1 Context
   The semantic gap
   Control-theoretical aspects
   Compilation aspects
   C code production

2 From real to floats
   Example of linear invariant system
   Numerical precision problems
   Machine representation of real numbers
   Alteration of constants
   Rounding errors
   Other systems

3 Closing the loop
   Closed-loop system
   Proof scheme
From Physics to Interrupt Handlers: The Real to Float Step

Vivien Maisonneuve
CRI, MINES ParisTech

Presentation at Toccata

November 23, 2012
Different levels of description

In control engineering, work on different levels to design and build a control system:

- **Format/high-level aspects**: system conception, modeling, possibly proof.
- **Concrete/low-level aspects**: creation of an object implementing the system.

**Quadricopter, DRONE Project, MINES ParisTech & ECP.**
Formal aspect

System definition:

- **Inputs**: sensors [accelerometer, sonar...] + references [operator instructions].
- **Outputs**: actions to act on environment [rotors rotation speed].
- Modeling in the form of equations to express relations between inputs and outputs: differential equations/transfer functions between IOs.
Formal aspect

System definition:

- **Inputs**: sensors [accelerometer, sonar...] + references [operator instructions].
  - Outputs: actions to act on environment [rotors rotation speed].
- Modeling in the form of equations to express relations between inputs and outputs: differential equations/transfer functions between IOs.

System requirements:

- **Stability** conditions [bounded rotation speed].
- **Pursuit** of reference input [try to reach the ordered position].
- **Perturbation** rejection [wind].
Concrete aspect

Creation of a real object implementing the system.
- **Electronic circuit** that physically computes the transfer function.
- With a **microcontroller**: a small system with processor, memory, I/O devices, that runs a **program** implementing the transfer function.

![Electronic circuit](image1.png)

**ATMEGA128**
- Frequency: 16 MHz
- RAM: 4 KB
- Prog. mem.: 128 KB
Semantic gap

Antagonism:
- Abstract, mathematical model.
- Microcontroller code: program written in C, then compiled. Long (thousands of LoC), low-level (elementary operations, hardware management, interruptions).

Series of transformations to go from abstract model to microcontroller code.
Semantic gap

Antagonism:
- Abstract, mathematical model.
- Microcontroller code: program written in C, then compiled.

Series of transformations to go from abstract model to microcontroller code.

Problem of proof transposition: Considering a model with correction proofs [stability], how to transpose down these proofs at the code level?

Interest: formally check the code, not only the model.

Difficulties: semantic gap, non-equivalent transformations (⇒ proofs must be checked).
Control-theoretical aspects

Produce a pseudocode from the abstract model:

- Solve the model differential equation, get a transfer function. (Laplace transform/Z transform, initial conditions problem.)
- If continuous-time model, discretization. (Problems with sampling, execution times.)

while transposing the proof.

Usual problems in control engineering.

Once done, discrete-time system with a loop on the transfer function ⇒ pseudocode [in MATLAB]. Proof: invariants on this code.
Compilation aspects

At the other end: compilation of C code to machine code.

Risks of error:
- Important changes in the code: elementary operations, management of registers and of memory stack, instruction jumps.
- Possible optimizations.

Solutions:
- “Existing C compilers are reliable enough.”
- Proof-preserving compilation [Barthe].
- Certified compilation [CompCert].
What’s between?

Opener question. Several challenges:

1. High-level mathematical operations $\leadsto$ series of elementary instructions [matrices, sinus].

2. Real values $\leadsto$ machine words with limited precision.

3. On a microcontroller, data/events acquisition raises interruptions (real-time answer, energy consumption) $\Rightarrow$ particular code structure.
Example system

Very simple, open-loop, linear system [Feron].

