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

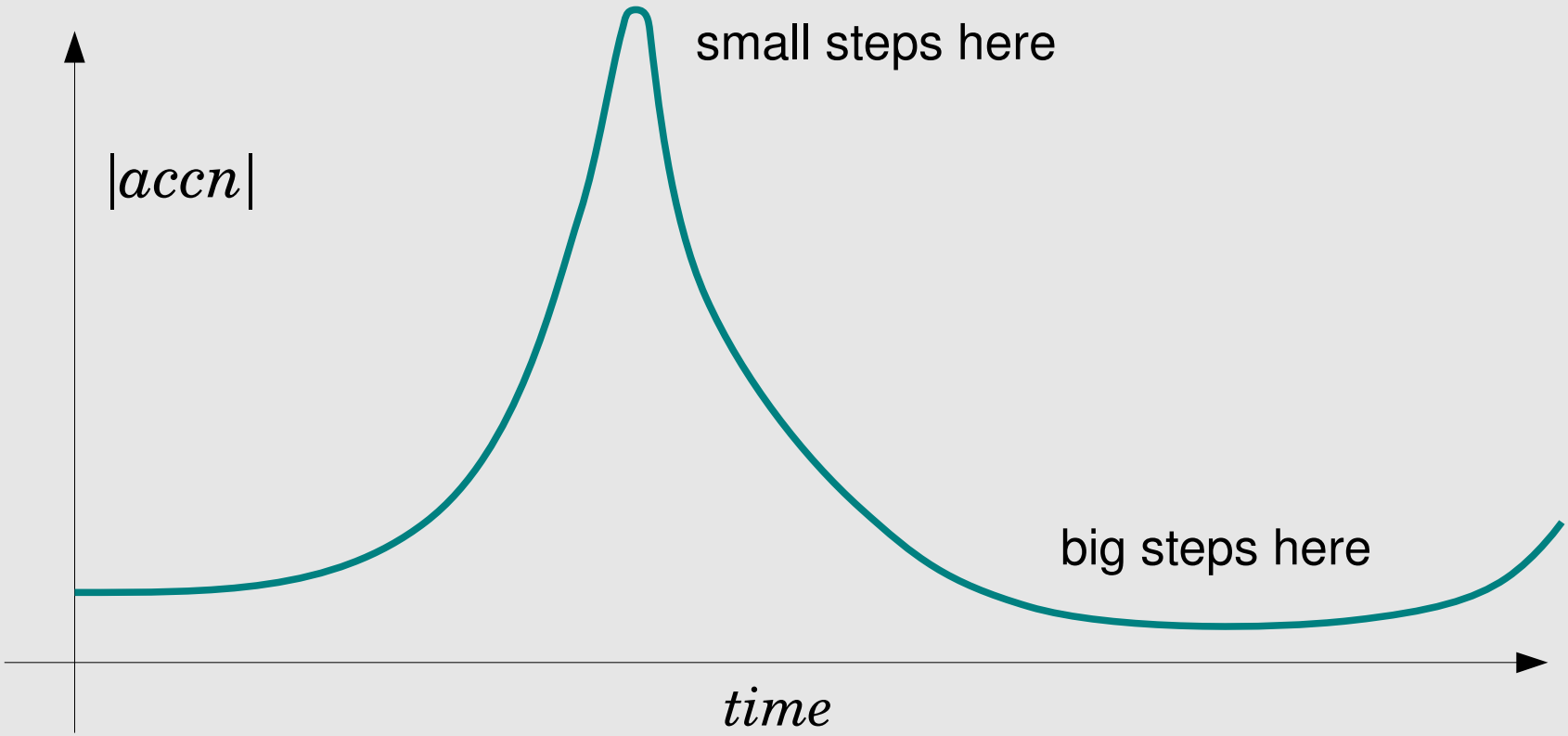
sim(*modelname*, *parameters*);

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.



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:

<u>data driven</u>

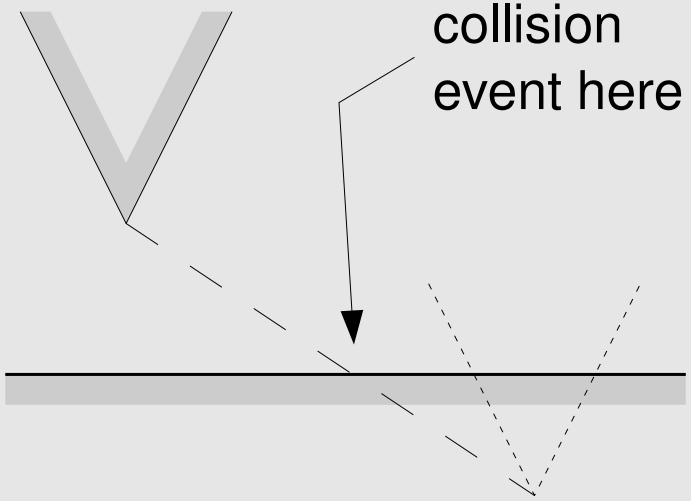
- gain/loss of contact
- stick/slip transitions
- hitting a saturation limit

<u>scheduled</u>

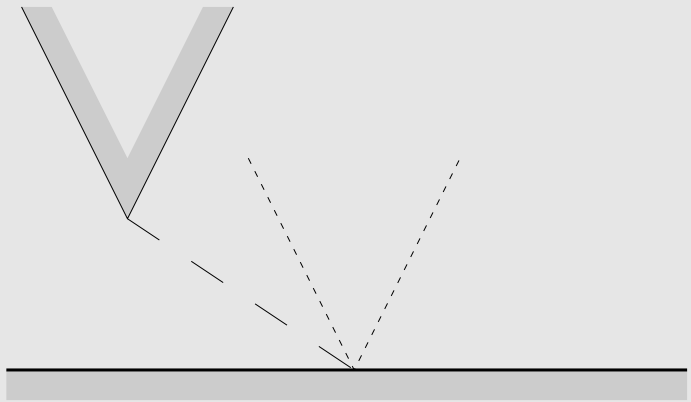
- scheduled finish time
- regular execution of controlling software (e.g. servo cycles)

