Cyber-Physical Systems

Modeling Physical Dynamics

UNIVERSITY AT ALBANY
State University of New York

IECE 553/453– Fall 2019

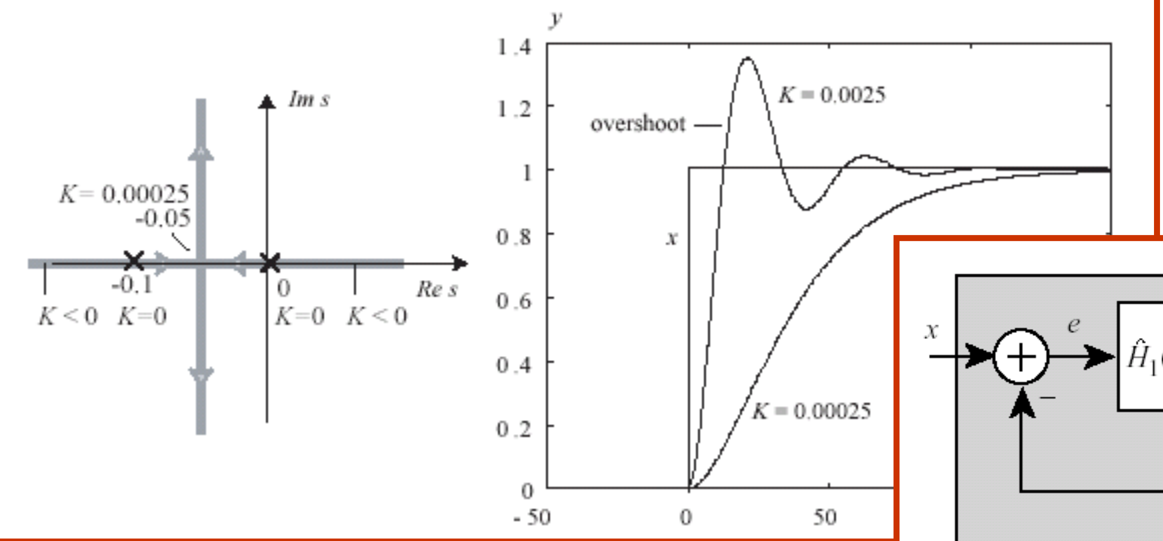
Prof. Dola Saha

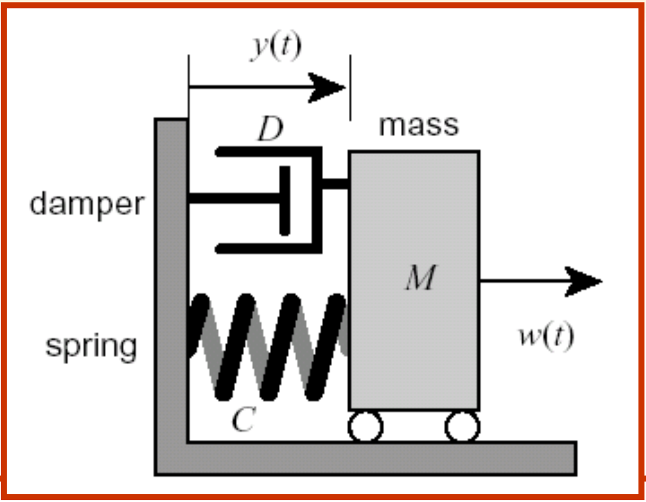
Modeling Techniques

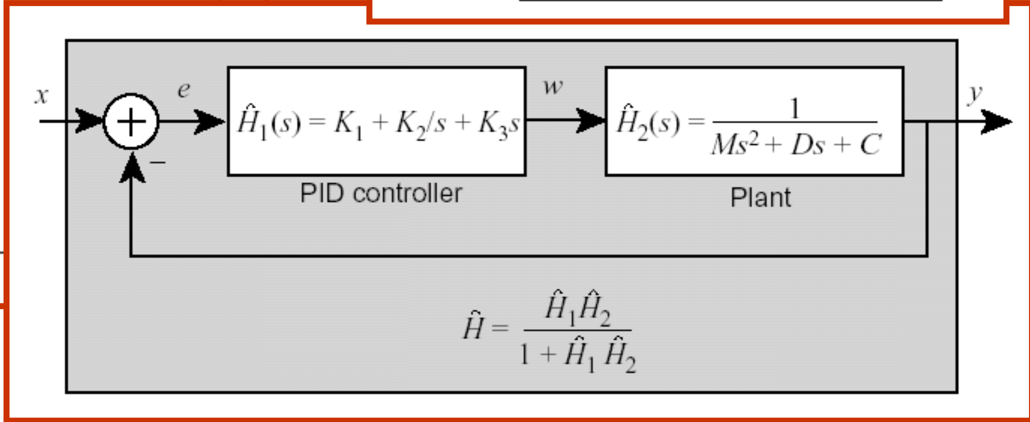
- Models that are abstractions of **system dynamics** (how system behavior changes over time)
  - Modeling physical phenomena – differential equations
  - Feedback control systems – time-domain modeling
  - Modeling modal behavior – FSMs, hybrid automata, …
  - Modeling sensors and actuators –calibration, noise, …
  - Hardware and software – concurrency, timing, power, …
  - Networks – latencies, error rates, packet losses, …

