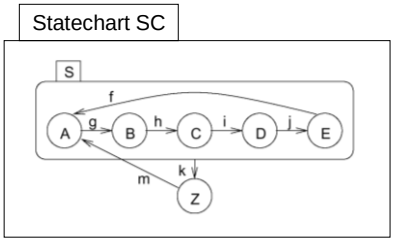
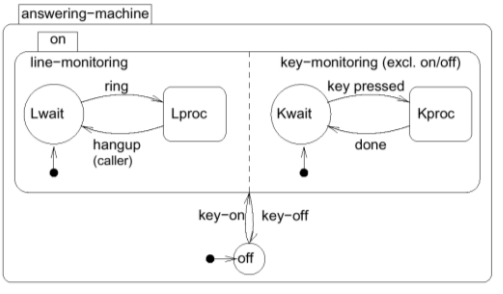
Embedded Systems 4

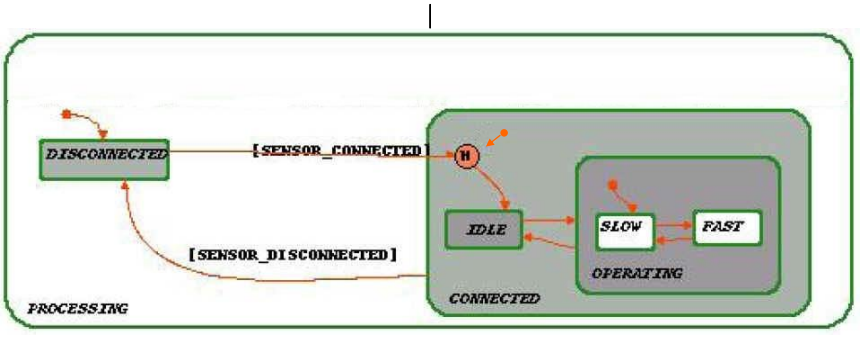
REVIEW: StateCharts

- Hierarchy
- Concurrency



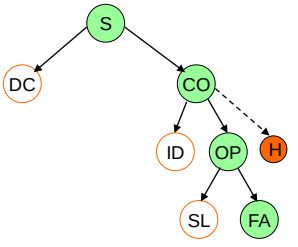


REVIEW: History and deep history



Default states

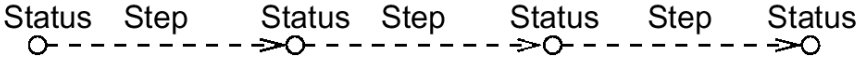
Active states



History connectors remember states **at the same level** as the history connector!

STATEMATE Semantics of StateCharts

- Execution of a StateChart model consists of a sequence of **steps**
- A step leads from one **status** to another



- One step:
  - Given:
    - Current system status s_i
    - Current time t
    - External changes Δ
  - Find:
    - New status s_{i+1}

Status of the system

The current status of the system is given by

- set of active states
- current values of variables
- the generated events from previous step
- the values of the history connectors
- set of all timeout events <tm(e, d), n> in the state chart with „emission times“ n (times n are initially set to 1)
- set of currently scheduled actions <sc(a, d), n> with their times n

External changes

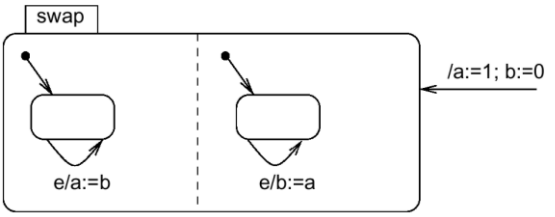
- External data and external events constitute the interface between system and environment.
- The environment provides external events at certain times and changes external data at certain times.
- External events not yet seen in the previous step and changes of external data not seen in the previous step are called **external changes** for the current step.

