Program Analysis and Synthesis for Control Applications

Indranil Saha

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

\[ \dot{x} = f(x, u) \]

\[ u = k(x) \]

Plant
Controller
Actuator Sensor
Control System

Indranil Saha
Program Analysis and Synthesis for Control Applications
Controller Software: The Weak Link

Plant \( x' = f(x, u) \)

Controller \( u = k(x) \)

Actuator
Sensor
Control System

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

1962 – Mariner I Space Probe Malfunction
1989 – Swedish Gripen Fighter Crash
Controller Software: The Weak Link

Plant \( \frac{dx}{dt} = f(x, u) \)

Controller \( u = k(x) \)

Actuator

Sensor

Control System

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 Multfunction

1989 – Swedish Gripen Fighter Crash

1995 – Ariane 5 Flight 501 Explosion

2005 – Toyota Prius Recall
Controller Software: The Weak Link

- Plant: $x' = f(x, u)$
- Controller: $u = k(x)$

1962 – Mariner I Space Probe Malfunction
1989 – Swedish Gripen Fighter Crash
1995 – Ariane 5 Flight 501 Explosion
2005 – Toyota Prius Recall

....
How can we develop more reliable controller software?
How can we develop more reliable controller software?

Control Theory + Program Analysis + Scheduling Theory
= Reliable Embedded Systems
Design

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

Mathematical Model of Physical System → Control Design → Mathematical Model of the Controller → Code Generation → Floating-point/Fixed-point C Code → Integration → Control System

Mathematical Analysis → Code Testing → System-level simulation
Gap between Design and Implementation


Infinite precision arithmetic
Negligible delay and computation time
Ideal network

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

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

Does the implemented system exhibit the same behavior as the mathematical model?
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 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
Code Testing
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.
<table>
<thead>
<tr>
<th>Stability</th>
<th>Verification of Controller Software [AntaMajumdarSTabuada, EMSOFT 2010]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Synthesis of Controller Software [MajumdarSZamani, EMSOFT 2012]</td>
</tr>
<tr>
<td></td>
<td>Optimization of Controller Software [DarulovaKuncakMajumdarS, under submission]</td>
</tr>
<tr>
<td>Schedulability</td>
<td>Synthesis of Static Scheduler [MajumdarSZamani, EMSOFT 2011]</td>
</tr>
<tr>
<td></td>
<td>Synthesis of Dynamic Scheduler [SMajumdar, under preparation]</td>
</tr>
<tr>
<td>Feasibility of Implementation</td>
<td>Memoization Based Implementation of Self-Triggered Controllers [SMajumdar, EMSOFT2012]</td>
</tr>
</tbody>
</table>
## Research Contribution

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

| Schedulability             | Synthesis of Static Scheduler       |
|----------------------------|--------------------------------------
|                            | [Majumdar S Zamani, EMSOFT 2011]    |
|                            | Synthesis of Dynamic Scheduler      |
|                            | [SMajumdar, under preparation]      |

<table>
<thead>
<tr>
<th>Feasibility of Implementation</th>
<th>Memoization Based Implementation of Self-Triggered Controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[SMajumdar, EMSOFT 2012]</td>
</tr>
</tbody>
</table>
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
Practical Stability

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
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 21.5°C and 22.5°C.
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 [Anta Majumdar STabuada 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)
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)
For linear system

\[ \dot{x} = Ax + Bu \]
\[ y = Cx \]

L2 gain \( \gamma \) is given by:

\[ \gamma = \max_{\psi \in [0, 2\pi]} \left\| C(e^{i\psi}1_{n \times n} - A)^{-1}B \right\| \]
For a nonlinear system

\[ \dot{x} = f(x, u) \]

with a feedback controller of the form

\[ u = k(x) \]

we need an ISS Lyapunov function \( V(x) \) that satisfies the following constraint:

\[
\frac{\delta V}{\delta x} f(x, kx + e) \leq -\lambda V(x) + \sigma \| e \|
\]

Guarantee on the state is given in terms of the value of the Lyapunov function in the steady state: \( V(x) \leq \gamma \times b \)

L2-Gain \( \gamma \) is given by

\[ \gamma = \frac{\sigma}{\lambda} \]
Bound on the Region of Practical Stability

**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)
Sources of Implementation Error

- Quantization error due to limited precision arithmetic
- Sharing of resources
- Effect of network (delay, packet drop)
Sources of Implementation Error

