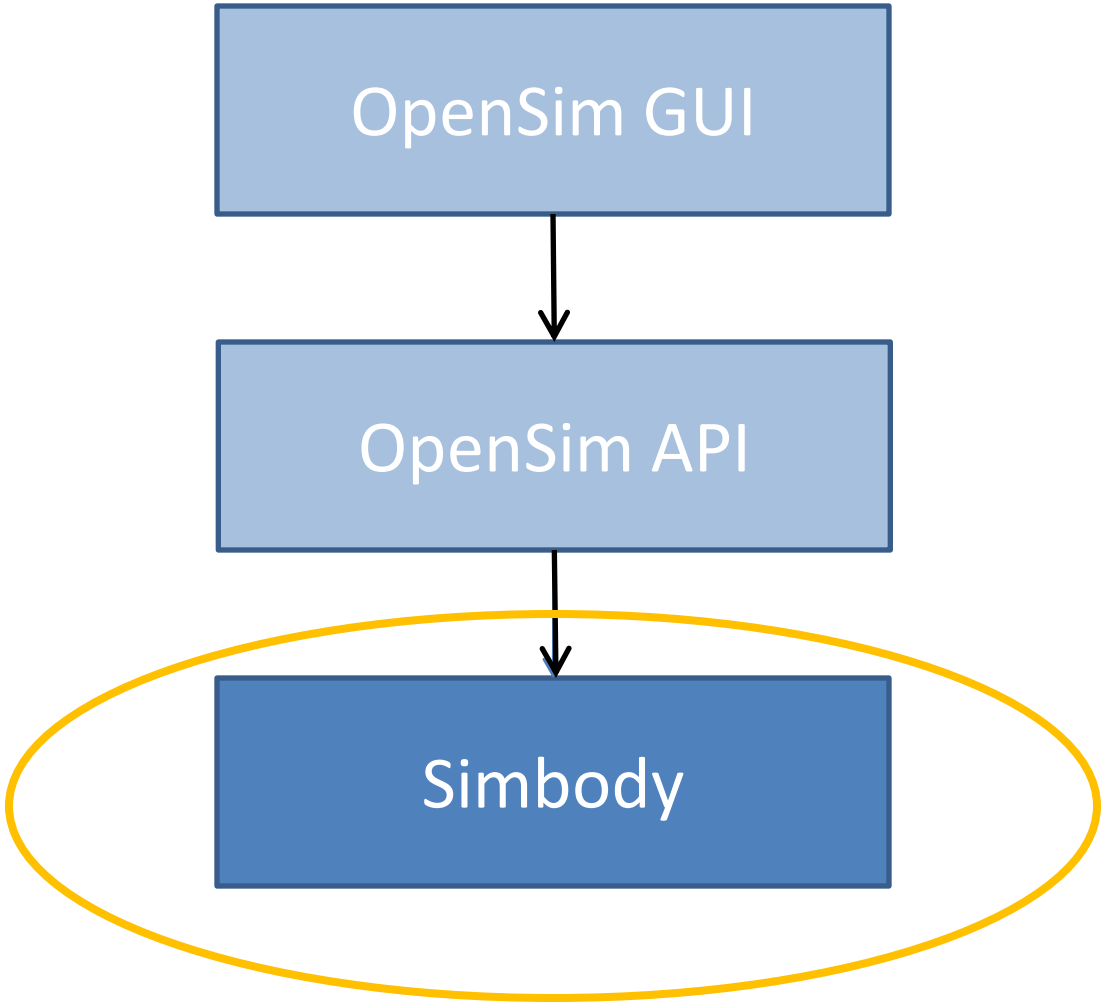
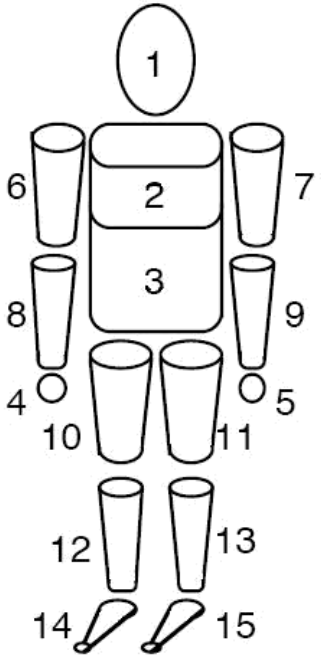
Simbody Concepts