StateMate Semantics

Three phases

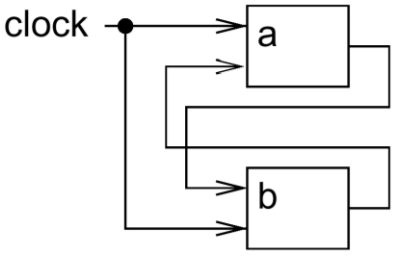
1. Effect of external changes on events and conditions is evaluated
2. The set of transitions to be made in the current step and right-hand side of assignments are computed
3. Transitions become effective, variables obtain new values

Example



- In part 2, variables a and b are assigned to temporary variables. In part 3, these are assigned to a and b. As a result, variables a and b are swapped.
- Without this separation, executing the left state first would assign the old value of b (=0) to a and b. Executing the right state first would assign the old value of a (=1) to a and b. The execution of parallel assignment would be nondeterministic.

Reflects model of clocked hardware



- In an actual clocked (synchronous) hardware system, both registers would be swapped as well.

Same separation into phases found in other languages as well, especially those that are intended to model hardware.

Other semantics

- Several other specification languages for hierarchical state machines (e.g., UML) do not include the three simulation phases
- Corresponds more to a software point of view without synchronous clocks.
- Some simulation tools can be run with optional multi-phased simulation.

Broadcast mechanism

- Values of variables are visible to all parts of the StateChart model.
- New values become effective in part 3 of the execution stage for the current step and are obtained by all parts of the model in the following step.

☞ StateCharts implicitly assumes a **broadcast** mechanism for variables.

☞ StateCharts is appropriate for local control systems (☺), but not for distributed applications for which updating variables might take some time (☹).

Time models

- External events and external changes of variables are associated with physical times.

- But how does time proceed internally?
- How many steps are performed before external changes are evaluated?

The synchronous time model

- A single step every time unit.
- If the current step is executed at time *t*, then the next step is executed at time *t+1*.

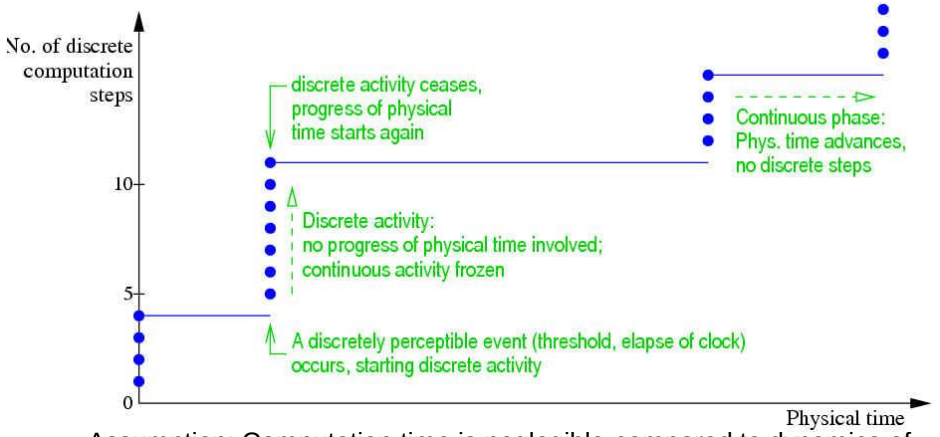
- Events and variable changes are communicated between different states during one time unit.
- External changes are only accumulated during one time unit.

The super-step time model (1)

- A step of the statechart does not need time.
- Super-steps are performed:
  - A super-step is a sequence of steps.
  - A super-step terminates when the status of the system is stable.
  - During a super-step the time does not proceed and thus external changes are not considered.
- After a super-step, physical time restarts running, i.e. activity of the environment will be possible again.
- The computation of the statechart is resumed when
  - external changes enable transitions in the statechart
  - Timeout events enable transitions of the statechart

The super-step time model (2)

- Two-dimensional time:



- Assumption: Computation time is neglegible compared to dynamics of the environment.

