Closing the Gap in Control System Implementations

Indranil Saha

Department of Computer Science
University of California, Los Angeles
The systems are mostly **life-critical** or **mission-critical**
Controller Software: The Weak Link
Controller Software: The Weak Link

Plant

\[ x' = f(x, u) \]

Controller

\[ u = k(x) \]

Actuator

Sensor

Control System

Plant

Controller

1962 – Mariner I Space Probe Malfunction
Controller Software: The Weak Link

Plant

\[ x' = f(x, u) \]

Controller

\[ u = k(x) \]

Actuator

Sensor

Control System

1962 – Mariner I Space Probe Malfunction

1989 – Swedish Gripen Fighter Crash
Controller Software: The Weak Link

1962 – Mariner I Space Probe Malfunction
1989 – Swedish Gripen Fighter Crash
1995 – Ariane 5 Flight 501 Explosion
Controller Software: The Weak Link

- 1962 – Mariner I Space Probe Malfunction
- 1989 – Swedish Gripen Fighter Crash
- 1995 – Ariane 5 Flight 501 Explosion

...
How can we develop more reliable control systems?
How can we develop more reliable control systems?

Control Theory + Program Analysis + Scheduling Theory = Reliable Embedded Systems
Control System Design and Implementation

Controller Design

Controller Implementation

Closing the Gap in Control System Implementations
Design

Mathematical Model of Physical System → Control Design → Mathematical Model of the Controller → Controller Implementation
Implementation

Mathematical Model of the Controller
Floating-point/Fixed-point C Code
Control System

Mathematical Model of Physical System
Control Design
Mathematical Model of the Controller

Mathematical Analysis
Code Generation
Floating-point/Fixed-point C Code
Integration
System-level simulation

Code Testing
Control System
Infinite precision arithmetic
Negligible delay and computation time
Ideal network

Finite precision arithmetic
Sharing of resources
Effect of network
Infinite precision arithmetic
Negligible delay and computation time
Ideal network

Does the implemented system exhibit the same behavior as the mathematical model?

Finite precision arithmetic
Sharing of resources
Effect of network
Gap between Design and Implementation

Mathematical Model of the Controller
Floating-point/Fixed-point C Code
Control System

Mathematical Model of Physical System
Control Design
Mathematical Model of the Controller

Mathematical Analysis
Code Generation
Floating-point/Fixed-point C Code
Integration
System-level simulation
Control System

Infinite precision arithmetic
Negligible delay and computation time
Ideal network

Finite precision arithmetic
Sharing of resources
Effect of network

Result of mathematical analysis does not carry forward from the design phase to the implementation phase
Gap between Design and Implementation

Mathematical Model of Physical System

Mathematical Model of the Controller

Control Design

Mathematical Analysis

Floating-point/Fixed-point C Code

Mathematical Model of the Controller

Code Generation

Floating-point/Fixed-point C Code

Integration

System-level simulation

Control System

Infinite precision arithmetic
Negligible delay and computation time
Ideal network

Finite precision arithmetic
Sharing of resources
Effect of network

Result of mathematical analysis does not carry forward from the design phase to the implementation phase

An end-to-end argument is missing
Combine the results of different analysis techniques to give formal guarantee on the behavior of the implementation.
Take into account the implementation constraints during the design of the controller
## Research Contribution

<table>
<thead>
<tr>
<th>Category</th>
<th>Research Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stability</strong></td>
<td>- Verification of Controller Software</td>
</tr>
<tr>
<td></td>
<td>[AntaMajumdarSTabuada, EMSOFT 2010]</td>
</tr>
<tr>
<td></td>
<td>- Synthesis of Controller Software</td>
</tr>
<tr>
<td></td>
<td>[MajumdarSZamani, EMSOFT 2012]</td>
</tr>
<tr>
<td></td>
<td>- Optimization of Controller Software</td>
</tr>
<tr>
<td></td>
<td>[DarulovaKuncakMajumdarS, under submission]</td>
</tr>
<tr>
<td><strong>Feasibility</strong></td>
<td>- Memoization Based Implementation of Self-Triggered Controllers</td>
</tr>
<tr>
<td></td>
<td>[SMajumdar, EMSOFT2012]</td>
</tr>
<tr>
<td><strong>Schedulability</strong></td>
<td>- Synthesis of Static Scheduler</td>
</tr>
<tr>
<td></td>
<td>[MajumdarSZamani, EMSOFT 2011]</td>
</tr>
<tr>
<td></td>
<td>- Synthesis of Dynamic Scheduler</td>
</tr>
<tr>
<td></td>
<td>[SMajumdar, under preparation]</td>
</tr>
</tbody>
</table>
| Stability | Verification of Controller Software  
[AntaMajumdarSTabuada, EMSOFT 2010] |
|-----------|-----------------------------------|
|           | Synthesis of Controller Software  
[MajumdarSZamani, EMSOFT 2012]       |
|           | Optimization of Controller Software  
[DarulovaKuncakMajumdarS, under submission] |

| Feasibility of Implementation | Memoization Based Implementation of Self-Triggered Controllers  
[SMajumdar, EMSOFT 2012] |

| Schedulability | Synthesis of Static Scheduler  
[MajumdarSZamani, EMSOFT 2011] |
|               | Synthesis of Dynamic Scheduler  
[SMajumdar, under preparation] |
**Definition:** The plant converges to a desired behavior under the actions of the controller.
Definition: The plant converges to a desired behavior under the actions of the controller