Modeling of Continuous Dynamics

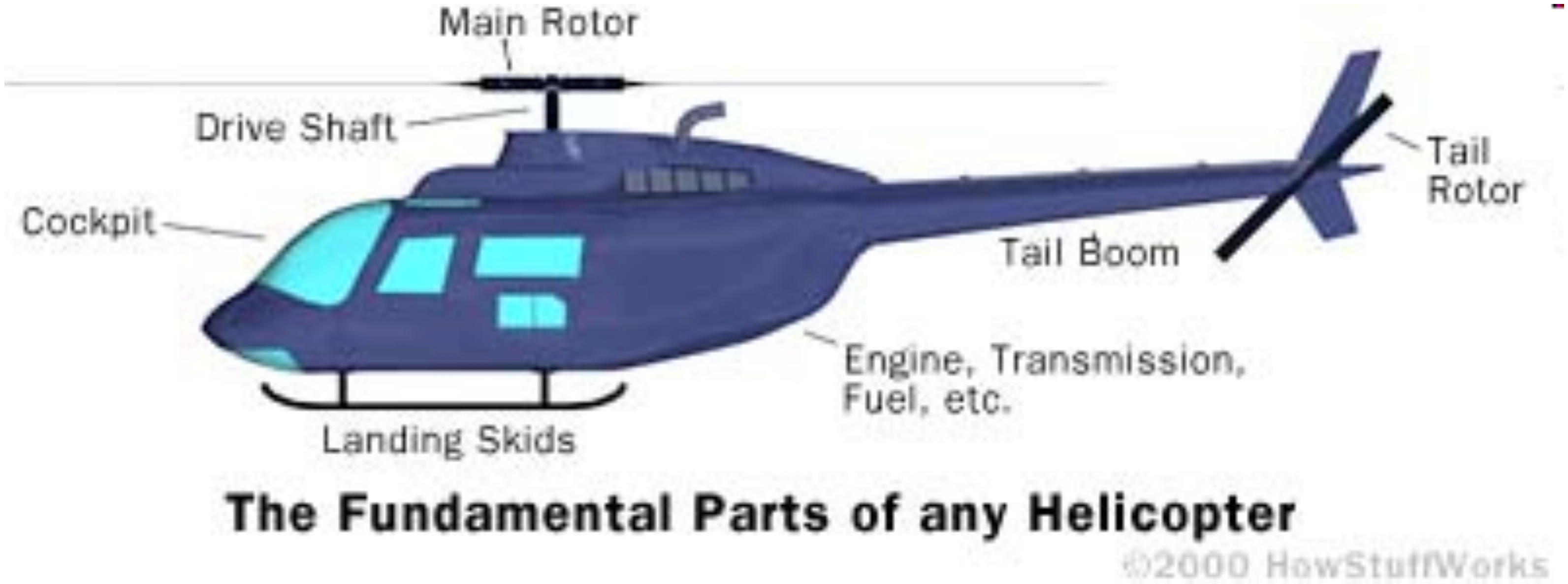
- Ordinary differential equations, Laplace transforms, feedback control models, …





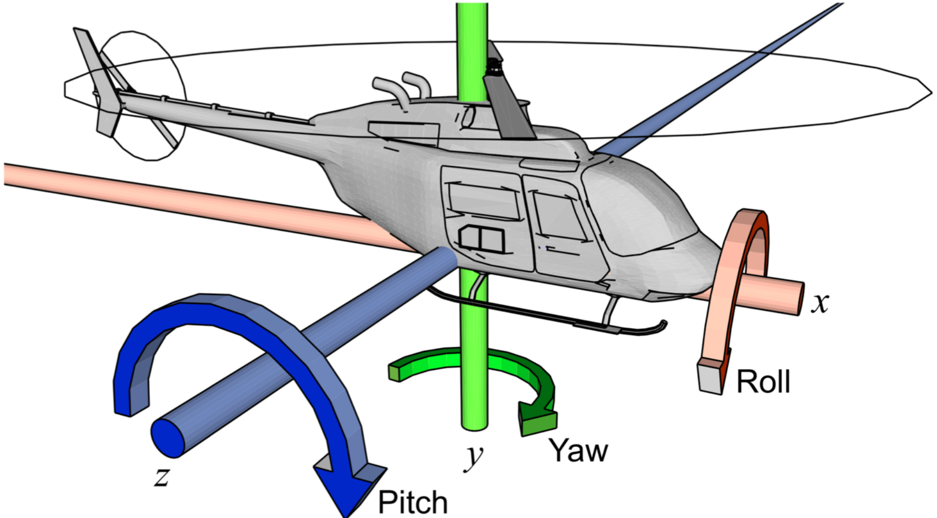


Example CPS System: Helicopter Dynamics



Modeling Physical Motion

- Six Degrees of Freedom
  - Position: x, y, z
  - Orientation: roll (θ_x), yaw (θ_y), pitch (θ_z)



Notation

Position is given by three functions:

$$x: \mathbb{R} \rightarrow \mathbb{R}$$
$$y: \mathbb{R} \rightarrow \mathbb{R}$$
$$z: \mathbb{R} \rightarrow \mathbb{R}$$

where the domain $\mathbb{R}$ represents time and the co-domain (range) $\mathbb{R}$ represents position along the axis. Collecting into a vector:

$$\mathbf{x}: \mathbb{R} \rightarrow \mathbb{R}^3$$

Position at time $t \in \mathbb{R}$ is $\mathbf{x}(t) \in \mathbb{R}^3$.

Orientation can be represented in the same form

- Functions of this form are known as continuous-time signals

Notation

Velocity

$$\dot{\mathbf{x}} : \mathbb{R} \to \mathbb{R}^3$$

is the derivative, $\forall\, t \in \mathbb{R}$,

$$\dot{\mathbf{x}}(t) = \frac{d}{dt}\mathbf{x}(t)$$

Acceleration $\ddot{\mathbf{x}} : \mathbb{R} \to \mathbb{R}^3$ is the second derivative,

$$\ddot{\mathbf{x}} = \frac{d^2}{dt^2}\mathbf{x}$$

Force on an object is $\mathbf{F} : \mathbb{R} \to \mathbb{R}^3$.

Newton's Second Law

Newton's second law states $\forall\, t \in \mathbb{R}$,

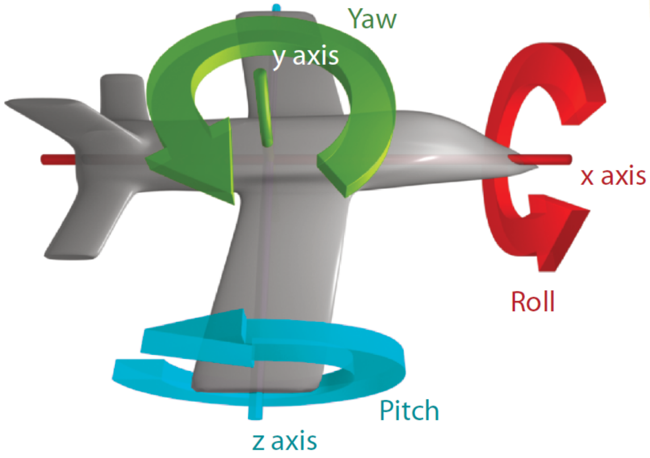
$$\mathbf{F}(t) = M\ddot{\mathbf{x}}(t)$$

where M is the mass. To account for initial position and velocity, convert this to an integral equation

$$\begin{aligned}\mathbf{x}(t) &= \mathbf{x}(0) + \int_0^t \dot{\mathbf{x}}(\tau)d\tau \\ &= \mathbf{x}(0) + t\dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \int_0^\tau \mathbf{F}(\alpha)d\alpha d\tau,\end{aligned}$$

