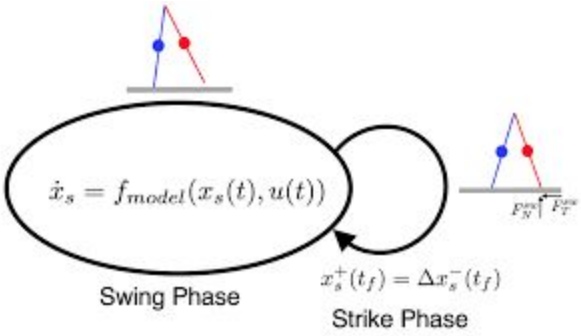
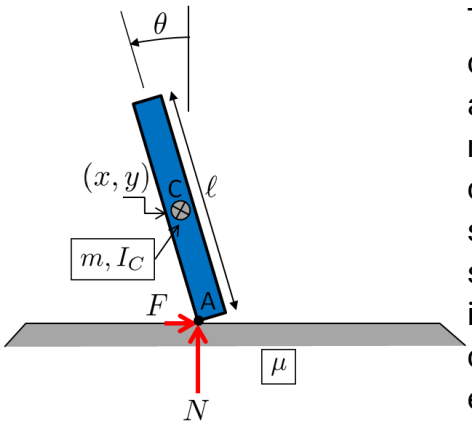
Understanding Hybrid Dynamical Systems:

What are hybrid systems?



These are dynamical systems that are a combination of continuous modes and discrete modes which are transition modes between the two continuous modes. Consider a two-link biped robot called "Acrobot". The control system of this model has two components: The single-phase component and the impact map. When the biped is supported by one of its legs and the other leg is moving, it is said to be in the swing/single-support phase. This is always a continuous phase described by differential equations. After the swing phase comes the impact phase. When the moving leg makes contact with the ground, it does so in instantaneous time (discrete transition) and then moves on to the second continuous phase. During the discrete phase the geometry of the system remains but the velocities of some of the components change creating a jump in the state phase, hence the term discrete phase.

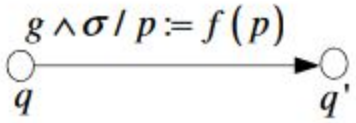


The time instant at which the transition takes place is called Event Time. Consider a toppling stick as another example. As the stick topples (assuming no-slip condition), the point of contact can be considered as a pivot and the mechanics can be solved using the lagrangian of the model. In the second phase, the stick touches the ground in instantaneous time which is modelled as elastic collision(discrete transition) and then moves on to either flight mode or toppling mode. Mainly there are 2 ways to simulate any hybrid system:

Event Based Simulation

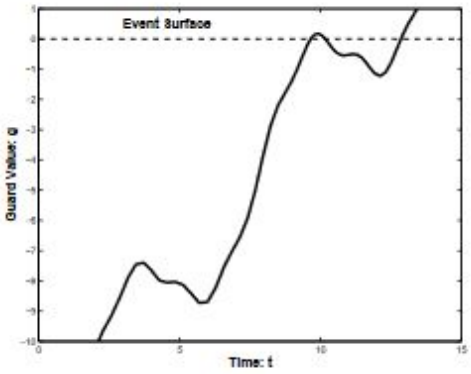
This method uses full finite state machine structure for the hybrid control. In each continuous state a set of differential equations governs the continuous dynamics of the system. The algorithm is basically a variable-step integrator that uses root finding to determine the precise instant at which the transition guards switch.

--Thanki Dhruv--



q, q': states
σ: event
g: guard
p: parameter

The transition shown is to be interpreted as follows: If at state q, the guard g is true and the event σ occurs, then the next state is q' and the values of variables will be updated to f(p).



Impulse Based Simulation (Time Stepping Simulation)