The super-step time model (3)

- During one super-step the number of communications between different states is not restricted. All communications are assumed to be performed in zero time.
- Simplified model for reality.
- Can only be realistic, if
  - Discrete computations are fast compared to dynamics of the environment.
  - Discrete computations will be stable after a restricted number of steps.
- Timeout events can reactivate a statechart

  ⇒ Possible to specify statecharts which permit progress of physical time after a limited number of steps and reactivate themselves via timeout events

Evaluation of StateCharts (1)

Pros:

- Hierarchy allows arbitrary nesting of AND- and OR-superstates.
- Formal semantics (defined in a follow-up paper to original paper).
- Large number of commercial simulation tools available (StateMate, StateFlow, BetterState, ...)
- Available „back-ends“ translate StateCharts into C or VHDL, thus enabling software or hardware implementations.

Evaluation of StateCharts (2)

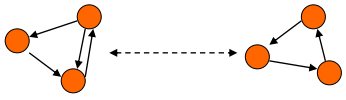
Cons:

- Generated C programs frequently inefficient,
- Not useful for distributed applications,
- No program constructs,
- No description of non-functional behavior,
- No object-orientation,
- No description of structural hierarchy.

Some general properties of languages
1. Synchronous vs. asynchronous languages

- Description of several (concurrent) processes in many languages non-deterministic:
  The order in which executable tasks are executed is not specified (may affect result).
- Synchronous languages: based on automata models. They describe concurrently operating automata. When automata are composed in parallel, a transition of the product is made of the "simultaneous" transitions of all of them.
- Synchronous languages implicitly assume the presence of a (global) clock. Each clock tick, all inputs are considered, new outputs and states are calculated and then the transitions are made.

Some general properties of languages
1. Synchronous vs. asynchronous languages



- This requires a broadcast mechanism for all parts of the model.
- Idealistic view of concurrency.
- Has the advantage of guaranteeing deterministic behavior.

- Statechart steps work synchronously.
  - Broadcast of events and variable changes during each step.
  - StateCharts are deterministic, if priority rules are introduced for transitions enabled at the same time.

Some general properties of languages
2. Properties of processes

- **Number of processes**
  static (suitable for hardware);
  dynamic (dynamically changed hardware architecture?)
- Nested **declaration of processes**
  or all declared at the same level
- ☞ **StateCharts comprises a static number of processes and nested declaration of processes.**

Some general properties of languages
3. Communication paradigms

- **Message passing**
  - **Asynchronous message passing = non-blocking communication**
    Sender does not have to wait until message has arrived; potential problem: buffer overflow
  - **Synchronous message passing = blocking communication, *rendez-vous*-based communication**
    Sender will wait until receiver is ready for receiving message ("point of communication")
  - **Extended *rendez-vous***
    Explicit acknowledge from receiver required. Receiver can do checking before sending acknowledgement.

Some general properties of languages
3. Communication paradigms

- **Shared memory**
  Variables accessible to several tasks
  - Problem: Concurrent write.
  - Critical sections = sections at which exclusive access to some resource *r* must be guaranteed.

☞ **StateCharts uses shared memory for communication between processes.**

Some general properties of languages
4. Specifying timing

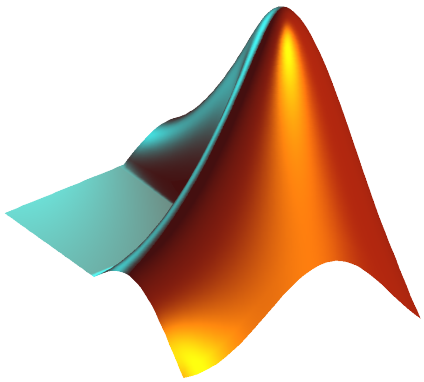
4 types of timing specs required [Burns, 1990]:

- **Measure elapsed time**
  Check, how much time has elapsed since last call
- **Means for delaying processes**
- **Possibility to specify timeouts**
  We would like to be in a certain state only a certain maximum amount of time.
- **Methods for specifying deadlines**
  With current languages not available or specified in separate control file.