OpenSim Developer's Week

Michael Sherman (Sherm), 16 July 2012

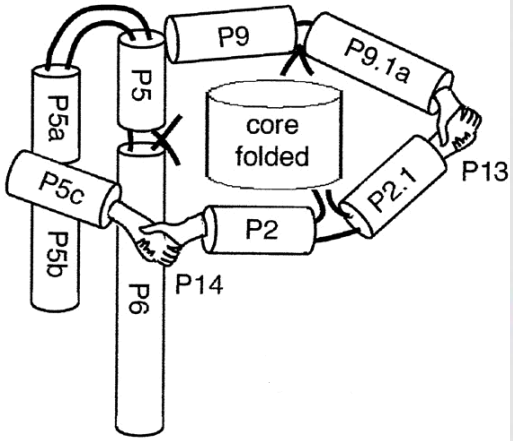

OpenSim GUI
OpenSim API
Simbody

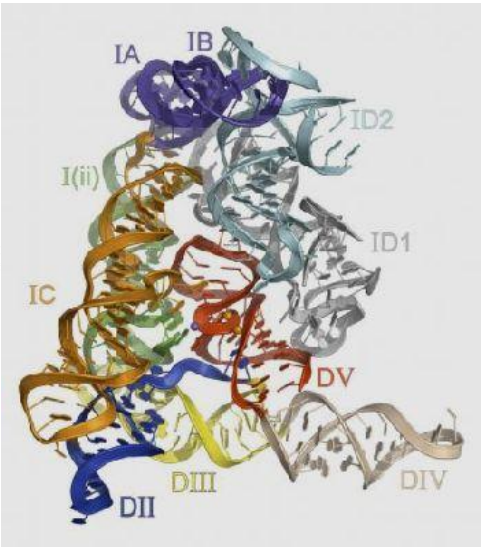
Similar models across multiple scales



Model of human

Huang, et al. Proc. 2nd Pacific Conf. Fundamentals Comp. Graphics, 1995





Model of RNA molecule

Zheng, et al. PNAS 98(7), 2001

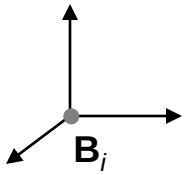
What is a multibody system?

Matter

- Mass
- Spatial distribution
- Motion

Abstract matter

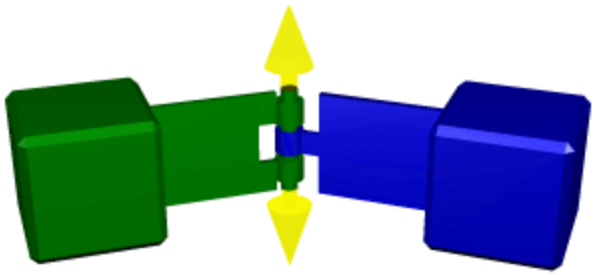
- The rigid body
- What is a rigid body?



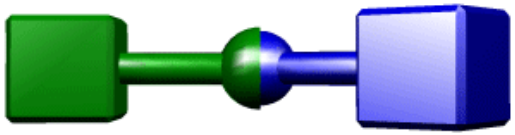
- Mass distribution: 10 *constants*
- Decorate w/geom & other props
- Ground is a (heavy) rigid body