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

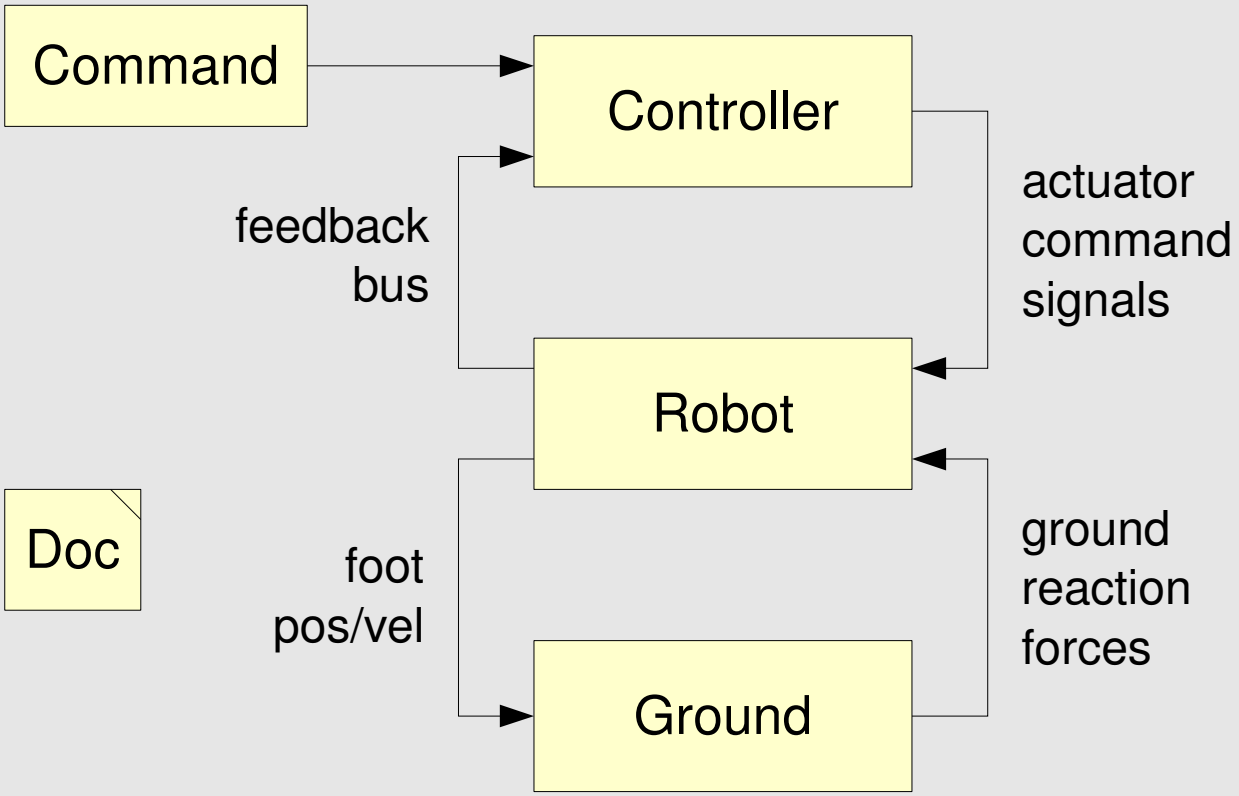


original integration step

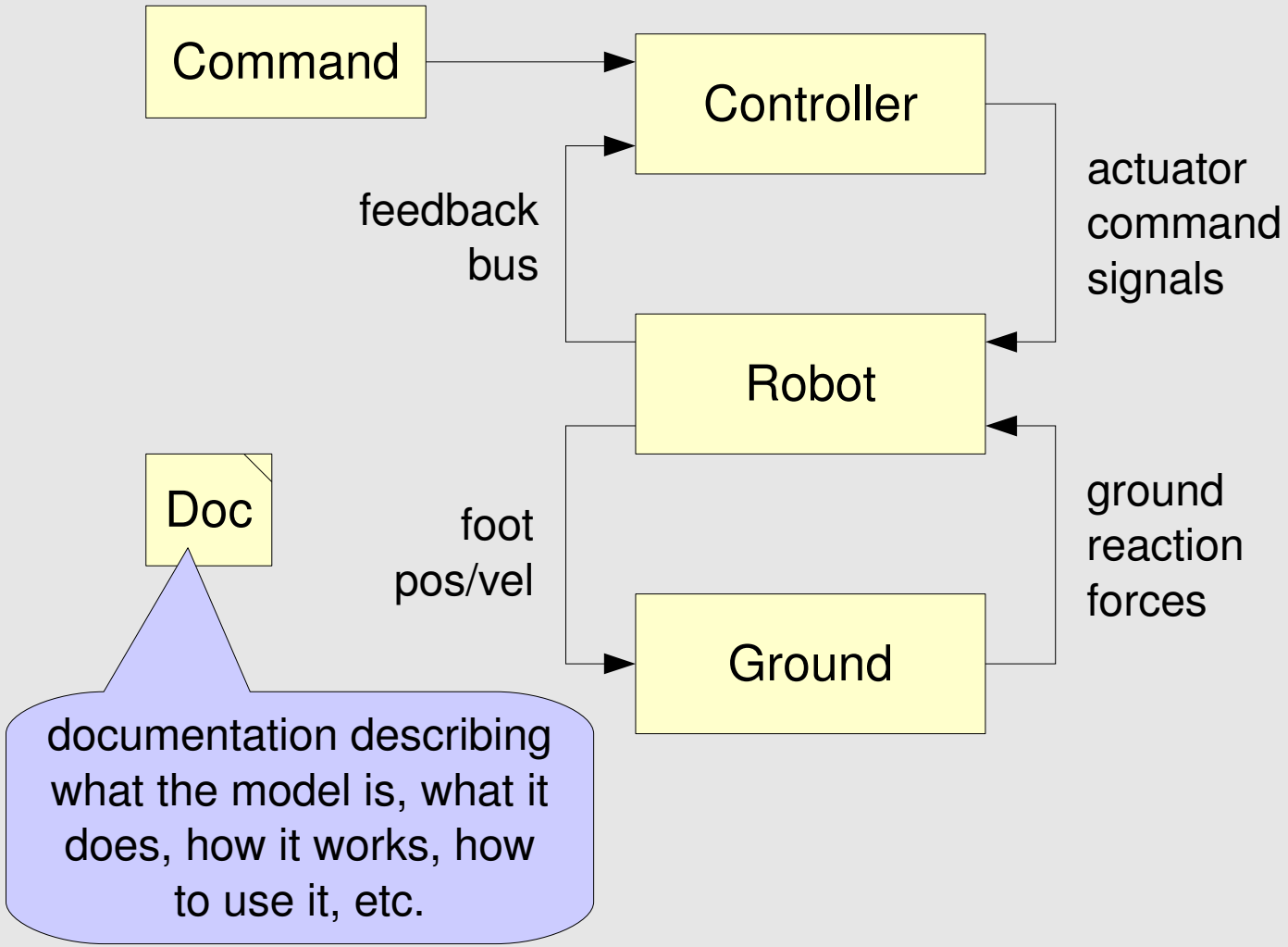


truncated integration step