☞ **StateCharts comprises a mechanism for specifying timeouts. Other types of timing specs are not supported.**

Matlab, Simulink & StateFlow

MATLAB - Matrix Laboratory

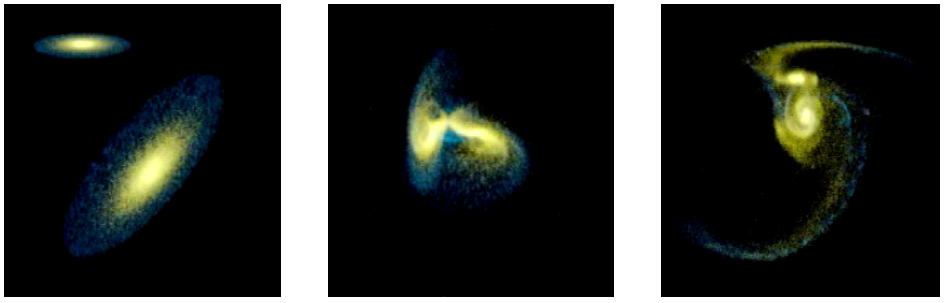


- Produced by Mathworks
- Used for simulation and numerical computation
- No (Maple-like) symbolical solving
- Industrial standard tool for developing embedded systems

MATLAB Structure

- MATLAB core: IDE for the MATLAB language
- Simulink: Graphical environment for continuous simulation
- Stateflow: Statecharts for Simulink
- Many other add-ons available...

Numerical Computing



- Some problems do not have a closed-form solution
- Approximative numerical solutions often suffice
- Simulation of the physical world

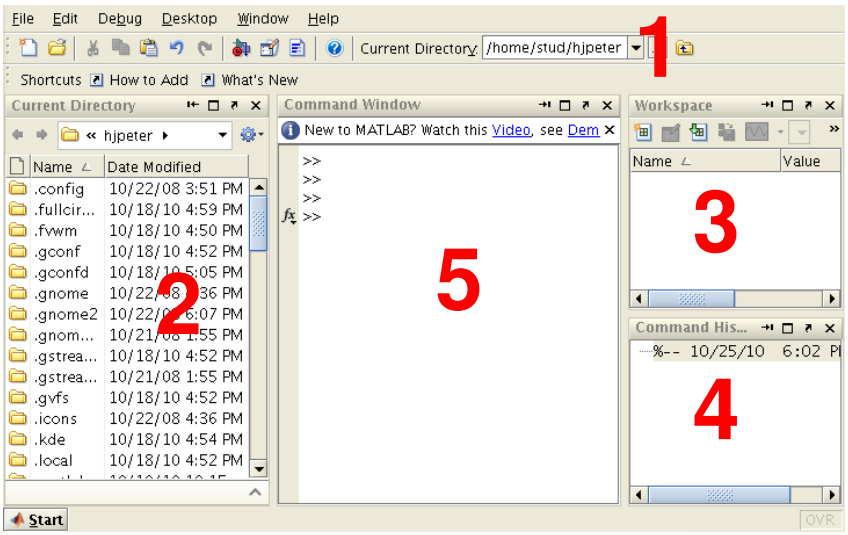
Starting MATLAB

1. `ssh -Y appsrv1.studcs.uni-saarland.de`
2. `matlab`

alternatively:

1. `http://sunray1.studcs.uni-sb.de`
2. Log in
3. Click on `MatLab`

MATLAB IDE



1. Current directory
2. Directory explorer
3. Workspace
4. Command history
5. Command window

The MATLAB Language

- Simplified C-like syntax
- Case sensitive
- Interactive shell: command window
- User defined functions: m-files
- Many built-in commands:
  - `lookfor <keyword>`
  - `help <function>`
  - `sprintf (<format str>, v1, v2, ...)`
  - `disp (<object>)`
  - `plot (Y)`
  - `plot (X, Y)`
  - `...`

