



Concurrent Models of Computation for Embedded Software

Edward A. Lee

Professor, UC Berkeley
EECS 290n – Advanced Topics in Systems Theory
Fall, 2004

Copyright © 2004, Edward A. Lee, All rights reserved

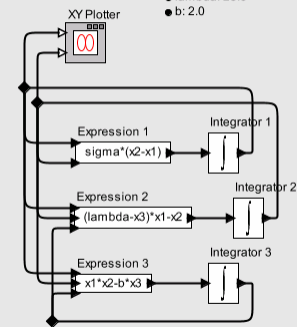
Lecture 20: Continuous-Time Models

Basic Continuous-Time Modeling

Continuous-Time (CT) Solver

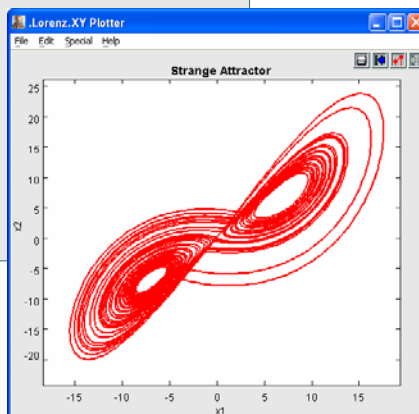


- sigma: 10.0
- lambda: 25.0
- b: 2.0



This model shows a nonlinear feedback system that exhibits chaotic behavior. It is modeled in continuous time. The CT director uses a sophisticated ordinary differential equation solver to execute the model. This particular model is known as a Lorenz attractor.

A basic continuous-time model describes an ordinary differential equation (ODE).



Basic Continuous-Time Modeling

Continuous-Time (CT) Solver

Author: Jie Liu

This model shows a nonlinear feedback system that exhibits chaotic behavior. It is modeled in continuous time. The CT director uses a sophisticated ordinary differential equation solver to execute the model. This particular model is known as a Lorenz attractor.

A basic continuous-time model describes an ordinary differential equation (ODE).

$$\dot{x}(t) = f(x(t), t)$$

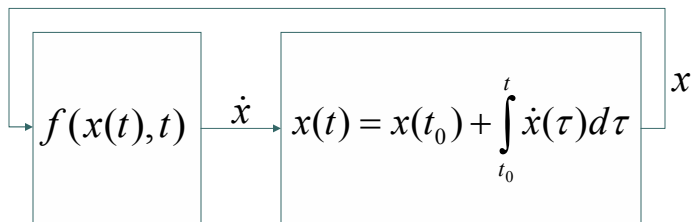
$$x(t) = x(t_0) + \int_{t_0}^t \dot{x}(\tau) d\tau$$

Lee 20: 3

Basic Continuous-Time Modeling

The state trajectory is modeled as a vector function of time,

$$x: T \rightarrow \mathbb{R}^n \quad T = [t_0, \infty) \subset \mathbb{R}$$



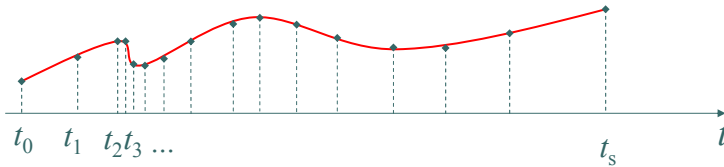
$$\dot{x}(t) = f(x(t), t)$$

$$f: \mathbb{R}^m \times T \rightarrow \mathbb{R}^m$$

ODE Solvers

Numerical solution approximates the state trajectory of the ODE by estimating its value at discrete time points:

$$\{t_0, t_1, \dots\} \subset T$$



Reasonable choices for these points depend on the function f .

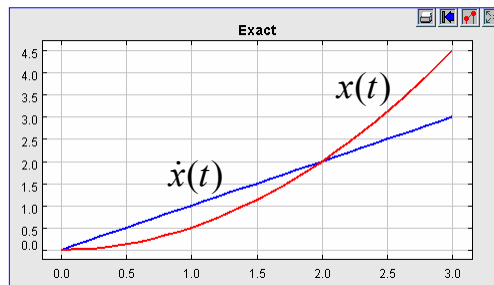
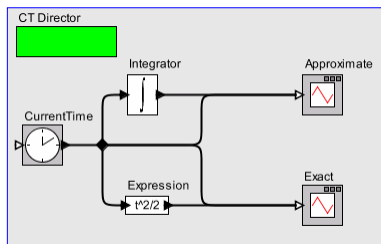
Using such solvers, signals are discrete-event signals.