Example: Thermostat
In the steady state, the room temperature will be at 22°C
Stability property is replaced by **practical stability**

**Definition:** The state of the plant eventually reaches a bounded region and remains there under the action of the controller.
Practical Stability

Mathematical Model

Software Implementation

Stability property is replaced by **practical stability**

**Definition:** The state of the plant eventually reaches a bounded region and remains there under the action of the controller

**Example: Thermostat**

In the steady state, the room temperature will be between 21.5C and 22.5C
Bound on the Region of Practical Stability

Mathematical Model

Software Implementation
Bound on the Region of Practical Stability

Mathematical Model

Software Implementation

**Theorem[AntaMajumdarSTabuada EMSOFT’10]** If $\gamma$ is the L2-Gain of a control system, $b$ is a bound on the implementation error, then

$$\rho \leq \gamma \times b$$
Bound on the Region of Practical Stability

Mathematical Model

Software Implementation

Theorem[AntaMajumdarSTabuada EMSOFT’10] If $\gamma$ is the L2-Gain of a control system, $b$ is a bound on the implementation error, then

$$\rho \leq \gamma \times b$$

Separation of concerns:

- Compute L2-gain from the mathematical model
  (standard problem in control theory)

- Compute the bound on implementation error
  (analysis of the implementation)
**Example of Controller Program**

**Control Law (Vehicle Steering):**

\[ u = 0.81 \times (\text{In1} - \text{In2}) - 1.017 \times \text{In3} \]

<table>
<thead>
<tr>
<th>Real-valued program</th>
<th>Fixed-point implementation (16-bit):</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static void output(void) {</code></td>
<td><code>static void output(void) {</code></td>
</tr>
<tr>
<td><code>    Subtract = In1 - In2;</code></td>
<td><code>    Subtract = (short int)(\text{In1} - \text{In2});</code></td>
</tr>
<tr>
<td><code>    Gain = 0.81 \times \text{Subtract};</code></td>
<td><code>    Gain = (short int)(26542 \times \text{Subtract} \gg 15);</code></td>
</tr>
<tr>
<td><code>    Gain2 = 1.017 \times \text{In3};</code></td>
<td><code>    Gain2 = (short int)(16663 \times \text{In3} \gg 14);</code></td>
</tr>
<tr>
<td><code>    Out1 = Gain - Gain2;</code></td>
<td><code>    Out1 = (short int)((Gain \ll 1) - \text{Gain2} \gg 1);</code></td>
</tr>
</tbody>
</table>
| `}` | `}`
Control Law (Vehicle Steering):
\[ u = 0.81 \times (\text{In1} - \text{In2}) - 1.017 \times \text{In3} \]

**Real-valued program**

```c
static void output(void) {
    Subtract = In1 - In2;
    Gain = 0.81 * Subtract;
    Gain2 = 1.017 * In3;
    Out1 = Gain - Gain2;
}
```

**Fixed-point implementation (16-bit):**

```c
short int In1, In2, In3;
short int Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = (short int)(In1 - In2);
    Gain = (short int)(26542 * Subtract >> 15);
    Gain2 = (short int)(16663 * In3 >> 14);
    Out1 = (short int)(((Gain << 1) - Gain2) >> 1);
}
```

**What is the bound on the error?**
Implementation: Costan

- An automatic tool to compute the bound on the region of practical stability

- Supports both linear and nonlinear controllers, for nonlinear controllers both polynomial implementation and lookup table based implementation
## Experimental Results