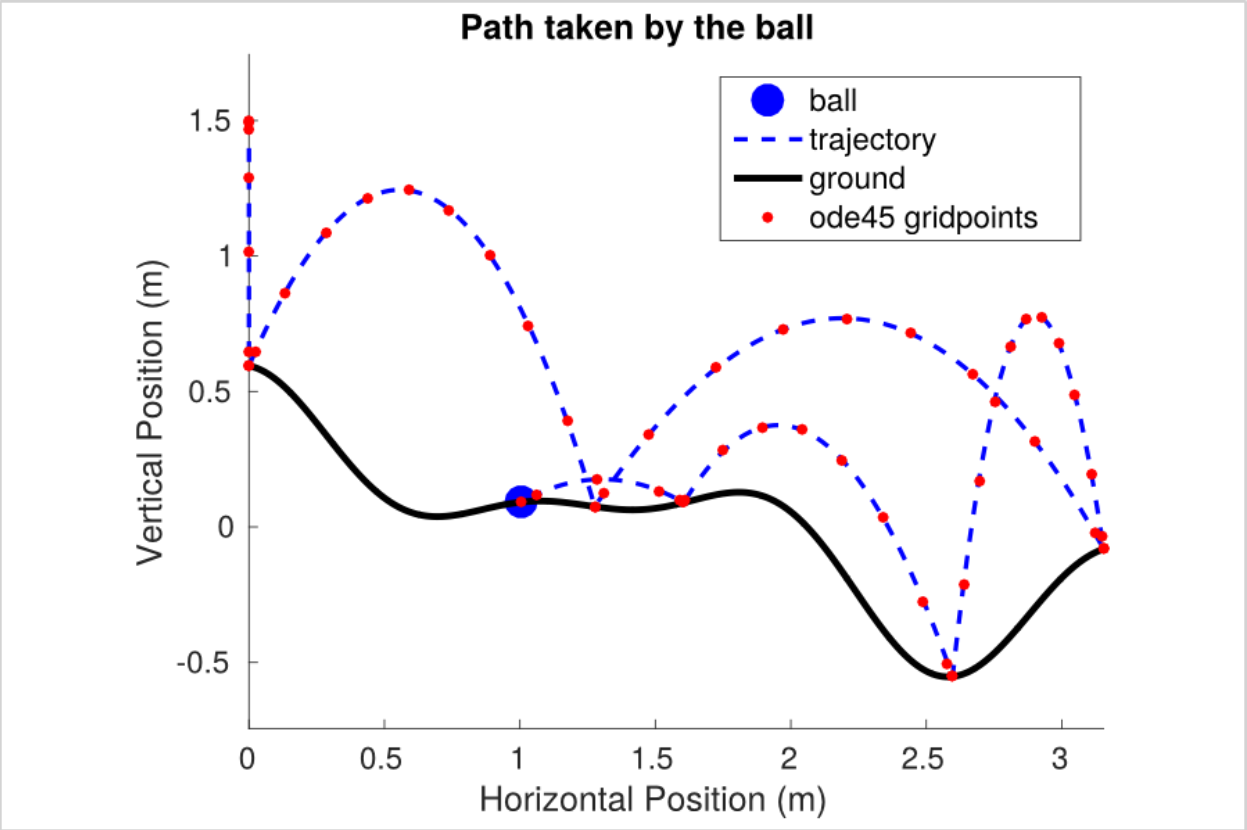
Impulse-based simulation was developed to deal with systems with arbitrarily complex contact mechanics. The main idea of this sort of simulation is that each rigid body is handled independently. At a given time step, all forces are considered as impulses when the dynamics of the system is formulated. At every time step, all the impulsive forces are solved as a part of a large optimisation problem. This optimisation problem finds a unique solution based on the following constraints for every pair where an impulse is detected:

$d_n > 0$ Contact separation
$J_t \leq \mu J_n$ Contact force in friction cone
$d_n J_t = 0$ Contact force when touching

These conditions are called linear complementarity problems (LCP).

--Thanki Dhruv--

Example: Bouncing Ball on Uneven Surface



The whole simulation code of bouncing ball can be broken down into 5 parts:

- System Dynamics.
- Geometrical representation of the terrain.
- Event function.
- Impact Map.
- Main Function.

Main Function:

There are 2 conditions that can be used to terminate the simulation:

- Either if the maximum number of bounces has happened.
- Or if the maximum time is reached.

Otherwise ODE45 continuously solves the dynamics iteratively in an infinite loop.

GroundHeight:

Contains a terrain which could be encoded in a series of sine waves. Any desired surface can be generated using a set of fourier series of sine waves.

--Thanki Dhruv--

(Ref:https://www.youtube.com/watch?v=r6sGWTCMz2k). It is also important to know the slope along the Terrain (I will just call it a curve from now on).