Lee 20: 5

Simple Example

This simple example integrates a ramp, generated by the CurrentTime actor. In this case, it is easy to find a closed form solution,

$$\dot{x}(t) = t \Rightarrow x(t) = t^2 / 2$$

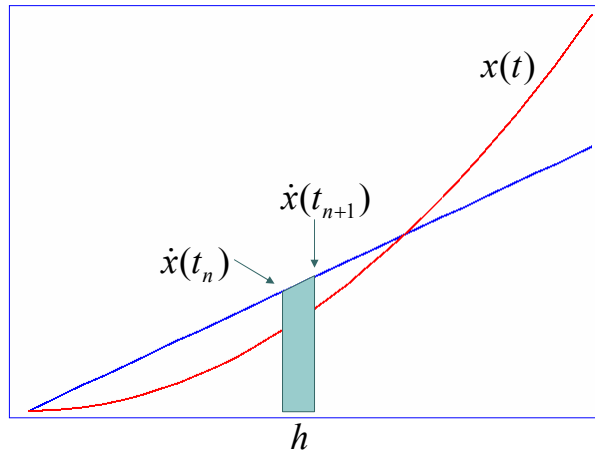


Lee 20: 6

Trapezoidal Method

Classical method estimates the area under the curve by calculating the area of trapezoids.

However, with this method, an integrator is only causal, not strictly causal or delta causal.

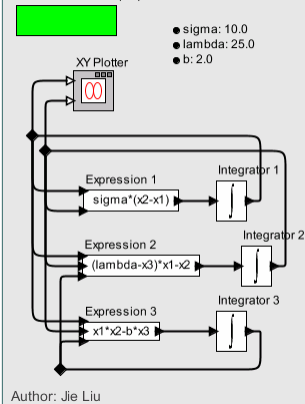


$$x(t_{n+1}) = x(t_n) + h(\dot{x}(t_n) + \dot{x}(t_{n+1}))/2$$

Lee 20: 7

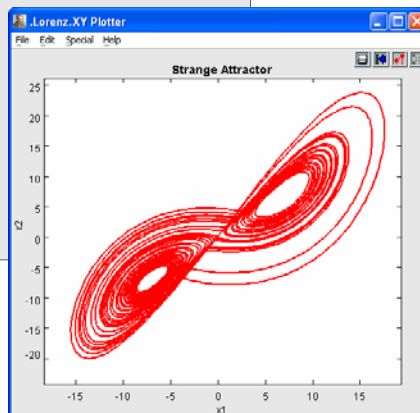
Trapezoidal Method is Problematic with Feedback

Continuous-Time (CT) Solver



This model shows a nonlinear feedback system that exhibits chaotic behavior. It is modeled in continuous time. The CT director uses a sophisticated ordinary differential equation solver to execute the model. This particular model is known as a Lorenz attractor.

We have no assurance of a unique fixed point, nor a method for constructing it.



Lee 20: 8

Forward Euler Solver

Given $x(t_n)$ and a time increment h , calculate:

$$t_{n+1} = t_n + h$$

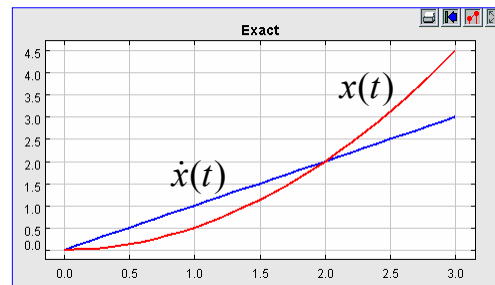
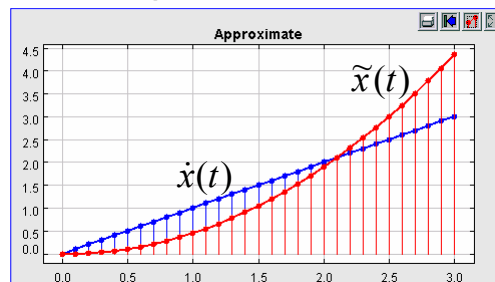
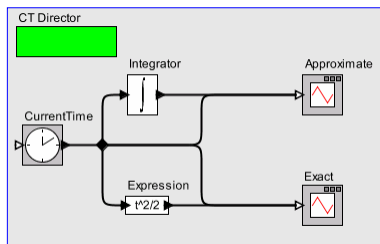
$$x(t_{n+1}) = x(t_n) + h f(x(t_n), t_n)$$

This method is strictly causal, or, with a lower bound on the step size h , delta causal. It can be used in feedback systems. The solution is unique and non-Zeno.

Lee 20: 9

Forward Euler on Simple Example

In this case, we have used a fixed step size $h = 0.1$. The result is close, but diverges over time.



Lee 20: 10

Runge-Kutta 2-3 Solver (RK2-3)

Given $x(t_n)$ and a time increment h , calculate

$$\begin{aligned}
 K_0 &= f(x(t_n), t_n) && \leftarrow \dot{x}(t_n) \\
 K_1 &= f(x(t_n) + 0.5hK_0, t_n + 0.5h) && \leftarrow \begin{array}{l} \text{estimate of} \\ \dot{x}(t_n + 0.5h) \end{array} \\
 K_2 &= f(x(t_n) + 0.75hK_1, t_n + 0.75h) && \leftarrow \begin{array}{l} \text{estimate of} \\ \dot{x}(t_n + 0.75h) \end{array}
 \end{aligned}$$

then let

$$t_{n+1} = t_n + h$$

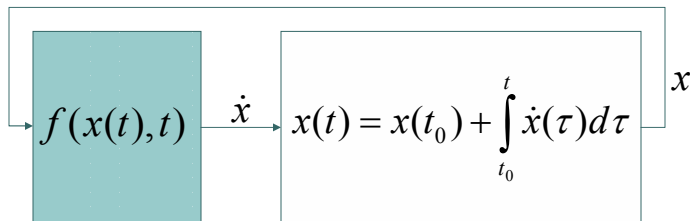
$$x(t_{n+1}) = x(t_n) + (2/9)hK_0 + (3/9)hK_1 + (4/9)hK_2$$

Note that this is strictly (delta) causal, but requires three evaluations of f at three different times with three different inputs.

Lee 20: 11

Operational Requirements

In a software system, the blue box below can be specified by a program that, given $x(t)$ and t calculates $f(x(t), t)$. But this requires that the program be functional (have no side effects).



$$\dot{x}(t) = f(x(t), t)$$

$$f : R^m \times T \rightarrow R^m$$

Lee 20: 12

Adjusting the Time Steps

For time step given by $t_{n+1} = t_n + h$, let

$$K_3 = f(x(t_{n+1}), t_{n+1})$$

$$\varepsilon = h((-5/72)K_0 + (1/12)K_1 + (1/9)K_2 + (-1/8)K_3)$$

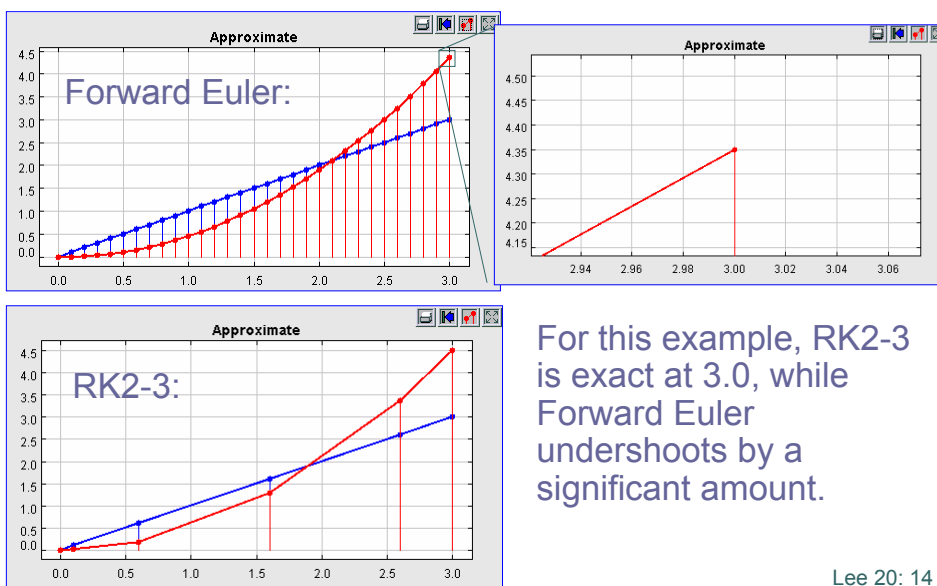
If ε is less than the “error tolerance” e , then the step is deemed “successful” and the next time step is estimated at:

$$h' = 0.8 \sqrt[3]{e/\varepsilon}$$

If ε is greater than the “error tolerance,” then the time step h is reduced and the whole thing is tried again.

Lee 20: 13

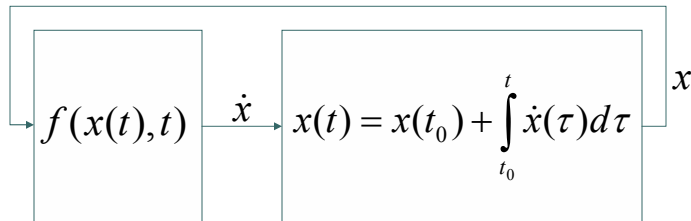
Comparing RK2-3 to Forward Euler



Lee 20: 14

Accumulating Errors

In feedback systems, the errors of FE accumulate more rapidly than those of RK2-3.

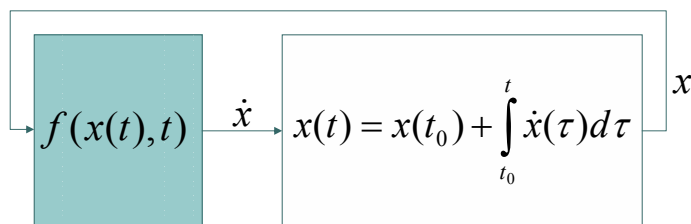


$$\dot{x}(t) = f(x(t), t)$$

$$f : R^m \times T \rightarrow R^m$$

Lee 20: 15

Examining This Computationally



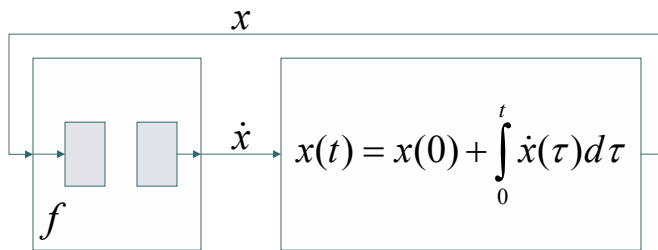
At each discrete time t_n , given a time increment $t_{n+1} = t_n + h$, we can estimate $x(t_{n+1})$ by repeatedly evaluating f with different values for the arguments. We may then decide that h is too large and reduce it and redo the process.

Lee 20: 16

How General Is This Model?

Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines?



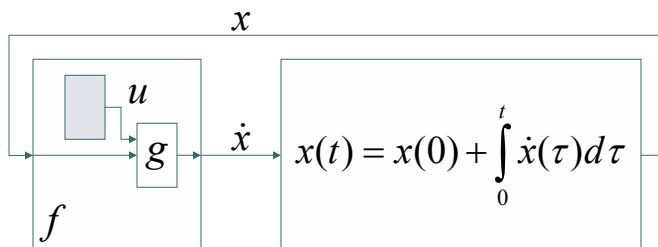
$$\dot{x}(t) = f(x(t), t)$$

Lee 20: 17

How General Is This Model?

Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines?



$$\dot{x}(t) = f(x(t), t) = g(u(t), x(t), t)$$

Lee 20: 18

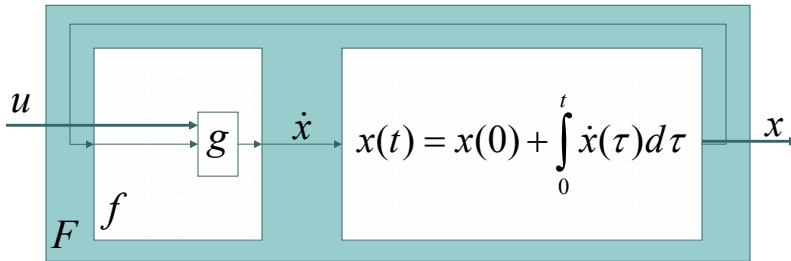
The Model Itself as a Function

Note that the model function has the form:

$$F : [T \rightarrow R^m] \rightarrow [T \rightarrow R^m]$$

Which does not match the form:

$$f : R^m \times T \rightarrow R^m$$



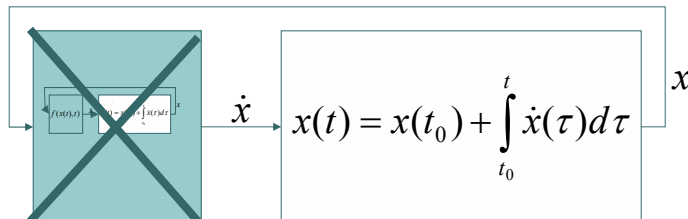
(This assumes certain technical requirements on f and u that ensure existence and uniqueness of the solution.)

Lee 20: 19

Consequently, the Model is Not Compositional!

In general, the behavior of the inside dynamical system cannot be given by a function of form:

$$f : R^m \times T \rightarrow R^m$$



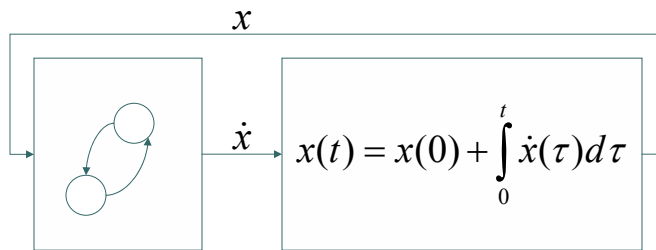
To see this, just note that the output must depend only on the current value of the input and the time to conform with this form.

Lee 20: 20

So How General Is This Model?

Does it handle:

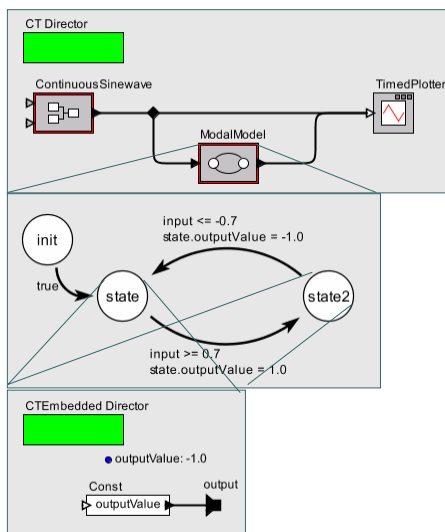
- External inputs?
- Systems without feedback?
- State machines? No... The model needs work...



Since this model is itself a state machine, the inability to put a state machine in the left box explains the lack of composability.