<table>
<thead>
<tr>
<th>Example</th>
<th>Error bound</th>
<th>Bound on $\rho$</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle steering (16bit)</td>
<td>0.0163</td>
<td>0.0375</td>
<td>1m</td>
</tr>
<tr>
<td>pendulum (16bit)</td>
<td>0.0508</td>
<td>0.1806</td>
<td>3m</td>
</tr>
<tr>
<td>dc motor (16bit)</td>
<td>0.0473</td>
<td>1.0889</td>
<td>2m</td>
</tr>
<tr>
<td>train car - 1 car (32bit)</td>
<td>5e-7</td>
<td>2.6080e-5</td>
<td>3m</td>
</tr>
<tr>
<td>train car - 2 cars (32bit)</td>
<td>1.5e-6</td>
<td>9.4000e-5</td>
<td>6m</td>
</tr>
<tr>
<td>train car - 3 cars (32bit)</td>
<td>8.5e-6</td>
<td>0.0010</td>
<td>10m</td>
</tr>
<tr>
<td>train car - 4 cars (32bit)</td>
<td>3.351e-5</td>
<td>0.0080</td>
<td>10m</td>
</tr>
<tr>
<td>train car - 5 cars (32bit)</td>
<td>1.655e-4</td>
<td>0.0627</td>
<td>20m</td>
</tr>
<tr>
<td>jet engine[poly] (16bit)</td>
<td>4e-3</td>
<td>0.0230</td>
<td>&lt;1m</td>
</tr>
<tr>
<td>jet engine[3 × 8]</td>
<td>6.40</td>
<td>37.0431</td>
<td>&lt;1m</td>
</tr>
<tr>
<td>jet engine[5 × 10]</td>
<td>4.48</td>
<td>25.9296</td>
<td>&lt;1m</td>
</tr>
<tr>
<td>jet engine[7 × 14]</td>
<td>2.73</td>
<td>15.8009</td>
<td>2m</td>
</tr>
<tr>
<td>jet engine[21 × 21]</td>
<td>1.25</td>
<td>7.2348</td>
<td>18m</td>
</tr>
<tr>
<td>jet engine[21 × 101]</td>
<td>0.88</td>
<td>5.0933</td>
<td>50m</td>
</tr>
<tr>
<td>jet engine[100 × 100]</td>
<td>0.33</td>
<td>1.9100</td>
<td>103m</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Example</th>
<th>Error bound</th>
<th>Bound on $\rho$</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vehicle steering</strong> (16bit)</td>
<td>0.0163</td>
<td>0.0375</td>
<td>1m</td>
</tr>
<tr>
<td>pendulum (16bit)</td>
<td>0.0508</td>
<td>0.1806</td>
<td>3m</td>
</tr>
<tr>
<td>dc motor (16bit)</td>
<td>0.0473</td>
<td>1.0889</td>
<td>2m</td>
</tr>
<tr>
<td>train car - 1 car (32bit)</td>
<td>5e-7</td>
<td>2.6080e-5</td>
<td>3m</td>
</tr>
<tr>
<td>train car - 2 cars (32bit)</td>
<td>1.5e-6</td>
<td>9.4000e-5</td>
<td>6m</td>
</tr>
<tr>
<td>train car - 3 cars (32bit)</td>
<td>8.5e-6</td>
<td>0.0010</td>
<td>10m</td>
</tr>
<tr>
<td>train car - 4 cars (32bit)</td>
<td>3.351e-5</td>
<td>0.0080</td>
<td>10m</td>
</tr>
<tr>
<td>train car - 5 cars (32bit)</td>
<td>1.655e-4</td>
<td>0.0627</td>
<td>20m</td>
</tr>
<tr>
<td>jet engine[poly] (16bit)</td>
<td>4e-3</td>
<td>0.0230</td>
<td>&lt;1m</td>
</tr>
<tr>
<td>jet engine[3 × 8]</td>
<td>6.40</td>
<td>37.0431</td>
<td>&lt;1m</td>
</tr>
<tr>
<td>jet engine[5 × 10]</td>
<td>4.48</td>
<td>25.9296</td>
<td>&lt;1m</td>
</tr>
<tr>
<td>jet engine[7 × 14]</td>
<td>2.73</td>
<td>15.8009</td>
<td>2m</td>
</tr>
<tr>
<td>jet engine[21 × 21]</td>
<td>1.25</td>
<td>7.2348</td>
<td>18m</td>
</tr>
<tr>
<td>jet engine[21 × 101]</td>
<td>0.88</td>
<td>5.0933</td>
<td>50m</td>
</tr>
<tr>
<td>jet engine[100 × 100]</td>
<td>0.33</td>
<td>1.9100</td>
<td>103m</td>
</tr>
</tbody>
</table>
The control objective is to make the vehicle stable parallel to the x-axis at a certain distance $d$. 

![Diagram showing vehicle parallel to x-axis with distance d]
The control objective is to make the vehicle stable parallel to the x-axis at a certain distance $d$

For vehicle steering, $\rho = 0.0375m$
Example: Vehicle Steering

The control objective is to make the vehicle stable parallel to the x-axis at a certain distance $d$

In the steady state the vehicle will be between $d - \rho$ and $d + \rho$ distance from the x-axis

For vehicle steering, $\rho = 0.0375m$
<table>
<thead>
<tr>
<th>Research Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stability</strong></td>
</tr>
<tr>
<td>Verification of Controller Software</td>
</tr>
<tr>
<td>[AntaMajumdarSTabuada, EMSOFT 2010]</td>
</tr>
<tr>
<td><strong>Synthesis of Controller Software</strong></td>
</tr>
<tr>
<td>[MajumdarSZamani, EMSOFT 2012]</td>
</tr>
<tr>
<td>Optimization of Controller Software</td>
</tr>
<tr>
<td>[DarulovaKuncakMajumdarS, under submission]</td>
</tr>
<tr>
<td><strong>Feasibility of Implementation</strong></td>
</tr>
<tr>
<td>Memoization Based Implementation of Self-Triggered Controllers</td>
</tr>
<tr>
<td>[SMajumdar, EMSOFT2012]</td>
</tr>
<tr>
<td><strong>Schedulability</strong></td>
</tr>
<tr>
<td>Synthesis of Static Scheduler</td>
</tr>
<tr>
<td>[MajumdarSZamani, EMSOFT 2011]</td>
</tr>
<tr>
<td>Synthesis of Dynamic Scheduler</td>
</tr>
<tr>
<td>[SMajumdar, under preparation]</td>
</tr>
</tbody>
</table>
Is it possible to synthesize a controller that minimizes the region of practical stability?
Is it possible to synthesize a controller that minimizes the region of practical stability?

Need to take into account other performance criteria as well e.g. LQR cost
Example

Model of a Vehicle Steering:

\[
\begin{bmatrix}
\dot{\xi}_1 \\
\dot{\xi}_2
\end{bmatrix}
= \begin{bmatrix}
0 & \frac{g}{h} \\
1 & 0
\end{bmatrix}
\begin{bmatrix}
\xi_1 \\
\xi_2
\end{bmatrix}
+ \begin{bmatrix}
1 \\
0
\end{bmatrix}(\nu + \omega)
\]

\[
\eta = \begin{bmatrix}
\frac{a v_0}{b h} & \frac{v_0^2}{b h}
\end{bmatrix}
\begin{bmatrix}
\dot{\xi}_1 \\
\dot{\xi}_2
\end{bmatrix}
+ \nu
\]

