Closing the Gap in Control System Implementations

Indranil Saha

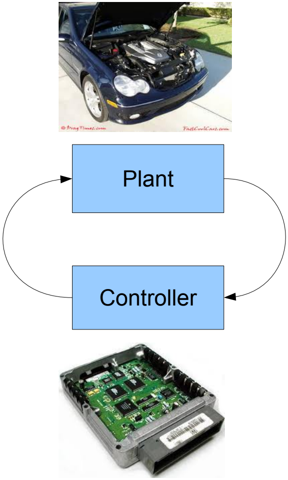
Department of Computer Science
University of California, Los Angeles

Application of Control Systems

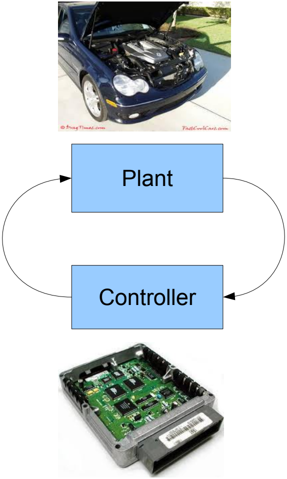
Application of Control Systems

The systems are mostly
life-critical or mission-critical

Controller Software: The Weak Link

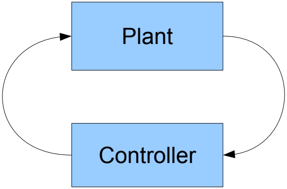


Controller Software: The Weak Link



1962 – Mariner I Space Probe Malfunction

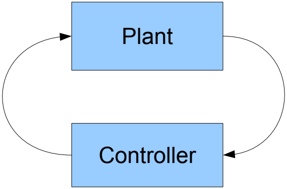
Controller Software: The Weak Link



1962 – Mariner I Space Probe Malfunction

1989 – Swedish Gripen Fighter Crash

Controller Software: The Weak Link

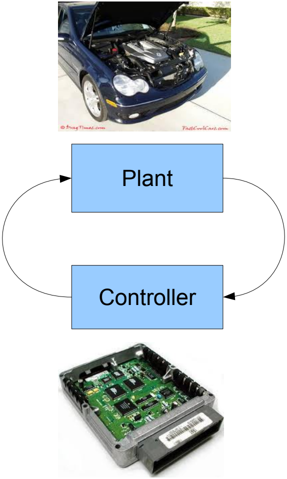


1962 – Mariner I Space Probe Malfunction

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

....

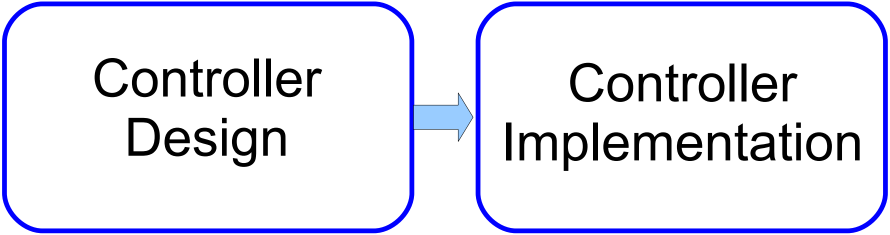
Research Goal

How can we develop
more reliable control systems?

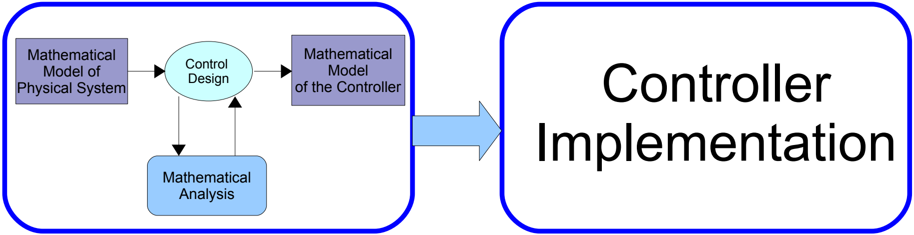
How can we develop
more reliable control systems?

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

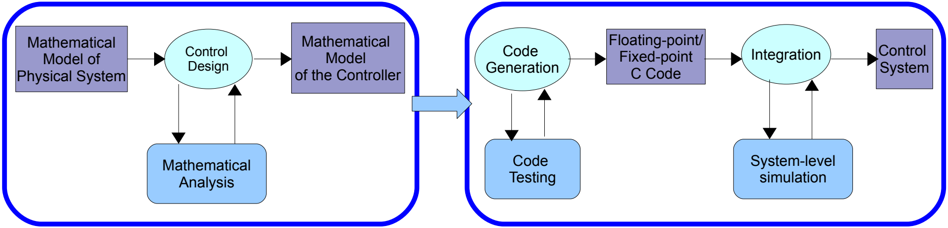
Control System Design and Implementation