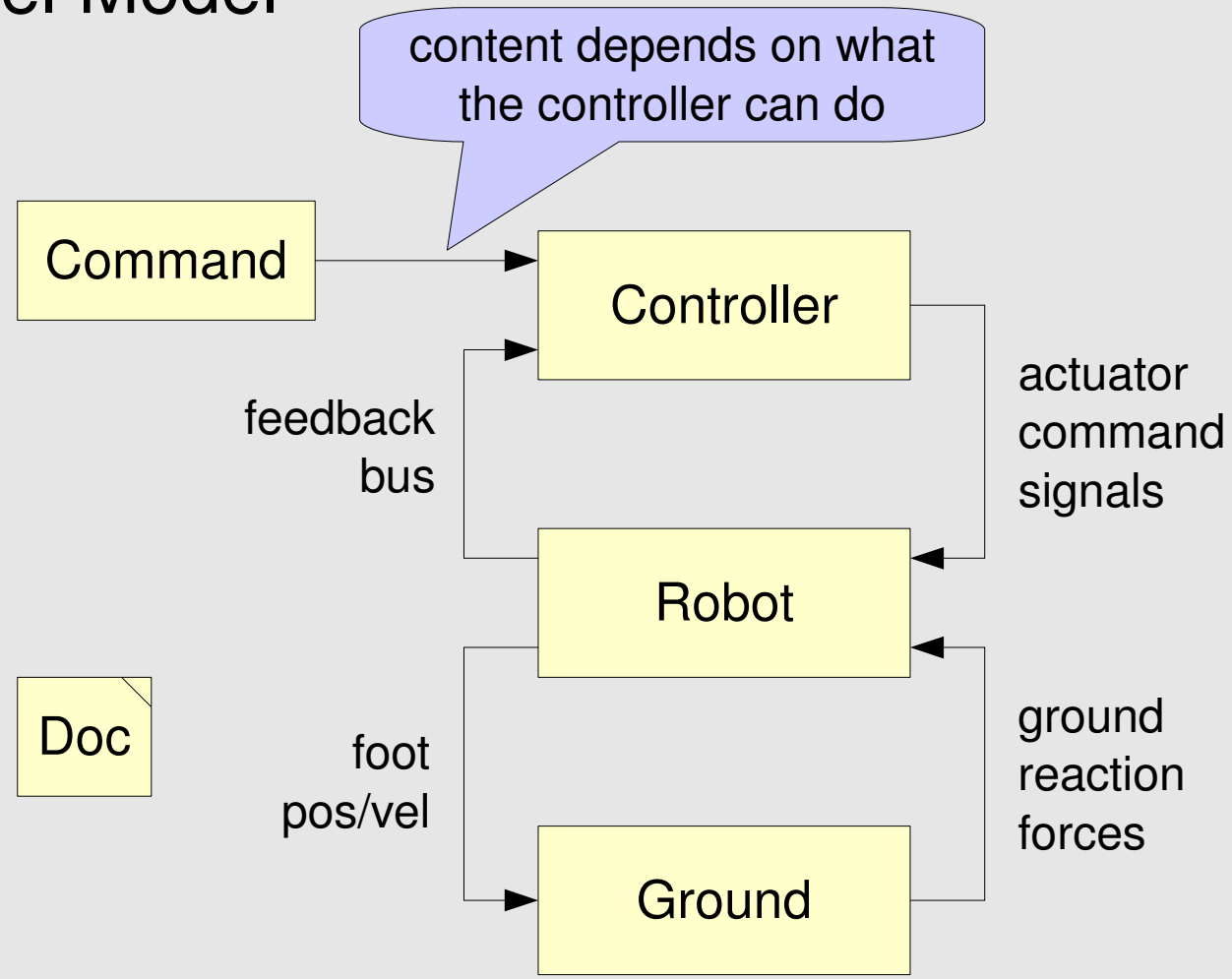
Top-Level Model



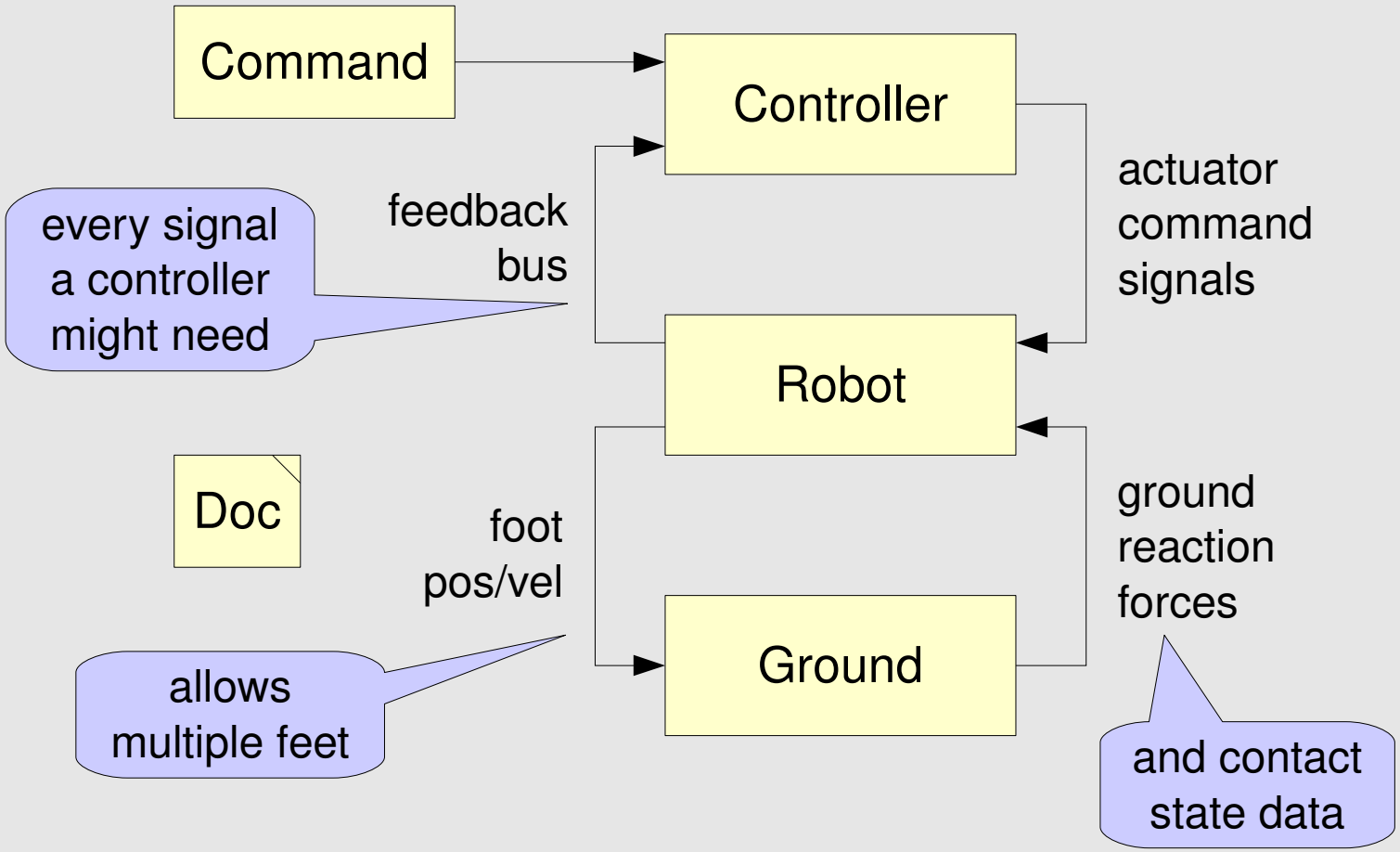
Top-Level Model



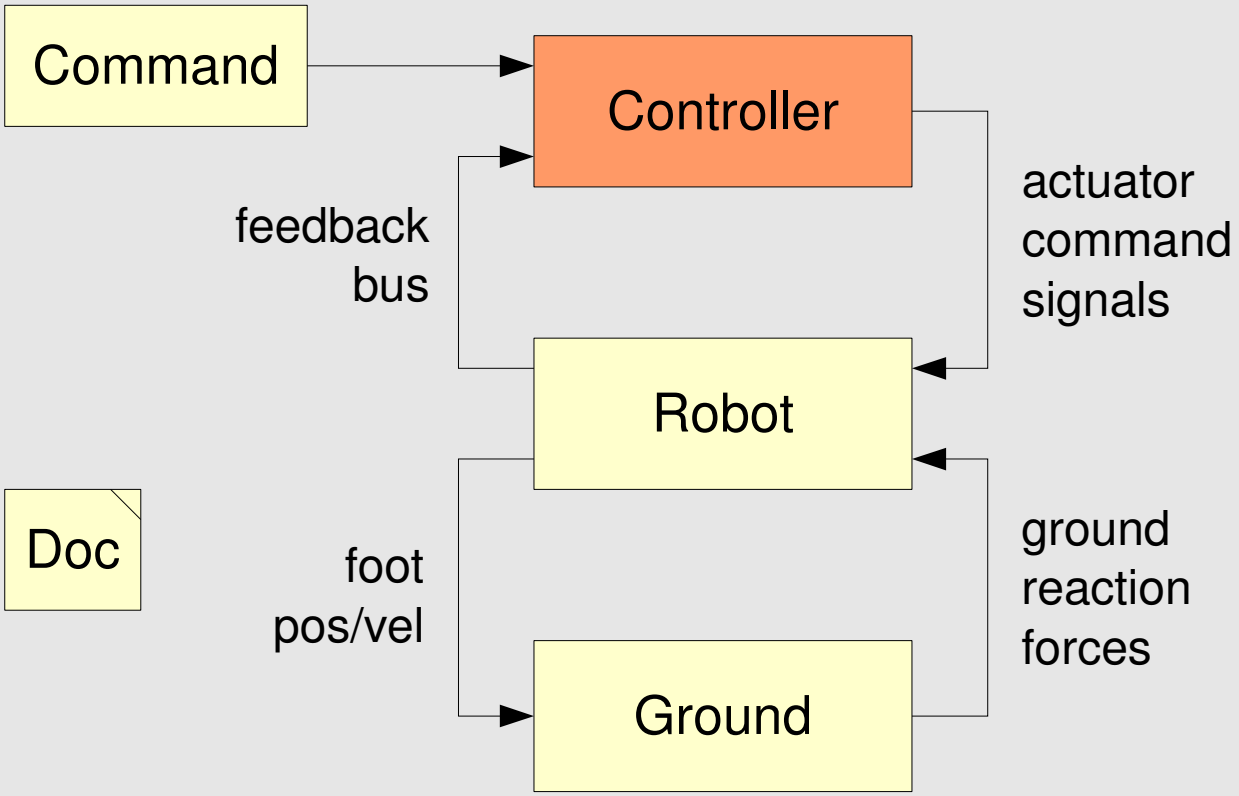
Top-Level Model



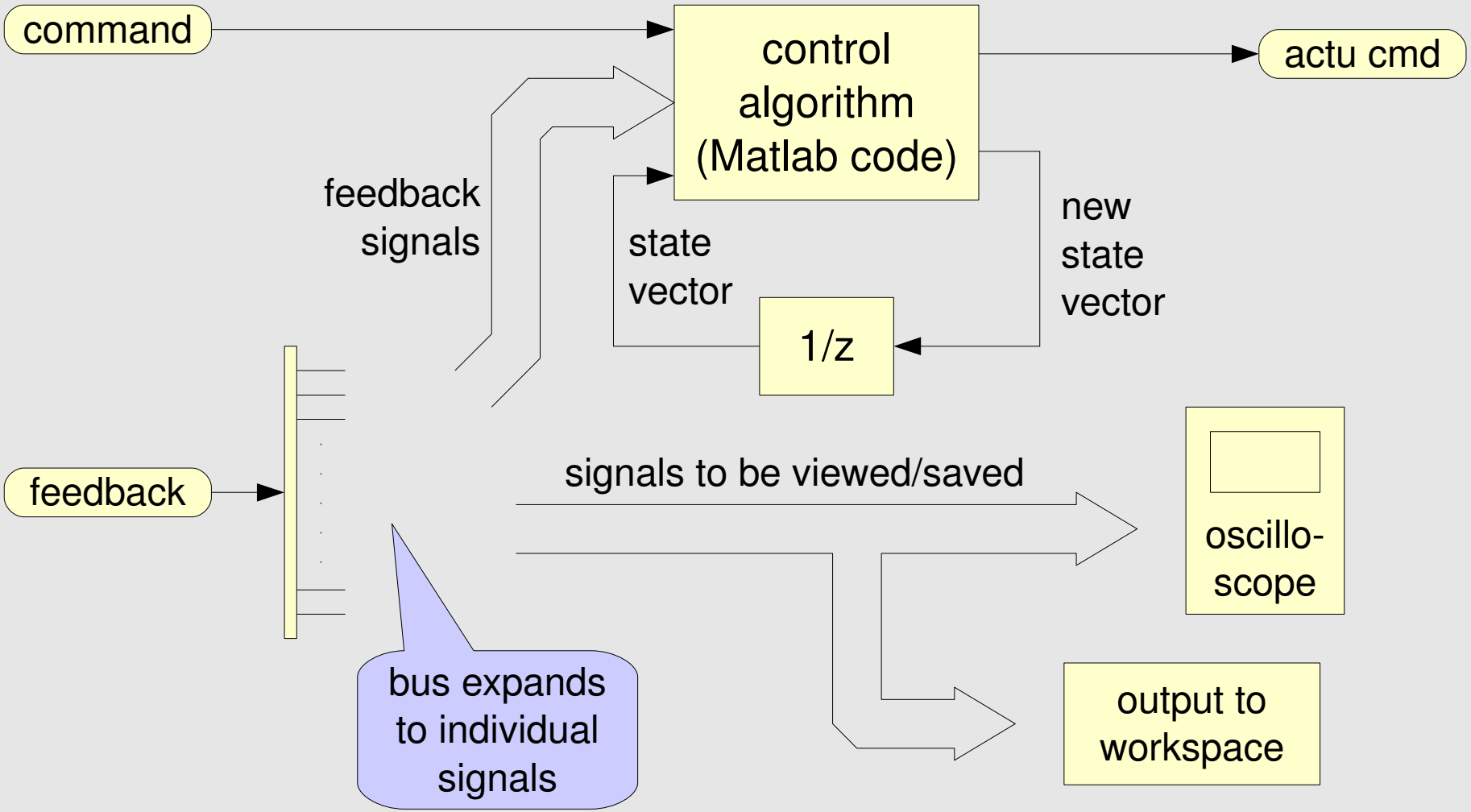