Pseudocode:

\[
\begin{align*}
Ac &= \begin{bmatrix} 0.4990, & -0.0500; 0.0100, & 1.0000 \end{bmatrix}; \\
Bc &= \begin{bmatrix} 1;0 \end{bmatrix}; \\
Cc &= \begin{bmatrix} 564.48, & 0 \end{bmatrix}; \\
Dc &= -1280; \\
xc &= \text{zeros}(2,1);
\end{align*}
\]

receive(y,2); receive(yd,3);

while 1

\[
\begin{align*}
yc &= \text{max}(\text{min}(y - yd,1),-1); \\
u &= Cc*xc + Dc*yc; \\
xc &= Ac*xc + Bc*yc; \\
send(u,1); \\
receive(y,2); \\
receive(yd,3);
\end{align*}
\]

end

state matrix (matrice de dynamique) 
input matrix (matrice de commande) 
output matrix (matrice d'observation) 
feedthrough matrix (matrice d'action directe) 
\[
\begin{align*}
x_c &= \begin{bmatrix} x_{c1} \\
x_{c2} \end{bmatrix} \in \mathbb{R}^2: \text{controller state} \\
y \in \mathbb{R}: \text{reference input}; y_d \in \mathbb{R}: \text{real position} \\
y_c \in [-1,1]: \text{bounded gap} \\
u \in \mathbb{R}: \text{action to be performed} \\
send, receive: \text{blocking, 2\textsuperscript{nd} arg. is channel id}
\end{align*}
\]
Lyapunov theory

(Lyapunov) stability: all reachable states \( x_c \) start near an equilibrium point \( x_e \) and stay in a neighborhood \( V \) of \( x_e \) forever.

Lyapunov theory: NSC on \( V \). On linear systems, provided as an equation that can be solved with LMIs, generally as an ellipsoid.

Here, show that \( x_c = \begin{pmatrix} x_{c1} \\ x_{c2} \end{pmatrix} \) belongs to the ellipse:

\[
E_P = \{ x \in \mathbb{R}^2 \mid x^T \cdot P \cdot x \leq 1 \}, \quad P = 10^{-3} \begin{pmatrix} 0.6742 & 0.0428 \\ 0.0428 & 2.4651 \end{pmatrix}.
\]

\[
x_c \in E_P \iff 0.6742x_{c1}^2 + 0.0856x_{c1}x_{c2} + 2.4651x_{c2}^2 \leq 1000.
\]
Stability proof

```
x_c = zeros(2,1);
x_c ∈ E_P
receive(y,2); receive(yd,3);
x_c ∈ E_P
while 1
  x_c ∈ E_P
  yc = max(min(y - yd,1),-1);
  x_c ∈ E_P, y_c^2 ≤ 1
  \begin{bmatrix} x_c \\ y_c \end{bmatrix} ∈ E_{Q_μ} \quad Q_μ = \begin{pmatrix} μP_{2×1} \\ 0_{1×2} \end{pmatrix}, μ = 0.9991
  u = Cc*xc + Dc*yc;
  \begin{bmatrix} x_c \\ y_c \end{bmatrix} ∈ E_{Q_μ}
  x_c = Ac*xc + Bc*yc;
  x_c ∈ E_{\tilde{P}} \quad \tilde{P} = \left[ (A_c \quad B_c) \cdot Q_{μ}^{-1} \cdot (A_c \quad B_c)^T \right]^{-1}
  send(u,1);
  x_c ∈ E_{\tilde{P}}
receive(y,2);
  x_c ∈ E_{\tilde{P}}
receive(yd,3);
  x_c ∈ E_{\tilde{P}}
  x_c ∈ E_P
end
```

Proof given as code invariants.

Implication (weakening) if two consecutive invariants.

Most of them easy to check, some depend on theorems.

Last implication: $E_{\tilde{P}} \subseteq E_P$ closes the loop. Validity relies on parameters $A_c, B_c, C_c, D_c, μ$: algebraic or numerical verification needed.
Digression: with C instructions

High level mathematical operations $\leadsto$ series of scalar elementary instructions.
Here, matrix operations are expanded: the instruction

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} \in \mathcal{E}_{Q_\mu}$$

$$x_c = A_c x_c + B_c y_c;$$

$$x_c \in \mathcal{E}_{\tilde{P}} \quad | \quad \tilde{P} = \left[ \begin{pmatrix} A_c & B_c \end{pmatrix} \cdot Q_\mu^{-1} \cdot \begin{pmatrix} A_c & B_c \end{pmatrix}^T \right]^{-1}$$

becomes:

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} \in \mathcal{E}_{Q_\mu}$$

$$xb[0] = xc[0];$$

$$xb[1] = xc[1];$$

$$xc[0] = A_c[0][0]*xb[0]+A_c[0][1]*xb[1]+yc;$$

$$xc[1] = A_c[1][0]*xb[0]+A_c[1][1]*xb[1];$$

???
Digression: with C instructions

High level mathematical operations $\leadsto$ series of scalar elementary instructions.