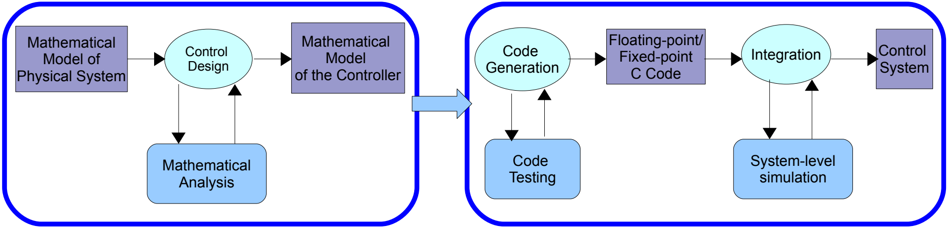
Design


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

Implementation



Gap between Design and Implementation



Infinite precision arithmetic

Negligible delay and computation time

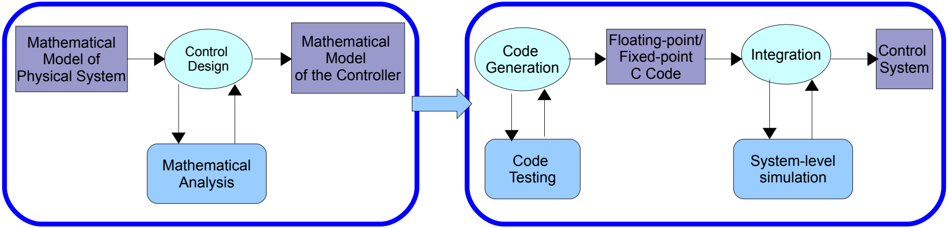
Ideal network

Finite precision arithmetic

Sharing of resources

Effect of network

Gap between Design and Implementation



Infinite precision arithmetic

Negligible delay and computation time

Ideal network

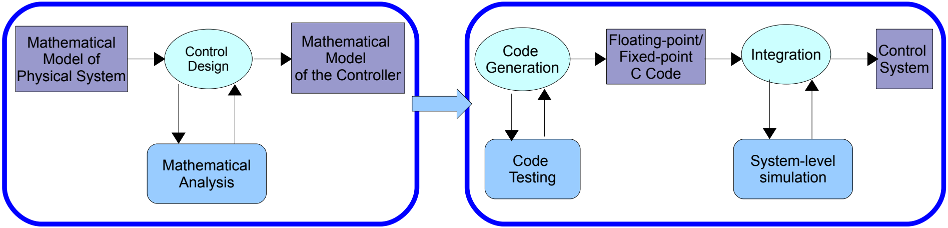
Finite precision arithmetic

Sharing of resources

Effect of network

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

Gap between Design and Implementation



Infinite precision arithmetic

Negligible delay and computation time

Ideal network

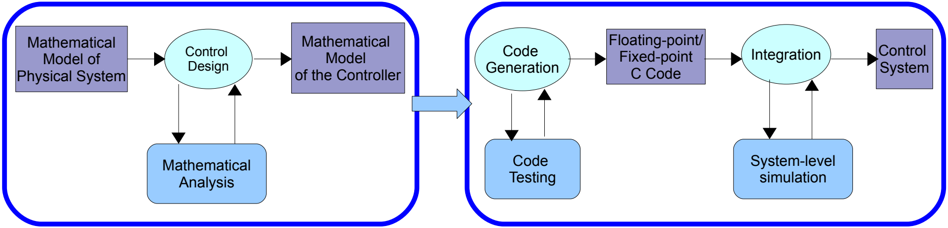
Finite precision arithmetic

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



Infinite precision arithmetic

Negligible delay and computation time

Ideal network

Finite precision arithmetic

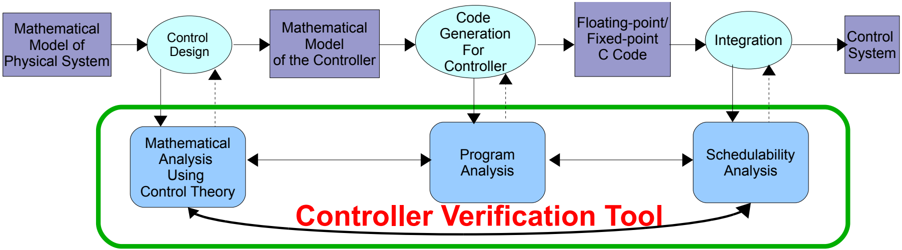
Sharing of resources

Effect of network

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

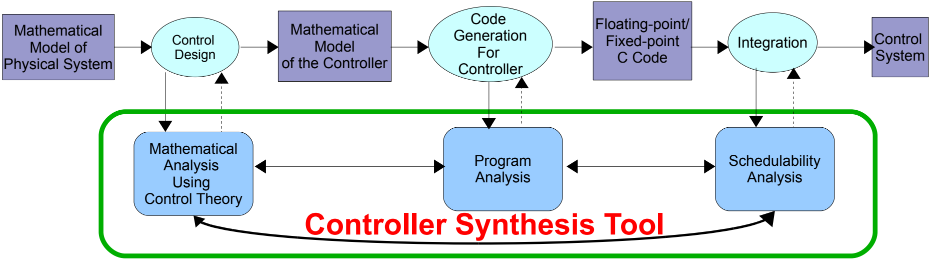
An end-to-end argument is missing

Formal Verification of the Implementation



Combine the results of different analysis techniques to give formal guarantee on the behavior of the implementation

Correct-by-construction Controller Synthesis



Take into account the implementation constraints during the design of the controller

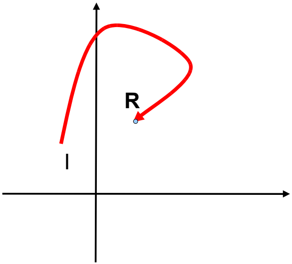
Research Contribution

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]

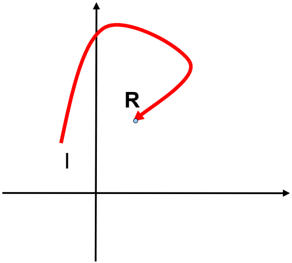
Research Contribution

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]

Definition: The plant converges to a desired behavior under the actions of the controller



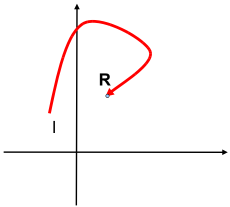
Definition: The plant converges to a desired behavior under the actions of the controller