Joints

- Defines relative mobility between 2 rigid bodies
- Examples

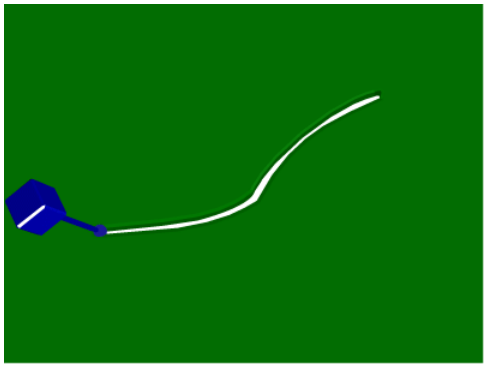


Pin



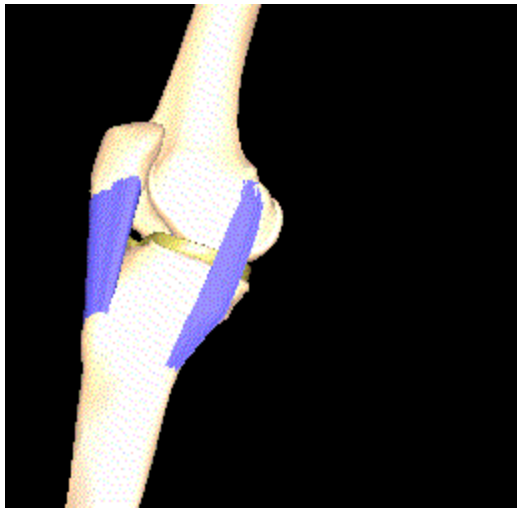
Ball

Slider

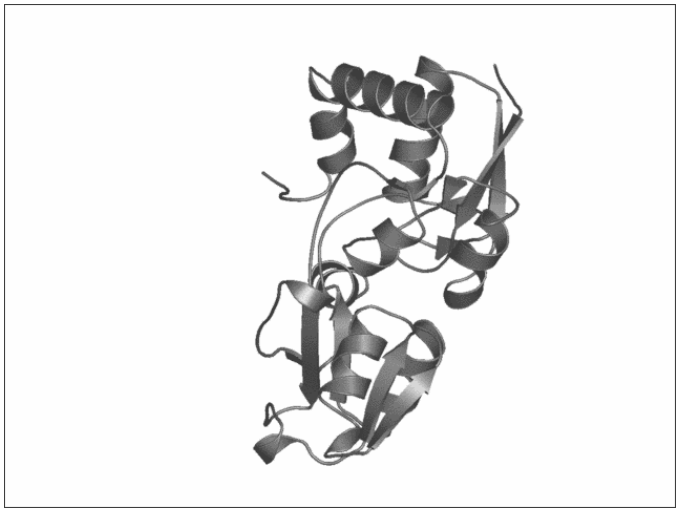


Pin-in-slot
(Custom)

(Biological joints are more complicated)



Human knee
Musculographics, Inc.



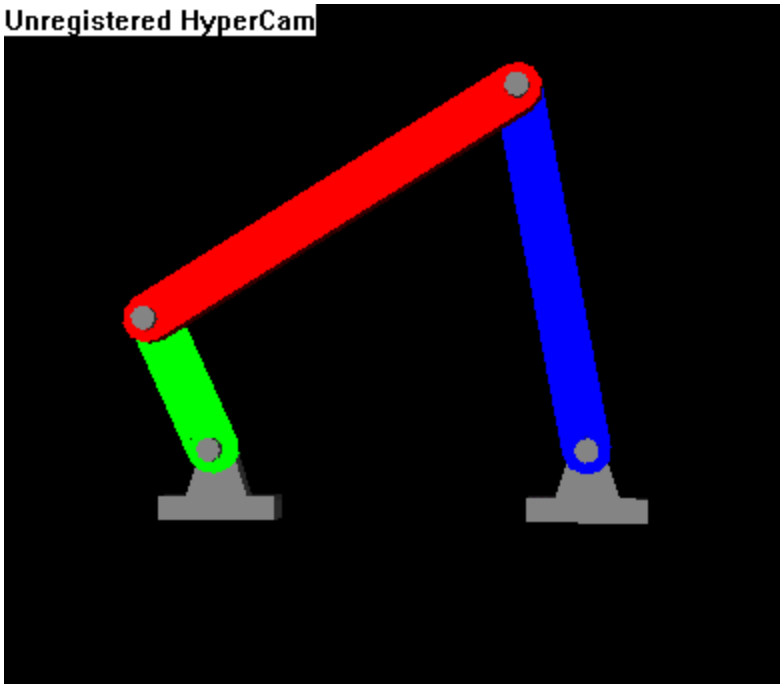
LAO binding protein
(1lst-2lao) Molmovdb.org

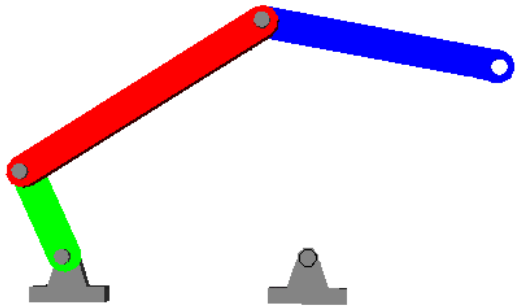
- Few degrees of freedom
- But, rotation & translation arbitrarily coupled

Simbody uses internal coordinates – why?

- Alternative: Cartesian coordinates
  - Each body has 6 dofs, joints are algebraic constraints
  - Large, sparse system: pin joint → **11 equations** (DAE)
  - Easy to implement
  - Used by almost all game physics engines (Baraff)
- Internal (generalized) coordinates
  - Select coords with physical meanings
  - Small, dense system: pin joint →**1 equation** (ODE)
  - Standard in biomechanics and robotics (e.g. Delp, Khatib)
  - Very fast, but difficult to implement efficiently (Featherstone)
- Internal coords contains Cartesian as a special case

Joints can *permit* or *restrict* motion

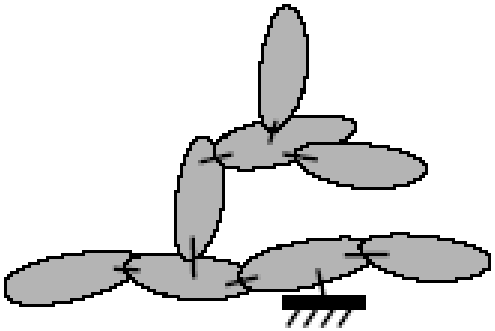




- 4-bar linkage has only 1 DOF
- But has 3 if you remove any one joint

Mobilizers
(joints that *permit* motion)