Event Function:

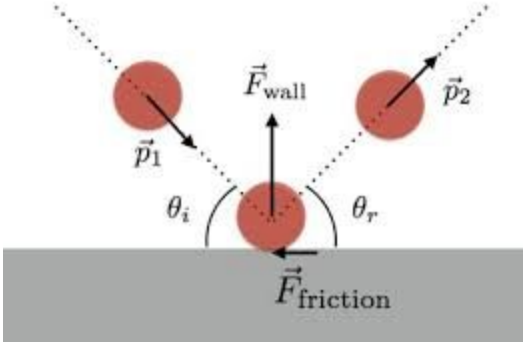
This function is given as options in the ode45 solver. It is used to check if a collision has occurred and whether the solver should be stopped or not. It checks if the two conditions given below are fulfilled or not:

- Also returns the difference in height between the falling ball and the ground.
- isterminal = true; (if the option function in ode45 solver returns isterminal as true: stop the solver.
- direction = -1; (Should only be coming to the ground from above).

Impact Map:

This function takes in the states before the collision, and returns states after a collision. How it does that is explained in the steps below:

- Find the slope of the curve at the point of collision.
- Change references from global to reference tangential and normal to the curve at that point using a rotation matrix.
- Only velocity components normal to the curve at that point are affected.
- Change back to the global reference.
- Return output velocity.



Additionally, another constraint can be added too: if the output velocity is lower than a certain threshold, the ball can be made to roll. This is done to avert the phenomena called Zeno behavior in which the ball bounces infinite times in a finite time length.

System Dynamics:

These are the set of equations that describe the evolution of the system under the effects of any external forces.

$\frac{d}{dt}(x) = v$

--Thanki Dhruv--

$\frac{d}{dt}(v) = -g$

For a simple system like the bouncing ball or the acrobot biped which is essentially a double pendulum with a few constraints keeping it away from chaos, they can be written by hand. If the system becomes complex like the 5-Link biped or potentially any other bipeds like the Cassie or MARLO, they are usually generated using a computation tool like MATLAB or Mathematica.

Lagrangian Mechanics:

To derive these equations we can use different tools like the Lagrangian Mechanics (generally used) or the Hamiltonian mechanics. First a Lagrangian of the system is calculated as below:

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

where the K stands for the total kinetic energy of the system, the V stands for the total potential energy of the system and the q stands for a set of variables of the system.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \Gamma,$$

To find the equations of motion of the system, we substitute it in the Lagrange's equation. Here Γ stands for the generalised torques and forces.

The resulting equation takes the following form:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \Gamma,$$

Isolating the $\frac{d^2}{dt^2}(q)$ from the above equations gives us the second part of the set of equations of the motion.

Hybrid System as a Directed Graph

$HS = (\Gamma, D, U, S, \Delta, FG)$

Where:

Γ (V,E) represents a directed graph i.e. (Vertices, Edges).

D represents a set of admissible domain, a subset of state space

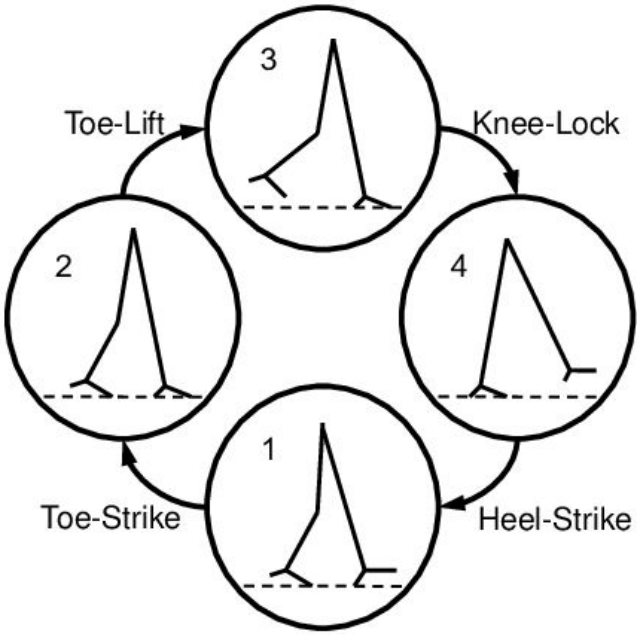
U represents a set of controllers

S represents a set of guards or switching surfaces

Δ represents a set of reset maps or impact map

FG represents the continuous and discrete dynamics

The directed graph, Γ can be pictorially represented as shown below:



References:

[1] Matthew P. Kelly. *Simulations with Sliding and Intermittent Contact*. July 29, 2014.

[2] Jessy Grizzle et. al. *Feedback Control of Dynamic Bipedal Robot Locomotion*.

[3] Ayonga Hereid and Aaron D. Ames. *FROST: Fast Robot Optimisation and Simulation Toolkit*.

--Thanki Dhruv--