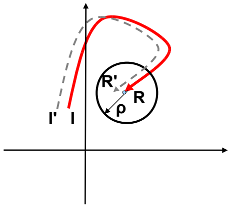
Example: Thermostat

In the steady state, the room temperature will be at 22C

Practical Stability



Mathematical Model

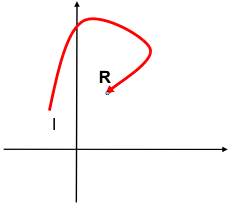


Software Implementation

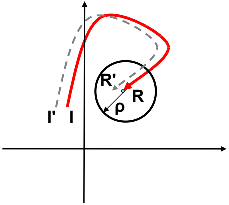
Stability property is replaced by **practical stability**

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

Practical Stability



Mathematical Model



Software Implementation

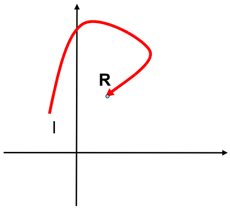
Stability property is replaced by **practical stability**

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

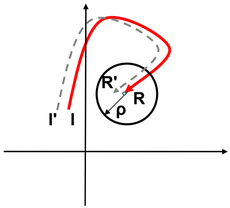
Example: Thermostat

In the steady state, the room temperature will be between 21.5C and 22.5C

Bound on the Region of Practical Stability

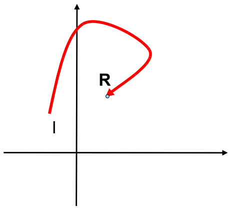


Mathematical Model

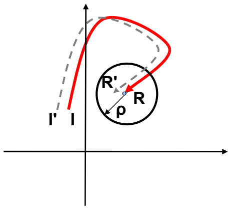


Software Implementation

Bound on the Region of Practical Stability



Mathematical Model

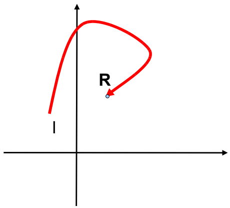


Software Implementation

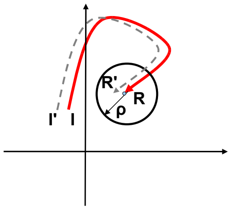
Theorem[AntaMajumdarSTabuada EMSOFT'10] If γ is the L2-Gain of a control system, b is a bound on the implementation error, then

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

Bound on the Region of Practical Stability



Mathematical Model



Software Implementation

Theorem[AntaMajumdarSTabuada EMSOFT'10] If γ 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) :

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

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);
}
```

Example of Controller Program

Control Law (Vehicle Steering) :

$$u = 0.81 \times (\mathrm{In1} - \mathrm{In2}) - 1.017 \times \mathrm{In3}$$

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):

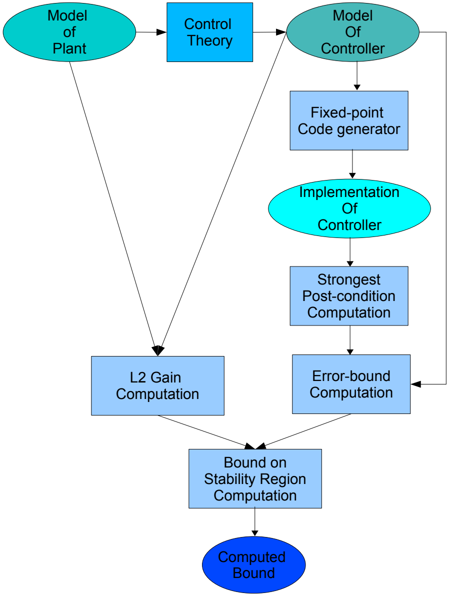
```
short int In1, In2, In3;
short int Subtract, Gain, Gain2, Out1;

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);
}
```

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

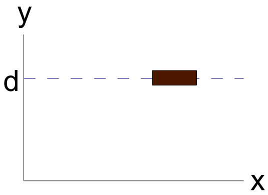
Example	Error bound	Bound on ρ	Run time
vehicle steering (16bit)	0.0163	0.0375	1m
pendulum (16bit)	0.0508	0.1806	3m
dc motor (16bit)	0.0473	1.0889	2m
train car - 1 car (32bit)	5e-7	2.6080e-5	3m
train car - 2 cars (32bit)	1.5e-6	9.4000e-5	6m
train car - 3 cars (32bit)	8.5e-6	0.0010	10m
train car - 4 cars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

Experimental Results

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

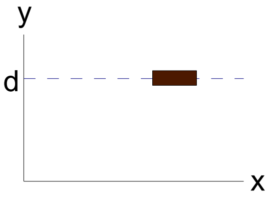
Example: Vehicle Steering

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



Example: Vehicle Steering

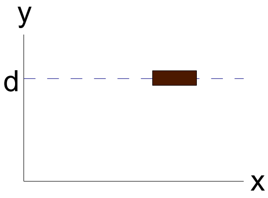
The control objective is to make the vehicle stable parallel to the x-axis at a certain distance *d*



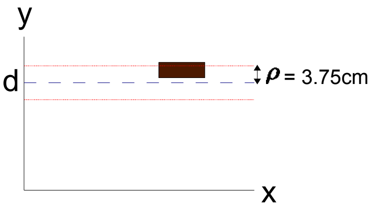
For vehicle steering, $\rho = 0.0375m$

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, $\rho = 0.0375m$

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]

Synthesis Question

Is it possible to synthesize a controller that minimizes the region of practical stability?

Synthesis Question

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:

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

$$\eta = \begin{bmatrix} \frac{av_0}{bh} & \frac{v_0^2}{bh} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \nu$$

LQR Controller:

$K_1 = [5.1538, \ 12.9724]$

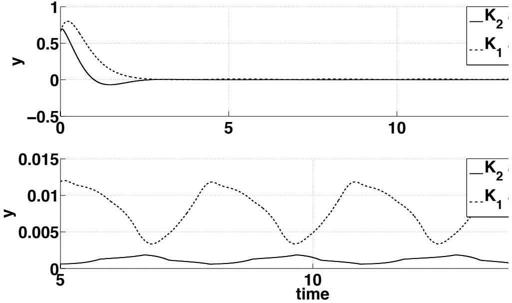
LQR cost function is 264.1908

Another Controller:

$K_2 = [3.0253, \ 12.6089]$

LQR cost function is 284.1578

Example (Cont.)



There is a trade-off between LQR cost and the region of practical stability

Problem Statement

Design a controller optimizing the following objectives:

- The LQR cost
- The bound on the region of practical stability

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

γ^* - 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:

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

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

γ^* - 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

γ^* - 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

Control systems	LQR cost ratio $\frac{\text{Synthesized Controller}}{\text{LQR Controller}}$	Steady state error ratio $\frac{\text{LQR Controller}}{\text{Synthesized Controller}}$
Bicycle	1.095	10.694
DC motor	1.3745	14.545
Pitch angle	1.005	5.88
Inverted Pendulum	1.244	5.023
Batch reactor	1.00029	2.554

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]

Control Law (Vehicle Steering) :

$$u = 0.81 \times (\mathrm{In1} - \mathrm{In2}) - 1.017 \times \mathrm{In3}$$

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);
}
```