- Quantization error due to limited precision arithmetic
- Sharing of resources
- Effect of network (delay, packet drop)
Sources of Implementation Error

- Quantization error due to limited precision arithmetic
- Sharing of resources
- Effect of network (delay, packet drop)

Our focus is fixed-point implementation of controller software
Fixed-Point Representation

An approximation of a real number represented as an integer

n
k
An approximation of a real number represented as an integer

Example:

10.99 ~ 0 0 1 0 1 0 1 1

n=8

k=2
Fixed-Point Representation

An approximation of a real number represented as an integer

$$n \quad k$$

Example:

10.99 ~

$$0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1$$

Error introduced due to truncation
Addition in Fixed-Point Arithmetic

Example: \( c = a + b \)

The fixed-point implementation:
\[
\hat{c} = \hat{a} + (\hat{b} \gg (k_2 - k_1))
\]
Given a mathematical controller, we consider its best possible fixed-point implementation.

Fixed-point datatypes are chosen in such a way that:

- there is no overflow
- error due to truncation is minimized
Best Fixed-point Implementation

Given a mathematical controller, we consider its best possible fixed-point implementation.

Fixed-point datatypes are chosen in such a way that:
- there is no overflow;
- error due to truncation is minimized.

**Example:**

\[
\begin{array}{cccccc}
1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[n=8, k=4\]

\[10.99 \sim 10.99\]
Example of Controller Program

Control Law:
\[ u = 0.81 \times (\ln_1 - \ln_2) - 1.017 \times \ln_3 \]

Real-valued program

```c
static void output(void) {
    Subtract = \ln_1 - \ln_2;
    Gain = 0.81 \times Subtract;
    Gain2 = 1.017 \times \ln_3;
    Out1 = Gain - Gain2;
}
```

Fixed-point implementation (16-bit):

```c
short int \ln_1, \ln_2, \ln_3;
short int Subtract, Gain, Gain2, Out1;

static void output(void) {
    Subtract = (short int)(\ln_1 - \ln_2);
    Gain = (short int)(26542 \times Subtract \gg 15);
    Gain2 = (short int)(16663 \times \ln_3 \gg 14);
    Out1 = (short int)(((Gain \ll 1) - Gain2) \gg 1);
}
```
Example of Controller Program

Control Law:

\[ u = 0.81 \times (\ln1 - \ln2) - 1.017 \times \ln3 \]

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?
Calculating the Bound on Quantization Error

Inputs:
- A real-valued function \( y = f(x) \)
- A program \( F \) implementing \( f \) using finite precision arithmetic
- Range \([l, u]\) for \( x \)

Question:
How far can the value \( f(x) \) be from the output of \( F(\hat{x}) \) when
- \( x \) is chosen from the range \([l, u]\) and
- \( \hat{x} \) is the best fixed-point representation of \( x \)
Solve the following optimization problem:

maximize $|y - \hat{y}|$ \hspace{1cm} // the difference in output

subject to

$x \in [l, u]$ \hspace{1cm} // the range of inputs

$|x - \hat{x}| < e$ \hspace{1cm} // the precision of the representation

$y = f(x)$ \hspace{1cm} // the actual controller output

$SP(F(\hat{x}, \hat{y}))$ \hspace{1cm} // the computed controller output

$SP(F(\hat{x}, \hat{y}))$ - strongest postcondition

a logical formula relating input $\hat{x}$ and output $\hat{y}$ of function $F$
Computing Strongest Postcondition

- Run the program with symbolic inputs

- Each execution maintains
  - A symbolic store: maps program variables to symbolic expressions
  - A path constraint: specifies constraints on inputs for the current path to be executed

- A satisfying assignment to the path constraint provides an input that guarantees execution along the path
Symbolic execution in practice is performed through concolic execution [GodefroidKlarlundSen,SenMarinovAgha,XuMajumdarGodefroid] where,

- start with a random concrete input
- program runs simultaneously on the concrete inputs and the symbolic inputs
- input for each consecutive execution is generated by appropriately modifying and solving the constraints generated during symbolic execution
float func(float b) {
    float x, y;
    x = b * 10;
    if (x > 100) {
        y = 100;
    } else {
        y = x;
    }
    return y;
}
float func(float b) {
    float x, y;
    x = b * 10;
    if (x > 100)
        y = 100;
    else
        y = x;
    return y;
}
float func(float b)
{
    float x, y;
    x = b * 10;
    if (x > 100)
        y = 100;
    else
        y = x;
    return y;
}