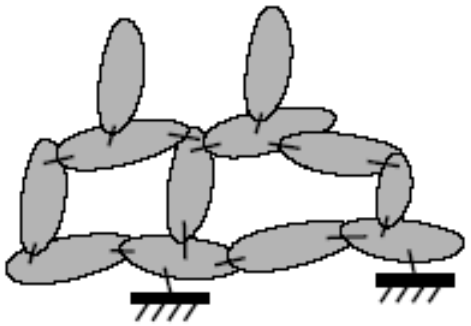
- In Simbody, bodies do not have inherent mobility
- Mobilizers precisely define the allowable motion relative to parent
- A mobilizer *always* increases the system's mobility
- These define *generalized coordinates q*, *generalized speeds u*: $\dot{q} = \mathrm{N}(q)u$ (usually $\dot{q} = u$)
- Bodies + mobilizers form a tree



$$\mathbf{M}\dot{u} = f$$

Constraints
(joints that *restrict* motion)

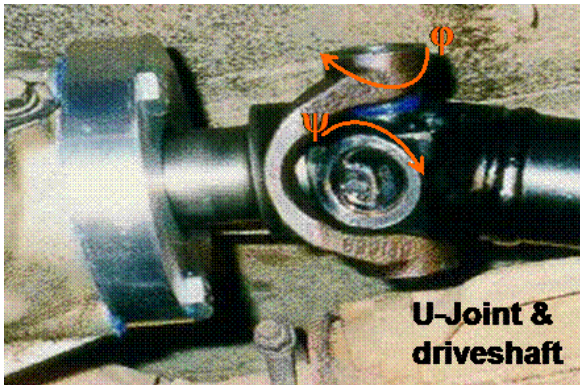
- Trees can be a little too floppy …
- *Constraints* introduce *constraint equations* (1 or more)
  - Restricts allowable motion – like negative mobility
  - E.g., ball constraint adds 3 constraint equations, -3 dofs
- Algebraic invariant relating q's and u's: g(q,u)=0
- But … might not be independent
- Must solve assembly problem before simulating
  - Find q such that g(q)=0
- Constraints permit loops, generate additional unknown forces

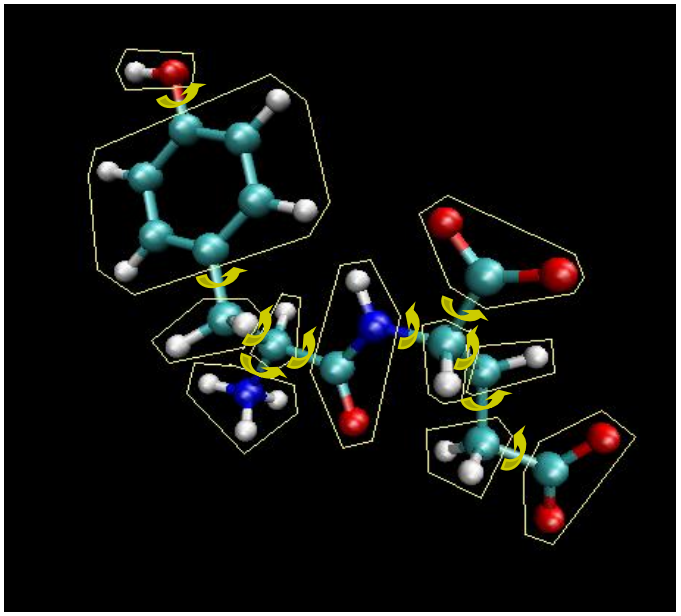


$$\mathbf{M}\dot{u} = f - f_c$$

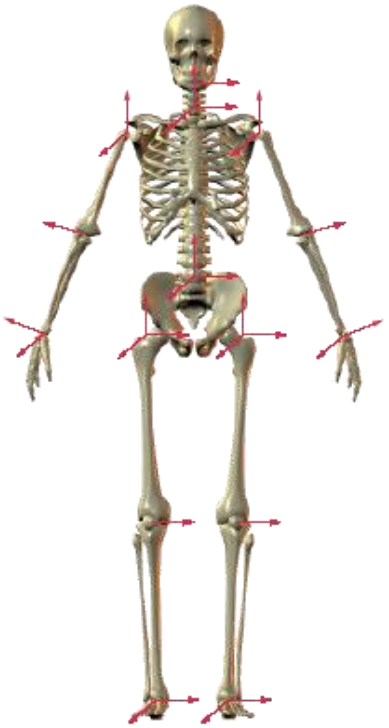
$$g(q) = 0$$

Multibody systems





- Rigid, immobile Bodies …
- … freed by Mobilizers
- … influenced by Forces
- … and restricted by Constraints.
- Key feature: motion is *localized*.



