage of the MonkeyBot's swinging properties. In the swinging state, the robot is assumed to rotate around the fixed end at  $(x, y)$ . Instead of applying constraints to the equations of 3.1, the system can be reduced to only two generalized coordinates,  $q = [\theta_1 \ \theta_2]^T$ . Solving this reduced system follows the procedure above (though the algebra is much simpler). The resultant matrices for the standardized form \*(1) in this swinging state are:

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\begin{aligned}
M(q) &= \\
&\begin{pmatrix} I_1 + I_2 + m_1 r_1^2 + m_2 l_1^2 + m_2 r_2^2 + 2m_2 r_2 l_1 c_2 & m_2 r_2^2 + m_2 r_2 l_1 c_2 + I_2 \\ m_2 r_2^2 + m_2 r_2 l_1 c_2 + I_2 & I_2 + m_2 r_2^2 \end{pmatrix} \\
C(q, \dot{q}) &= \\
&\begin{pmatrix} -m_2 r_2 l_1 s_2 \dot{\theta}_2 & -m_2 r_2 l_1 s_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ m_2 r_2 l_1 s_2 \dot{\theta}_1 & 0 \end{pmatrix} \\
G(q) &= \\
&\begin{pmatrix} g(m_1 r_1 + m_2 l_1) s_1 + g m_2 r_2 s_{12} \\ g m_2 r_2 s_{12} \end{pmatrix}
\end{aligned}$$

This swinging state is applicable to any brachiating type motion. The robot might swing with only one magnet attached to the wall - a situation that is described by these reduced equations. At the end of the swing, the robot could release the magnet and enter a free-flight phase, which would instead be described by the full 4 degree of freedom system in section 3.1. Smooth transitions between these states is critical to an accurate simulation.

### 3.3 Impact Dynamics

To move any significant distance across the wall, the MonkeyBot needs to transition its holding points by changing which magnet is fixed. The transitions involve impact and corresponding loss of energy as the magnet attaches to the wall surface. It is assumed the friction and holding force of the magnets is high, so we model these collisions as purely plastic.

If both magnets are simultaneously active, then the solution to this impact is trivial and uninteresting – all motion and velocities are halted. As such, we will only consider impacts in the case of transition from free-flight to swinging dynamics (or, equivalently, an instantaneous change from one magnet to the other).

Using the end of the top link as the attachment point, we approximate the impact as an (unknown) finite impulse applied at the  $(x,y)$  position. The impulse is constrained so that the fixed end velocities in the post-impact state are zero (plastic collision).<sup>ii</sup>

$$J(q) \dot{q}^+ = 0 \quad (*)$$

$J(q)$  is a Jacobian matrix that transforms the generalized velocities  $(dq)$  into a secondary coordinate system consisting only of the velocities we want to become zero:  $[dx \ dy]^T$ .  $dx$  and  $dy$  are identical in both systems, so  $J(q)$  is simply:

$$J(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Impulse applied to a system causes a change in momentum (mass\*velocity). The change in momentum is reflected by a change in velocity,  $\Delta dq = (dq^+ - dq^-)$ , where  $dq^-$  and  $dq^+$  are the pre- and post- impact velocities respectively.

$$M(q)\dot{\Delta q} = J(q)^T \lambda \quad (**)$$

The parameter  $\lambda$  in (\*\*) represents the (unknown) impulse applied to the fixed end during.

Combining equations (\*) and (\*\*), and solving to eliminate the parameter  $\lambda$ , gives us the solution to our impact problem. The post-impact velocities are in terms of the pre-impact state and a projection matrix  $P(q)$ :

$$\begin{aligned} \dot{q}^+ &= P(q)\dot{q}^- \\ P(q) &= I - M^{-1}J^T(JM^{-1}J^T)^{-1}J \end{aligned} \quad *(2)$$

The above equation applies when the end at (x,y) becomes fixed. In the alternate case where we desire to activate the other magnet (at the end of the second/bottom link), we can simply transform the coordinates so that (x,y) are located about the desired point. Applying geometry to  $q$  and  $dq$ , the bottom and top links are effectively interchanged. These transformations are:

$$\begin{aligned} q_{flipped} &= \begin{bmatrix} x \\ y \\ \theta_1 \\ \theta_2 \end{bmatrix}_{flipped} = \begin{bmatrix} x + L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \\ y - L_1 \cos(\theta_1) - L_2 \cos(\theta_1 + \theta_2) \\ \pi + \theta_1 + \theta_2 \\ -\theta_2 \end{bmatrix} \\ \dot{q}_{flipped} &= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}_{flipped} = \begin{bmatrix} \dot{x} + L_1 \cos(\theta_1)\dot{\theta}_1 + L_2 \cos(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{y} + L_1 \sin(\theta_1)\dot{\theta}_1 + L_2 \sin(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{\theta}_1 + \dot{\theta}_2 \\ -\dot{\theta}_2 \end{bmatrix} \end{aligned}$$

