Synthesis of Minimal-Error Control Software

Rupak Majumdar\textsuperscript{1,2}, Indranil Saha\textsuperscript{1} and Majid Zamani\textsuperscript{3}

\textsuperscript{1}University of California, Los Angeles

\textsuperscript{2}Max Planck Institute for Software Systems

\textsuperscript{3}Delft University of Technology

EMSOFT 2012
October 8, 2012
Discrete-Time Linear Control System

\[ x[r + 1] = A \tau x[r] + B \tau u[r] + \bar{B} \tau d[r] \]
\[ y[r] = Cx[r] + v[r] \]

- \( x \) - state of the plant
- \( y \) - output of the plant as observed by the controller
- \( u \) - control signal generated by the controller
- \( d \) - disturbance input
- \( v \) - measurement noise
Discrete-Time Linear Control System

Plant

\[
x[r + 1] = A_\tau x[r] + B_\tau u[r] + B_\tau d[r]
\]
\[
y[r] = Cx[r] + v[r]
\]

Actuator

Controller

Sensor

\[
\hat{x}[r + 1] = (A_\tau - B_\tau K - LC)\hat{x}[r] + LCx[r] + Lv[r]
\]
\[
u[r] = -K\hat{x}[r]
\]
Synthesizing a controller deals with finding out suitable values for $K$ and $L$ so that
- stability of the plant is ensured
- desired control performance is achieved
The LQR cost function to be minimized is given by:

\[ J_{LQR} = \sum_{r=0}^{+\infty} \{ x[r]^T Q x[r] + u[r]^T R u[r] \} \]

for some chosen weight matrices \( Q \) and \( R \) that are positive definite and of appropriate dimensions.

The LQG cost function to be minimized is given by:

\[ J_{LQG} = \lim_{r \to +\infty} \mathbb{E} \left[ \| e[r] \|^2 \right] \]

where \( \mathbb{E} \) stands for expected value and \( e \) is the estimation error whose dynamic is given by:

\[ e[r] = x[r] - \hat{x}[r] \]

The controller that minimizes the LQR and LQG cost functions is referred as **LQR/LQG controller**.
Now the controller has to be implemented as a software program...

Use of finite precision arithmetic introduces quantization error at the output of the controller.
Now the controller has to be implemented as a software program...

Use of finite precision arithmetic introduces quantization error at the output of the controller.
Now the controller has to be implemented as a software program...

Use of finite precision arithmetic introduces quantization error at the output of the controller

What is the effect of the quantization error on the behavior of the control system?
The fixed point implementation of the controller introduces quantization error in the control signal.

The overall dynamics of the controller implementation:

\[
\begin{align*}
\hat{x}[r+1] &= (A_\tau - B_\tau K - LC)\hat{x}[r] + LCx[r] + Lv[r] + e_{q1} \\
u[r+1] &= K\hat{x}[r] + e_{q2}
\end{align*}
\]

- $e_{q1}$ - quantization error in the implementation of the observer dynamics
- $e_{q2}$ - quantization error in the implementation of the control signal computation
Proposition [Anta et al. EMSOFT 2010] Consider the discrete-time linear system:

\[ x[r + 1] = Ax[r] \]

and assume that the origin is an asymptotically stable equilibrium point. Then, for any signal \( d : \mathbb{N}_0 \to \mathbb{R}^m \) satisfying \( \|d[r]\| \leq b(d) \) for any \( r \in \mathbb{N}_0 \) and some constant \( b(d) \in \mathbb{R}^+_0 \), the output \( y =Cx \) of the system:

\[ x[r + 1] = Ax[r] + Bd[r] \]

is guaranteed to converge to the set:

\[ \mathcal{A}_y = \{ y \in \mathbb{R}^p \mid \|y\| \leq \gamma_y b(d) \} \]

with:

\[ \gamma_y = \max_{\theta \in [0, 2\pi]} \left\| C \left( e^{i\theta} I_n - A \right)^{-1} B \right\| \]
The overall dynamics of the control system is:

\[
\begin{align*}
    x[r + 1] &= A_\tau x[r] - B_\tau K \hat{x}[r] + \overline{B}_\tau d[r] + B_\tau e_{q2} \\
    \hat{x}[r + 1] &= (A_\tau - B_\tau K - LC)\hat{x}[r] + LCx[r] + Lv[r] + e_{q1}
\end{align*}
\]