Top-Level Model



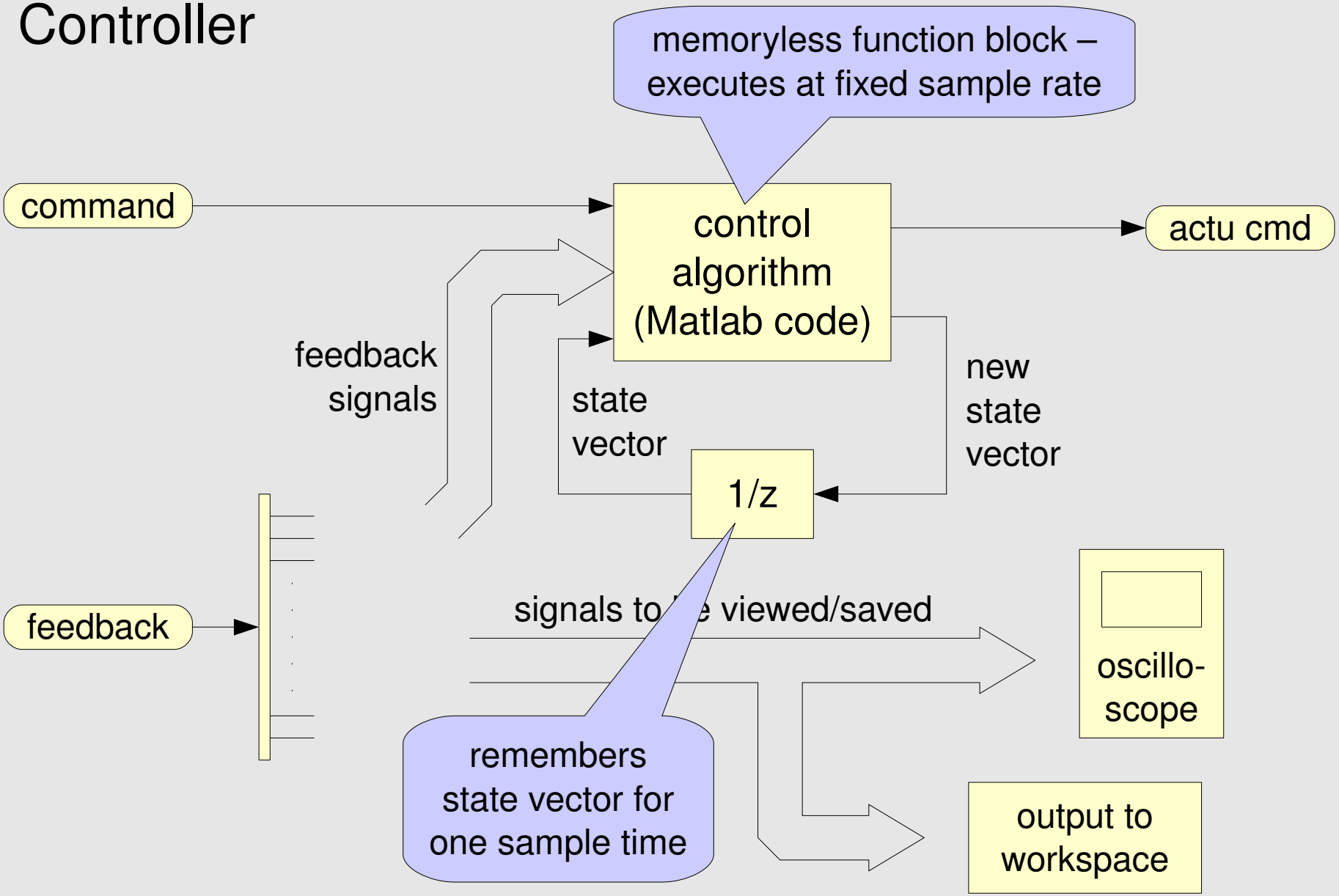
Controller


Command
Controller
feedback
bus
actuator
command
signals
Robot
Doc
foot
pos/vel
ground
reaction
forces
Ground