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

$$
\begin{aligned}
\text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\
& (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\
& (-0.0003) * \text{y1} + 0.0020 * \text{y2}
\end{aligned}
$$

Example: Batch Reactor Process

$$\begin{aligned} \text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\ & (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\ & (-0.0003) * \text{y1} + 0.0020 * \text{y2} \end{aligned}$$

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

Example: Batch Reactor Process

$$\begin{aligned}\text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\ & (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\ & (-0.0003) * \text{y1} + 0.0020 * \text{y2}\end{aligned}$$

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

$$\begin{aligned}\text{out} =& ((0.9052 * \text{state2}\ ) + (((\text{state3}\ \ * -0.0181) + \\ & (-0.0078 * \text{state1}\ )) + (((-0.0392 * \text{state4}\ ) + \\ & (-0.0003 * \text{y1}\ )) + (0.002 * \text{y2}\ ))))\end{aligned}$$

Example: Batch Reactor Process

$$\begin{aligned}\text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\ & (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\ & (-0.0003) * \text{y1} + 0.0020 * \text{y2}\end{aligned}$$

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

$$\begin{aligned}\text{out} =& ((0.9052 * \text{state2}\ ) + (((\text{state3}\ * -0.0181) + \\ & (-0.0078 * \text{state1}\ )) + (((-0.0392 * \text{state4}\ ) + \\ & (-0.0003 * \text{y1}\ )) + (0.002 * \text{y2}\ ))))\end{aligned}$$

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

Example: Batch Reactor Process

$$\begin{aligned} \text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\ & (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\ & (-0.0003) * \text{y1} + 0.0020 * \text{y2} \end{aligned}$$

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

$$\begin{aligned} \text{out} =& ((0.9052 * \text{state2}\ ) + (((\text{state3}\ * -0.0181) + \\ & (-0.0078 * \text{state1}\ )) + (((-0.0392 * \text{state4}\ ) + \\ & (-0.0003 * \text{y1}\ )) + (0.002 * \text{y2}\ )))) \end{aligned}$$

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

Improvement   55%, without requiring any extra hardware

Example: Batch Reactor Process

$$\begin{aligned} \text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\ & (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\ & (-0.0003) * \text{y1} + 0.0020 * \text{y2} \end{aligned}$$

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

$$\begin{aligned} \text{out} =& ((0.9052 * \text{state2}\ ) + (((\text{state3}\ * -0.0181) + \\ & (-0.0078 * \text{state1}\ )) + (((-0.0392 * \text{state4}\ ) + \\ & (-0.0003 * \text{y1}\ )) + (0.002 * \text{y2}\ )))) \end{aligned}$$

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

$$\begin{aligned}\text{out} =& (-0.0078) * \text{state1} + 0.9052 * \text{state2} + \\ & (-0.0181) * \text{state3} + (-0.0392) * \text{state4} + \\ & (-0.0003) * \text{y1} + 0.0020 * \text{y2}\end{aligned}$$

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

$$\begin{aligned}\text{out} =& ((0.9052 * \text{state2}\ ) + (((\text{state3}\ * -0.0181) + \\ & (-0.0078 * \text{state1}\ )) + (((-0.0392 * \text{state4}\ ) + \\ & (-0.0003 * \text{y1}\ )) + (0.002 * \text{y2}\ ))))\end{aligned}$$

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

Results

Control systems	Region of Practical Stability			Improvement (%)	
	Baseline	Improved	Optimal	Improved	Optimal
bicycle	7.85e-02	7.70e-02	6.99e-02	1.93	10.96
dc motor	1.64e-02	1.44e-02	9.80e-03	12.14	40.24
pitch angle	1.08e-02	8.87e-03	5.15e-03	18.00	52.32
pendulum	3.11e-04	2.64e-04	2.51e-04	14.76	19.26
batch reactor	2.59e-01	2.24e-01	2.07e-01	13.31	20.08