Concrete inputs $b = 20$
Symbolic input $b_0$

Constraint: $b_0 \times 10 > 100$

Negate the constraint: $b_0 \times 10 \leq 100$
float func(float b) {
    float x, y;
    x = b * 10;
    if (x > 100) {
        y = 100;
    } else {
        y = x;
    }
    return y;
}

Concrete inputs $b = 20$
Symbolic input $b_0$

Negate the constraint:
$b_0 * 10 \leq 100$

Solve the constraints:
$b_0 = 5$
float func(float b)
{
    float x, y;
    x = b * 10;
    if (x > 100)
        y = 100;
    else
        y = x;
    return y;
}

Concrete inputs $b = 5$
Symbolic input $b_0$

Constraint: $b_0 \times 10 \leq 100$

Symbolic output: $b_0 \times 10$
float func(float $b$)
{
    float $x$, $y$;
    $x = b \times 10$;
    if ($x > 100$)
        $y = 100$;
    else
        $y = x$;
    return $y$;
}

Concrete inputs $b = 5$
Symbolic input $b_0$

Constraint: $b_0 \times 10 \leq 100$
Symbolic output: $b_0 \times 10$

Strongest Postcondition:

$$(((b_0 \times 10 > 100 \land y = 100) \lor (b_0 \times 10 \leq 100 \land y = b_0 \times 10)))$$
Solve the following optimization problem:

maximize \( |y - \hat{y}| \) \hspace{1cm} // the difference in output

subject to

\( x \in [l, u] \) \hspace{1cm} // the range of inputs
\( |x - \hat{x}| < e \) \hspace{1cm} // the precision of the representation
\( y = f(x) \) \hspace{1cm} // the actual controller output
\( SP(F(\hat{x}, \hat{y})) \) \hspace{1cm} // the computed controller output

\( SP(F(\hat{x}, \hat{y})) \) - strongest postcondition
a logical formula relating input \( \hat{x} \) and output \( \hat{y} \) of function \( F \)
The bound on the quantization error can be computed using:

- off-the-shelf decision procedure and bisection method
- mixed-integer linear-programming-based optimization technique
- abstract interpretation based on interval arithmetic and affine arithmetic

Our experiments show that:

- SMT solvers are slowest, but produces the most accurate bound
- abstract interpretation based computation is the fastest, but the least accurate
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>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>
The control objective is to make the vehicle stable parallel to the x-axis at a certain distance $d$. 

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

For vehicle steering, $\rho = 0.0375m$.
The control objective is to make the vehicle stable parallel to the x-axis at a certain distance $d$.

For vehicle steering, $\rho = 0.0375m$.

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

$\rho = 3.75cm$
<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>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>
<tr>
<td><strong>Feasibility of Implementation</strong></td>
</tr>
<tr>
<td>Memoization Based Implementation</td>
</tr>
<tr>
<td>of Self-Triggered Controllers</td>
</tr>
<tr>
<td>[SMajumdar, EMSOFT2012]</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
Illustration of LQR Performance Criteria

State and Control Input for an LQR controller

State and Control Input for another stabilizing controller
Consider the following simple physical model of a bicycle [Astrom and Murray 2008]:

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

\[\eta = \begin{bmatrix}
\frac{av_0}{bh} & \frac{v_0^2}{bh}
\end{bmatrix} \begin{bmatrix}
\dot{\xi}_1 \\
\dot{\xi}_2
\end{bmatrix} + \nu\]

LQR Controller \((K_1)\)

\[
K_1 = [5.1538, 12.9724]
\]

LQR cost function is 264.1908

Another Controller \((K_2)\)

\[
K_2 = [3.0253, 12.6089]
\]

LQR cost function is 284.1578

The second controller has 7.58% more LQR cost
Figure: Evolution of the output $y$ with initial state $(0.5, 0.5)^T$ for the pair of gains $K_1$ and $K_2$ using 16-bit implementation.