The control system can be rewritten as follows:

\[
    w[r + 1] = Gw[r] + H_1 e_1[r] + H_2 e_2[r]
\]

with:

\[
    w = \begin{bmatrix} x \\ \hat{x} \end{bmatrix}, \quad e_1 = \begin{bmatrix} d \\ v \end{bmatrix}, \quad e_2 = \begin{bmatrix} e_{q1} \\ e_{q2} \end{bmatrix}
\]

\[
    G = \begin{bmatrix} A_\tau & -B_\tau K \\ LC & A_\tau - B_\tau K - LC \end{bmatrix}, \quad H_1 = \begin{bmatrix} \overline{B}_\tau & 0_{n \times p} \\ 0_{n \times q} & L \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0_{n \times n} & B_\tau \\ I_n & 0_{n \times m} \end{bmatrix}
\]
For any input $e_1$ and $e_2$ satisfying $\|e_1[r]\| \leq b(e_1)$ and $\|e_2[r]\| \leq b(e_2)$ for any $r \in \mathbb{N}_0$ and some constants $b(e_1), b(e_2) \in \mathbb{R}_0^+$, the output $y = Cx$ is guaranteed to converge to the set:

$$A_y = \{ y \in \mathbb{R}^p \mid \|y\| \leq \gamma_1 y b(e_1) + \gamma_2 y b(e_2) \}$$

where $\gamma_1 y$ and $\gamma_2 y$ are given by:

$$\gamma_{jy} = \max_{\theta \in [0, 2\pi]} \left\| \begin{bmatrix} C & 0_{p \times n} \end{bmatrix} \left( e^{i\theta} l_{2n} - G \right)^{-1} H_j \right\| \quad \text{for } j = 1, 2$$

$\gamma_1 y$ and $\gamma_2 y$ are called $\mathcal{L}_2$ Gain of the control system.

The set $A_y$ is called the region of practical stability.
For any input $e_1$ and $e_2$ satisfying $\|e_1[r]\| \leq b(e_1)$ and $\|e_2[r]\| \leq b(e_2)$ for any $r \in \mathbb{N}_0$ and some constants $b(e_1), b(e_2) \in \mathbb{R}^+_0$, the output $y = Cx$ is guaranteed to converge to the set:

$$A_y = \{ y \in \mathbb{R}^p | \|y\| \leq \gamma_1 y b(e_1) + \gamma_2 y b(e_2) \}$$

where $\gamma_1 y$ and $\gamma_2 y$ are given by:

$$\gamma_{jy} = \max_{\theta \in [0, 2\pi]} \left\| \begin{bmatrix} C & 0_{p \times n} \end{bmatrix} \left( e^{i\theta} l_{2n} - G \right)^{-1} H_j \right\| \quad \text{for } j = 1, 2$$

$\gamma_1 y$ and $\gamma_2 y$ are called $\mathcal{L}_2$ Gain of the control system.

The set $A_y$ is called the region of practical stability.
For any input $e_1$ and $e_2$ satisfying $\|e_1[r]\| \leq b(e_1)$ and $\|e_2[r]\| \leq b(e_2)$ for any $r \in \mathbb{N}_0$ and some constants $b(e_1), b(e_2) \in \mathbb{R}_0^+$, the output $y = Cx$ is guaranteed to converge to the set:

$$A_y = \{y \in \mathbb{R}^p \mid \|y\| \leq \gamma_1 b(e_1) + \gamma_2 b(e_2)\}$$

where $\gamma_1 y$ and $\gamma_2 y$ are given by:

$$\gamma_{jy} = \max_{\theta \in [0, 2\pi]} \left\| [C \ 0_{p \times n}] \left( e^{i\theta} I_{2n} - G \right)^{-1} H_j \right\| \text{ for } j = 1, 2$$

$\gamma_1 y$ and $\gamma_2 y$ are called $\mathcal{L}_2$ Gain of the control system.

The set $A_y$ is called the region of practical stability.
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/LQG Controller \((K_1, L_1)\)**

\[
K_1 = [5.1538, 12.9724]
\]

\[
L_1 = [0.0317, 0.0118]^T
\]

LQR cost function is **264.1908**

LQG cost function is **0.0229**

**Another Controller \((K_2, L_2)\)**

\[
K_2 = [3.0253, 12.6089]
\]

\[
L_2 = [0.0132, 0.1021]^T
\]

LQR cost function is **284.1578**

LQG cost function is **0.0246**

The second controller has **7.58%** more LQR cost and **7.42%** more LQG cost
Figure: Evolution of the output $y$ with initial state $(0.5, 0.5)^T$ for the pair of gains $(K_1, L_1)$ and $(K_2, L_2)$ using 16-bit implementation.