Orientation

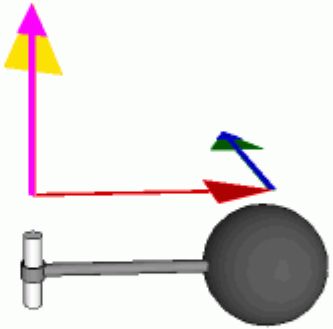
- Orientation: $\theta : \mathbb{R} \rightarrow \mathbb{R}^3$
- Angular velocity: $\dot{\theta} : \mathbb{R} \rightarrow \mathbb{R}^3$
- Angular acceleration: $\ddot{\theta} : \mathbb{R} \rightarrow \mathbb{R}^3$
- Torque: $\mathbf{T} : \mathbb{R} \rightarrow \mathbb{R}^3$



$$\theta(t) = \begin{bmatrix} \dot{\theta}_x(t) \\ \dot{\theta}_y(t) \\ \dot{\theta}_z(t) \end{bmatrix} = \begin{bmatrix} \text{roll} \\ \text{yaw} \\ \text{pitch} \end{bmatrix}$$

Torque: Angular version of Force

- radius of the arm: $r \in \mathbb{R}$
- force orthogonal to arm: $f \in \mathbb{R}$
- mass of the object: $m \in \mathbb{R}$



$T_y(t) = r\, f(t)$

angular momentum, momentum

Just as force is a push or a pull, a torque is a twist.
Units: newton-meters/radian, Joules/radian

Rotational Version of Newton's Law

$$\mathbf{T}(t) = \frac{d}{dt}\left(I(t)\dot{\theta}(t)\right),$$

where $I(t)$ is a 3×3 matrix called the moment of inertia tensor.

$$\begin{bmatrix} T_x(t) \\ T_y(t) \\ T_z(t) \end{bmatrix} = \frac{d}{dt}\left(\begin{bmatrix} I_{xx}(t) & I_{xy}(t) & I_{xz}(t) \\ I_{yx}(t) & I_{yy}(t) & I_{yz}(t) \\ I_{zx}(t) & I_{zy}(t) & I_{zz}(t) \end{bmatrix}\begin{bmatrix} \dot{\theta}_x(t) \\ \dot{\theta}_y(t) \\ \dot{\theta}_z(t) \end{bmatrix}\right)$$

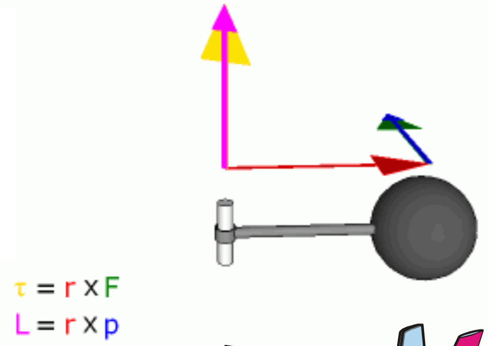
Here, for example, $T_y(t)$ is the net torque around the y axis (which would cause changes in yaw), $I_{yx}(t)$ is the inertia that determines how acceleration around the x axis is related to torque around the y axis.

If the object is spherical, this reluctance is the same around all axes, so it reduces to a constant scalar $\mathbf{I}$ (or equivalently, to a diagonal matrix $\mathbf{I}$ with equal diagonal elements I).

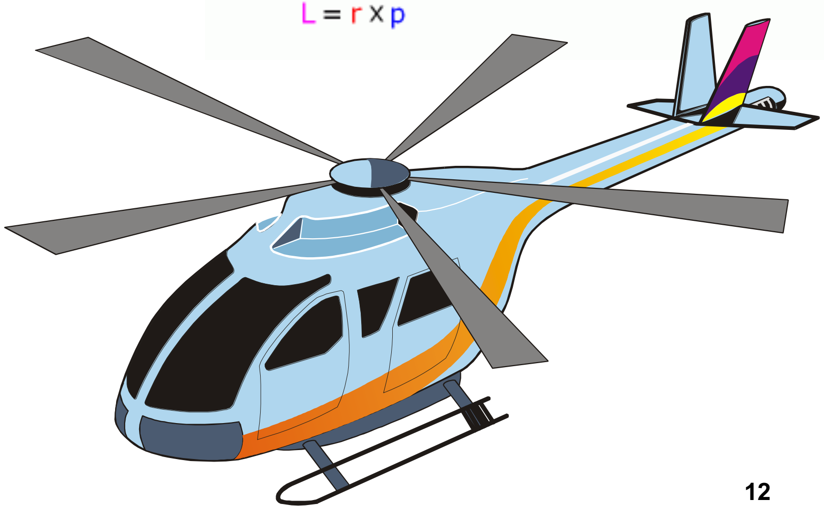
$$\mathbf{T}(t) = I\ddot{\theta}(t)$$

Feedback Control Problem

A helicopter without a tail rotor, like the one below, will spin uncontrollably due to the torque induced by friction in the rotor shaft.



Control system problem: Apply torque using the tail rotor to counterbalance the torque of the top rotor.



For a spherical object

Rotational velocity is the integral of acceleration,

$$\dot{\theta}(t) = \dot{\theta}(0) + \int_0^t \ddot{\theta}(\tau)d\tau,$$

$$\dot{\theta}(t) = \dot{\theta}(0) + \frac{1}{I}\int_0^t \mathbf{T}(\tau)d\tau.$$

Orientation is the integral of rotational velocity,

$$\begin{aligned}\theta(t) &= \theta(0) + \int_0^t \dot{\theta}(\tau)d\tau \\ &= \theta(0) + t\dot{\theta}(0) + \frac{1}{I}\int_0^t \int_0^\tau \mathbf{T}(\alpha)d\alpha d\tau\end{aligned}$$

Simplified Model

- Model-order Reduction

Yaw dynamics:

$$T_y(t) = I_{yy}\ddot{\theta}_y(t)$$

To account for initial angular velocity, write as

$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}}\int_0^t T_y(\tau)d\tau.$$

Simplified Model of Helicopter

- the force produced by the tail rotor must counter the torque produced by the main rotor
- Assumptions:
  - helicopter position is fixed at the origin
  - helicopter remains vertical, so pitch and roll are fixed at zero
- the moment of inertia reduces to a scalar that represents a torque that resists changes in yaw

$$\ddot{\theta}_y(t) = T_y(t)/I_{yy}$$

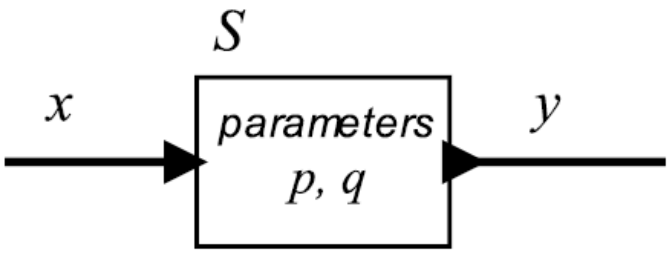
$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}}\int_0^t T_y(\tau)d\tau$$

Actor Model

- Mathematical Model of Concurrent Computation
- Actor is an unit of computation
- Actors can
  - Create more actors
  - Send messages to other actors
  - Designate what to do with the next message
- Multiple actors may execute at the same time

Actor Model of Systems

- A *system* is a function that accepts an input *signal* and yields an output signal.
- The domain and range of the system function are sets of signals, which themselves are functions.
- Parameters may affect the definition of the function *S*.