Lee 20: 21

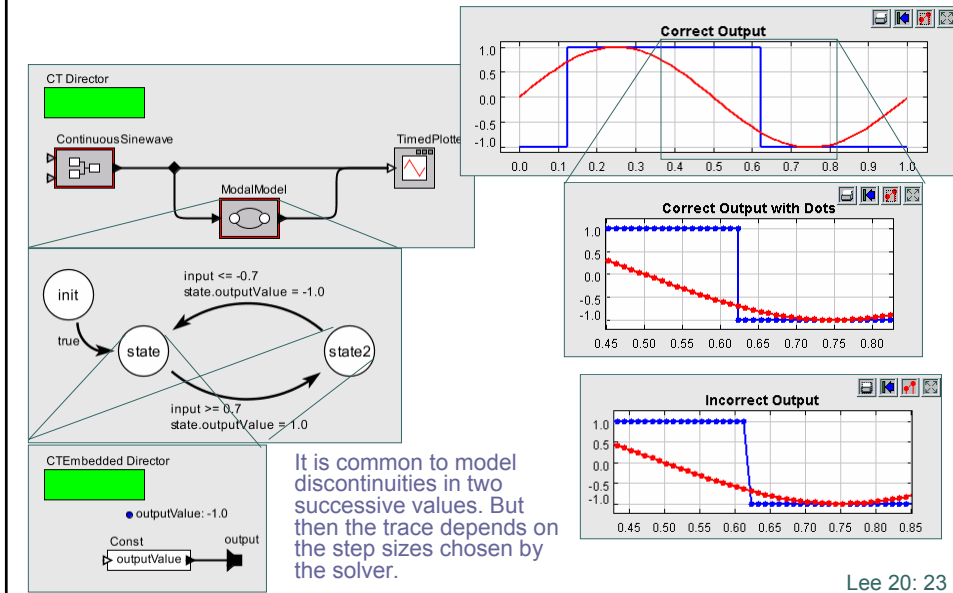
Start with Simple State Machines Hysteresis Example



This model shows the use of a two-state FSM to model hysteresis. Semantically, the output of the ModalModel block is discontinuous. If transitions take zero time, this is modeled as a signal that has two values *at the same time*, and *in a particular order*.

Lee 20: 22

Hysteresis Example



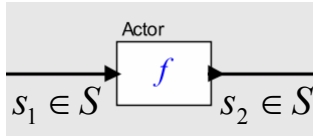
Requirements

The hysteresis example illustrates two requirements:

- A signal may have more than one value at a particular time, and the values it has have an order.
- The times at which the solver evaluates signals must precisely include the times at which interesting events happen, like a guard becoming true.

Both Requirements Are Dealt With By an Abstract Semantics

Previously

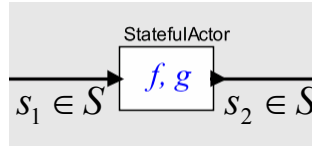


$$S = [T \rightarrow R]$$

$$f : R^m \times T \rightarrow R^m$$

$$\forall t \in T, \quad s_2(t) = f(s_1(t), t)$$

Now we need:



$$S = [T \times N \rightarrow R]$$

$$f : \Sigma \times R^m \times T \rightarrow R^m$$

$$g : \Sigma \times R^m \times T \rightarrow \Sigma$$

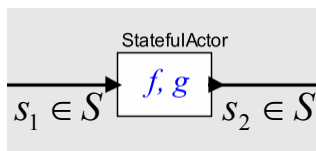
state space

$$\forall (t, n) \in T \times N, \quad s_2(t, n) = ?$$

The new function f gives outputs in terms of inputs and the current state. The function g updates the state at the specified time.

Lee 20: 25

Abstract Semantics



$$S = [T \times N \rightarrow R]$$

$$f : \Sigma \times R^m \times T \rightarrow R^m$$

$$g : \Sigma \times R^m \times T \rightarrow \Sigma$$

At each $t \in T$ the output is a *sequence* of one or more values where given the current state $\sigma(t) \in \Sigma$ and the input $s_1(t)$ we evaluate the procedure

$$s_2(t, 0) = f(\sigma(t), s_1(t, 0), t)$$

$$\sigma_1(t) = g(\sigma(t), s_1(t, 0), t)$$

$$s_2(t, 1) = f(\sigma_1(t), s_1(t, 1), t)$$

$$\sigma_2(t) = g(\sigma_1(t), s_1(t, 1), t)$$

...

until the state no longer changes. We use the final state on any evaluation at later times.

This deals with the first requirement.

Lee 20: 26

Conclusion and Open Issues

- The basic model assumed by many ODE solvers does not lend itself easily to reasonable software architectures.
- A generalized model supports signals with multiple, ordered values at a time value.
- An abstract semantics for components can be defined that supports these multiple values and also is amenable to reasonable software realizations.
- Compositionality remains an open issue.