Simbody architecture

Some hard problems

- Models are hierarchical, resources are flat
- Implementation should be opaque, but we want to precisely record & restore state
- We need abstraction to handle complexity, but can't afford copying or recomputation
- Model development is extremely error-prone: e.g., user-managed "already computed this" flags
- Need "hints" for efficiency (e.g. previous contact solutions) but need to make trial evaluations which can be complete backed out
- Nothing is ever fast enough!

Worst problem: out-of-date results

- Typical error: use results from previous step or rejected trial step
- These are qualitatively just like the right results!
- No hint of error, plausible-looking results
  - Just pessimistic conclusions about modeling & stability
- Many simulations have subtle timing bugs that are never noticed
- These are not physics-based simulations
  - Not approximations; just plain wrong (or “not even wrong”)
- My experience: humans can’t get the hard cases right reliably

Built for speed & correctness

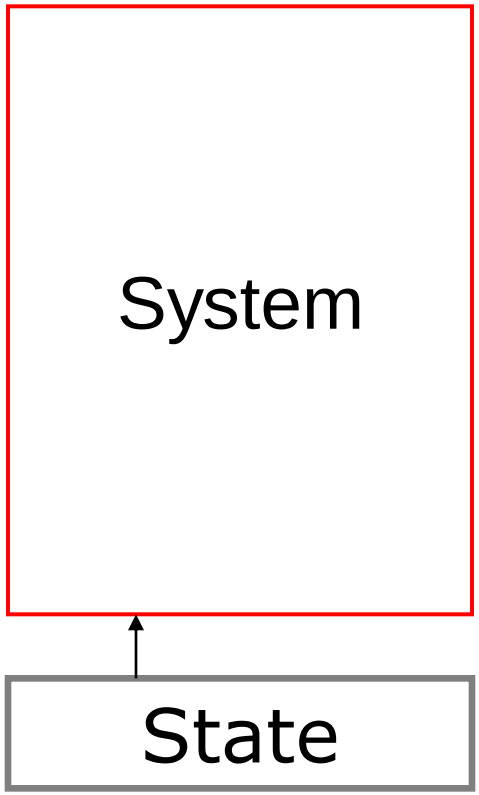
Physical world

Modeling

System

- A “System” is a computational embodiment of a mathematical model

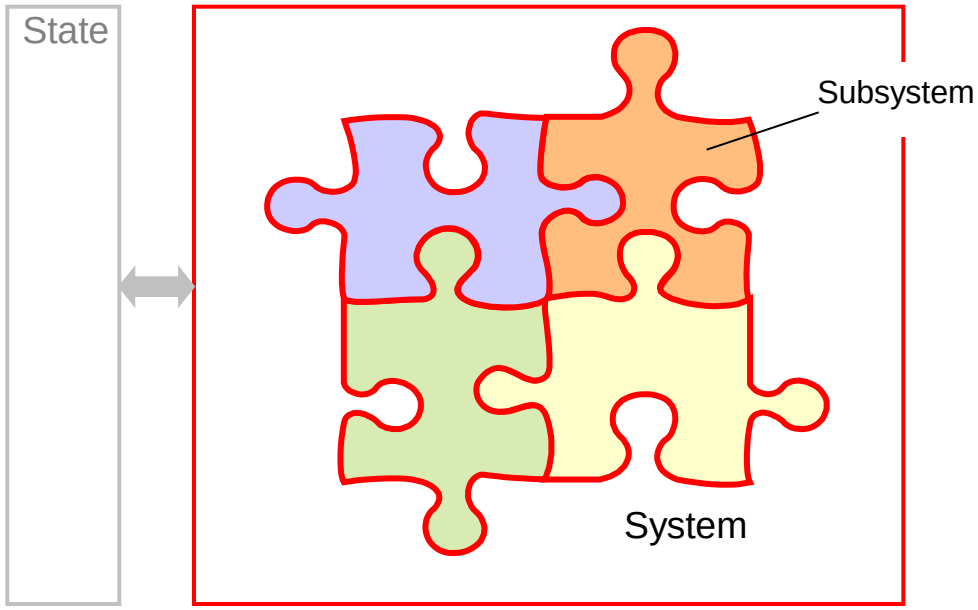
Properties of a Simbody System



- *Defines* its parameterization
- But … it is *stateless.*
- Given a *State*, performs useful computations