Variables

- Each numerical variable is a matrix
- Scalars = 1 $\times$ 1 matrices
- No explicit declarations / dynamic typing
- Polymorphism
- Removing variables:
  - `clear <variable>`
  - `clear`

Working with Matrices

- `a = 4`
- `b = [4 8 15 ; 16 23 42 ; 1 2 3]`
- `c = b'`
- `d = ones(4)`
- `e = eye(3)`
- `f = b*b`
- `g = b.*b`
- `h = 0:10`
- `i = 0:0.01:2*pi`

Script Files

- m-files
- Must be located in
  - the current directory or
  - the global search path
- Can be executed from the command window
- Can also define functions

Control Structures

- Conditional

```
if <cond>
  <statements>
[else
  <statements>]
end
```

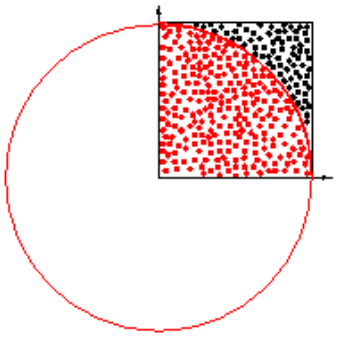
- While loop

```
while <cond>
  <statements>
end
```

- For loop

```
for v = <from>:[<step>:]<to>
  <statements>
end
```

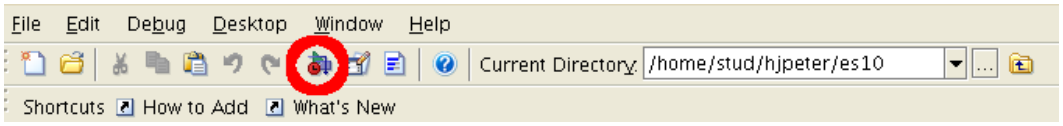
Example: Computing π

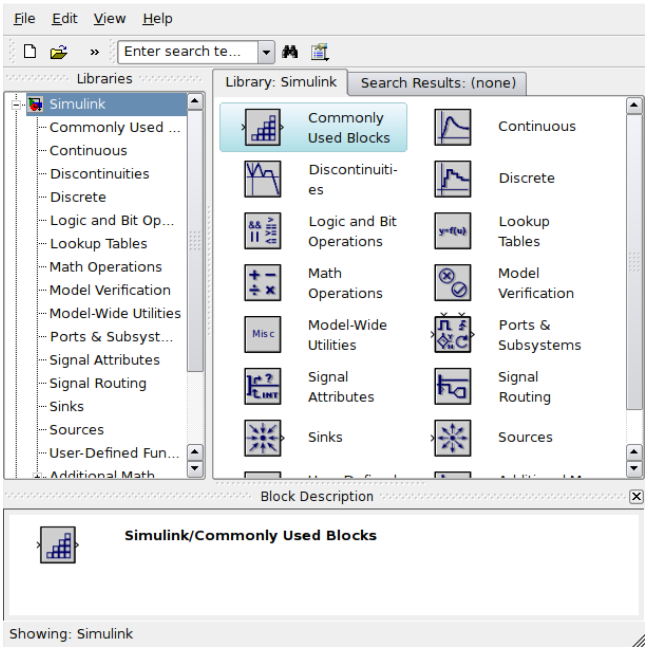


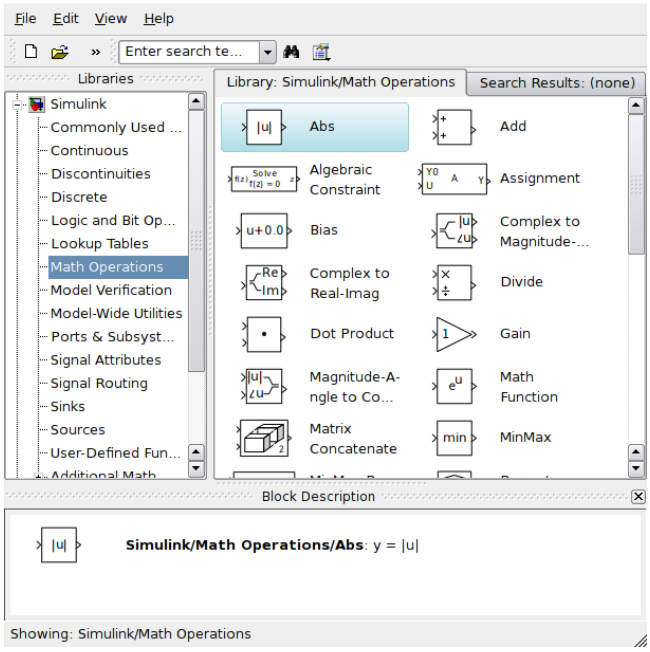
- Monte Carlo method for computing π

$$\frac{\text{points inside}}{\text{points total}} \approx \frac{\pi}{4}$$

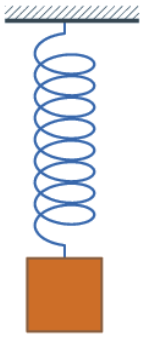
Simulink







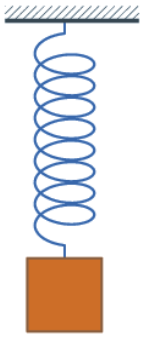
Harmonic Oscillator



Hooke's Law: $F = -ky$

- F: restoring force
- k: positive constant that characterizes the oscillator
- y: amplitude or displacement

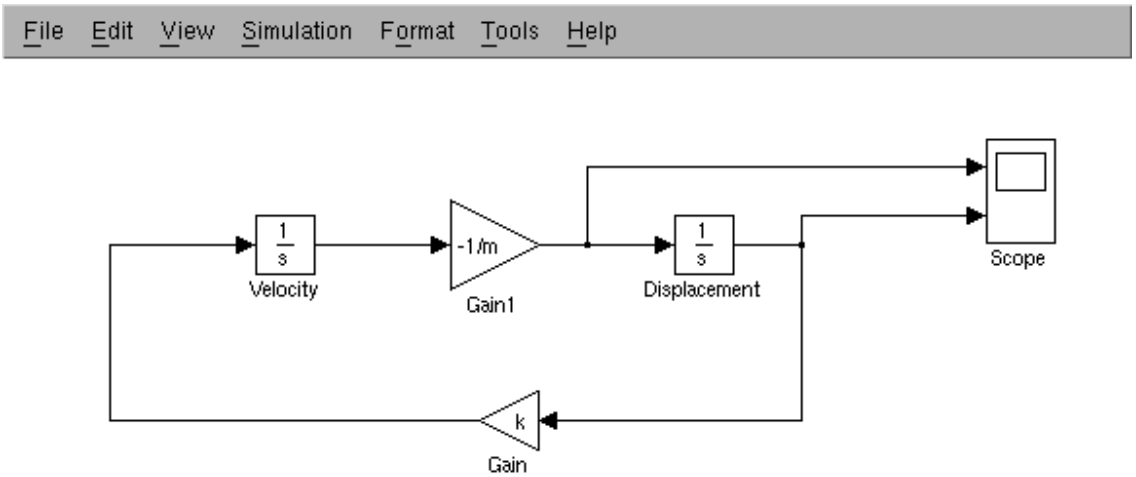
Harmonic Oscillator (2)