Baseline - Controller synthesized by Ocsyn

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

Optimal - Controller synthesized by Ocsyn+

Moral

Control Theory + Software Analysis/Synthesys

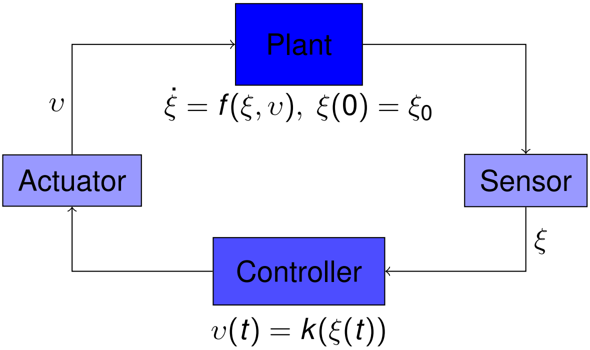
can provide

Reliability Guarantee on the Implemented System

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]

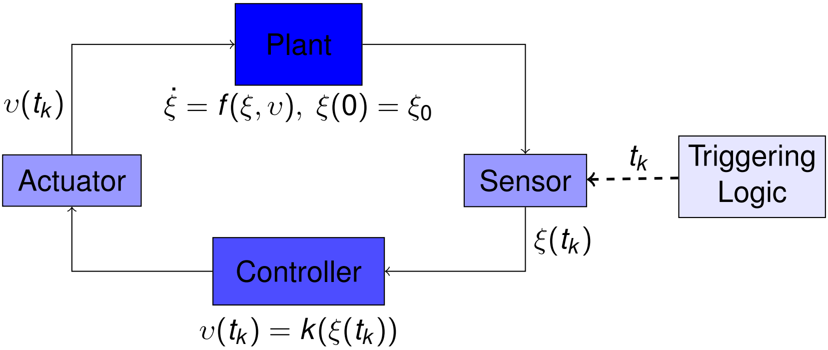
A Control System



ξ - State of the plant

υ - Control signal generated by the controller

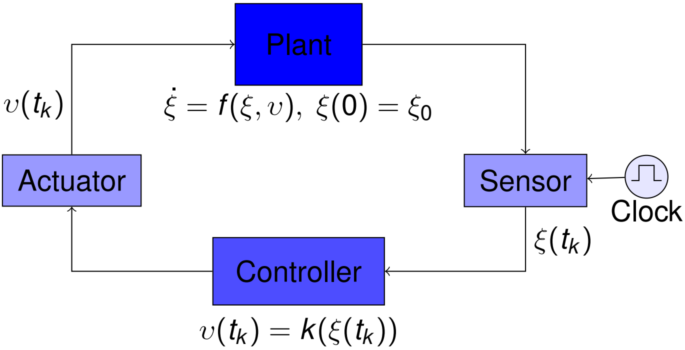
Implementation of a Control System



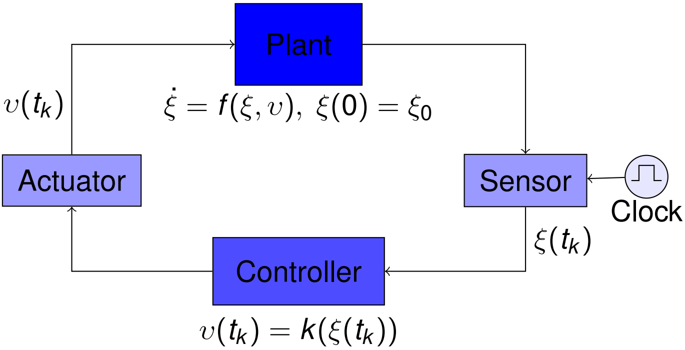
To implement the control law on a digital computer, the state of the plant is sampled at a sequence of time instants $t_0 = 0,\ t_1,\ t_2, \ldots$

The time instant t_k is called the trigger time

Time-Triggered Implementation



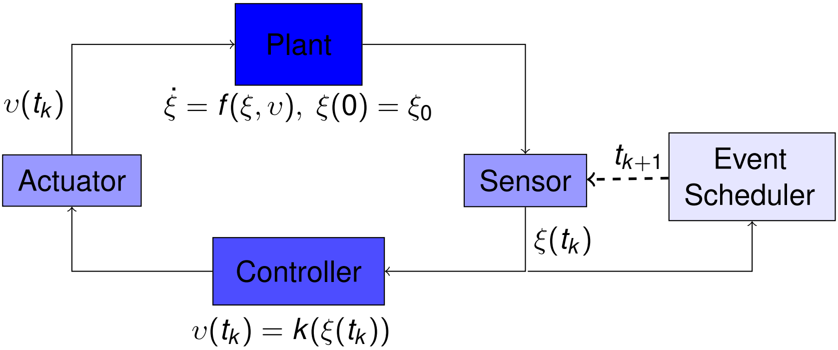
Time-Triggered Implementation



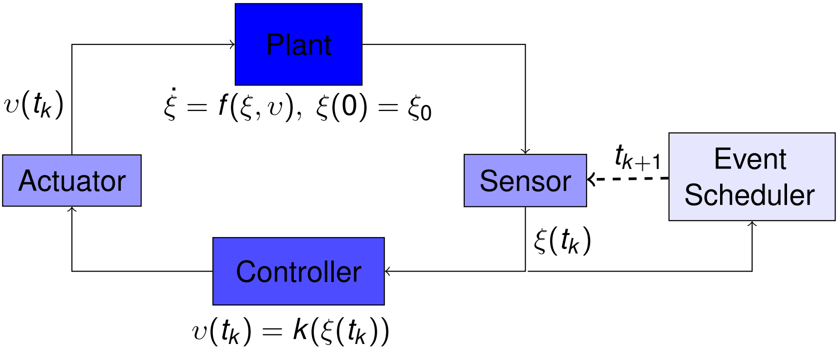
Sampling period is selected based on the worst case scenario