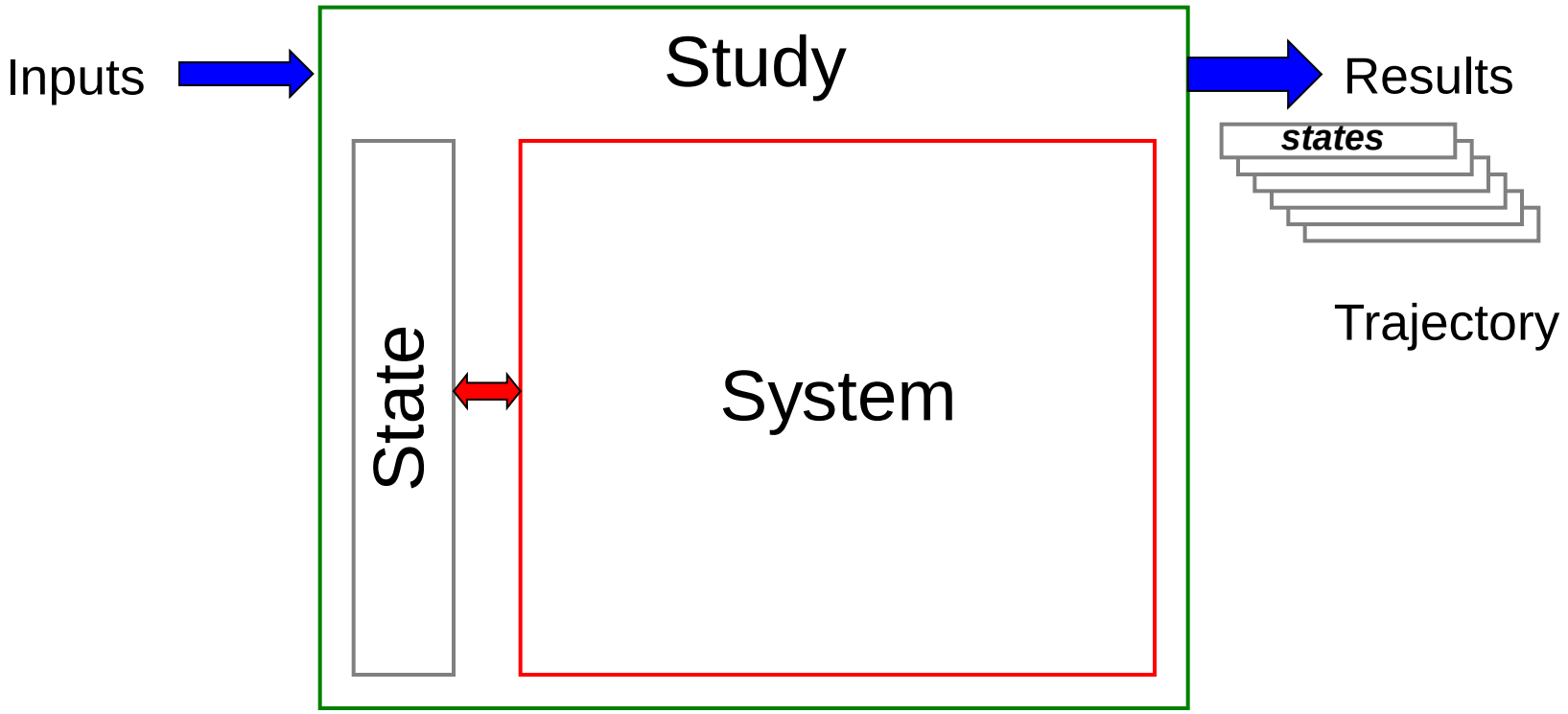
- Requires a very general notion of “state”

Systems are composed from Subsystems



- Interlocking computations
- System provides the “edge pieces”

