Continuous + Discrete = Hybrid (1)

Mixture of ... continuous & discrete inputs, outputs, states, dynamics

$\mathbb{R}^n + Q \approx \{1, 2, \ldots, N\} = \mathbb{R}^n \times Q$
Continuous + Discrete = Hybrid (2)

Mixture of ... differential equations and discrete events / switching
Continuous + Discrete = Hybrid (3)

Mixture of ... continuous physical process with finite-state logic

Force-guided robotic assembly [Branicky-Chhatpar, HSCC, 2002]
Continuous + Discrete = Hybrid (4)

Mixture of ... control theory and computer science

Autonomous vehicle DEXTER [urbanchallenge.case.edu]
Hybrid Systems All Around Us

They drive on our streets, work in our factories, fly in our skies, ...
Networked Control Systems (1)

Sensors, actuators, and controllers connected over a network ... with feedback loops controlling physical systems closed among them

- continuous plants
- asynchronous or *event-driven* data transmission
  - sampling, varying transmission delay, packet loss
- discrete implementation of network/protocols
  - data packets, queuing, routing, scheduling, etc.
Networked Control Systems (2)

Co-simulation and co-design [Branicky-Liberatore-Phillips, ACC, 2003]
Other Examples

- systems with relays, switches, and hysteresis
- computer disk drives
- constrained robotic systems (locomotion, assembly, etc.)
- vehicle powertrains, transmissions, stepper motors
- mode-switched flight control, vehicle management systems
- automated highway systems (AHS)
- multi-vehicle formations and coordination
- power electronics
- analog/digital circuit co-design and verification
- biological applications
Systems with Switches and Relays

HVAC control with a thermostat:

$$\dot{x} = f(x, H(x - x_0), u)$$

- $x$, room temperature
- $x_0$, desired temperature
- $f$, dynamics of temperature
- $u$, control signal (e.g., the fuel burn rate)

Hysteresis Function, $H$  

Associated Finite Automaton
Hard Disk Drive

HS for main hard disk drive functionality [Gollu-Varaiya, CDC, 1989]
Finite state machine for disk drive activities [Gollu-Varaiya, 1989]
Fast Run [Mayer, 1886]

\[ \ddot{x} = -g \]
Flight

\[ \ddot{x} = -g + \frac{n}{x} \]
Compression

\[ \ddot{x} = -g + \gamma \]
Thrust

\[ \ddot{x} = -g + \frac{\text{ret}}{x} \]
Decompression

Hopping robot: dynamic phases and controller [Raibert, 1986]
Vehicle Powertrains / Cruise Control

<table>
<thead>
<tr>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throttle</td>
<td>Gear Position</td>
</tr>
<tr>
<td>Engine RPM</td>
<td>Cylinder Phases</td>
</tr>
<tr>
<td>Fuel/Air Mixture</td>
<td>Cylinder Firings</td>
</tr>
<tr>
<td>Belts, Cams</td>
<td>Microprocessors</td>
</tr>
<tr>
<td>Elevation</td>
<td>Road Condition</td>
</tr>
</tbody>
</table>

$I, O$ are discrete (i.e., countable) sets of symbols
$U, Y$ are continuums
Flight Vehicle Mgmt. Systems

[George Meyer, Plenary Lecture, CDC, 1994]
2. The DC/DC buck converter

An elementary buck converter can be implemented in a circuit as represented in Fig. 1. This consists of a basic RLC circuit, a diode and a switching element. The aim of the circuit is to maintain a desired voltage, lower than that provided by the input battery $E$, across the load resistance $R$. This can be achieved by appropriately turning on and off the switch $S$, so that the circuit is repeatedly forced by the external forcing voltage source $E$. The switching action is usually implemented through a Pulse

![Diagram of the DC/DC buck converter](image)

Figure 1. The DC/DC buck converter.

Width Modulated (PWM) feedback law. Namely a linear combination of the two system states, $v_c(t) = g_1 i(t) + g_2 v(t)$ (in the present work $g_1 = 1, g_2 = 0$), is compared with a given asymmetric sawtooth (ramp) signal of assigned period $T$ as shown in Fig. 2. The circuit switch, $S$, is then turned on whenever the ramp signal becomes greater than the combination of the two states and turned off when the ramp signal falls below this combination. Hence, a switching occurs whenever $v_c(t)$ crosses the ramp (either within the ramp period or at $t = nT$, $n = 1, 2, ...$).