LQR Controller:

\[K_1 = [5.1538, 12.9724]\]

LQR cost function is 264.1908

Another Controller:

\[K_2 = [3.0253, 12.6089]\]

LQR cost function is 284.1578
There is a trade-off between LQR cost and the region of practical stability
Design a controller optimizing the following objectives:

- The LQR cost
- The bound on the region of practical stability
Synthesize a controller minimizing the following objective function:

\[
J(K) = w_1 \frac{S(K)}{S^*} + w_2 \frac{\gamma(K)b_e(K)}{\gamma^* b_e^*}
\]

where \(w_1\) and \(w_2\) are weighting factors

- \(S(K)\) - LQR cost of the controller \(K\)
- \(S^*\) - LQR cost of the LQR controller
- \(\gamma(K)\) - \(L_2\)-gain of the controller \(K\)
- \(\gamma^*\) - \(L_2\)-gain of the LQR controller
- \(b_e(K)\) - Bound on the implementation error for controller \(K\)
- \(b_e^*\) - Bound on the implementation error for the LQR controller
Objective Function for Controller Synthesis

Synthesize a controller minimizing the following objective function:

\[ J(K) = w_1 \frac{S(K)}{S^*} + w_2 \frac{\gamma(K)b_e(K)}{\gamma^*b_e^*} \]

where \( w_1 \) and \( w_2 \) are weighting factors

- \( S(K) \) - LQR cost of the controller \( K \)
- \( S^* \) - LQR cost of the LQR controller
- \( \gamma(K) \) - \( L_2 \)-gain of the controller \( K \)
- \( \gamma^* \) - \( L_2 \)-gain of the LQR controller
- \( b_e(K) \) - Bound on the implementation error for controller \( K \)
- \( b_e^* \) - Bound on the implementation error for the LQR controller

Computation of \( S(K) \) requires solving a convex optimization problem
Synthesize a controller minimizing the following objective function:

$$\mathcal{J}(K) = w_1 \frac{S(K)}{S^*} + w_2 \frac{\gamma(K)b_e(K)}{\gamma^*b_e^*}$$

where $w_1$ and $w_2$ are weighting factors.

$S(K)$ - LQR cost of the controller $K$

$S^*$ - LQR cost of the LQR controller

$\gamma(K)$ - $L_2$-gain of the controller $K$

$\gamma^*$ - $L_2$-gain of the LQR controller

$b_e(K)$ - Bound on the implementation error for controller $K$

$b_e^*$ - Bound on the implementation error for the LQR controller

Computation of $S(K)$ requires solving a convex optimization problem.

The cost function $\mathcal{J}$ is not necessarily convex with respect to the feedback gain $K$. 
Controller Synthesis Tool: Ocsyn

- Employs a stochastic optimization method to solve the synthesis problem
- Searches for the optimal controller in a chosen search space
- Uses a static analyzer that
  - synthesizes a fixed-point program for a controller
  - computes the bound on the fixed-point implementation error
## Experimental Results

<table>
<thead>
<tr>
<th>Control systems</th>
<th>LQR cost ratio</th>
<th>Steady state error ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Synthesized Controller</td>
<td>LQR Controller</td>
</tr>
<tr>
<td>Bicycle</td>
<td>1.095</td>
<td>10.694</td>
</tr>
<tr>
<td>DC motor</td>
<td>1.3745</td>
<td>14.545</td>
</tr>
<tr>
<td>Pitch angle</td>
<td>1.005</td>
<td>5.88</td>
</tr>
<tr>
<td>Inverted Pendulum</td>
<td>1.244</td>
<td>5.023</td>
</tr>
<tr>
<td>Batch reactor</td>
<td>1.00029</td>
<td>2.554</td>
</tr>
<tr>
<td>Research Contribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification of Controller Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[AntaMajumdarSTabuada, EMSOFT 2010]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesis of Controller Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[MajumdarSZamani, EMSOFT 2012]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Optimization of Controller Software</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[DarulovaKuncakMajumdarS, under submission]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Feasibility of Implementation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memoization Based Implementation of Self-Triggered Controllers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SMajumdar, EMSOFT2012]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Schedulability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesis of Static Scheduler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[MajumdarSZamani, EMSOFT 2011]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesis of Dynamic Scheduler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SMajumdar, under preparation]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Control Law (Vehicle Steering) :
\[ u = 0.81 \times (\text{In1} - \text{In2}) - 1.017 \times \text{In3} \]

Real-valued program

```c
static void output(void) {
    Subtract = In1 - In2;
    Gain = 0.81 * Subtract;
    Gain2 = 1.017 * In3;
    Out1 = Gain - Gain2;
}
```

Fixed-point implementation (16-bit):

```c
short int In1, In2, In3;
short int Subtract, Gain, Gain2, Out1;
static void output(void) {
    Subtract = (short int)(In1 - In2);
    Gain = (short int)(26542 * Subtract >> 15);
    Gain2 = (short int)(16663 * In3 >> 14);
    Out1 = (short int)(((Gain << 1) - Gain2) >> 1);
}
```
Optimization Question

Control Law (Vehicle Steering) :
\[ u = 0.81 \times (\text{In1} - \text{In2}) - 1.017 \times \text{In3} \]

Real-valued program

```
static void output(void) {
    Subtract = In1 - In2;
    Gain = 0.81 * Subtract;
    Gain2 = 1.017 * In3;
    Out1 = Gain - Gain2;
}
```