The controller $(K_2, L_2)$ has similar LQR and LQG cost as the LQR/LQG controller $(K_1, L_1)$, but improves the region of practical stability significantly.
Design a controller optimizing the following objectives:

- The LQR and the LQG costs for performance
- Error caused by disturbance and measurement noise
- The implementation error given by a fixed-precision encoding
For a stabilizing controller $K$ and any initial state $x[0]$, the upper bound of the LQR cost is given by

$$J_{LQR} = \|S(K)\| \times \|x[0]\|^2$$

$S(K) \in \mathbb{R}^{n \times n}$ is a positive definite matrix that is the unique solution of the Lyapunov equation for $S$:

$$(A_\tau - B_\tau K)^T S (A_\tau - B_\tau K) - S + Q + K^T R K = 0$$
For an Observer gain $L$, the LQG cost function can be rewritten as

$$J_{LQG} = \| P(L) \|$$

$P(L) \in \mathbb{R}^{n \times n}$ is a positive definite matrix that is the unique solution for $P$ to the Lyapunov equation:

$$(A_T - LC) P (A_T - LC)^T - P + B_T \hat{Q} B_T^T + L \hat{R} L^T = 0$$
A fixed-point representation of a real number is a triple \( \langle s, n, m \rangle \) consisting of
- a sign bit indicator \( s \in \{1, 0\} \) (for signed and unsigned)
- a length \( n \in \mathbb{N} \)
- a length of the fractional part \( m \in \mathbb{N} \)

An integer variable \( \hat{x} \) that represents a fixed-point variable with type \( \langle 0, n, m \rangle \) can be interpreted as the rational number \( 2^{-m} \hat{x} \)

The maximum truncation error in a fixed-point variable with representation \( \langle s, n, m \rangle \) is \( 2^{-m} \)

If the range for a variable is known, its best fixed-point datatype can be uniquely determined
A fixed-point representation of a real number is a triple $\langle s, n, m \rangle$ consisting of

- a sign bit indicator $s \in \{1, 0\}$ (for signed and unsigned)
- a length $n \in \mathbb{N}$
- a length of the fractional part $m \in \mathbb{N}$

An integer variable $\hat{x}$ that represents a fixed-point variable with type $\langle 0, n, m \rangle$ can be interpreted as the rational number $2^{-m}\hat{x}$

- The maximum truncation error in a fixed-point variable with representation $\langle s, n, m \rangle$ is $2^{-m}$
- If the range for a variable is known, its best fixed-point datatype can be uniquely determined
A fixed-point representation of a real number is a triple \( \langle s, n, m \rangle \) consisting of

- a sign bit indicator \( s \in \{1, 0\} \) (for signed and unsigned)
- a length \( n \in \mathbb{N} \)
- a length of the fractional part \( m \in \mathbb{N} \)

An integer variable \( \hat{x} \) that represents a fixed-point variable with type \( \langle 0, n, m \rangle \) can be interpreted as the rational number \( 2^{-m} \hat{x} \)

The maximum truncation error in a fixed-point variable with representation \( \langle s, n, m \rangle \) is \( 2^{-m} \)

If the range for a variable is known, its best fixed-point datatype can be uniquely determined.
A fixed-point representation of a real number is a triple \( \langle s, n, m \rangle \) consisting of

- a sign bit indicator \( s \in \{1, 0\} \) (for signed and unsigned)
- a length \( n \in \mathbb{N} \)
- a length of the fractional part \( m \in \mathbb{N} \)

An integer variable \( \hat{x} \) that represents a fixed-point variable with type \( \langle 0, n, m \rangle \) can be interpreted as the rational number \( 2^{-m}\hat{x} \)

The maximum truncation error in a fixed-point variable with representation \( \langle s, n, m \rangle \) is \( 2^{-m} \)

If the range for a variable is known, its best fixed-point datatype can be uniquely determined
Fixed-Point Arithmetic

Example: \( s: c = a + b \)

\( a: \langle 0, n, k_1 \rangle, \ b: \langle 0, n, k_2 \rangle, \ c: \langle 0, n, k_3 \rangle \)

\( k_2 > k_1, \ k_3 = k_1 \)

The fixed-point implementation:

\[
fp(s) : \hat{c} = \hat{a} + (\hat{b} \gg (k_2 - k_1))
\]
Given:

- A linear expression: \( u = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n, \quad c_i \in \mathbb{R} \)
- A compact subset \( y_i \subset \mathbb{R} \)
- Number of bits in the fixed-point datatype

Compute:

The bound on the quantization error in the fixed-point implementation of the expression
Apply a mixed-integer linear-programming-based optimization technique

The error computation is compositional

\[ u = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n \]
Apply a mixed-integer linear-programming-based optimization technique

The error computation is compositional

\[ u = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n \]
Apply a mixed-integer linear-programming-based optimization technique