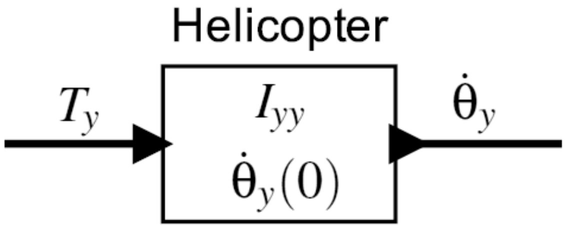
$$x\colon \mathbb{R} \to \mathbb{R}, \quad y\colon \mathbb{R} \to \mathbb{R}$$

$$S\colon X \to Y$$

$$X = Y = (\mathbb{R} \to \mathbb{R})$$

Actor Model of the Helicopter

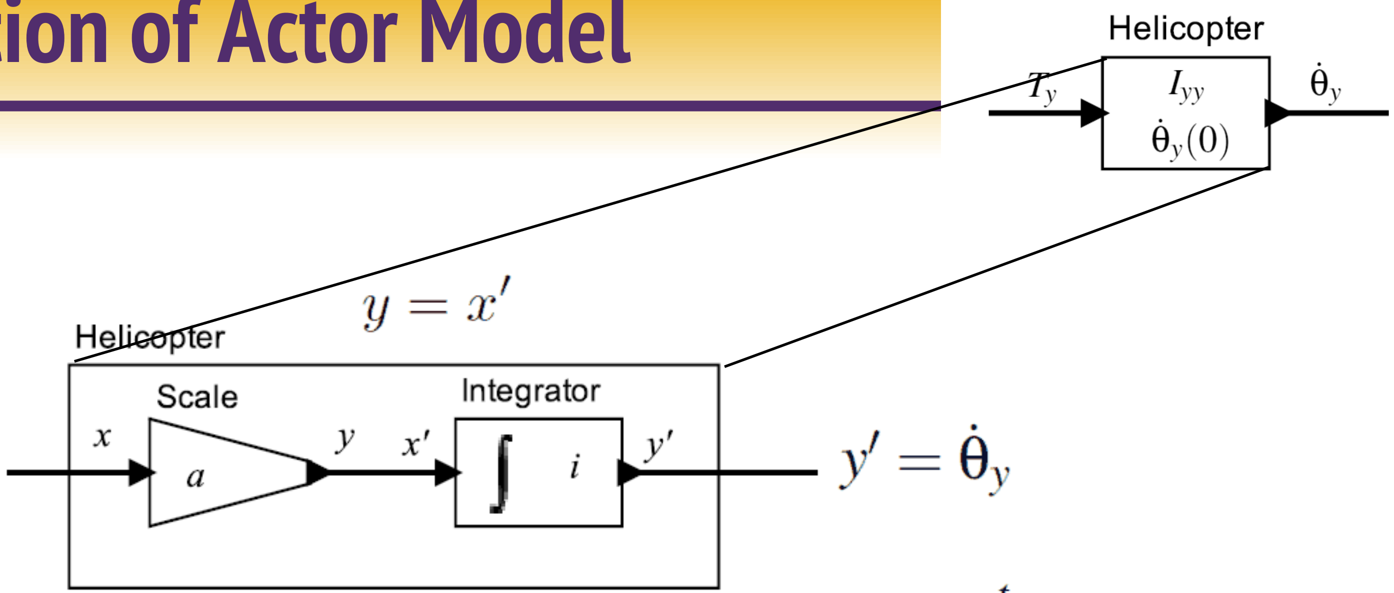
- Input is the net torque of the tail rotor and the top rotor. Output is the angular velocity around the y-axis.



- Parameters of the model are shown in the box. The input and output relation is given by the equation to the right.