Fixed-point implementation (16-bit):

```
short int In1, In2, In3;
short int Subtract, Gain, Gain2, Out1;
static void output(void) {
    Subtract = (short int)(In1 - In2);
    Gain = (short int)(26542 * Subtract >> 15);
    Gain2 = (short int)(16663 * In3 >> 14);
    Out1 = (short int)(((Gain << 1) - Gain2) >> 1);
}
```

If we implement a different expression, say,
\[ u = 0.81 \times \text{In1} - 1.017 \times \text{In3} - 0.81 \times \text{In2} \]
can we improve the bound on the error?
Example: Batch Reactor Process

\[ \text{out} = (-0.0078) \times \text{state1} + 0.9052 \times \text{state2} + (-0.0181) \times \text{state3} + (-0.0392) \times \text{state4} + (-0.0003) \times y1 + 0.0020 \times y2 \]
Example: Batch Reactor Process

\[
\text{out} = (-0.0078) \times \text{state1} + 0.9052 \times \text{state2} + \\
(-0.0181) \times \text{state3} + (-0.0392) \times \text{state4} + \\
(-0.0003) \times \text{y1} + 0.0020 \times \text{y2}
\]

Error bound in the best fixed-point implementation (16 bits): 3.9e-3
Example: Batch Reactor Process

\[
\text{out} = \begin{align*}
&= (-0.0078) \times \text{state1} + 0.9052 \times \text{state2} + \\
&\quad (-0.0181) \times \text{state3} + (-0.0392) \times \text{state4} + \\
&\quad (-0.0003) \times y1 + 0.0020 \times y2 
\end{align*}
\]

Error bound in the best fixed-point implementation (16 bits): $3.9\times10^{-3}$

\[
\text{out} = \begin{align*}
&= (0.9052 \times \text{state2} ) + (((\text{state3} \times -0.0181) + \\
&\quad (-0.0078 \times \text{state1} )) + (((-0.0392 \times \text{state4} )+ \\
&\quad (-0.0003 \times y1 )) + (0.002 \times y2 )))
\end{align*}
\]
Example: Batch Reactor Process

\[
\text{out} = (-0.0078) \ast \text{state1} + 0.9052 \ast \text{state2} + \]
\[
(-0.0181) \ast \text{state3} + (-0.0392) \ast \text{state4} + \]
\[
(-0.0003) \ast \text{y1} + 0.0020 \ast \text{y2} \]

Error bound in the best fixed-point implementation (16 bits): 3.9e-3

\[
\text{out} = ((0.9052 \ast \text{state2} ) + (((\text{state3} \ast -0.0181)+ \]
\[
(-0.0078 \ast \text{state1 } )) + (((-0.0392 \ast \text{state4 })+ \]
\[
(-0.0003 \ast \text{y1 } )) + (0.002 \ast \text{y2 } )))\]

Error bound in the best fixed-point implementation (16 bits): 1.39e-03
Example: Batch Reactor Process

\[
\text{out} = (-0.0078) \times \text{state1} + 0.9052 \times \text{state2} + \\
(−0.0181) \times \text{state3} + (−0.0392) \times \text{state4} + \\
(−0.0003) \times y1 + 0.0020 \times y2
\]

Error bound in the best fixed-point implementation (16 bits): 3.9e-3

\[
\text{out} = ((0.9052 \times \text{state2} ) + (((\text{state3 } \times −0.0181) + \\
(−0.0078 \times \text{state1 } )) + (((−0.0392 \times \text{state4 } )+ \\
(−0.0003 \times y1 )) + (0.002 \times y2 )))
\]

Error bound in the best fixed-point implementation (16 bits): 1.39e-03

Improvement   55%, without requiring any extra hardware
Example: Batch Reactor Process

\[ \text{out} = (-0.0078) \times \text{state1} + 0.9052 \times \text{state2} + \\
(-0.0181) \times \text{state3} + (-0.0392) \times \text{state4} + \\
(-0.0003) \times y1 + 0.0020 \times y2 \]

Error bound in the best fixed-point implementation (16 bits): 3.9e-3

\[ \text{out} = ((0.9052 \times \text{state2}) + (((\text{state3} \times -0.0181) + \\
(-0.0078 \times \text{state1}) + (((-0.0392 \times \text{state4}) + \\
(-0.0003 \times y1) + (0.002 \times y2)))))) \]

Error bound in the best fixed-point implementation (16 bits): 1.39e-03

Improvement 55%, without requiring any extra hardware

**Question:** How to find the best expression?
Example: Batch Reactor Process

\[
\text{out} = (-0.0078) \times \text{state1} + 0.9052 \times \text{state2} + \\
(-0.0181) \times \text{state3} + (-0.0392) \times \text{state4} + \\
(-0.0003) \times \text{y1} + 0.0020 \times \text{y2}
\]

Error bound in the best fixed-point implementation (16 bits): 3.9e-3

\[
\text{out} = ((0.9052 \times \text{state2 } ) + (((\text{state3 } \times -0.0181) + \\
(-0.0078 \times \text{state1 } )) + (((-0.0392 \times \text{state4 } ) + \\
(-0.0003 \times \text{y1 } )) + (0.002 \times \text{y2 } )))
\]