We must also ensure to swap the parameters for the links, to ensure the “flipped” system is consistent with the old one. The impact equations \*(2) can then be correctly applied to  $dq_{flipped}$ .

## 4. Simulation

### 4.1 Code High-Level Overview

The primary objective of the code was to provide a simulation for MonkeyBot. The code, written in Matlab, handles equations of motions for all possible motion types: free-flight, swinging, and fixed. It also smoothly transitions between motion types, modeling impacts as necessary. Parameters for the MonkeyBot are user modifiable.

The simulation function, *monkeySim* is called with the initial conditions and system parameters. It will return results of the simulation at each time step (.001seconds). Numerical integration is done with Matlab's *ode45*, using the standardized equation  $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$  and the matrices applicable to the motion state / magnet configuration.

Solving the equations of motion for each time step is handled by the function *DiffEqWrapper*. The control algorithm is (mostly) modular and contained with *controlAlgorithm.m*. It can be easily replaced or modified.

The attached .zip file contains a folder with the simulation code, and a sample script file *main.m* to demonstrate its use.

The overall algorithm is as follows:

1. Setup script: *main.m* – sets up the initial conditions, system parameters, and controller parameters
2. Simulation function: *monkeySim* is then called with all necessary parameters
3. Values for generating the matrices  $M$ ,  $C$ , and  $G$  are pre-calculated to improve speed
4. *ode45* is called to numerically integrate the equations of motion
5. For each time step in *ode45*, the equations of motion are solved by *DiffEqWrapper*
  - a. Reduced matrices (2 DOF) used when swinging with only one magnet attached
  - b. Full matrices (4 DOF) used when neither magnet is attached.
6. For each time step specified (.001seconds), a monitoring function, *DiffEqMonitor* is called by *ode45*
  - a. Plots the location and state of the MonkeyBot, so that user can watch simulation progress
  - b. Control algorithm is called to check current state
    - i. Control algorithm receives current system state  $Q$
    - ii. Can adjust motor torque or change magnet states
7. Integration continues, but is halted when magnet states are transitioned
  - a. System state is flipped (bottom link becomes top, end point becomes new base x/y)
  - b. Projection matrix calculates post impact velocities from free-flight dynamics.
8. Repeat integration with new initial conditions (from post-impact velocities)



## 4.2 Adjusting Control Algorithm and System Parameters

System parameters available for modification include  $m$ ,  $I$ ,  $L$ , and  $r$  for each link, as well as the gravity constant  $g$ . These parameters are passed in vector form (e.g.  $M=[m_1 \ m_2]$ ) to the simulation function. Demonstration of these parameters is shown in *main.m*.

The control algorithm setup was designed to be isolated from the rest of the simulation. The function *controlAlgorithm* is called for each time step, and is passed the current time and system state. Any changes made to the global control variables (*motorState*, *magnetState*) will be reflected as the simulation progresses.

The current code package includes a simple time-based (open-loop) controller. Modification of *initial\_t*, *motor\_t*, *swap\_t*, and *maxMotorPower* will give demonstration of varying simulation results. See *main.m* and *controlAlgorithm.m*.

Additional parameters can be fine tuned in various parts of the program:

- Step size (currently .001) can be changed in *monkeySim.m* to change the controller resolution or number of reported values. Note: This does not affect the accuracy of the integration as *ode45* chooses its own step size then interpolates to return values at the desired points.
- Friction (currently 0). *DiffEqWrapper* includes parameters to apply both constant and viscous friction to the joints at  $\theta_1$  and  $\theta_2$ .
- Frequency of graphing can be changed in *DiffEqMonitor*. This has a large impact on simulation time.

---

<sup>i</sup>  $M(q)$ ,  $C(q,dq)$ ,  $G(q)$  matrices for both dynamic systems (free-flight/4 DOF and swinging/2 DOF) obtained from Nelson Rosa.

<sup>ii</sup> Equations for impact dynamics obtained from Kevin Lynch.