Inefficient usage of computational resource and communication bandwidth

Self-Triggered Implementation



Self-Triggered Implementation



Has been shown to reduce the number of control computations significantly with respect to its time-triggered counterpart

Trigger Time Computation

Trigger time is computed based on two parameters:

- τ_{min} : minimum trigger time
  - The trigger-time which works in the worst case scenario
  - Can be computed from the parameters of the control system
- τ_{max}: maximum trigger time
  - The maximum duration the plant can be kept open loop
  - A design parameter

$$(t_k + \tau_{min}) \leq t_{k+1} \leq (t_k + \tau_{max})$$

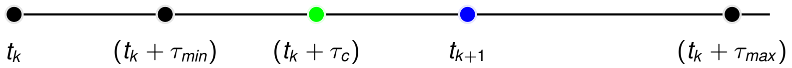
Computation Time of Trigger Time

τ_c - The time required to compute the trigger time

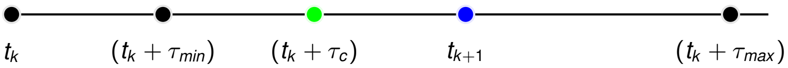
The self-triggered implementation scheme is feasible if and only if

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

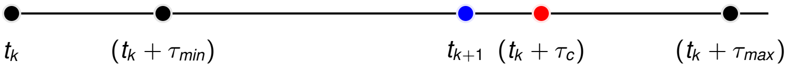
The Problem



The Problem

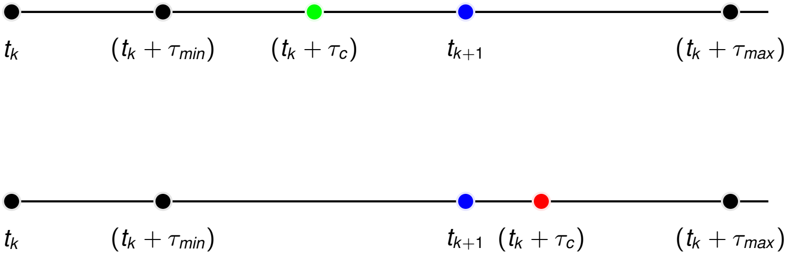


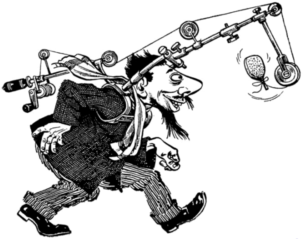
t_k
$(t_k + \tau_{min})$
$(t_k + \tau_c)$
t_{k+1}
$(t_k + \tau_{max})$



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

The Problem


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}
$(t_k + \tau_c)$
$(t_k + \tau_{max})$



An Example

The model of a batch reactor process is given by

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

The feedback controller

$$\upsilon = - \begin{bmatrix} 0.1006 & -0.2469 & -0.0952 & -0.2447 \\ 1.4099 & -0.1966 & 0.0139 & 0.0823 \end{bmatrix} \xi$$

renders the system exponentially stable.

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

An Example

The model of a batch reactor process is given by

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

The feedback controller

$$\upsilon = - \begin{bmatrix} 0.1006 & -0.2469 & -0.0952 & -0.2447 \\ 1.4099 & -0.1966 & 0.0139 & 0.0823 \end{bmatrix} \xi$$

renders the system exponentially stable.

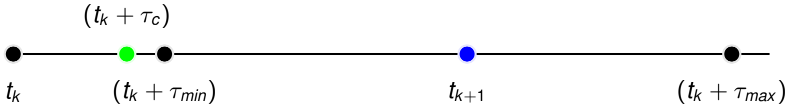
For this system, $\tau_{\text{min}} = 18ms$
Following literature we chose $\tau_{\text{max}} = 358ms$
On a Leon 2 processor with frequency $100MHz$, the WCET of the trigger time computation is $29.793ms$

It is possible that $(\mathbf{t_k} + \tau_\mathbf{c}) > \mathbf{t_{k+1}}$

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

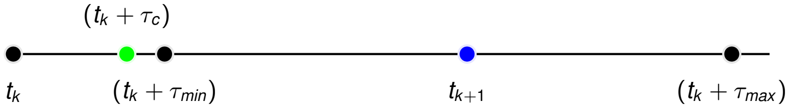
Proposed Solution

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



Proposed Solution

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



- Continue trigger-time computation as a background task, and memoize the result of the computation
  - Trigger-time is computed based on quantized state

Memoization of Trigger Time

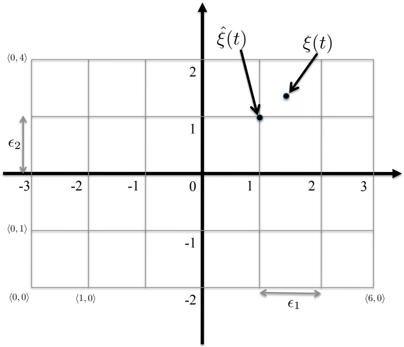


Figure: Memoization region and table

The state $(1.4, 1.3)$ is quantized to $(1, 1)$

The trigger time corresponding to the state $(1, 1)$ is stored in $Memo[4, 3]$

Effect of State Quantization

- 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

Moral

- Program analysis is helpful in detecting infeasibility of implementation
- Classical software engineering techniques can be helpful in the implementation of control systems