Error bound in the best fixed-point implementation (16 bits): 1.39e-03

Improvement 55%, without requiring any extra hardware

**Question:** How to find the best expression?

The problem is **NP-hard**
Implementation: Ocsyn+

- Modifies the objective function used in Ocsyn
- Considers the error bound in the best possible expression for a given controller
- Apply genetic programming based search for optimal expression for a chosen controller
<table>
<thead>
<tr>
<th>Control systems</th>
<th>Region of Practical Stability</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>Improved</td>
</tr>
<tr>
<td>bicycle</td>
<td>7.85e-02</td>
<td>7.70e-02</td>
</tr>
<tr>
<td>dc motor</td>
<td>1.64e-02</td>
<td>1.44e-02</td>
</tr>
<tr>
<td>pitch angle</td>
<td>1.08e-02</td>
<td>8.87e-03</td>
</tr>
<tr>
<td>pendulum</td>
<td>3.11e-04</td>
<td>2.64e-04</td>
</tr>
<tr>
<td>batch reactor</td>
<td>2.59e-01</td>
<td>2.24e-01</td>
</tr>
</tbody>
</table>

Baseline - Controller synthesized by Ocsyn

Improved - Applied the genetic programming based expression search on the controller synthesized by Ocsyn

Optimal - Controller synthesized by Ocsyn+
Control Theory + Software Analysis/Synthesys can provide Reliability Guarantee on the Implemented System
| Research Contribution | Verification of Controller Software  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[AntaMajumdarSTabuada, EMSOFT 2010]</td>
</tr>
</tbody>
</table>
| Stability             | Synthesis of Controller Software  
|                       | [MajumdarSZamani, EMSOFT 2012]          |
|                       | Optimization of Controller Software  
|                       | [DarulovaKuncakMajumdarS, under submission] |
| Feasibility of 
Implementation | Memoization Based Implementation of Self-Triggered Controllers 
|                       | [SMajumdar, EMSOFT2012]                |
| Schedulability        | Synthesis of Static Scheduler  
|                       | [MajumdarSZamani, EMSOFT 2011]         |
|                       | Synthesis of Dynamic Scheduler  
|                       | [SMajumdar, under preparation]         |
A Control System

\[ \dot{\xi} = f(\xi, \upsilon), \quad \xi(0) = \xi_0 \]

- \( \xi \) - State of the plant
- \( \upsilon \) - Control signal generated by the controller
To implement the control law on a digital computer, the state of the plant is sampled at a sequence of time instants \( t_0 = 0, \ t_1, \ t_2, \ldots \)

The time instant \( t_k \) is called the trigger time
\[ \begin{align*}
\dot{\xi} &= f(\xi, v), \quad \xi(0) = \xi_0 \\
v(t_k) &= k(\xi(t_k))
\end{align*} \]
\[ \dot{\xi} = f(\xi, \upsilon), \quad \xi(0) = \xi_0 \]

Sampling period is selected based on the worst case scenario

Inefficient usage of computational resource and communication bandwidth
Self-Triggered Implementation

$$\dot{\xi} = f(\xi, \nu), \quad \xi(0) = \xi_0$$

$$\nu(t_k) = k(\xi(t_k))$$

Actuator $\rightarrow$ Plant $\rightarrow$ Controller $\rightarrow$ Sensor

Event Scheduler

$t_{k+1}$
Self-Triggered Implementation

\[ \dot{\xi} = f(\xi, \nu), \quad \xi(0) = \xi_0 \]

\[ \nu(t_k) = k(\xi(t_k)) \]

Has been shown to **reduce the number of control computations significantly** with respect to its time-triggered counterpart.
Trigger time is computed based on two parameters:

- $\tau_{min}$: minimum trigger time
  - The trigger-time which works in the worst case scenario
  - Can be computed from the parameters of the control system

- $\tau_{max}$: maximum trigger time
  - The maximum duration the plant can be kept open loop
  - A design parameter

\[(t_k + \tau_{min}) \leq t_{k+1} \leq (t_k + \tau_{max})\]
$\tau_c$ - The time required to compute the trigger time

The self-triggered implementation scheme is feasible if and only if

$$(t_k + \tau_c) \leq t_{k+1}$$
The Problem

\[ t_k \quad (t_k + \tau_{min}) \quad (t_k + \tau_c) \quad t_{k+1} \quad (t_k + \tau_{max}) \]
The Problem

\[ t_k (t_k + \tau_{min}) (t_k + \tau_c) t_{k+1} (t_k + \tau_{max}) \]

\[ t_k (t_k + \tau_{min}) t_{k+1} (t_k + \tau_c) (t_k + \tau_{max}) \]
The Problem

Indranil Saha

Closing the Gap in Control System Implementations
The model of a batch reactor process is given by

\[
\dot{\xi} = \begin{bmatrix}
1.38 & -0.20 & 6.71 & -5.67 \\
-0.58 & -4.29 & 0 & 0.67 \\
1.06 & 4.27 & -6.65 & 5.89 \\
0.04 & 4.27 & 1.34 & -2.10
\end{bmatrix} \xi + \begin{bmatrix}
0 & 0 \\
5.67 & 0 \\
1.13 & -3.14 \\
1.13 & 0
\end{bmatrix} v.
\]

The feedback controller

\[
v = -\begin{bmatrix}
0.1006 & -0.2469 & -0.0952 & -0.2447 \\
1.4099 & -0.1966 & 0.0139 & 0.0823
\end{bmatrix} \xi
\]

renders the system exponentially stable.

