Treatment of Discontinuities

• Today, we shall look at the problem of dealing with discontinuities in models.
• Models from engineering often exhibit discontinuities that describe situations such as switching, limiters, dry friction, impulses, or similar phenomena.
• The modeling environment must deal with these problems in special ways, since they influence strongly the numerical behavior of the underlying differential equation solver.

Table of Contents

• Numerical differential equation solvers
• Discontinuities in state equations
• Integration across discontinuities
• State events
• Event handling
• Multi-valued functions
• The electrical switch
• The ideal diode
• Friction
Numerical Differential Equation Solvers

- All of the *differential equation solvers* that are currently on the market operate on *polynomial extrapolation*.
- The value of a state variable $x$ at time $t+h$, where $h$ is the current *integration step size*, is approximated by fitting a *polynomial of $n$th order* through known supporting values of $x$ and $dx/dt$ at the current time $t$ as well as at past instances of time.
- The value of the extrapolation polynomial at time $t+h$ represents the approximated solution of the differential equation.
- In the case of *implicit integration algorithms*, the state derivative at time $t+h$ is also used as a supporting value.

Examples

*Explicit Euler Integration Algorithm of 1st Order:*

$$x(t+h) = x(t) + h \cdot \dot{x}(t)$$

*Implicit Euler Integration Algorithm of 1st Order:*

$$x(t+h) = x(t) + h \cdot \dot{x}(t+h)$$
Discontinuities in State Equations

- Polynomials are always continuous and continuously differentiable functions.
- Therefore, when the state equations of the system:
  \[ \dot{x}(t) = f(x(t), t) \]
  exhibit a discontinuity, the polynomial extrapolation is a very poor approximation of reality.
- Consequently, integration algorithms with a fixed step size exhibit a large integration error, whereas integration algorithms with a variable step size must reduce the step size dramatically in the vicinity of the discontinuity.

Integration Across Discontinuities

- An integration algorithm of variable step size reduces the step size at every discontinuity.
- After passing the discontinuity, the step size is only slowly enlarged again, as the integration algorithm cannot distinguish between a discontinuity on one hand and a point of large local stiffness (with a large absolute value of the derivative) on the other.
The State Event

- These problems can be avoided by telling the integration algorithm explicitly, when and where discontinuities are contained in the model description.

**Example: Limiter Function**

\[
\begin{align*}
  f &= f_m & \text{if } x < x_m \\
  f &= m \cdot x & \text{if } x_m < x < x_p \\
  f &= f_p & \text{else if } x \geq x_p
\end{align*}
\]

\[
m = \tan(\alpha)
\]

\[
f = \begin{cases} 
  f_m & \text{if } x < x_m \\
  m \cdot x & \text{if } x_m < x < x_p \\
  f_p & \text{else if } x \geq x_p
\end{cases}
\]

---

**Event Handling I**

- Model switching
- Iteration
- Step size reduction during process of iteration
Event Handling II

Step size as function of time
without event handling

Step size as function of time
with event handling

November 17, 2003

Representation of Discontinuities

\[ f = \begin{cases} fm & \text{if } x < xm \ \\
  m \ast x & \text{if } x < xp \ \\
 fp & \text{else} \end{cases} \]

- In Modelica, discontinuities are represented as if-statements.
- In the process of translation, these statements are transformed into correct event descriptions (sets of models with switching conditions).
- The modeler does not need to concern him or herself with the mechanisms of event descriptions. These are hidden behind the if-statements.
Problems

• The modeler needs to take into account that the discontinuous solution is temporarily left during iteration.

\[ q = \sqrt{|\Delta p|} \]

\[ \Delta p = p_1 - p_2; \]

\[ abs \Delta p = \text{noEvent}( \text{if } \Delta p > 0 \text{ then } \Delta p \text{ else } -\Delta p ); \]

\[ q = \sqrt{abs \Delta p}; \]

may be dangerous, since \( abs \Delta p \) can become temporarily negative.

\[ \Rightarrow \Delta p = p_1 - p_2; \]

\[ abs \Delta p = \text{noEvent}( \text{if } \Delta p > 0 \text{ then } \Delta p \text{ else } -\Delta p ); \]

\[ q = \sqrt{abs \Delta p}; \]

solves this problem.

The “noEvent” Construct

• The noEvent construct has the effect that if-statements or Boolean expressions, which normally would be translated into simulation code containing correct event handling instructions, are handed over to the integration algorithm untouched.

• Thereby, management of the simulation across these discontinuities is left to the step size control of the numerical Integration algorithm.
Multi-valued Functions I

- The language constructs that have been introduced so far don’t suffice to describe multi-valued functions, such as the dry hysteresis function shown below.

- When $x$ becomes greater than $x_p$, $f$ must be switched from $f_m$ to $f_p$.
- When $x$ becomes smaller than $x_m$, $f$ must be switched from $f_p$ to $f_m$.

Multi-valued Functions II

- When initial() then
  - reinit($f$, $f_p$);
- end when;
- when $x > x_p$ or $x < x_m$ then
  - $f = $ if $x > 0$ then $f_p$ else $f_m$;
- end when;

These statements are only executed, when either $x$ becomes larger than $x_p$, or when $x$ becomes smaller than $x_m$.