$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau) d\tau$$

Composition of Actor Model



$$y = x'$$

$$x = T_y$$

$$y' = \dot{\theta}_y$$

$$\forall\, t \in \mathbb{R}, \quad y(t) = ax(t)$$

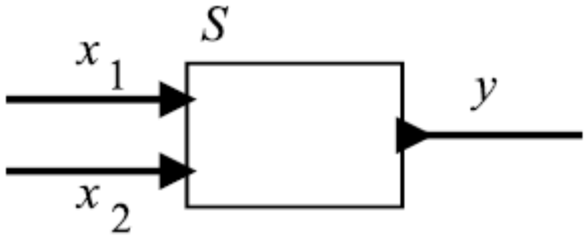
$$y'(t) = i + \int_0^t x'(\tau)d\tau$$

$$y = ax$$

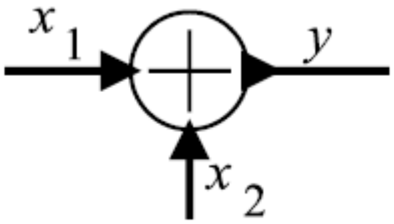
$$i = \dot{\theta}_y(0)$$

$$a = 1/I_{yy}$$

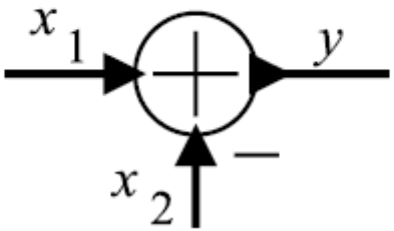
Actor Models with Multiple Inputs



$$S\colon (\mathbb{R} \to \mathbb{R})^2 \to (\mathbb{R} \to \mathbb{R})$$



$$\forall\, t \in \mathbb{R}, \quad y(t) = x_1(t) + x_2(t)$$

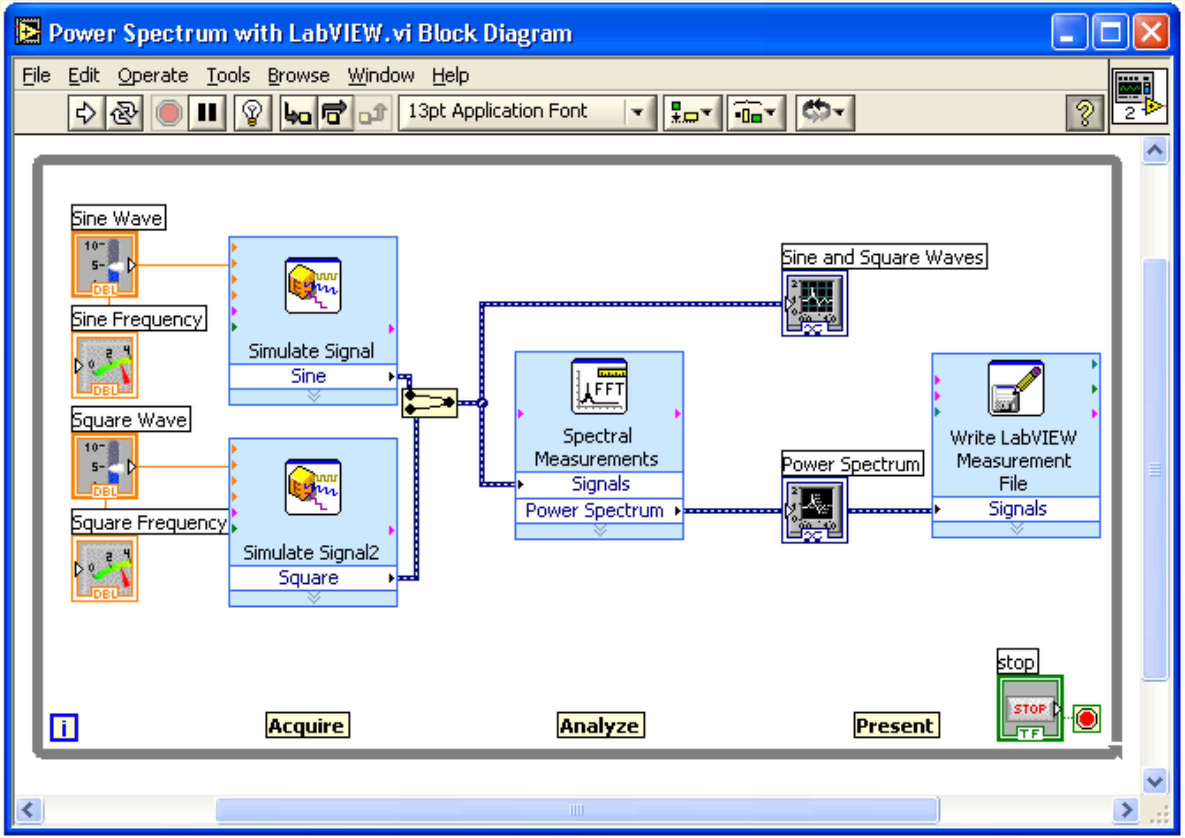


$$(S(x_1, x_2))(t) = y(t) = x_1(t) - x_2(t)$$

Modern Actor Based Platforms

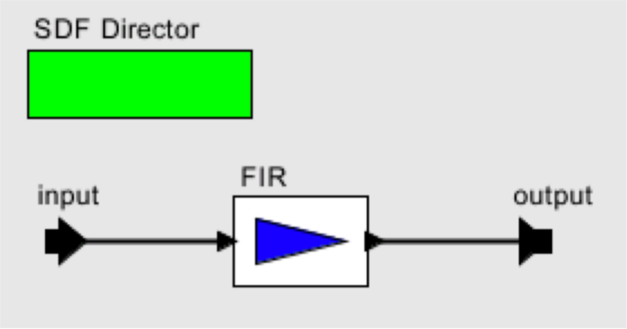
- Simulink (The MathWorks)
- Labview (National Instruments)
- Modelica (Linkoping)
- OPNET (Opnet Technologies)
- Polis & Metropolis (UC Berkeley)
- Gabriel, Ptolemy, and Ptolemy II (UC Berkeley)
- OCP, open control platform (Boeing)
- GME, actor-oriented meta-modeling (Vanderbilt)
- SPW, signal processing worksystem (Cadence)
- System studio (Synopsys)
- ROOM, real-time object-oriented modeling (Rational)
- Easy5 (Boeing)
- Port-based objects (U of Maryland)
- I/O automata (MIT)
- VHDL, Verilog, SystemC (Various)

Example LabVIEW Screenshot



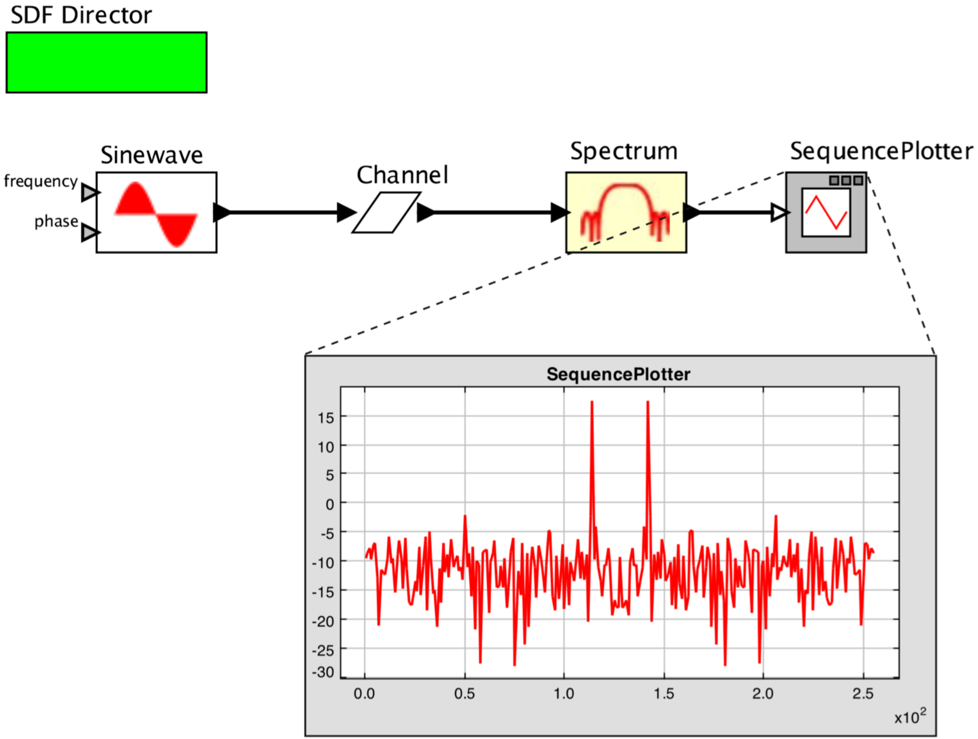
Synchronous Dataflow (SDF)

- Specialized model for dataflow
- All actors consume input tokens, perform their computation and produce outputs in one atomic operation
- Flow of control is known (predictable at compile time)
- Statically scheduled domain
- Useful for synchronous signal processing systems
- Homogeneous SDF: one token is usually produced for every iteration

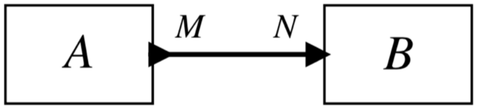


Multirate SDF Model

- The firing rates of the actors are not identical
- The Spectrum actor requires 256 tokens to fire, so one iteration of this model requires 256 firings of Sinewave, Channel, and SequencePlotter, and one firing of Spectrum.