Research Contribution

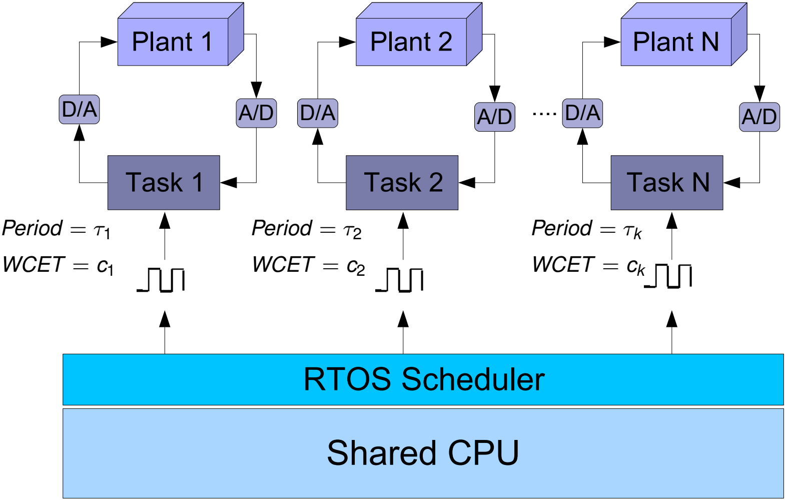
Stability	Verification of Controller Software [AntaMajumdarSTabuada, EMSOFT 2010] Synthesis of Controller Software [MajumdarSZamani, EMSOFT 2012] Optimization of Controller Software [DarulovaKuncakMajumdarS, under submission]
Feasibility of Implementation	Memoization Based Implementation of Self-Triggered Controllers [SMajumdar, EMSOFT2012]
Schedulability	**Synthesis of Static Scheduler** [MajumdarSZamani, EMSOFT 2011] Synthesis of Dynamic Scheduler [SMajumdar, under preparation]

Integrated Architectures for Complex Cyber-Physical Systems

- 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



Hard Real-Time Scheduling

- 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 → Implement

  System not schedulable → Send back to designer

  Or: Throw more resources at it

Not-So-Hard Real-Time Scheduling

- 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

Not-So-Hard Real-Time Scheduling

- 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?

Relate Successful Computation Rate to Stability

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

Relate Successful Computation Rate to Performance

- 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

Relate Successful Computation Rate to Performance

- 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

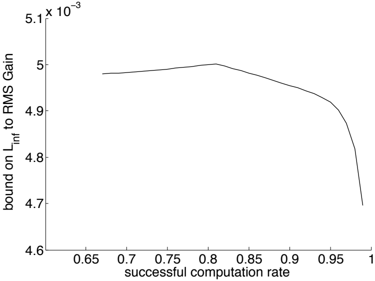
Theorem: The bound on the $\mathcal{L}_\infty$ to RMS gain of the discrete time LTI control system attains the minimum value for the successful computation rate to be either at $r_{\min}$ or at $r_{\max}$

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 $\mathcal{L}_\infty$ to RMS Gain is minimized

The problem is NP-Hard

- Reduction is from Multiple-Choice Knapsack Problem

Our Approach

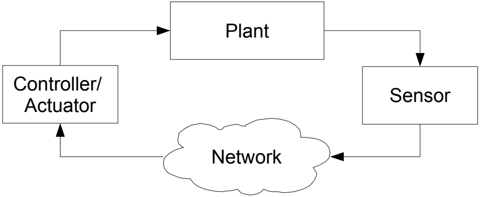
- Find r_{min} for each control system
- Find r_{max} for all control systems
  - Maximize weighted sum of successful computation rates
  - Weights are based on the priorities of the control systems
- Select $r_{opt} \in [r_{min}, r_{max}]$ such that the performance is the best
- Synthesize a scheduler based on the selected rates

  - We provide an constraint solving based static scheduler synthesis algorithm

Research Contribution

Stability	Verification of Controller Software [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]

Networked Control Systems



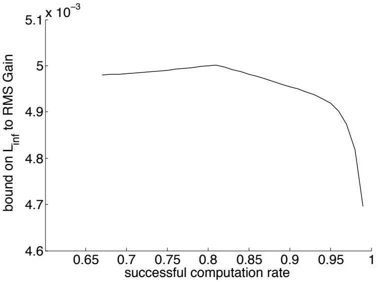
Network introduces delay and packet dropout

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

Static scheduler is not feasible

$\gamma_m(r)$ - the mean of the $\mathcal{L}_\infty$ 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 $[r_{\min} + r_{net}, r_{\max}]$

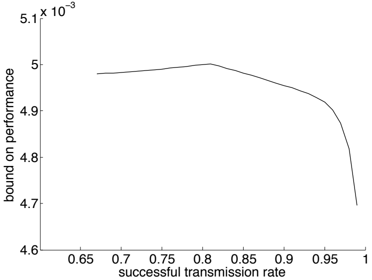


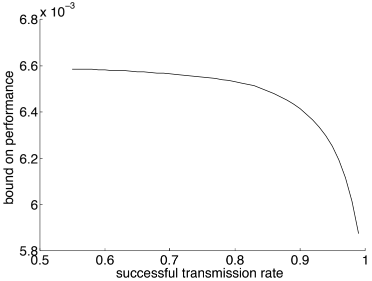
Theorem: The operating successful computation rate (r_{opr}) is either $r_{\max}$ or $r_{\min} + r_{net}$

Dynamic Scheduler

- Follows EDF strategy
- Maintains the successful computation rate in the range $[r_{opr} - r_{net}, r_{opr}]$ by suitably dropping control computation