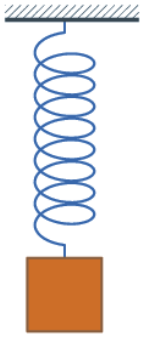
- m: mass constant
- k: spring constant
- y_0: initial displacement
- y: current displacement
- $v = \dot{y}$: current velocity
- $a = \dot{v} = \ddot{y}$: current acceleration

$$
\begin{aligned}
F = ma &= -ky \\
\Leftrightarrow \quad ma + ky &= 0 \\
\Leftrightarrow \quad m\ddot{y} + ky &= 0 \\
\Leftrightarrow \quad m\dot{v} + ky &= 0
\end{aligned}
$$

Harmonic Oscillator in Simulink



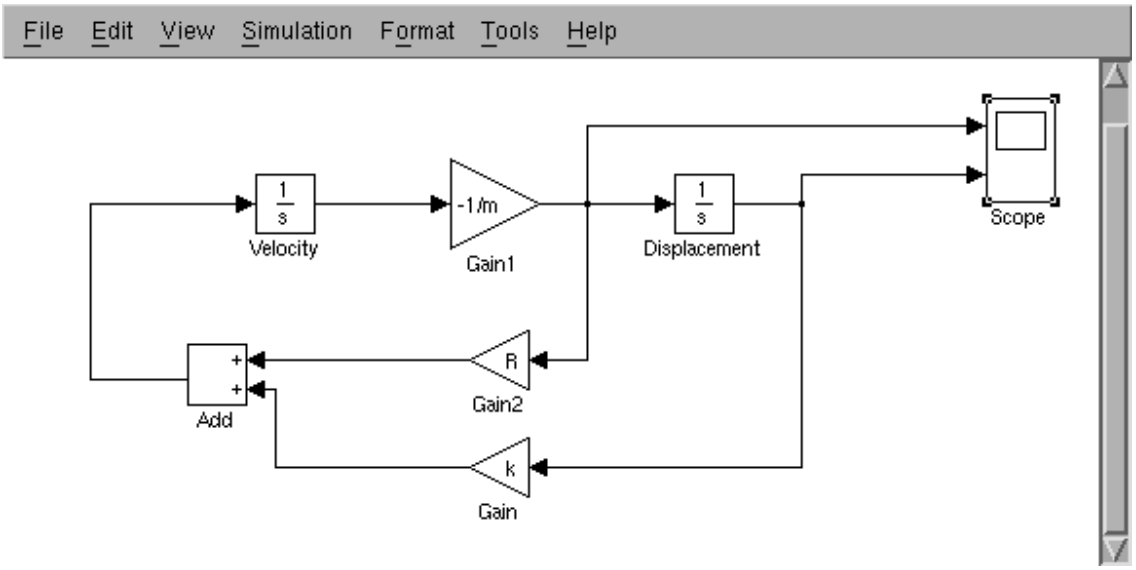
Damped Harmonic Oscillator



- m = mass constant
- R = damper constant
- k: spring constant
- y_0: initial displacement
- y: current displacement
- $v = \dot{y}$: current velocity
- $a = \dot{v} = \ddot{y}$: current acceleration

$$m\ddot{y} + R\dot{y} + ky = 0$$
$$\Leftrightarrow \quad m\dot{v} + Rv + ky = 0$$

Damped Harmonic Oscillator in Simulink



Semantics: Statemate vs. Stateflow

Standard (Statemate)

- Any finite number of active events.
- Emitted events are collected and then passed to the entire chart.

Stateflow

- At most one active event.
- Emitted events are immediately passed to the receiver.

Semantics: Statemate vs. Stateflow (2)

