Simulating Mobile Robots Using Simulink

Roy Featherstone

School of Engineering
Australian National University
Why Use Simulink?

- widely used simulator
- good user interface
- good documentation
- tightly coupled with Matlab
- general discrete/continuous event simulator
- variable-step integrators
- both interactive and programmable
- supports model libraries, sharing and hierarchical model decomposition
Why Not Use Simulink?

- expensive
- slow (compared with best special-purpose simulators)
- designed for systems with unidirectional signals
Programmability

From inside a Matlab function you can call

```matlab
sim( modelname, parameters );
```

to make simulink run a specified simulation with specified parameters. Results can be output to the Matlab workspace.

This is useful (e.g.) for automatic optimization of parameter values.
Variable-Step Integrators

Rigid-body dynamics tends to produce spiky accelerations. A variable-step integrator can improve both efficiency and accuracy by adapting the step size.

\[ |\text{accn}| \]

\[ \text{time} \]

small steps here

big steps here
Discrete/Continuous Simulation

- **discrete events** are instantaneous – they take place at instants in time

- **continuous dynamics** is simulated (by numerical integration) in the time intervals between discrete events

- there are two types of discrete event:
  - **data driven**
    - gain/loss of contact
    - stick/slip transitions
    - hitting a saturation limit
  - **scheduled**
    - scheduled finish time
    - regular execution of controlling software (e.g. servo cycles)
Discrete/Continuous Simulation

whenever a discrete event occurs, the current integration step is truncated (if necessary) so as to end at exactly the instant when the event occurs.

original integration step

truncated integration step

collision event here
Top-Level Model

Command → Controller

Controller → Robot

Robot → Ground

Ground → Controller

Doc

feedback
bus

actuator
command
signals

foot
pos/vel

ground
reaction
forces
Top-Level Model

Documentation describing what the model is, what it does, how it works, how to use it, etc.
Top-Level Model

- Command
- Controller
- Robot
- Ground

- content depends on what the controller can do
- feedback bus
- actuator command signals
- foot pos/vel
- ground reaction forces

Doc

The content depends on what the controller can do, with feedback through the bus and actuator command signals. The robot receives foot position/velocity and sends ground reaction forces.
Top-Level Model

every signal a controller might need

feedback bus

actuator command signals

ground reaction forces

and contact state data

allows multiple feet

foot pos/vel

Doc

Command

Controller

Robot

Ground
Controller

Command

Controller

feedback
bus

Robot

actuator
command
signals

Ground

foot
pos/vel

ground
reaction
forces
Controller

- Command
- Feedback signals
- Feedback
- Bus expands to individual signals
- Control algorithm (Matlab code)
- State vector
- New state vector
- 1/z
- Output to workspace
- Oscilloscope
- Signals to be viewed/saved
Controller

memoryless function block – executes at fixed sample rate

command

control algorithm (Matlab code)

actu cmd

feedback signals

state vector

new state vector

1/z

signals to be viewed/saved

remembers state vector for one sample time

oscilloscope

output to workspace

feedback

...
Robot

Command \rightarrow Controller

Controller \rightarrow Robot

Robot \rightarrow Ground

Ground \rightarrow Doc

Doc \rightarrow Command

feedback bus

actuator command signals

ground reaction forces

support bus

foot pos/vel
Robot

actuator dynamics

forward dynamics (Matlab code)

1/s

q

qd

foot kinematics

foot pos/vel

support

contact status and forces

all useful signals collected here

feedback
Ground

Command → Controller

Controller → Robot

Robot → Ground

actuator command signals

support bus

foot pos/vel

Doc
Ground

- foot pos/vel
  - position
  - tangent
  - normal
  - velocity
  - normal

friction model
- f_T
- f_N
- sticking
- in_contact

normal contact model
- support
- f_ext
Ground

- A foot is the union of one or more points or spheres.
- The model accepts data on multiple feet.

```
foot pos/vel
  position
    tangent
      normal
        tan
          velocity
            normal
              normal contact model
                  sticking
                    in_contact
                        support

friction model
  f_N
    f_T
      combine
          f_ext
```
Ground

- **foot pos/vel**
  - position
  - tangent
  - normal
  - velocity
  - normal

- **friction model**
  - f_T

- **normal contact model**
  - f_N
  - sticking
  - in_contact

- **combine**
  - f_ext

- **support**

- detect contact and calculates contact normal forces
Ground

detects sticking vs. sliding and calculates Coulomb friction forces

foot pos/vel

position

tangent

friction model

f_T

combine

f_ext

normal

tan

normal contact model

normal

velocity

sticking

support

in_ctact
Ground

- **Foot pos/vel**
- **Position**
  - **Tangent**
  - **Normal**
- **Friction model**
  - **f_T**
- **Combine**
  - **fExt**
- **Normal contact model**
  - **f_N**
  - **Sticking**
  - **In_contact**
- **Support**

Calculates ground reaction forces from normal and tangent components.
Results so Far

- students have implemented Raibert-style controllers on hoppers and runners in 2D and 3D
- new, improved contact normal model
- realistic models of lossy electric motors and drive trains
- 3D animation software for off-line analysis of simulation results
- S-functions (C/C++) for faster simulations
Future Work

- develop an *energy audit* facility to track all energy flows in a simulation
- use the software to support research in highly dynamic legged locomotion
- make the software available on the web