Balance Equations

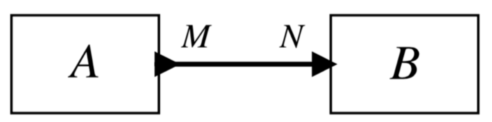


- When A fires, it produces M tokens on its output port
- When B fires, it consumes N tokens on its input port
- M and N are non-negative integers
- Suppose that A fires q_A times and B fires q_B times
- All tokens that A produces are consumed by B if and only if the following **balance equation** is satisfied

$$q_A M = q_B N$$

- The system remains in balance if and only if the balance equation is satisfied

Example



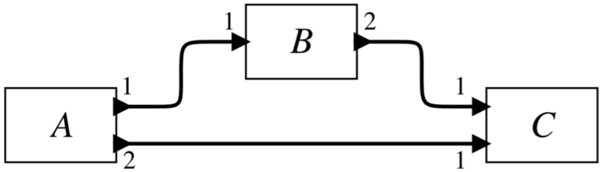
- Suppose M=2, N=3
- Possible Solution:
  - $q_A=3$, $q_B=2$
  - Example Schedule : {A, A, A, B, B} OR {A, B, A, A, B}
- Another Possible Solution:
  - $q_A=6$, $q_B=4$
  - Example Schedule: {A,A,A,A,A,A,B,B,B,B}

Strategy for firing

- Streaming applications: arbitrarily large number of tokens
- Naive strategy: fire actor A an arbitrarily large number q_A times, and then fire actor B q_B times
  - Why naive?
- Better strategy:
  - smallest positive q_A and q_B that satisfy the balance equation
- Unbounded execution with bounded buffers

Solving the Balance Equation

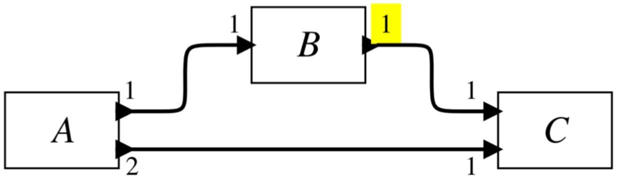
- Every connection between actors results in a balance equation
- The model defines a system of equations, and the goal is to find the least positive integer solution



$$
\begin{aligned}
q_A &= q_B \\
2q_B &= q_C \\
2q_A &= q_C
\end{aligned}
$$

- The least positive integer solution to these equations is
  - $q_A = q_B = 1$, and $q_C = 2$
- The schedule {A, B, C, C} can be repeated forever to get an unbounded execution with bounded buffers

Inconsistent SDF

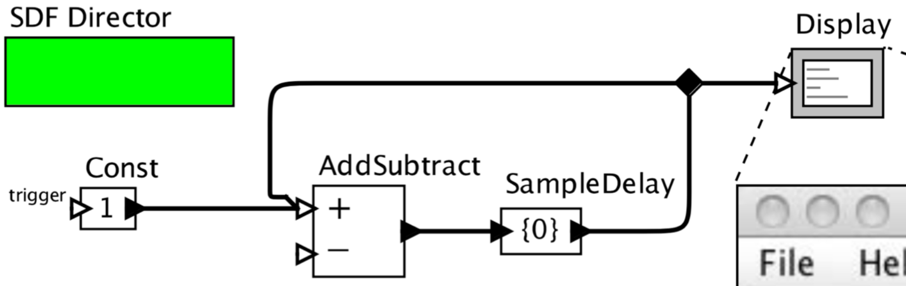


$$q_A = q_B = q_C = 0$$

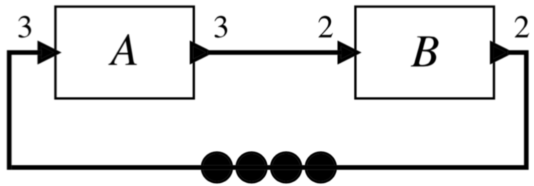
- An SDF model that has a non-zero solution to the balance equations is said to be consistent.
- If the only solution is zero, then it is inconsistent.
- An inconsistent model has no unbounded execution with bounded buffers.

Feedback Loop

- A feedback loop in SDF must include at least one instance of the SampleDelay actor
- Without this actor, the loop would deadlock
  - actors in the feedback loop would be unable to fire because they depend on each other for tokens.
- The initial tokens enable downstream actors to fire and break the circular dependencies that would otherwise result from a feedback loop



Example Feedback Loop



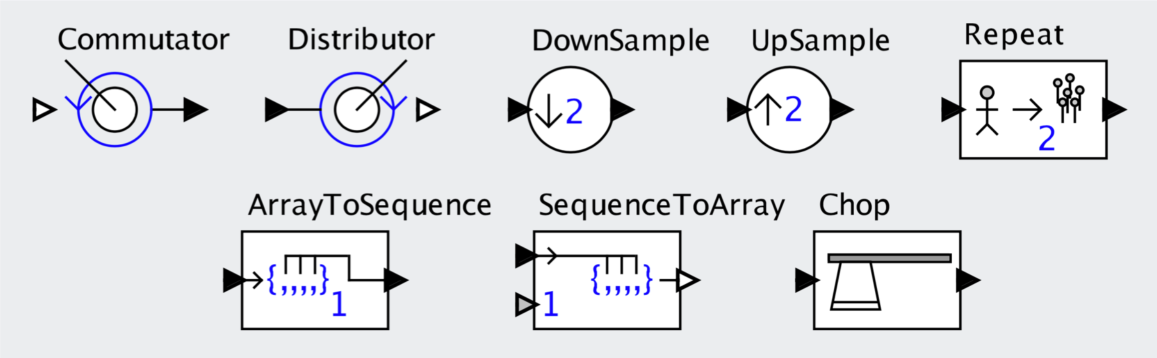
$$3q_A = 2q_B$$
$$2q_B = 3q_A$$

$$A, B, A, B, B$$

- The least positive integer solution is
  - qA = 2, qB = 3, so the model is consistent.
- With 4 initial tokens: consistent
- With 3 initial tokens: deadlock