The controller $K_2$ has similar LQR as the LQR controller $K_1$, but improves the region of practical stability significantly.
Design a controller optimizing the following objectives:

- The LQR cost
- 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
Objective Function for Controller Synthesis

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 \( K \)
Solving the Optimization Problem

We solve the non-convex optimization problem using particle swarm optimization (PSO)

- A population-based stochastic optimization approach
- Iteratively solves an optimization problem by maintaining a population of candidate solutions called particles
- Particles move around in the search-space of possible solutions, trying to minimize the objective function
- The algorithm stops after a fixed number of iterations or if the value of the global best solution does not change for long enough time
We solve the non-convex optimization problem using particle swarm optimization (PSO)

- A population-based stochastic optimization approach
- Iteratively solves an optimization problem by maintaining a population of candidate solutions called particles
- Particles move around in the search-space of possible solutions, trying to minimize the objective function
- The algorithm stops after a fixed number of iterations or if the value of the global best solution does not change for long enough time

In our setting, a particle represents gain parameters $K$ for a controller
Particle Swarm Optimization

\[ P = \text{Particle\_Initialization}() \]
\[ \text{For } i = 1 \text{ to } \text{it\_max} \]
\[ \quad \text{For each particle } p \text{ in } P \text{ do} \]
\[ \quad \quad fp = f(p) \]
\[ \quad \quad \text{If } fp \text{ is better than } f(p\text{Best}) \]
\[ \quad \quad \quad p\text{Best} = p \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ g\text{Best} = \text{best } p \text{ in } P \]
\[ \text{update the position of each particle} \]
\[ \text{end} \]
The tool is implemented in Matlab.

The tool uses a PSO function in Matlab by Ebbesen et al. [ACC 2012].

The tool uses a static analyzer written in OCaml that:
- synthesizes the best 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>
</tbody>
</table>
| Verification of Controller Software  
[AntaMajumdarSTabuada, EMSOFT 2010] |
| Synthesis of Controller Software  
[MajumdarSZamani, EMSOFT 2012] |
| **Optimization of Controller Software**  
[DarulovaKuncakMajumdarS, under submission] |
| **Schedulability** |
| Synthesis of Static Scheduler  
[MajumdarSZamani, EMSOFT 2011] |
| Synthesis of Dynamic Scheduler  
[SMajumdar, under preparation] |
| **Feasibility of Implementation** |
| Memoization Based Implementation of Self-Triggered Controllers  
[SMajumdar, EMSOFT 2012] |
Given: A real-valued polynomial expression $e$

Find: An expression $e_0$ that is
- equivalent to $e$ over the reals
- whose fixed-point implementation minimizes the error between the fixed-point value and the value of $e$ over the space of all inputs
Example

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

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

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

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

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

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

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

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
Example

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

Error bound in the best fixed-point implementation (16 bits): $3.9 \times 10^{-3}$

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

Error bound in the best fixed-point implementation (16 bits): $1.39 \times 10^{-3}$

Improvement 55%, without requiring any extra hardware
Example

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

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

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

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?

The problem is NP-hard
We Use Genetic Programming

**Objective function:** The bound on the error at the output of the fixed-point implementation of the synthesized expression.

**Mutation** and **Crossover** functions are defined on the AST of the controller expression.

The operations preserve the mathematical equivalence of the old expression and the new expressions.
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 particle
## Results

<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+
Future Plan
Exploring specific domain

E.g. Robot manipulators

**Question**: How to relate the implementation errors to the quality of tracking?
The duration for which a controller should be active is called **dwell time**

How is the dwell time affected by quantization error in the software implementation?
**Problem:** Synthesize a controller that has the optimal resiliency with respect to specific faults and attacks

Sources of faults:

- Single-Event-Upset in FPGA implementation
- Attack on controller software
Control Theory
- Real Analysis
- Convex Analysis

Program Analysis
- Symbolic Simulation
- Abstract Interpretation

Scheduling Theory
- Schedulability Analysis
- WCET Analysis
Summary

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

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

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

Indranil Saha  
Program Analysis and Synthesis for Control Applications  55/56
Summary

Control Theory
- Real Analysis
- Convex Analysis

Program Analysis
- Symbolic Simulation
- Abstract Interpretation

Scheduling Theory
- Schedulability Analysis
- WCET Analysis

Reliable Embedded Systems
Control Theory
- Real Analysis
- Convex Analysis
...

Program Analysis
- Symbolic Simulation
- Abstract Interpretation
...

Scheduling Theory
- Schedulability Analysis
- WCET Analysis
...

Reliable Embedded Systems

http://www.cs.ucla.edu/~indranil
Thank You!!