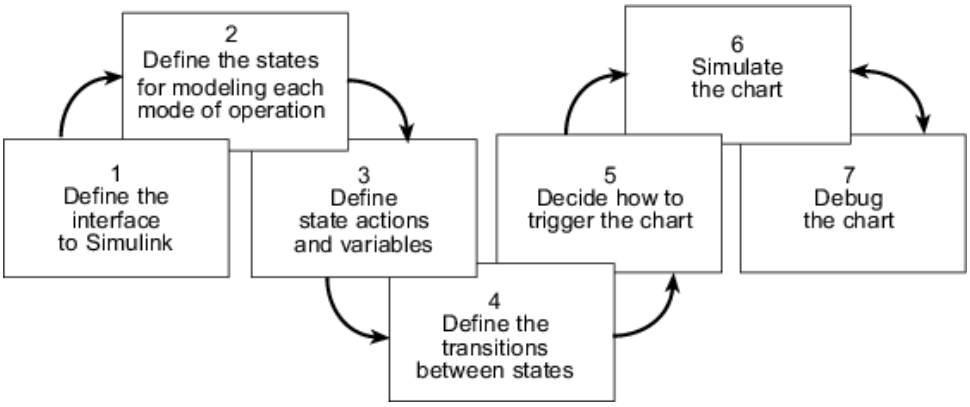
Standard (Statemate)

- Non-determinism is allowed.
- Synchronous execution of AND-states.
- Variable changes at the end of the step.

Stateflow

- Non-determinism is not allowed.
- Sequential execution of AND-states.
- Immediate variable changes.

Simulink/Stateflow Development

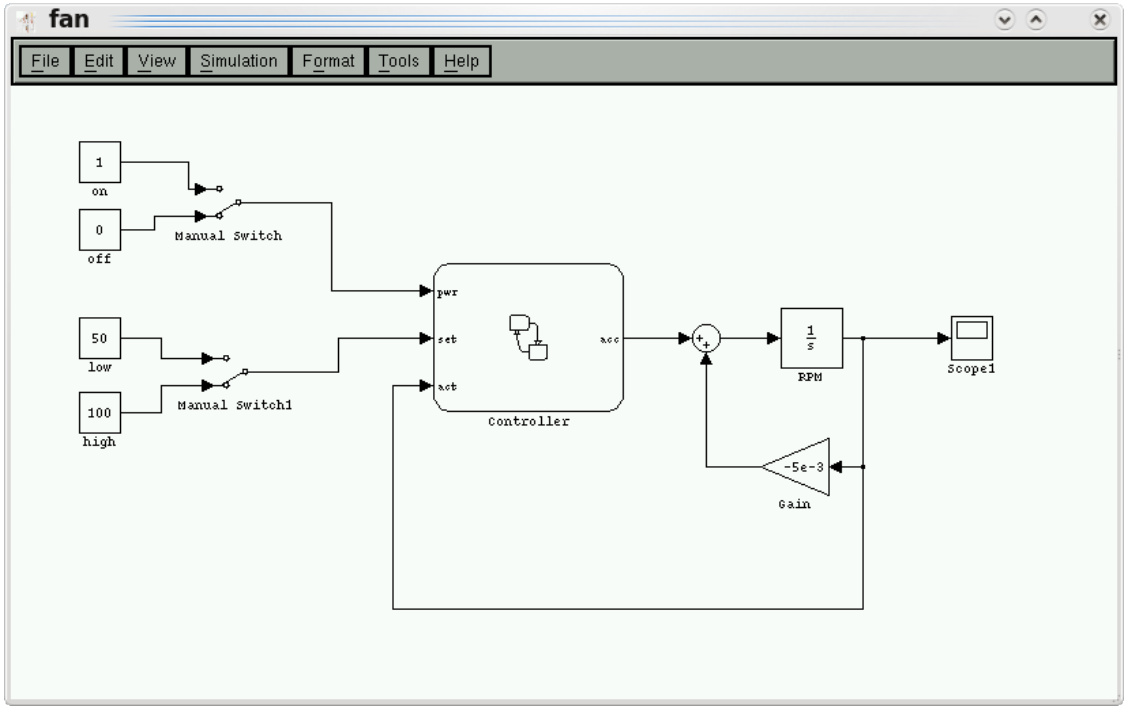


Example: Fan Controller

Specification

- Turn on / off
- Two modes: low / high
- Can only accelerate
- Damped
- Feedback

Fan Controller: Simulink Model



Fan Controller: Statechart