Here, matrix operations are expanded: the instruction

$$
\begin{pmatrix}
    x_c \\
    y_c
\end{pmatrix} \in E_{Q_\mu}
$$

\[ x_c = A_c x_c + B_c y_c; \]

\[ x_c \in E_\tilde{P} \quad | \quad \tilde{P} = \left[ \left( A_c \quad B_c \right) \cdot Q_\mu^{-1} \cdot \left( A_c \quad B_c \right)^T \right]^{-1} \]

becomes:

$$
\begin{pmatrix}
    x_c \\
    y_c
\end{pmatrix} \in E_{Q_\mu}
$$

\[ x_b[0] = x_c[0]; \quad x_b[1] = x_c[1]; \]

\[ x_c[0] = A_c[0][0]*x_b[0]+A_c[0][1]*x_b[1]+y_c; \]

\[ x_c[1] = A_c[1][0]*x_b[0]+A_c[1][1]*x_b[1]; \]

\[ x_c \in E_\tilde{P} \quad | \quad \tilde{P} = \left[ \left( A_c \quad B_c \right) \cdot Q_\mu^{-1} \cdot \left( A_c \quad B_c \right)^T \right]^{-1} \]

Same computation: output invariant can be found [Feron].
Numerical precision problems

To produce C code: \textit{real} numbers $\rightarrow$ binary finite-length machine words (32 b. or 64 b.).

$\Rightarrow$ Loss in accuracy, two consequences:

1. Constant values are slightly altered.
2. Rounding errors during computations.
Machine representation of real numbers

1. Floating point: IEEE 754.
   Not usual on microcontrollers.

   \[
   \text{number} = \text{sign} \times 2^{\text{exponent} + \text{cst. offset}} \times \text{fraction}
   \]

   Correct rounding for base operations: +, −, *, /.
   ⇒ If [bounds on] operands are known, not special, far enough from extremal values, then rounding error is bounded for +, −, * (not /).

2. Fixed point.
   If operands are not special, far enough from extremal values, then rounding error is bounded for +, −, *.

3. Two integers.
Machine representation of real numbers

1. Floating point.
2. Fixed point.
   - Base behavior: +, -, *, / follow rational definition + fraction simplification:

   \[
   \frac{p_1}{q_1} + \frac{p_2}{q_2} = \text{simpl} \left( \frac{p_1q_2 + p_2q_1}{q_1q_2} \right), \text{ etc.}
   \]

   No rounding error.
   Problem: numerator value can easily exceed integer bounds.
   - Approximated behavior to ensure bounded numerator.
Alteration of constants

With IEEE 754, 32 bits, constants

\[
\begin{align*}
A_c &= [0.4990, -0.0500; 0.0100, 1.0000]; \\
B_c &= [1;0]; \\
C_c &= [564.48, 0]; \\
D_c &= -1280;
\end{align*}
\]

become

\[
\begin{align*}
A_c &\approx [0.49900001287460327, -0.05000000074505806; \\
&0.009999999776482582, 1.0000]; \\
B_c &\approx [1;0]; \\
C_c &\approx [564.47998046875, 0]; \\
D_c &\approx -1280;
\end{align*}
\]
Effect on proof

\[ xc = \text{zeros}(2,1); \]
\[ xc \in \mathcal{E}_P \]
\[ \text{receive}(y,2); \text{receive}(yd,3); \]
\[ xc \in \mathcal{E}_P \]
\[ \text{while } 1 \]
\[ xc \in \mathcal{E}_P \]
\[ yc = \max(\min(y - yd,1),-1); \]
\[ xc \in \mathcal{E}_P, \; y_c^2 \leq 1 \]
\[ (x_c, y_c) \in \mathcal{E}_{Q_\mu} \]
\[ u = Cc*xc + Dc*yc; \]
\[ (x_c, y_c) \in \mathcal{E}_{Q_\mu} \]
\[ xc = Ac*xc + Bc*yc; \]
\[ xc \in \tilde{\mathcal{P}} \]
\[ \tilde{\mathcal{P}} = \left[ \begin{pmatrix} A_c & B_c \end{pmatrix} \cdot Q_\mu^{-1} \cdot \begin{pmatrix} A_c & B_c \end{pmatrix}^T \right]^{-1} \]
\[ \text{send}(u,1); \]
\[ xc \in \tilde{\mathcal{P}} \]
\[ \text{receive}(y,2); \]
\[ xc \in \tilde{\mathcal{P}} \]
\[ \text{receive}(yd,3); \]
\[ xc \in \tilde{\mathcal{P}} \]
\[ xc \in \mathcal{E}_P \]
\[ xc \in \mathcal{E}_P \]