Controller Scheduler Co-design





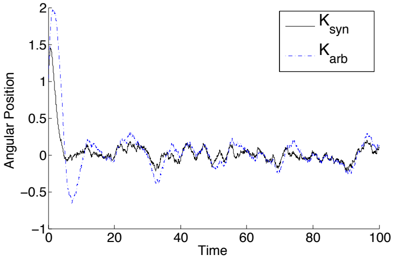
Performance profile of two controllers may be quite different

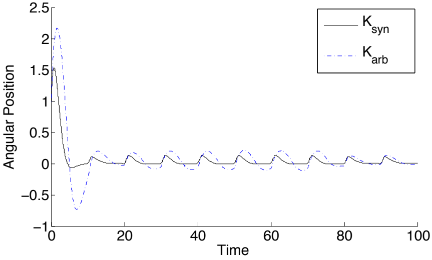
Problem: Given the scheduling constraints, synthesize a controller to achieve optimal performance

Controller is synthesized using stochastic local search

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

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





Moral

Control Theory + Schedulability Analysis

gives

better end-to-end performance

Summary

Control Theory
- Real Analysis
- Convex Analysis

....

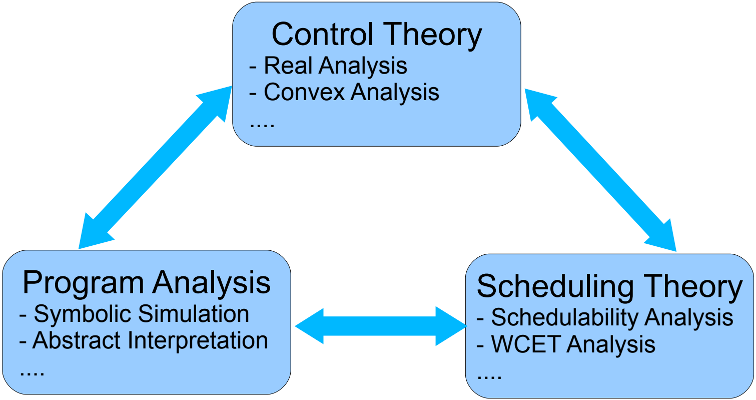
Program Analysis
- Symbolic Simulation
- Abstract Interpretation

....

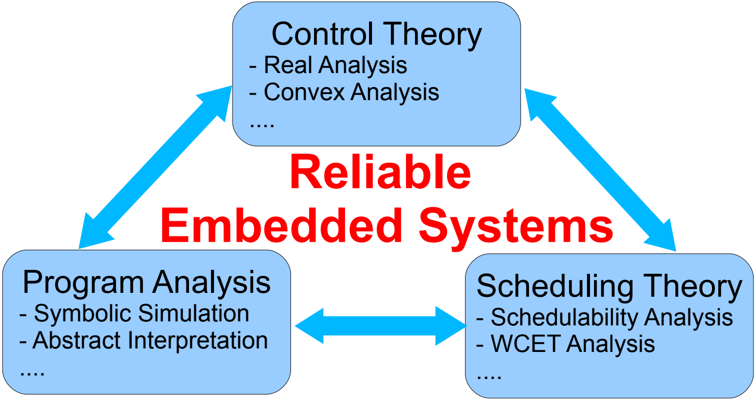
Scheduling Theory
- Schedulability Analysis
- WCET Analysis

....

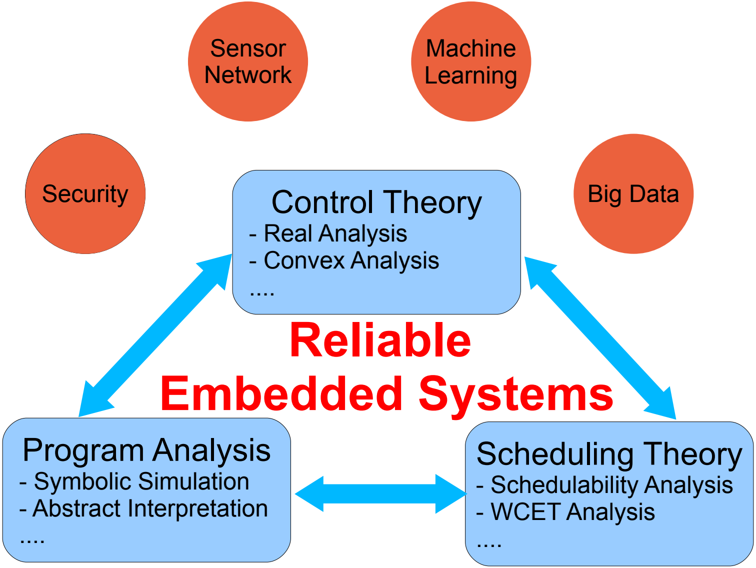
Summary


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
....
Reliable
Embedded Systems
Program Analysis
- Symbolic Simulation
- Abstract Interpretation
....
Scheduling Theory
- Schedulability Analysis
- WCET Analysis
....

Looking Ahead



Publications

- Adolfo Anta, Rupak Majumdar, Indranil Saha, Paulo Tabuada. Automatic Verification of Control System Implementations. **EMSOFT 2010**. Best Paper Award
- Rupak Majumdar, Indranil Saha, Majid Zamani. Performance-Aware Scheduler Synthesis for Control Systems. **EMSOFT 2011**.
- Rupak Majumdar, Indranil Saha, Majid Zamani. Synthesis of Minimal Error Control Software. **EMSOFT 2012**. Best Paper Nomination
- 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.
- Indranil Saha and Rupak Majumdar. Performance aware synthesis of networked control systems. Ready for 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