2.1. System Modelling

Whether the switch is on or off, the buck converter can always be described as a second order linear system, whose states are the voltage $v$ across the capacitor, and the current $i$ along the inductor. The equations take the form

$$\frac{dv}{dt} = -v/RC + i/C,$$

$$\frac{di}{dt} = -i/L + \begin{cases} 0 & \text{if } v > v_r(t) = \gamma + \eta(t \mod T) \\ E/L & \text{if } v < v_r(t) = \gamma + \eta(t \mod T) \end{cases} \quad \text{(ON)}$$

![Graph showing the PWM operating conditions](image)

Figure 2. Standard DC/DC converter PWM operating conditions: one impact per period.
As a baby monitor, in this artist's rendering of a production version of the Gecko, the video decoder task [far left and bottom on circuit board] that is displaying a full-screen image of a baby [on handheld, below] runs on one of three "tiles" on an FPGA [on circuit board] to provide the best resolution at the highest frame rate.

Places traded: The game runs at maximum quality on the FPGA [on circuit board, far right] and takes up most of the display area. But the baby monitor still functions. The video decoder restarts as software on the processor [below, second from right] and the baby is displayed smaller than before, at lower resolution, and at a slower frame rate, in a corner of the screen [below].

Playing games: To switch to a game, the game decoder stored as software in the processor's flash memory [below, second from left] swaps places with the video decoder on the FPGA.

The Internet. Tasks run in the tiles and communicate with other tasks on other tiles, on the microprocessor, or on an ASIC, by sending messages via the network. Since the interconnect network and the tile interfaces are fixed, tasks can be dynamically created and deleted without affecting those running in other tiles.

Keeping tabs on tasks:
The key to a great user experience with a device like Gecko lies in making the transitions from one function to another as smooth as possible. This responsibility falls to the real-time operating system, which manages all these complex transitions.

We based Gecko's Operating System for Reconfigurable Systems (OS4RS) on a real-time version of Linux. The OS4RS manages the dynamic creation of hardware tasks and handles communications among them. It also determines when and on which resource to schedule newly created tasks. When switching between tiles on the FPGA or between the FPGA and the StrongARM microprocessor, OS4RS must suspend certain tasks that are running so that other tasks can take a turn. To do so, it must remember the state each task was in when it stopped so that each task can restart from the same state.

To find a way to seamlessly and automatically switch a task running in software on the microprocessor to the FPGA tiles, we looked at traditional microprocessors and operating systems. These solved the software half of the problem a long time ago.
CILIA: A Computational Environment for Programmable Surfaces

- Field Editor for specifying individual vector fields
- Object Editor for specifying the geometry of parts
- Script Editor for sequential composition of fields over time
Control Problem
[STEIN, 1992]

- Normal acceleration, $n_z$, should track pilot input
- Angle of attack, $\alpha$, must be limited $\alpha \leq \alpha_{\text{lim}}$

A successful controller would satisfy both objectives simultaneously (to the extent that this is possible):

We desire good tracking of the pilot's input without violating the constraint

Switching System: Max Controller

$\delta = \text{max} \left\{ \delta_1, \delta_2 \right\}$

Tracking Controller

Max Controller

(a) normal acceleration, $n_z$ (solid); desired $n_z$, $r$ (dashed)
(b) angle of attack $\alpha$ (solid); $\alpha$'s limit (dashed)

$\dot{x} = f(x, q)$, $q \in \{1, 2\}$
DYNAMICAL PICK-AND-PLACE

BURRIDGE, RIZZI, KODITSCHEK (1995)