For this system, \( \tau_{\text{min}} = 18 \text{ms} \)
Following literature we chose \( \tau_{\text{max}} = 358 \text{ms} \)
On a Leon 2 processor with frequency 100\text{MHz}, the WCET of the trigger time computation is 29.793\text{ms}
An Example

The model of a batch reactor process is given by

\[ \dot{\xi} = \begin{bmatrix} 1.38 & -0.20 & 6.71 & -5.67 \\ -0.58 & -4.29 & 0 & 0.67 \\ 1.06 & 4.27 & -6.65 & 5.89 \\ 0.04 & 4.27 & 1.34 & -2.10 \end{bmatrix} \xi + \begin{bmatrix} 0 & 0 \\ 5.67 & 0 \\ 1.13 & -3.14 \\ 1.13 & 0 \end{bmatrix} \nu. \]

The feedback controller

\[ \nu = -\begin{bmatrix} 0.1006 & -0.2469 & -0.0952 & -0.2447 \\ 1.4099 & -0.1966 & 0.0139 & 0.0823 \end{bmatrix} \xi \]

renders the system exponentially stable.

For this system, \( \tau_{\text{min}} = 18\, ms \)
Following literature we chose \( \tau_{\text{max}} = 358\, ms \)
On a Leon 2 processor with frequency 100 MHz, the WCET of the trigger time computation is 29.793 ms

It is possible that \((t_k + \tau_c) > t_{k+1}\)
Proposed Solution

- Fall back to the time-triggered implementation when trigger time is not guaranteed to be computed before the trigger time
Proposed Solution

- Fall back to the time-triggered implementation when trigger time is not guaranteed to be computed before the trigger time

\[(t_k + \tau_c)\]

\[t_k\] \hspace{1cm} \[t_k + \tau_{\text{min}}\] \hspace{1cm} \[t_{k+1}\] \hspace{1cm} \[(t_k + \tau_{\text{max}})\]
Proposed Solution

- Fall back to the time-triggered implementation when trigger time is not guaranteed to be computed before the trigger time.

\[
(t_k + \tau) \\
(t_k + \tau_{\text{min}}) \\
(t_k + 1) \\
(t_k + \tau_{\text{max}})
\]

- Continue trigger-time computation as a background task, and memoize the result of the computation.

- Trigger-time is computed based on quantized state.

Indranil Saha
Closing the Gap in Control System Implementations 42/68
The state \((1.4, 1.3)\) is quantized to \((1, 1)\)

The trigger time corresponding to the state \((1, 1)\) is stored in \(Memo[4, 3]\)
The effect of state quantization can be modeled as a bounded disturbance added at the input of the plant.

Guarantee on region of practical stability - the controller can render the states of the plant exponentially in a region around the origin.

- The size of the region of practical stability depends on the quantization factor.
Program analysis is helpful in detecting infeasibility of implementation

Classical software engineering techniques can be helpful in the implementation of control systems
## Research Contribution

| Stability                  | Verification of Controller Software  
|                           | [AntaMajumdarSTabuada, EMSOFT 2010] |
|                           | Synthesis of Controller Software  
|                           | [MajumdarSZamani, EMSOFT 2012] |
|                           | Optimization of Controller Software  
|                           | [DarulovaKuncakMajumdarS, under submission] |

| Feasibility of Implementation | Memoization Based Implementation of Self-Triggered Controllers  
|                              | [SMajumdar, EMSOFT2012] |

| Schedulability               | Synthesis of Static Scheduler  
|                              | [MajumdarSZamani, EMSOFT 2011] |
|                              | Synthesis of Dynamic Scheduler  
|                              | [SMajumdar, under preparation] |
Today’s complex cyber-physical systems have many control units

- Modern motor vehicles have up to 80 ECUs

Automotive and Avionics industries are moving from federated architecture to integrated architecture

- Multiple control loops need to be implemented on a single processor
Multiple Control Systems with Shared Resources

- Task 1
  - A/D
  - Plant 1
  - D/A
  - Period = $\tau_1$
  - WCET = $c_1$

- Task 2
  - A/D
  - Plant 2
  - D/A
  - Period = $\tau_2$
  - WCET = $c_2$

- Task N
  - A/D
  - Plant N
  - D/A
  - Period = $\tau_k$
  - WCET = $c_k$

- RTOS Scheduler
- Shared CPU
Given tasks with worst case execution times and periods, is there a way to execute them so that all tasks finish executing before their deadlines?

- System schedulable $\rightarrow$ Implement
  
  System not schedulable $\rightarrow$ Send back to designer
  
  Or: Throw more resources at it
Suppose we relax the scheduler:

- In some rounds, the scheduler can decide not to execute a task
- The control input generated in the previous cycle is applied to the plant
- Scheduling problem becomes easier
Suppose we relax the scheduler:

- In some rounds, the scheduler can decide not to execute a task
- The control input generated in the previous cycle is applied to the plant
- Scheduling problem becomes easier

But what happens to the controlled system?

- If we ignore a control task too many times, the system may become unstable
- Even if the system is stable, what happens to the performance?
**Theorem:** For a discrete-time linear time-invariant (LTI) control system, there exists a successful computation rate $r_{min}$, such that the LTI control system with dropout, with no disturbance, is exponentially stable for all $r > r_{min}$

$r_{min}$: Minimal successful computation rate

- can be computed from the parameter of the LTI control system
Performance Criteria: $L_\infty$ to RMS Gain
- captures the effect of the disturbance on the output of the plants

The Lower is the gain, the better is the performance

The value of the gain depends on successful computation rate

For a given rate an upper bound on the $L_\infty$ to RMS Gain can be computed by solving a convex optimization problem
Performance Criteria: $\mathcal{L}_\infty$ to RMS Gain
- captures the effect of the disturbance on the output of the plants