Rest of the code and proof sketch unchanged.

\[ \tilde{\mathcal{P}} \] depends on \( A_c, B_c, C_c, D_c \), is altered.

\[ \Rightarrow \text{Check that } \mathcal{E}_{\tilde{\mathcal{P}}} \subseteq \mathcal{E}_P \text{ still holds.} \]
Rounding errors

With real numbers, the implication

\[
\begin{pmatrix} x_c \\ y_c \end{pmatrix} \in \mathcal{E}_Q \\
\begin{align*}
x_c &= A_c x_c + B_c y_c; \\
x_c &\in \mathcal{E}_{\tilde{P}} \quad \tilde{P} = \left[ (A_c \ B_c) \cdot Q^{-1}_\mu \cdot (A_c \ B_c)^T \right]^{-1}
\end{align*}
\]

holds.

With floats, + and * introduce rounding errors.

As \( x_c, \ y_c \) belong to an ellipsoid, they are bounded so the rounding error on \( x_c \) can be bounded by \((e_1, e_2)\).
Super-ellipsoid

Let $\mathcal{E}_{\tilde{F}} \supset \mathcal{E}_{\tilde{P}}$ an ellipse s.t.

$$\forall x_c \in \mathcal{E}_{\tilde{P}}, \forall x'_c \in \mathbb{R}^2, |x'_{c_1} - x_{c_1}| \leq e_1 \land |x'_{c_2} - x_{c_2}| \leq e_2 \implies x'_c \in \mathcal{E}_{\tilde{F}} \quad (\ast)$$

Then:

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} \in \mathcal{E}_{Q_\mu}$$

$$x_c = A c \ast x_c + B c \ast y_c; \quad x_c \in \mathcal{E}_{\tilde{F}}$$

$\mathcal{E}_{\tilde{F}}$ can be the smallest magnification of $\mathcal{E}_{\tilde{P}}$ s.t. (\ast) holds.

Can be computed, whatever number of dimensions.
Effect on proof

\[
\begin{align*}
xc &= \text{zeros}(2,1); \\
xc &\in \mathcal{E}_P \\
\text{receive}(y,2); \text{receive}(yd,3); \\
x_c &\in \mathcal{E}_P \\
\text{while } 1 \\
&\quad \text{ } x_c \in \mathcal{E}_P \\
yc &= \max(\min(y - yd, 1), -1); \\
x_c &\in \mathcal{E}_P, \quad y_c^2 \leq 1 \\
\begin{pmatrix}
  x_c \\
  y_c
\end{pmatrix} &\in \mathcal{E}_{Q\mu} \quad | \quad Q\mu = \begin{pmatrix}
  \mu P & 0_{2\times1} \\
  0_{1\times2} & 1 - \mu
\end{pmatrix}, \mu = 0.9991 \\
u &= C_c*x_c + D_c*y_c; \\
\begin{pmatrix}
  x_c \\
  y_c
\end{pmatrix} &\in \mathcal{E}_{Q\mu} \\
xc &= A_c*x_c + B_c*y_c; \\
x_c &\in \mathcal{E}_{\tilde{F}} \\
\text{send}(u,1); \\
x_c &\in \mathcal{E}_{\tilde{F}} \\
\text{receive}(y,2); \\
x_c &\in \mathcal{E}_{\tilde{F}} \\
\text{receive}(yd,3); \\
x_c &\in \mathcal{E}_{\tilde{F}} \\
x_c &\in \mathcal{E}_P \\
\text{end}
\end{align*}
\]

Replace $\mathcal{E}_P$ by $\mathcal{E}_{\tilde{F}}$ in proof sketch.

$\Rightarrow$ Check that $\mathcal{E}_{\tilde{F}} \subseteq \mathcal{E}_P$ holds.

Here it works: system stable with floats 😊.
Other functions

Elementary operations $+$, $\ast$ are sufficient for linear, invariant systems. The method applies if the proof sketch fits: no tight assumptions, complex operations on weakened invariants.