Studying a system



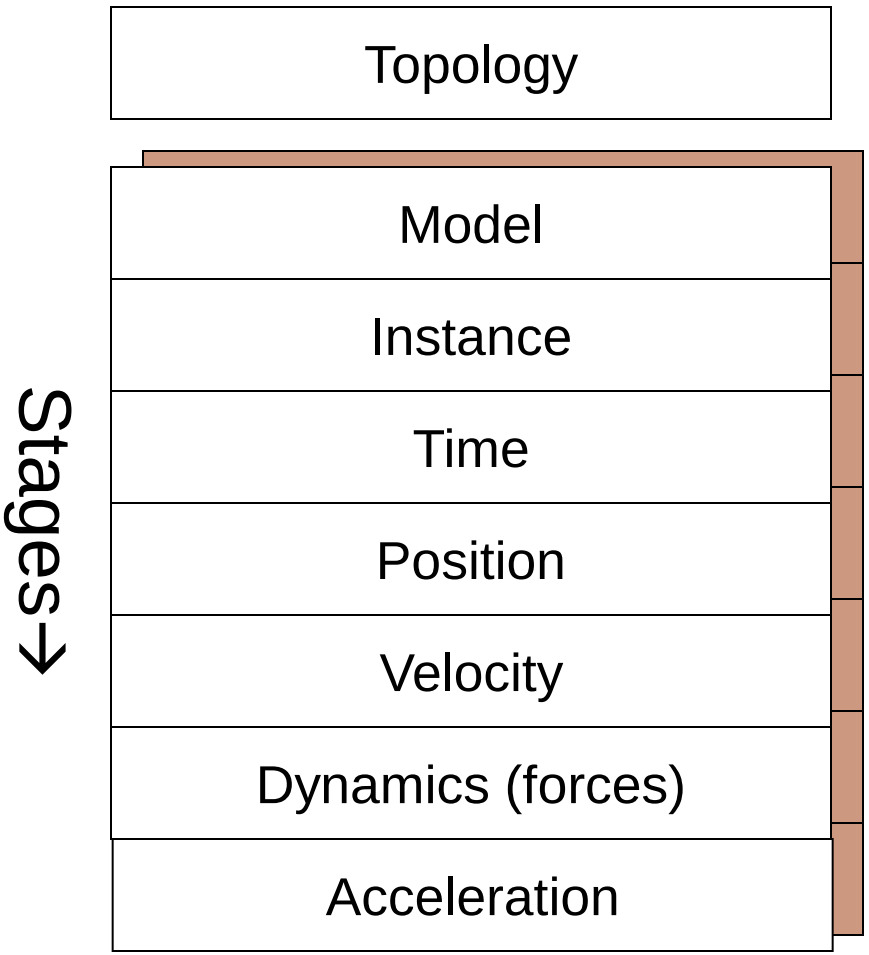
- *Any* study can be cast in this form
  - because only State can change

Contents of State

- Usual time, position, velocity variables t,q,u
  - And other continuous states z
- Parameters (mass, length, stiffness, …)
  - “instance” variables
- Discrete variables (event memory, samples)
- Flags, modeling choices, previous solutions

- And, results cache for speed & correctness

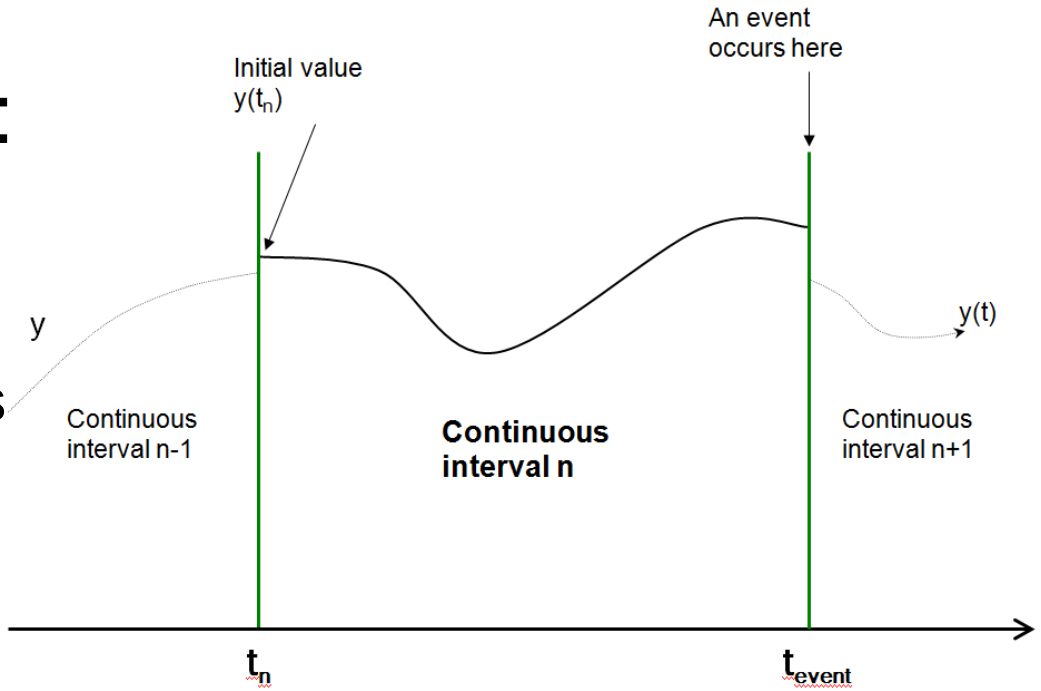
Managing state/computation coherency



- State is “shadowed” by per-stage cache
- State is “realized” into cache, by stage
- Coherency enforced automatically
- Cache maximizes performance, *never* affects the answer
- Eliminates buggy user-managed flags