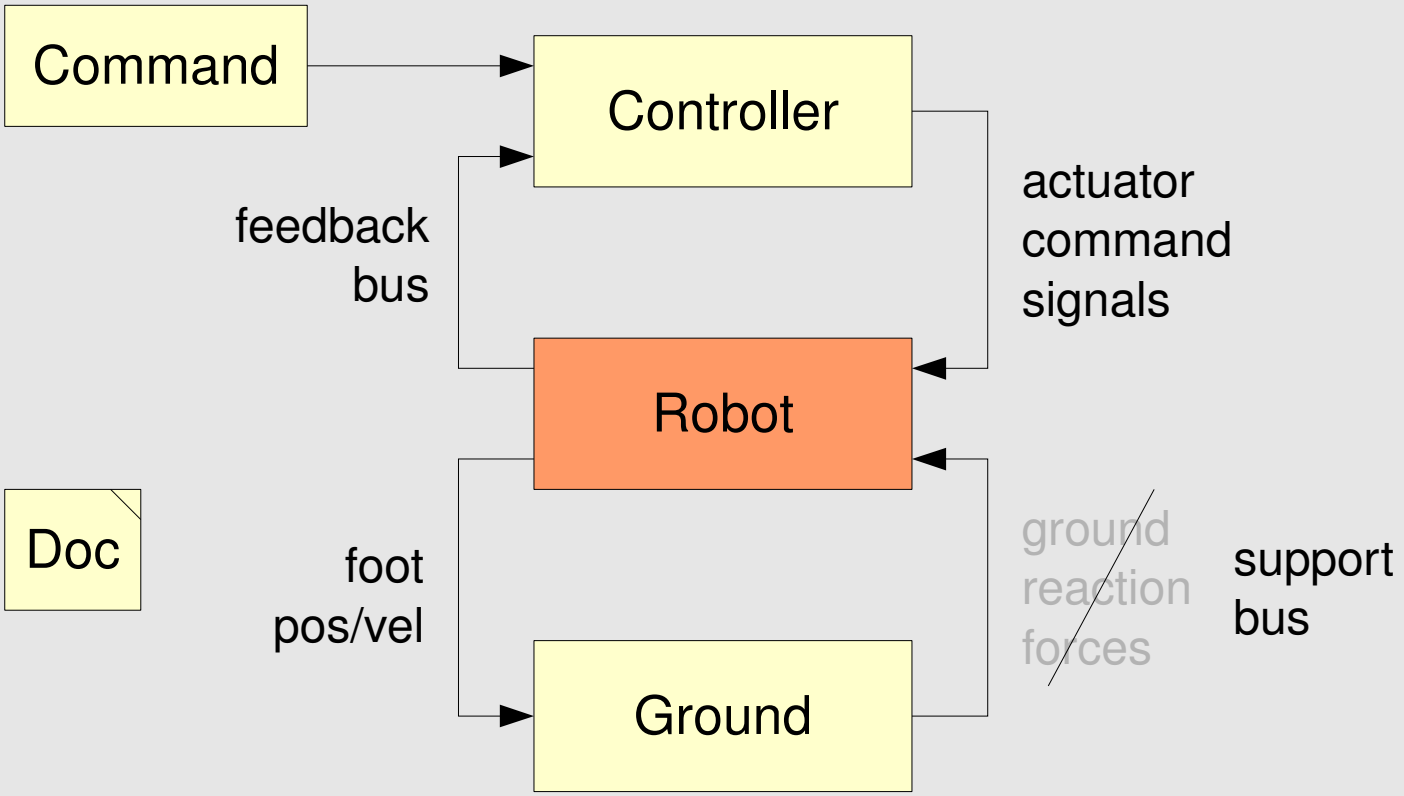
Controller


command
control
algorithm
(Matlab code)
actu cmd
feedback
signals
state
vector
new
state
vector
1/z
feedback
signals to be viewed/saved
oscillo-
scope
bus expands
to individual
signals
output to
workspace

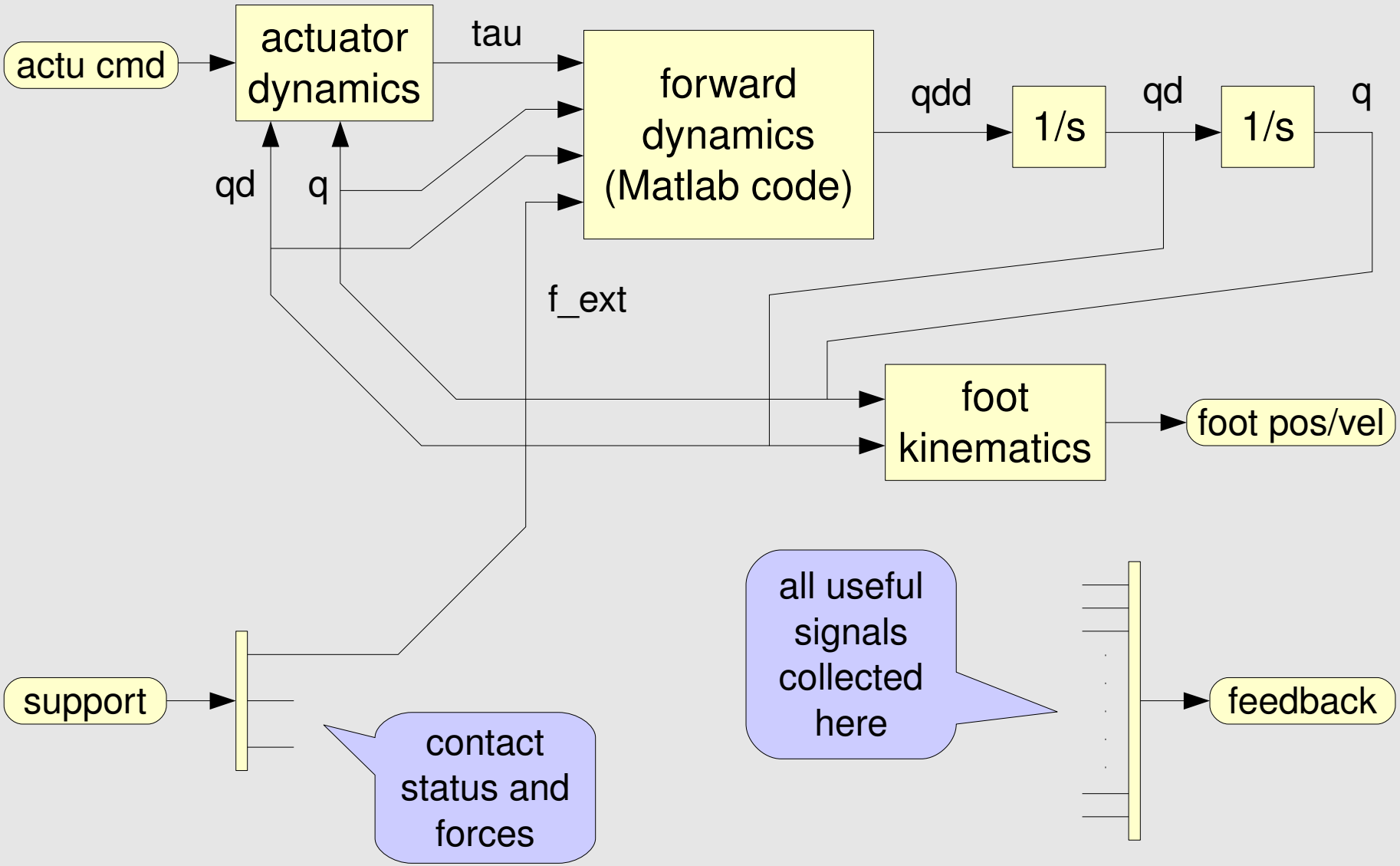
Controller


memoryless function block – executes at fixed sample rate
command
control algorithm (Matlab code)
actu cmd
feedback signals
state vector
new state vector
1/z
feedback
signals to be viewed/saved
oscillo-scope
remembers state vector for one sample time
output to workspace