The Lower is the gain, the better is the performance

The value of the gain depends on successful computation rate

For a given rate an upper bound on the $\mathcal{L}_\infty$ to RMS Gain can be computed by solving a convex optimization problem

**Theorem:** The bound on the $\mathcal{L}_\infty$ to RMS gain of the discrete time LTI control system attains the minimum value for the successful computation rate to be either at $r_{\text{min}}$ or at $r_{\text{max}}$
Performance Profile

Captures how performance varies between $r_{min}$ and $r_{max}$

- $r_{max}$ is decided by the scheduling constraints

Example: Pendulum

An end-to-end argument can give a better overall system performance, even with lower resources
Optimal Performance Scheduler Synthesis Problem

Choose:

successful computation rates for the controllers

Such that

1. the system is schedulable
2. the weighted sum of the bound on the $L_\infty$ to RMS Gain is minimized

The problem is NP-Hard

- Reduction is from Multiple-Choice Knapsack Problem
Our Approach

- Find $r_{min}$ for each control system
- Find $r_{max}$ for all control systems
  - Maximize weighted sum of successful computation rates
  - Weights are based on the priorities of the control systems
- Select $r_{opt} \in [r_{min}, r_{max}]$ such that the performance is the best
- Synthesize a scheduler based on the selected rates
  - We provide an constraint solving based static scheduler synthesis algorithm
## Research Contribution

| Stability            | Verification of Controller Software  
|----------------------|-------------------------------------
|                      | [Anta, Majumdar, Tabuada, EMSOFT 2010] |
| Synthesis of Controller Software  
| [Majumdar, Zamani, EMSOFT 2012] |
| Optimization of Controller Software  
| [Darulova, Kuncak, Majumdar, under submission] |

| Feasibility of Implementation | Memoization Based Implementation of Self-Triggered Controllers  
|-----------------------------|---------------------------------------------------------------
|                             | [Majumdar, EMSOFT 2012]                                      |

| Schedulability | Synthesis of Static Scheduler  
|----------------|----------------------------------
|                | [Majumdar, Zamani, EMSOFT 2011] |

| Synthesis of Dynamic Scheduler  
| [Majumdar, under preparation] |
Network introduces delay and packet dropout

- Bounded rate of packet dropout ($r_{\text{net}}$)
- There is no deterministic mechanism of modeling the drop of individual

Static scheduler is not feasible
Operating Successful Computation Rate

$\gamma_m(r)$ - the mean of the $L_\infty$ to RMS gains for $r' \in [r - r_{net}, r]$

Operating Successful Computation Rate - the successful computation rate $r$ so that $\gamma_m(r)$ is minimized among all $r$ in the range $[r_{\text{min}} + r_{\text{net}}, r_{\text{max}}]$

**Theorem:** The operating successful computation rate ($r_{\text{opr}}$) is either $r_{\text{max}}$ or $r_{\text{min}} + r_{\text{net}}$
Follows EDF strategy

Maintains the successful computation rate in the range $[r_{opr} - r_{net}, r_{opr}]$ by suitably dropping control computation
Performance profile of two controllers may be quite different

**Problem:** Given the scheduling constraints, synthesize a controller to achieve optimal performance
Controller Synthesis for an Inverted Pendulum

Controller is synthesized using stochastic local search

The objective function is $\gamma_m(r_{opr})$

$r_{opr}$ - operating successful computation rate satisfying scheduling constraints

![Graph 1: Angular Position vs. Time](Image1)

![Graph 2: Angular Position vs. Time](Image2)
Control Theory + Schedulability Analysis gives better end-to-end performance
Summary

Control Theory
- Real Analysis
- Convex Analysis
...

Program Analysis
- Symbolic Simulation
- Abstract Interpretation
...

Scheduling Theory
- Schedulability Analysis
- WCET Analysis
...

Indranil Saha
Closing the Gap in Control System Implementations  63/68
Summary

Control Theory
- Real Analysis
- Convex Analysis
....

Program Analysis
- Symbolic Simulation
- Abstract Interpretation
....

Scheduling Theory
- Schedulability Analysis
- WCET Analysis
....

Reliable Embedded Systems

Indranil Saha
Closing the Gap in Control System Implementations
Looking Ahead

Control Theory
- Real Analysis
- Convex Analysis

Program Analysis
- Symbolic Simulation
- Abstract Interpretation

Scheduling Theory
- Schedulability Analysis
- WCET Analysis

Reliable Embedded Systems

Sensor Network
Machine Learning
Security
Big Data

Indranil Saha
Closing the Gap in Control System Implementations
Adolfo Anta, Rupak Majumdar, Indranil Saha, Paulo Tabuada. Automatic Verification of Control System Implementations. **EMSOFT 2010. Best Paper Award**


Indranil Saha and Rupak Majumdar. Trigger Memoization in Self-Triggered Control. **EMSOFT 2012.**

Eva Darulova, Viktor Kuncak, Rupak Majumdar and Indranil Saha Synthesis of fixed-point programs. Under submission.

Acknowledgement: Research Collaborators

- Adolfo Anta
- Eva Darulova
- Viktor kuncak
- Rupak Majumdar
- Paulo Tabuada
- Majid Zamani
Acknowledgement: Funding Sources

- DARPA award HR0011-09-1-0037
- NSF awards 0720881, 0953994, and 1035916
- MPI-SWS
- SRI International
- TEMA TTC
- UCLA Dissertation Year Fellowship
Thank You!!

http://www.cs.ucla.edu/~indranil