1-var, differentiable, periodic functions can be computed
- with an abacus and a polyhedral interpolation function
- with a polyhedral approximation

with a bounded error $(\sin, \cos)$.

Idem for 1-var, differentiable functions restricted to a finite range. OK if proof ensures the operand is bounded to the range.
Closing the loop

Modeling the result of the effects of the action on the environment, with feedback.

Design: here, two parallel, synchronized programs:

controller + plant (abstract).

\[
\begin{align*}
Ac &= \begin{bmatrix} 0.4990, & -0.0500; \\ 0.0100, & 1.0000 \end{bmatrix}; \\
Bc &= \begin{bmatrix} 1; 0 \end{bmatrix}; \\
Cc &= \begin{bmatrix} 564.48, & 0 \end{bmatrix}; \\
Dc &= -1280; \\
xc &= \text{zeros}(2,1); \\
\text{receive}(y,2); \text{receive}(yd,3); \\
\text{while 1} \\
\quad yc &= \text{max}(\text{min}(y - yd,1),-1); \\
\quad u &= Cc*xc + Dc*yc; \\
\quad xc &= Ac*xc + Bc*yc; \\
\quad \text{send}(u,1); \\
\quad \text{receive}(y,2); \\
\quad \text{receive}(yd,3); \\
\text{end}
end
\end{align*}
\]

\[
\begin{align*}
Ap &= \begin{bmatrix} 1.000, & 0.0100; \\ -0.0100, & 1.000 \end{bmatrix}; \\
Bp &= \begin{bmatrix} 0.00005; & 0.01 \end{bmatrix}; \\
Cp &= \begin{bmatrix} 1, & 0 \end{bmatrix}; \\
while (1) \\
\quad yp &= Cp * xp; \\
\quad \text{send}(yp,2); \\
\quad \text{receive}(up,1); \\
\quad xp &= Ap * xp + Bp * up; \\
\text{end}
\end{align*}
\]

System is not linear.
Proving the system

Lyapunov stability: global state \((x_c, x_p)\) in some ellipsoid \(E_P\).
\[\Rightarrow\] + Boundedness of variables in physical system.

Difficulties:

- Non-linearity issues: trickier to find a suitable \(E_P\), post-condition to \(y_c\) definition.
  Usual case here, has been dealt.
- Handling concurrency in invariants: switch between system and plant analysis.
Proving the system

\[
\begin{align*}
Ac &= \begin{bmatrix} 0.4990 & -0.0500 \\ 0.0100 & 1.0000 \end{bmatrix}; \\
Bc &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \\
Cc &= \begin{bmatrix} 564.48 \\ 0 \end{bmatrix}; \\
Dc &= -1280; \\
xc &= \text{zeros}(2,1); \\
receive(y,2); & \text{receive}(yd,3);
\end{align*}
\]

\[
\begin{align*}
\text{while 1} & \\
& yc = \max(\min(y - yd,1),-1); \\
u &= Cc*xc + Dc*yc; \\
xc &= Ac*xc + Bc*yc; \\
send(u,1); \\
& \text{receive}(y,2); \\
& \text{receive}(yd,3); \\
& \text{while (1)} \\
& \hspace{1em} yp = Cp \ast xp; \\
& \hspace{1em} send(yp,2); \\
& \hspace{1em} receive(up,1); \\
& \hspace{1em} xp = Ap \ast xp + Bp \ast up; \\
& \hspace{1em} end
\end{align*}
\]
Proving the system

Lyapunov stability: global state \((x_c, x_p)\) in some ellipsoid \(E_P\).

\[ \Rightarrow + \] Boundedness of variables in physical system.

Difficulties:

- Non-linearity issues: trickier to find a suitable \(E_P\), post-condition to \(y_c\) definition.
  Usual case here, has been dealt.

- Handling concurrency in invariants: switch between system and plant analysis.

- Invariants of greater dimension: cannot test algebraically invariant inclusion, fails with floats.

- C code with interrupts.

```c
SIGNAL(2) SIGNAL(3) while(1) {
    y = ... yd = ...
    ... ... 
    sleep();
}
```
From Physics to Interrupt Handlers: The Real to Float Step

Vivien Maisonneuve
CRI, MINES ParisTech

Presentation at Toccata

November 23, 2012