Executed at the beginning of the simulation.
The Electrical Switch I

When the switch is open, the current is \( i=0 \).
When the switch is closed, the voltage is \( u=0 \).

\[
0 = \text{if open then } i \text{ else } u;
\]

The if-statement in Modelica is a-causal. It is being sorted together with all other statements.

The Electrical Switch II

\[
0 = s \cdot i + (1-s) \cdot u
\]

Switch open:

\[
\text{SF} \quad f = 0 \quad \Rightarrow \quad s \quad \text{Sw} \quad e
\]

Switch closed:

\[
\text{SE} \quad e = 0
\]

The causality of the switch element is a function of the value of the control signal \( s \).
The Ideal Diode I

When $u < 0$, the switch is open. No current flows through.

When $u > 0$, the switch is closed. Current may flow. The ideal diode behaves like a short circuit.

```
open = u < 0 ;
0 = if open then i else u ;
```

The Ideal Diode II

• Since current flowing through a diode cannot simply be interrupted, it is necessary to slightly modify the diode model.

```
open = u <= 0 and not i > 0 ;
0 = if open then i else u ;
```

• The variable $open$ must be declared as $Boolean$. The value to the right of the Boolean expression is assigned to it.
The Friction Characteristic I

- More complex phenomena, such as friction characteristics, must be carefully analyzed case by case.
- The approach is discussed here by means of the friction example.

When \( v \neq 0 \), the friction force is a function of the velocity.
When \( v = 0 \), the friction force is computed such that the velocity remains 0.

The Friction Characteristic II

- We distinguish between five situations:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v = 0 )</td>
<td><strong>Sticking:</strong> The friction force compensates the sum of all forces attached, except if (</td>
<td>\Sigma</td>
</tr>
<tr>
<td>( a = 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( v &gt; 0 )</td>
<td><strong>Moving forward:</strong> The friction force is computed as: ( f_B = R_v \cdot v + R_m ).</td>
<td></td>
</tr>
<tr>
<td>( v &lt; 0 )</td>
<td><strong>Moving backward:</strong> The friction force is computed as: ( f_B = R_v \cdot v - R_m ).</td>
<td></td>
</tr>
<tr>
<td>( v = 0 )</td>
<td><strong>Beginning of forward motion:</strong> The friction force is computed as: ( f_B = R_m ).</td>
<td></td>
</tr>
<tr>
<td>( a &gt; 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( v = 0 )</td>
<td><strong>Beginning of backward motion:</strong> The friction force is computed as: ( f_B = -R_m ).</td>
<td></td>
</tr>
<tr>
<td>( a &lt; 0 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The State Transition Diagram

- The set of events can be described by a *state transition diagram*.

![State Transition Diagram](image)

The Friction Model I

```plaintext
model Friction;
  parameter Real R0, Rm, Rv;
  parameter Boolean ic=false;
  Real fB, fc;
  Boolean Sticking (final start = ic);
  Boolean Forward (final start = ic), Backward (final start = ic);
  Boolean StartFor (final start = ic), StartBack (final start = ic);

  fB = if Forward then Rv*v + Rm else
      if Backward then Rv*v - Rm else
      if StartFor then Rm else
      if StartBack then -Rm else fc;

  0 = if Sticking or initial() then a else fc;
```

November 17, 2003

© François E. Cellier, Ph.D.
The Friction Model II

when Sticking and not initial() then
  reinit(v,0);
end when;

\[
\begin{align*}
\text{Forward} &= \text{initial()} \quad \text{and} \quad v > 0 \quad \text{or} \\
  &\quad \text{pre(StartFor)} \quad \text{and} \quad v > 0 \quad \text{or} \\
  &\quad \text{pre(Forward)} \quad \text{and not} \quad v \leq 0; \\
\text{Backward} &= \text{initial()} \quad \text{and} \quad v < 0 \quad \text{or} \\
  &\quad \text{pre(StartBack)} \quad \text{and} \quad v < 0 \quad \text{or} \\
  &\quad \text{pre(Backward)} \quad \text{and not} \quad v \geq 0;
\end{align*}
\]

The Friction Model III

\[
\begin{align*}
\text{StartFor} &= \text{pre(Sticking)} \quad \text{and} \quad fc > R_0 \quad \text{or} \\
  &\quad \text{pre(StartFor)} \quad \text{and not} \quad (v > 0 \quad \text{or} \quad a \leq 0 \quad \text{and not} \quad v > 0); \\
\text{StartBack} &= \text{pre(Sticking)} \quad \text{and} \quad fc < -R_0 \quad \text{or} \\
  &\quad \text{pre(StartBack)} \quad \text{and not} \quad (v < 0 \quad \text{or} \quad a \geq 0 \quad \text{and not} \quad v < 0); \\
\text{Sticking} &= \text{not} \quad (\text{Forward} \quad \text{or} \quad \text{Backward} \quad \text{or} \quad \text{StartFor} \quad \text{or} \quad \text{StartBack});
\end{align*}
\]

end Friction;
References I


References II