The error computation is compositional

\[ u = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n \]
Apply a mixed-integer linear-programming-based optimization technique

The error computation is compositional

\[ u = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n \]
Apply a mixed-integer linear-programming-based optimization technique

The error computation is compositional

\[ u = c_1 y_1 + c_2 y_2 + \ldots + c_n y_n \]
Computing Quantization Error

\( s : a = b \ op \ c, \) where \( \op \in \{+, -, \ast\} \)

**Ranges:** \( a : [l_a, u_a], \) \( b : [l_b, u_b], \) and \( c : [l_c, u_c] \)

**Fixed-point representation:** \( a : \langle 1, n_a, m_a \rangle, \) \( b : \langle 1, n_b, m_b \rangle, \)
\( c : \langle 1, n_c, m_c \rangle \)

The optimization problem to find the bound on the error is:

\[
\text{maximize} \quad |a - 2^{-m_a} \hat{a}|
\]
\[
\text{subject to} \quad l_b \leq b \leq u_b, \quad l_c \leq c \leq u_c
\]
\[
|b - 2^{-m_b} \ast \hat{b}| \leq b(e_b)
\]
\[
|c - 2^{-m_c} \ast \hat{c}| \leq b(e_c)
\]
\( a = b \ op \ c \)
\( \Phi(fp(s)) \)

\( \Phi(fp(s)) \) denotes a logical formula that relates the inputs and outputs of the fixed-point representation \( s \)
Synthesize a controller minimizing the following objective function:

\[ J(K, L) = w_1 \frac{\|S(K)\|}{\|S^*\|} + w_2 \frac{\|P(L)\|}{\|P^*\|} + w_3 \frac{\gamma_{1y}}{\gamma_{1y}^*} + w_4 \frac{\gamma_{2y} b(e_2)}{\gamma_{2y}^* b^*(e_2)} \]

where \( w_1, \ldots, w_4 \) are weighting factors.

The cost function \( J \) is not necessarily convex with respect to the feedback and observer gains \( K \) and \( L \).
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, L) for a controller.
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, L) for a controller
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
- solves the mixed-integer linear programming problems using lp_solve
### Experimental Results

<table>
<thead>
<tr>
<th>Control systems</th>
<th># bits</th>
<th>Synthesized gains</th>
<th>L</th>
<th>Time cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bicycle</strong></td>
<td>16</td>
<td>[3.0253, 12.6089]</td>
<td>[0.0132, 0.1021]'</td>
<td>1h36m41s</td>
</tr>
<tr>
<td><strong>DC motor position</strong></td>
<td>16</td>
<td>[0.1129, 0.0211, 0.0093]</td>
<td>[0.0390, 0.3700, -0.0175]'</td>
<td>1h39m06s</td>
</tr>
<tr>
<td><strong>Pitch angle control</strong></td>
<td>32</td>
<td>[-0.1202, 42.5655, 1.0001]</td>
<td>[0.0001, 0.0000, 0.0017]'</td>
<td>8h31m53s</td>
</tr>
<tr>
<td><strong>Inverted pendulum</strong></td>
<td>32</td>
<td>[-1.5362, -2.0254, 16.5192, 2.7358]</td>
<td>[0.0017, 0.0021, 0.0012, 0.0000]</td>
<td>9h54m17s</td>
</tr>
<tr>
<td><strong>Batch reactor process</strong></td>
<td>16</td>
<td>[0.0583, 0.9093, 0.3258, 0.8721]</td>
<td>[0.0774, -0.0022, 0.0267, 0.0356]'</td>
<td>3h08m29s</td>
</tr>
</tbody>
</table>

Table: Synthesized gains and required time for synthesizing them.

The experiments were run on a machine running MAC OS with CPU Intel dual core 2.4 GHZ, and RAM 2GB.

Maximum number of iterations is set to 100.
### Experimental Results

<table>
<thead>
<tr>
<th>Control systems</th>
<th>lub of LQR cost</th>
<th>LQG cost</th>
<th>Steady state error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LQR/LQG</td>
<td>LQR/LQG</td>
<td>Synthesized K</td>
</tr>
<tr>
<td>Bicycle</td>
<td>3956.3(</td>
<td>x</td>
<td>^2)</td>
</tr>
<tr>
<td>DCmotor position</td>
<td>1001.6(</td>
<td>x</td>
<td>^2)</td>
</tr>
<tr>
<td>Pitchangle</td>
<td>2.9732 (10^6)</td>
<td>0.0013</td>
<td>2.6781(b(e_1)) + 0.4746</td>
</tr>
<tr>
<td>Inverted</td>
<td>4.2988 (10^4)</td>
<td>0.3600</td>
<td>83.4217(b(e_1))+ 0.0432</td>
</tr>
<tr>
<td>Batchreactor</td>
<td>223.1773 (</td>
<td>x</td>
<td>^2)</td>
</tr>
<tr>
<td>process</td>
<td>223.1825 (</td>
<td>x</td>
<td>^2)</td>
</tr>
</tbody>
</table>

| Chosen weights in the objective function: | \(w_1 = w_2 = w_3 = 1, w_4 = 5\) |
Experimental Results

- The synthesized controller worsens the LQR and LQG performances by at most 1.38 times (for Pitch angle control).