Multirate Dataflow Actors

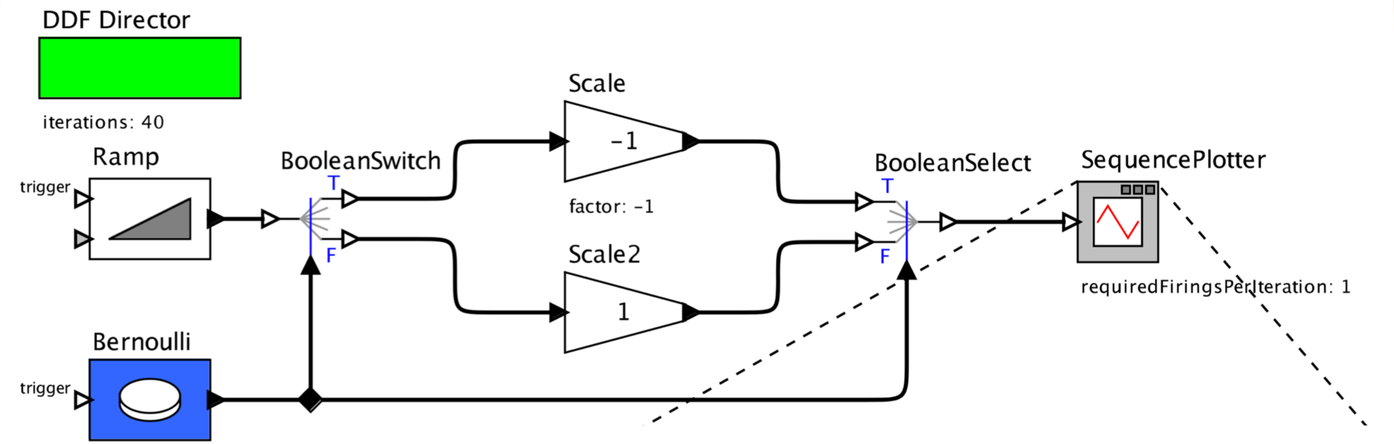
- actors that produce and/or consume multiple tokens per firing on a port



Dynamic Dataflow (DDF)

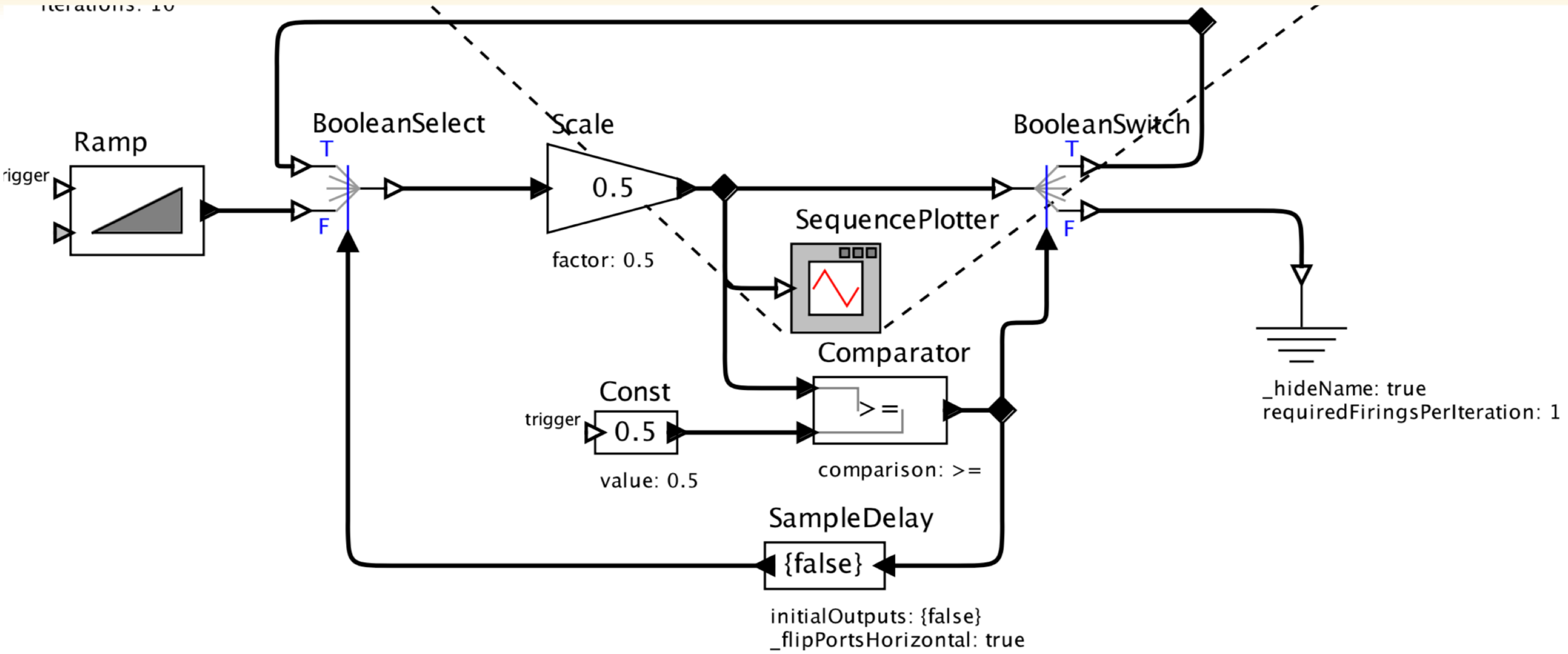
- SDF cannot express conditional firing: an actor fires only if a token has a particular value
- DDF: Firing Rule is required to be satisfied for firing
- Number of tokens produced can vary
- Example DDF Actor: Select
- Similar to Go To in Imperative Programming

Example DDF (Conditional Firing)