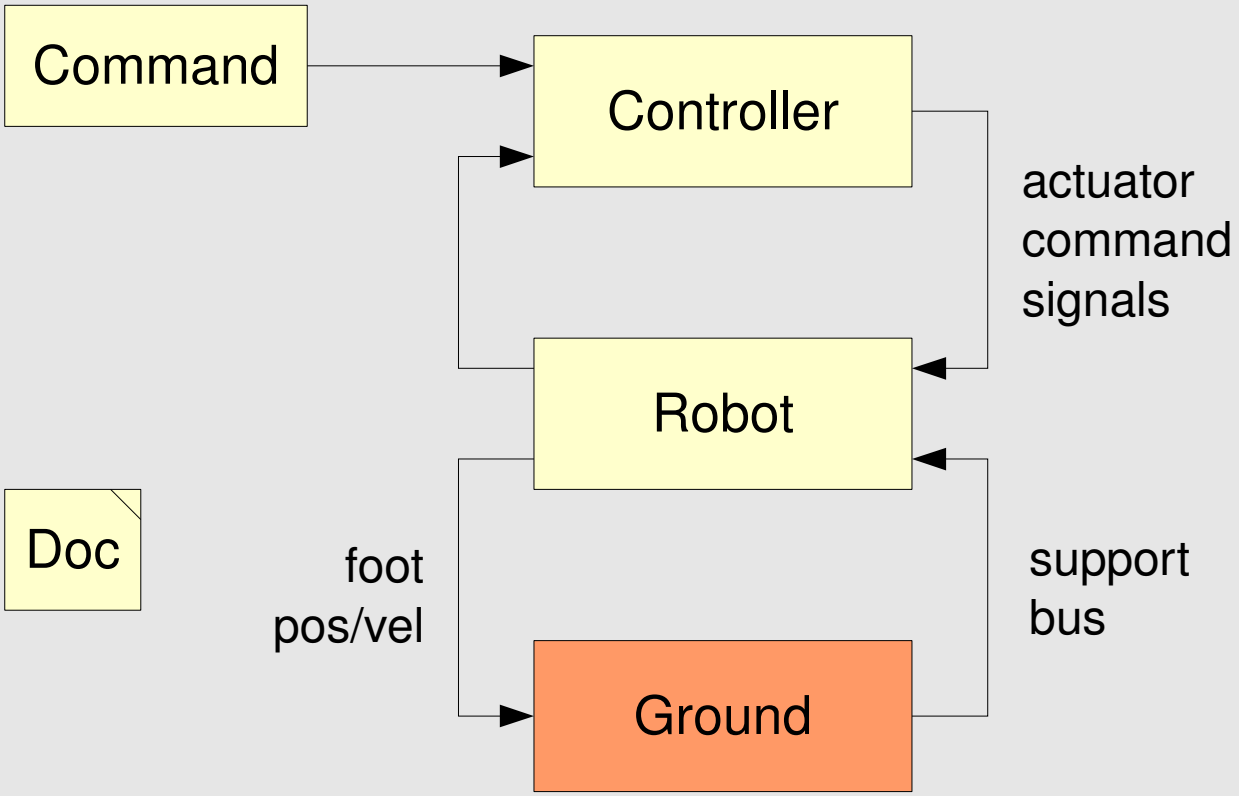
Robot


Command
Controller
feedback
bus
actuator
command
signals
Robot
Doc
foot
pos/vel
ground
reaction
forces
support
bus
Ground

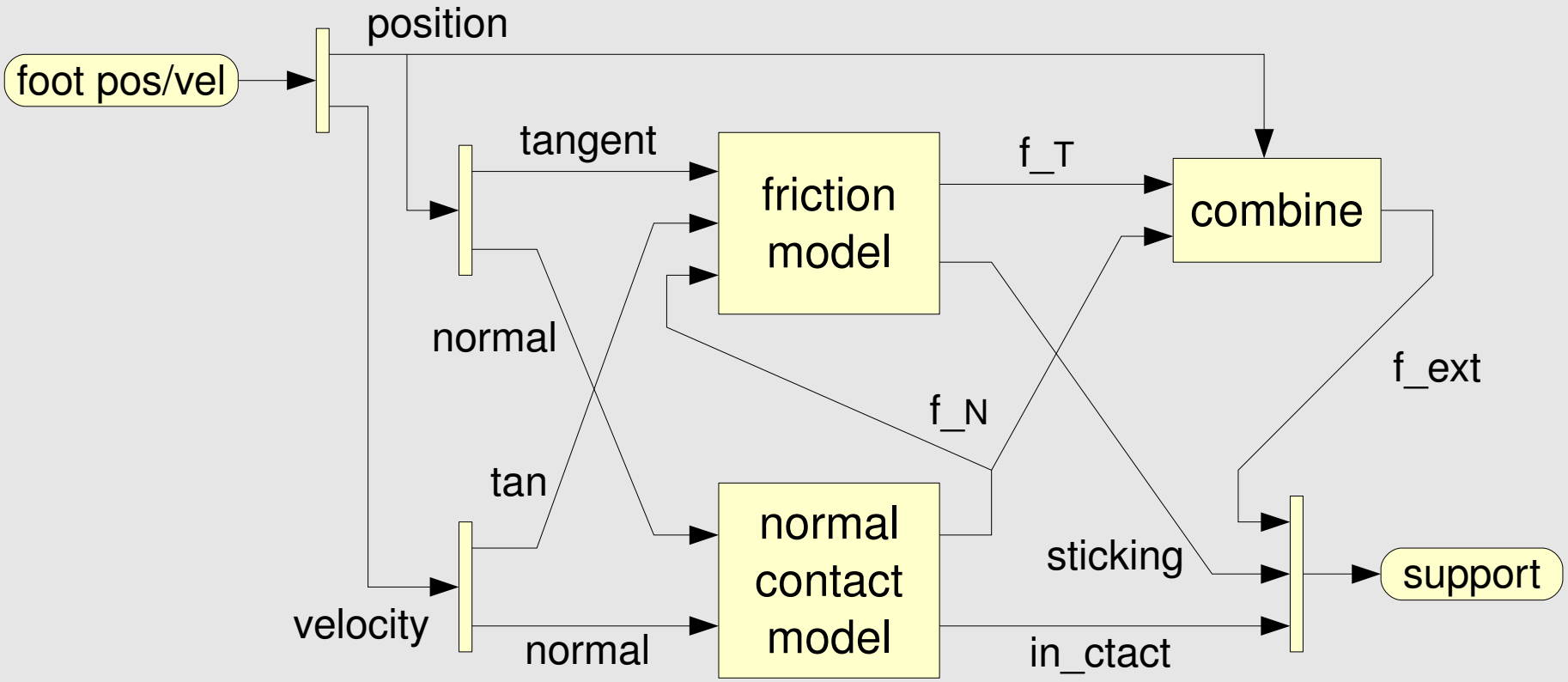
Robot


actu cmd
actuator dynamics
tau
forward dynamics (Matlab code)
qdd
1/s
qd
1/s
q
qd
q
f_ext
foot kinematics
foot pos/vel
support
contact status and forces
all useful signals collected here
feedback