- It improves the size of the region of practical stability by at least 2.55 times.

- For certain examples, the improvement goes beyond the factor of 10.

  - For the bicycle: 10.69
  - For DC motor position control: 14.55
PID controllers are widely used in the industries.

A common performance criteria for PID controllers are **Gain Margin** and **Phase Margin**.

We synthesize the parameters of the PID controllers ($K_P$, $K_D$ and $K_I$) for both gain and phase margin and fixed-point implementation error.
We present a generic methodology to search for optimal controller implementations that minimize the effect of implementation errors in addition to traditional controller performance criteria.

Our algorithm is more generally applicable to other performance criteria and other sources of implementation error.

Bridges the gap in controllers synthesis - gap between theory and implementation.
Finally, the end product..
float output(float yin) {
    static int x1 = x10; // fixdt(1,16,14)
    static int x2 = x20; // fixdt(1,16,14)
    int x1_new; // fixdt(1,16,14)
    int x2_new; // fixdt(1,16,14)
    int u; // fixdt(1,16,11)

    // Intermediate variables
    int Gain1; // fixdt(1,16,15)
    int Gain2; // fixdt(1,16,15)
    int Gain3; // fixdt(1,16,15)
    int Add1; // fixdt(1,16,14)
    int Gain4; // fixdt(1,16,15)
    int Gain5; // fixdt(1,16,15)
    int Gain6; // fixdt(1,16,15)
    int Add2; // fixdt(1,16,15)
    int Gain7; // fixdt(1,16,13)
    int Gain8; // fixdt(1,16,11)

    y = convert_to_fixedpoint(yin);
    Gain1 = (31499 * x1) >> 14;
    Gain2 = (-3145 * x2) >> 14;
    Add1 = (Gain1 + Gain2) >> 1;
    Gain3 = (432 * y) >> 14;
    x1_new = ((Add1 << 1) + Gain3) >> 1;
    Gain4 = (-1907 * x1) >> 14;
    Gain5 = (23835 * x2) >> 14;
    Add2 = Gain4 + Gain5;
    Gain6 = (3345 * y) >> 14;
    x2_new = (Add1 + Gain6) >> 1;
    Gain7 = (24783 * x1_new) >> 14;
    Gain8 = (25823 * x2_new) >> 14;
    u = (Gain7 + (Gain8 << 2)) >> 2;
    return(float(u));
}
float output(float yin)
{
    static int x1 = x10;  // fixdt(1,16,14)
    static int x2 = x20;  // fixdt(1,16,14)
    int x1_new;           // fixdt(1,16,14)
    int x2_new;           // fixdt(1,16,14)
    int u;                // fixdt(1,16,11)

    // Intermediate variables
    int Gain1;            // fixdt(1,16,15)
    int Gain2;            // fixdt(1,16,15)
    int Gain3;            // fixdt(1,16,15)
    int Add1;             // fixdt(1,16,14)
    int Gain4;            // fixdt(1,16,15)
    int Gain5;            // fixdt(1,16,15)
    int Gain6;            // fixdt(1,16,15)
    int Gain7;            // fixdt(1,16,15)
    int Gain8;            // fixdt(1,16,11)

    y = convert_to_fixedpoint(yin);
    Gain1 = (31499 * x1) >> 14;
    Gain2 = (-3145 * x2) >> 14;
    Add1 = (Gain1 + Gain2) >> 1;
    Gain3 = (432 * y) >> 14;
    x1_new = ((Add1 << 1) + Gain3) >> 1;
    Gain4 = (-1907 * x1) >> 14;
    Gain5 = (23835 * x2) >> 14;
    Add2 = Gain4 + Gain5;
    Gain6 = (3345 * y) >> 14;
    x2_new = (Add1 + Gain6) >> 1;
    Gain7 = (24783 * x1_new) >> 14;
    Gain8 = (25823 * x2_new) >> 14;
    u = (Gain7 + (Gain8 << 2)) >> 2;
    return(float(u));
}

Thanks for your attention!!