Time stepper

- Numerical method for advancing a hybrid dynamic system model through time
- Repeat until done:
  1. Advance continuous system until discrete update required
  2. Make discrete changes
  3. Set up new initial conditions for continuous system

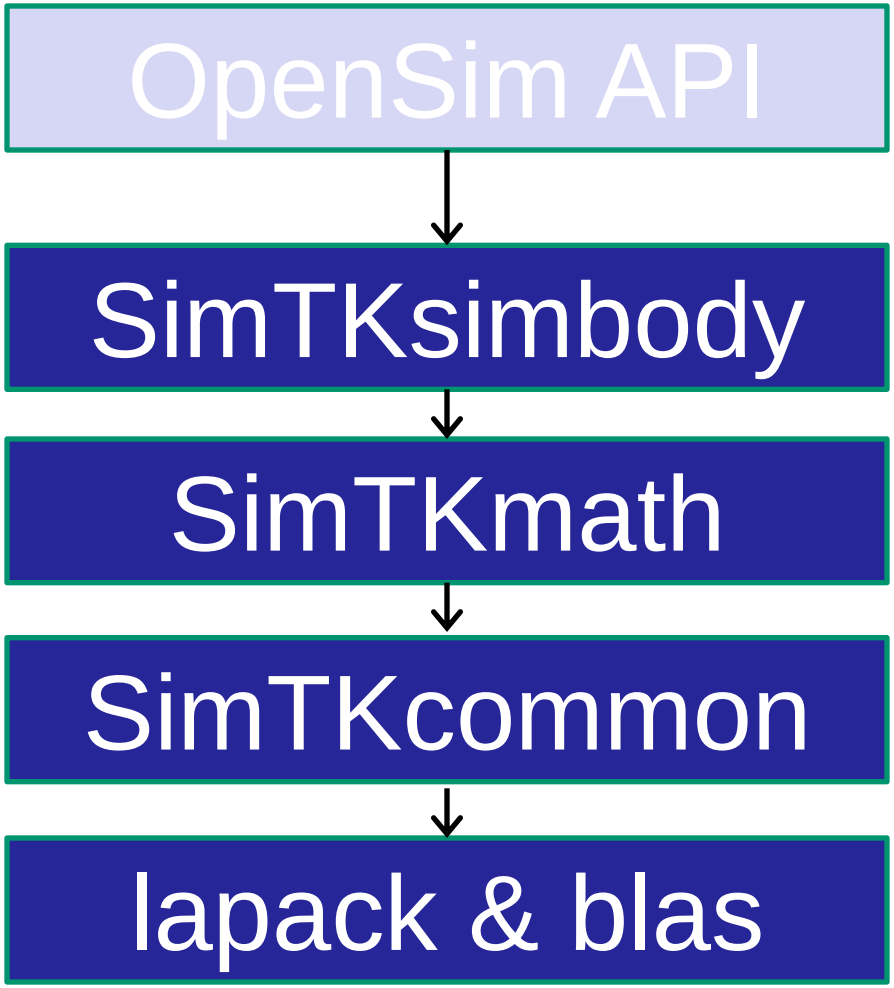


- Uses *any* numerical integrator to advance continuous system

Concept summary

- Computations proceed in ordered *stages*
- Generalized state, independent of model
  - Every study is defined by its effect on the state
  - Same state → same answer, always
  - Rigorous handling of stage computations in *realization cache*
- Simbody Systems are built from Subsystems
  - Subsystems are stateless
  - *Realize* external states, by stage
  - Permits each logical entity to contribute to each stage
- Time stepper uses integrator & event detection
  - Advance mixed continuous/discrete event system through time

Simbody libraries



order(n) multibody dynamics & visualizer

object-oriented numerical methods

Vector, Matrix, State, System, Subsystem, …

linear algebra

More information

- Sherman, et al. *Procedia IUTAM* 2, 2011.
  - Technical overview of Simbody
- Seth, et al. *Nonlinear Dynamics* 62, 2010.
  - Novel formulation of internal coordinate joints in Simbody
- https://simtk.org/home/simbody
  - User's Guide, Advanced User's Guide, Theory Manual, papers
  - API reference (Doxygen)
  - Forum, mailing lists, bug & feature reporting

Sherm: msherman@stanford.edu

Discussion

https://simtk.org/home/simbody

msherman@stanford.edu