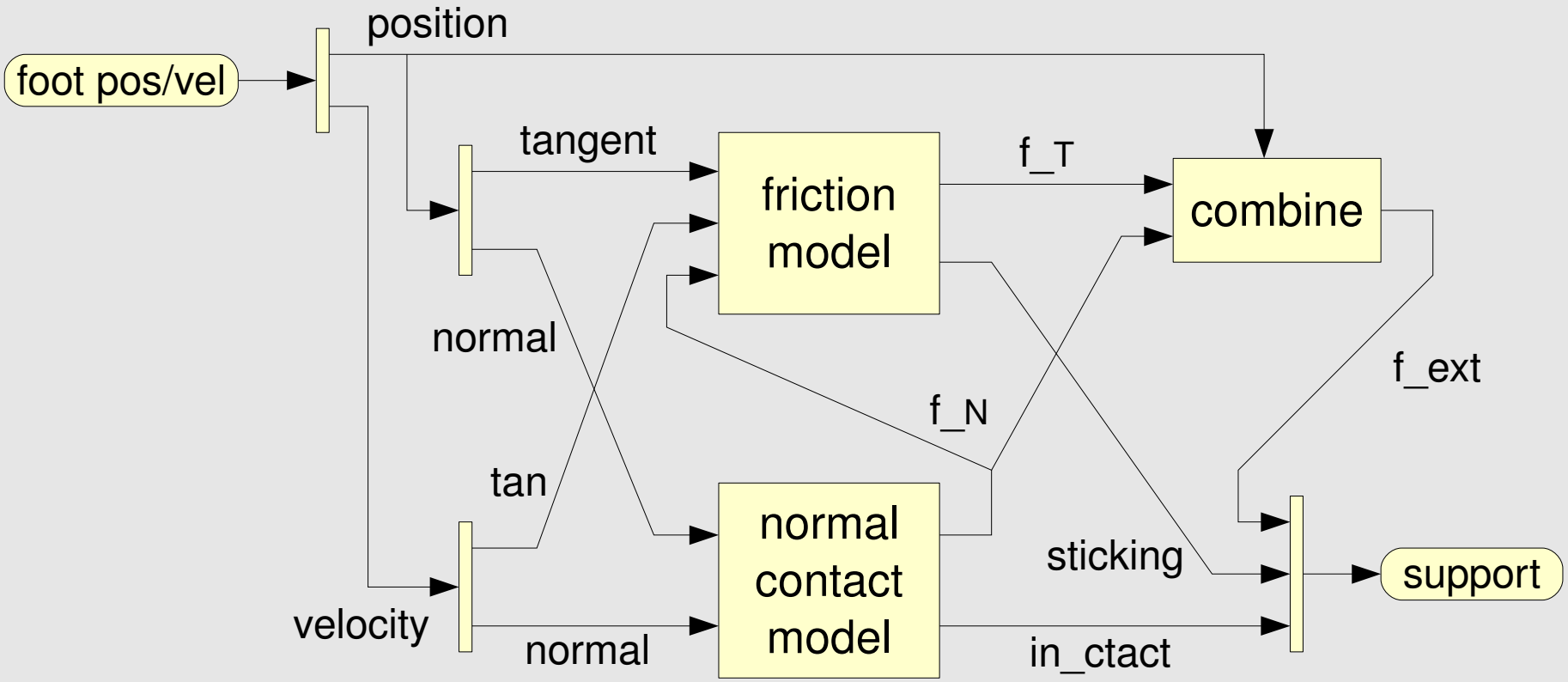
Ground


Command
Controller
actuator
command
signals
Robot
Doc
foot
pos/vel
support
bus
Ground

Ground

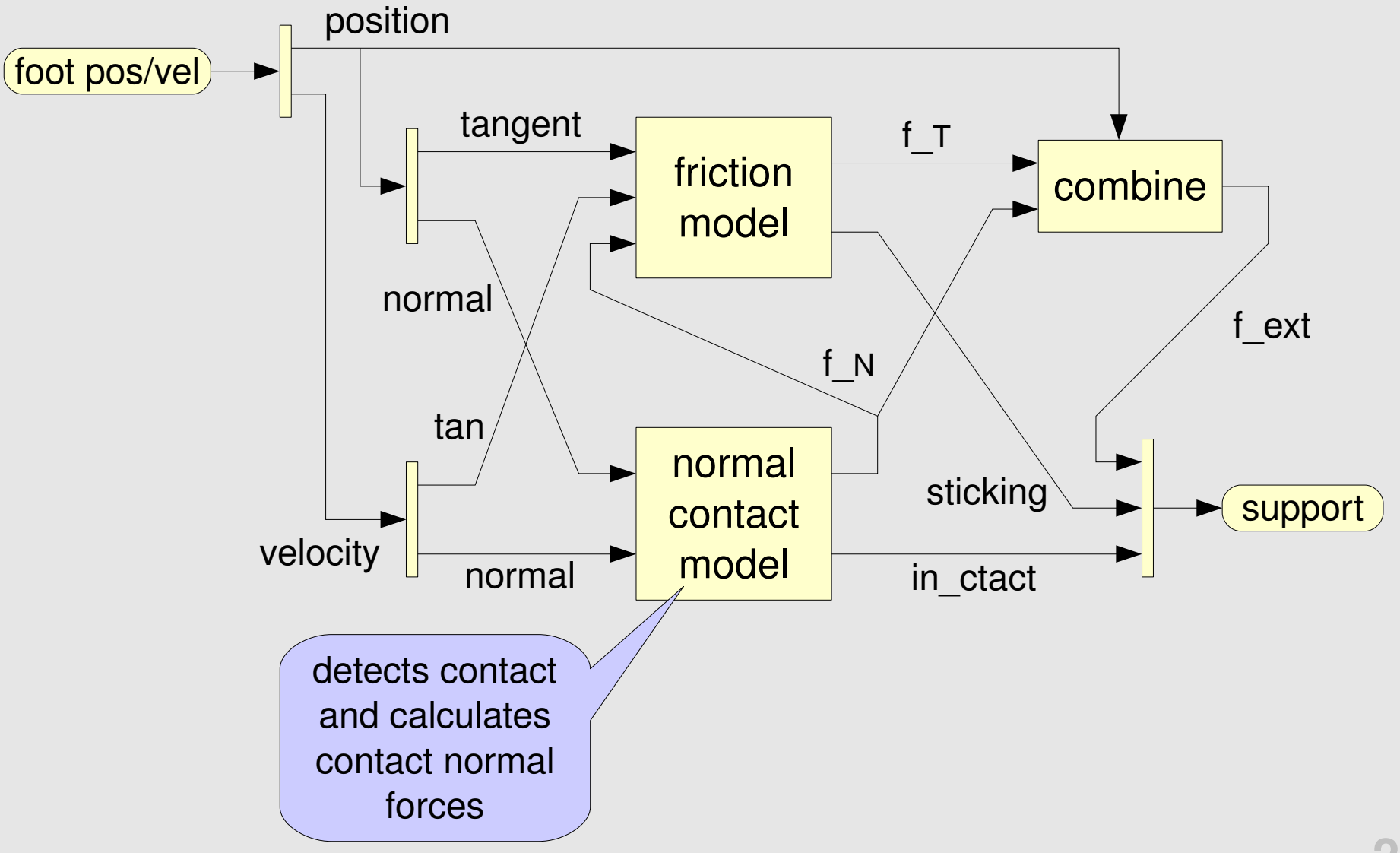

foot pos/vel
position
tangent
friction model
f_T
combine
normal
f_N
f_ext
tan
normal contact model
sticking
support
velocity
normal
in_ctact