JUGGLING ROBOT

GOAL: BRING BALL TO REST ON PADDLE AT A SPECIFIED LOCATION FROM "RANDOMLY" THROWN INITIAL BALL STATES

<table>
<thead>
<tr>
<th>BEHAVIOR</th>
<th>NAME</th>
<th>GOAL</th>
<th>DOMAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Φp</td>
<td>PALM</td>
<td>TASK GOAL POINT</td>
<td>ALL BALL STATES W/ LOW VERT. ENERGY</td>
</tr>
<tr>
<td>Φc</td>
<td>CATCH</td>
<td>TASK GOAL POINT</td>
<td>… W/ LOW HORIZ. POS. ERRS., LOW VEL. ERRS., BOUNDED VERT. ENERGY</td>
</tr>
<tr>
<td>Φj</td>
<td>JUGGLE</td>
<td>0.8m ABOVE TASK GOAL PT.</td>
<td>ENTIRE STATE SPACE</td>
</tr>
</tbody>
</table>

Figure 1: Behavior switching. (a) A quick example. (b) Several mode switches. (c) Four attempts before success.
Figure 2: Hybrid model for a dumbbell network with TCP-SACK congestion control. In this figure $q := \sum_{f \in \mathcal{F}} q_f$, $r := \sum_{f \in \mathcal{F}} r_f$, and $RTT = T^e + \frac{q}{B}$. 

\[ w_f := 1 \]
\[
\text{slow-start/queue-not-full:} \quad \dot{w}_f = \frac{\log 2}{RTT} w_f, \quad \dot{r}_f = \frac{w_f}{RTT} \\
\dot{q} = r - B \]

\[ t_{\text{tim}} := RTT \]

\[ q = q_{\text{max}}, r > B? \]

\[ \dot{t}_{\text{tim}} < 0, \frac{r}{2} \leq B? \]

\[ \frac{r}{2} > B? \]

\[ \text{slow-start/queue-full:} \quad \dot{w}_f = \frac{\log 2}{RTT} w_f, \quad \dot{r}_f = \frac{w_f}{RTT} \\
\dot{q} = 0, \quad \dot{t}_{\text{tim}} = -1 \]

\[ t_{\text{tim}} < 0, \frac{r}{2} \leq B? \]

\[ \frac{r}{2} > B? \]

\[ \text{fast-recover./queue-not-full:} \quad \dot{w}_f = 0, \quad \dot{r}_f = \frac{w_f}{2RTT} \\
\dot{q} = r - B, \quad \dot{t}_{\text{tim}} = -1 \]

\[ t_{\text{tim}} < 0? \]

\[ \text{fast-recover./queue-full:} \quad \dot{w}_f = 0, \quad \dot{r}_f = \frac{w_f}{2RTT} \\
\dot{q}_f = 0 \]

\[ t_{\text{tim}} := RTT \]

\[ w_f := \frac{w_f}{2} \]

\[ \text{cong.-avoid./queue-not-full:} \quad \dot{w}_f = \frac{1}{RTT}, \quad \dot{r}_f = \frac{w_f}{RTT} \\
\dot{q} = r - B \]

\[ t_{\text{tim}} := RTT \]

\[ q = q_{\text{max}}, r > B? \]

\[ t_{\text{tim}} < 0, \frac{r}{2} \leq B? \]

\[ \frac{r}{2} > B? \]

\[ \text{cong.-avoid./queue-full:} \quad \dot{w}_f = \frac{1}{RTT}, \quad \dot{r}_f = \frac{w_f}{RTT} \\
\dot{q} = 0, \quad \dot{t}_{\text{tim}} = -1 \]
To verify the formulas in Theorem 2, we simulated the dumbbell topology of Figure 1, using the ns-2 network simulator [9]. Figure 4 summarizes the results obtained for a network with the following parameters: $B = \frac{10^7 \text{bits/sec}}{8 \text{bits/char} \times 1000 \text{char/packet}} = 1250 \text{packets/sec}$, $T_p = .04 \text{sec}$, $q_{\text{max}} = 250 \text{packets}$. As seen in the figure, the theoretical predictions given by (6)-(8) match the simulation results quite accurately. Note the comparison in the rightmost plot of the theoretical prediction obtained by (8) and that obtained using the standard formula (1). One can see that the model derived here is valid over a wider range of traffic conditions (almost one order of magnitude larger).

Figure 4: Comparison between the predictions obtained from the hybrid model and the results from ns-2 simulations.