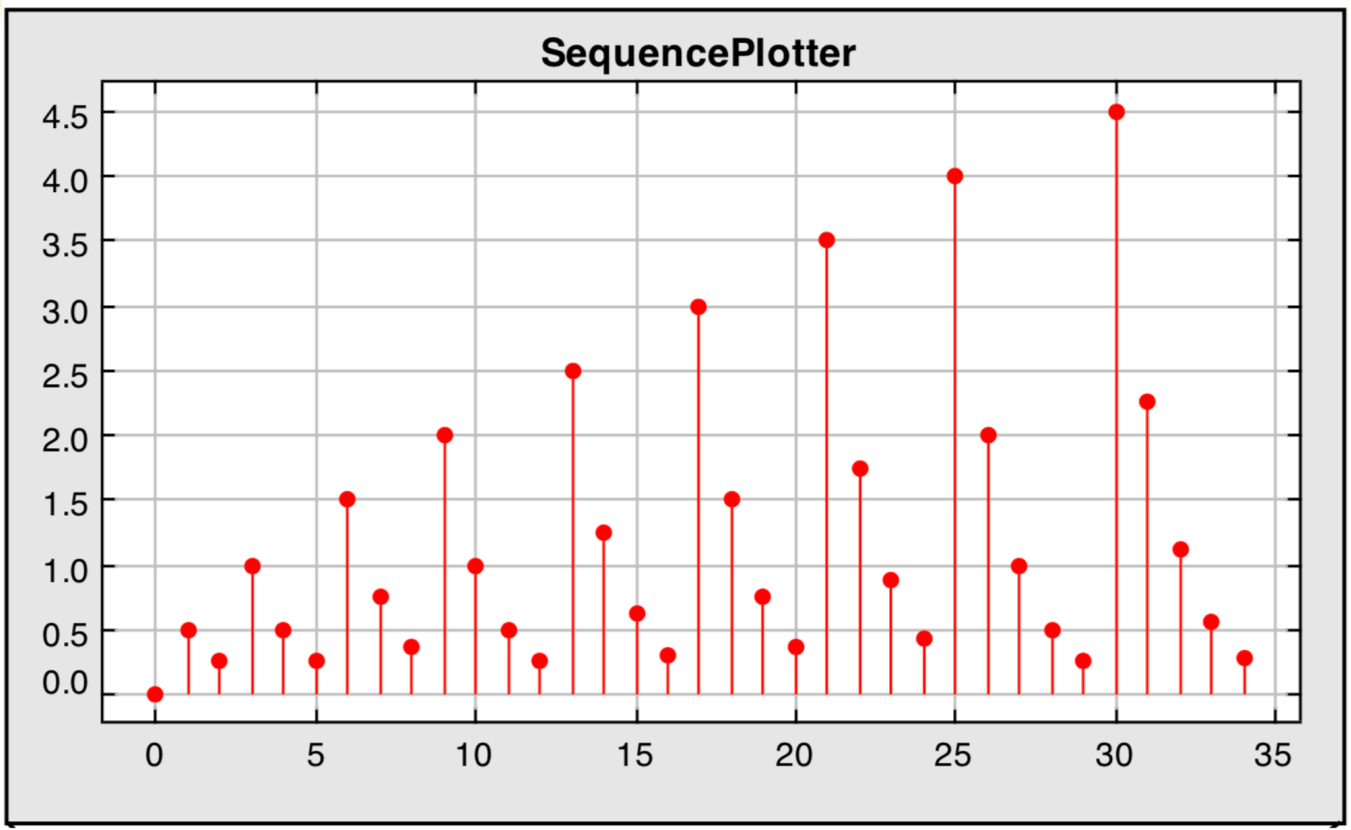


When Bernoulli produces true, the output of the Ramp actor is multiplied by −1

Data Dependent Iteration

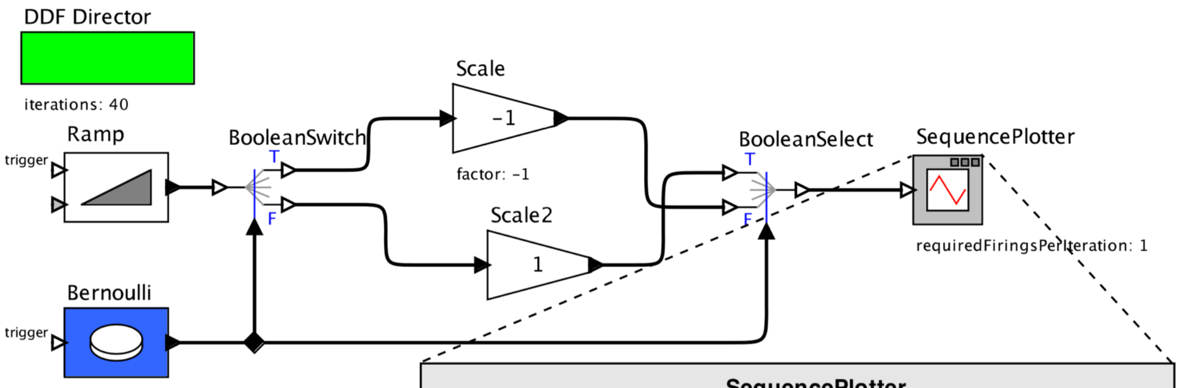


Conditional Firing Output



Unbounded Buffer Schedule

- The Bernoulli actor is capable of producing an arbitrarily long sequence of true-valued tokens, during which an arbitrarily long sequence of tokens may build up on input buffer for the *false* port of the BooleanSelect, thus potentially overflowing the buffer.

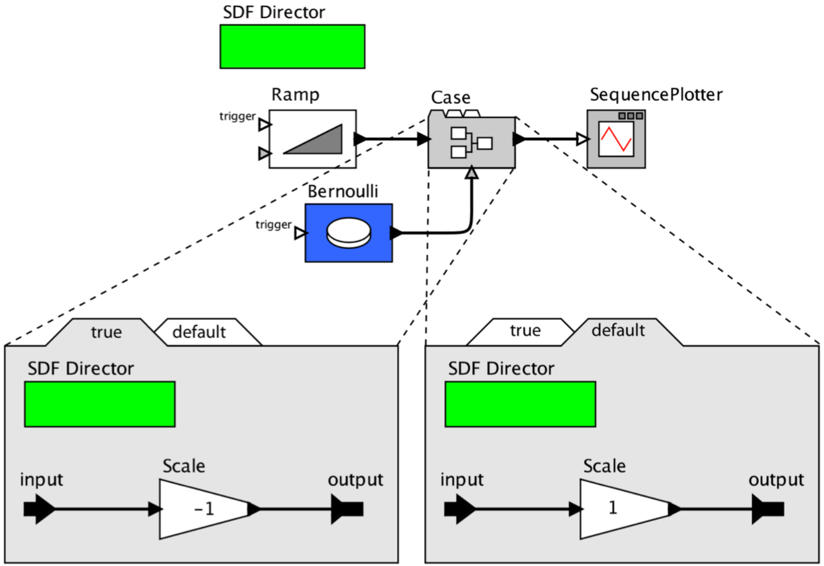


DDF

- It may not be possible to determine a schedule with bounded buffers
- Not always possible to ensure that the model will not deadlock
- Buck (1993) showed that bounded buffers and deadlock are undecidable for DDF models.
- DDF models are not as readily analyzed.
- Structured dataflow & higher order actors are used

Structured Dataflow

- Higher order actor: combine multiple actors as components
- Example Case: 2 sub-models
  - true that contains a Scale actor with a parameter of −1, and
  - default that contains a Scale actor with a parameter of 1.
  - When the control input to the Case actor is true, the true refinement executes one iteration. For any other control input, the default refinement executes.



Actor Model Implementation

- Multiple clocks
- Multiple domains
- Buffer: Queue
- Message: Interprocess communication