Ground

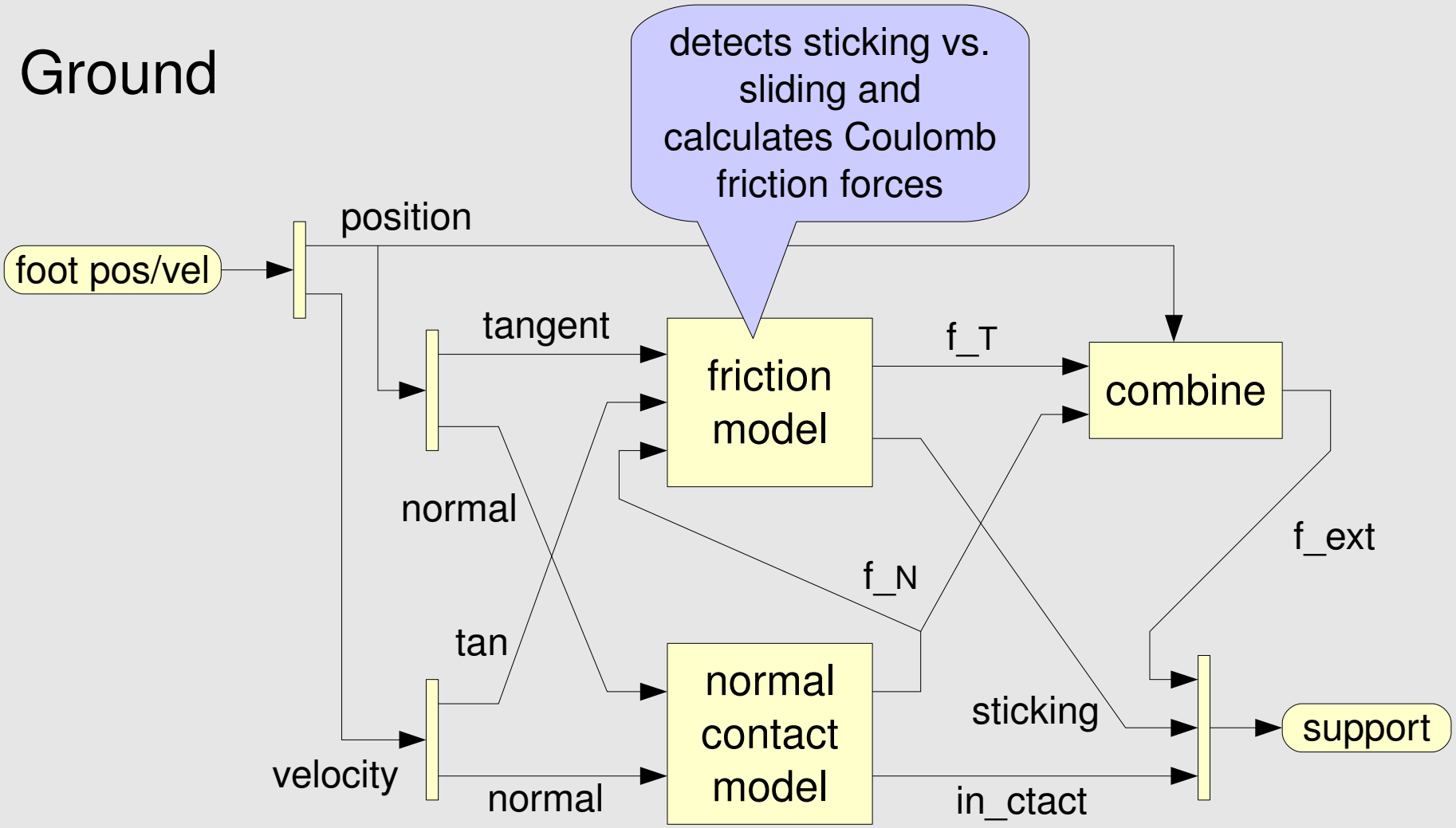


- a foot is the union of one or more points or spheres
- the model accepts data on multiple feet

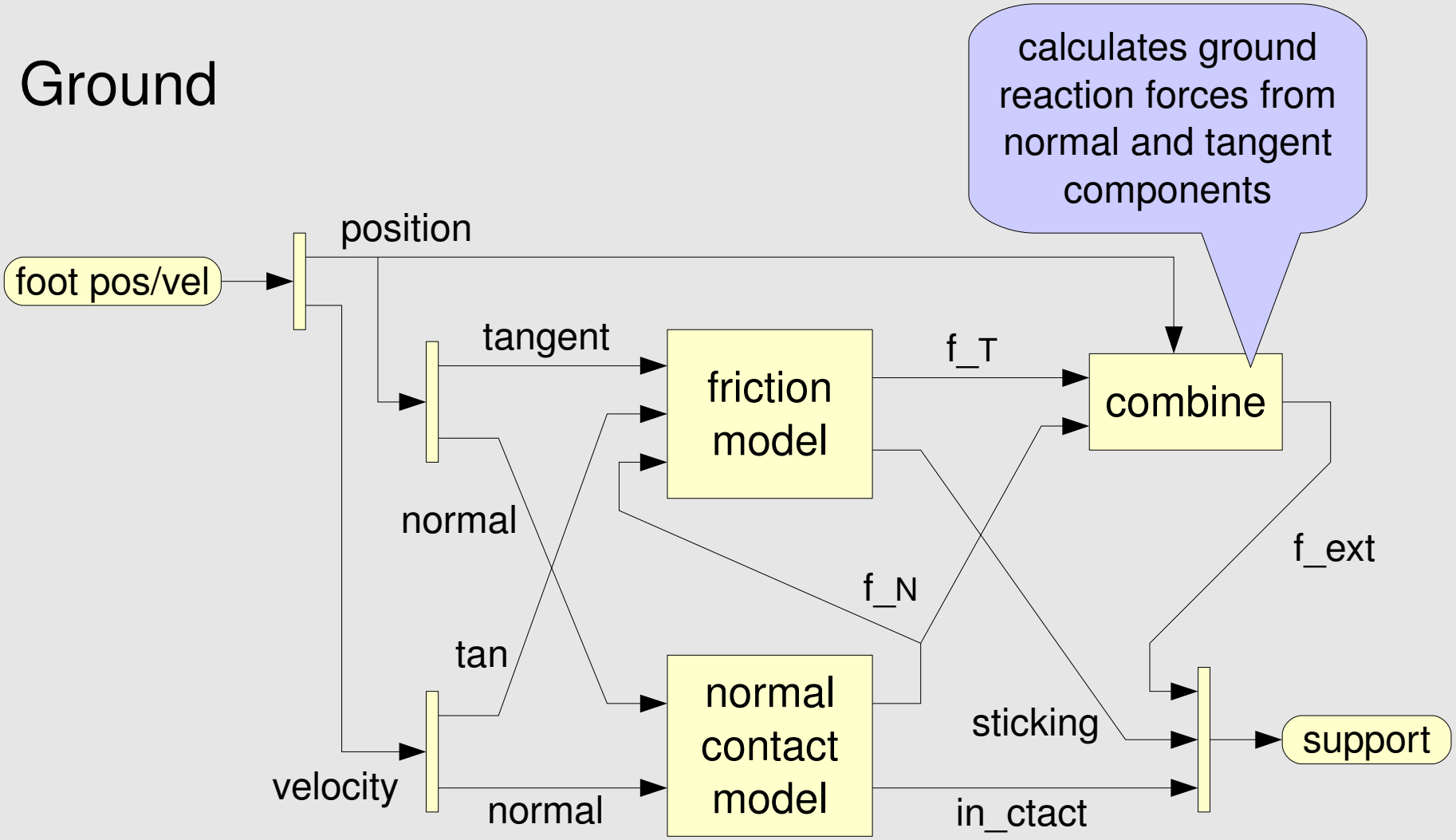
Ground


foot pos/vel
position
tangent
friction model
f_T
combine
normal
f_N
f_ext
tan
normal contact model
sticking
support
velocity
normal
in_ctact
detects contact and calculates contact normal forces

Ground


detects sticking vs. sliding and calculates Coulomb friction forces
position
foot pos/vel
tangent
friction model
f_T
combine
normal
f_N
f_ext
tan
normal contact model
sticking
support
velocity
normal
in_ctact

Ground


calculates ground reaction forces from normal and tangent components
position
foot pos/vel
tangent
friction model
f_T
combine
normal
f_N
f_ext
tan
normal contact model
sticking
support
velocity
normal
in_ctact

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