Microcontroller Implementation of a Multi Objective Genetic Algorithm for Real-Time Intelligent Control
• Introduction and Motivation
• Proposed Hybrid Control Scheme
• Implementation
• Results
• Conclusions and Future Work
Introduction and Motivation

- **Soft-Computing** → excellent for complex problems 1
  - Fuzzy Logic
  - Neurocomputing
  - Evolutionary Computation

- Research line
  - Hybridation of Intelligent Techniques 3
  - Implementation on high performance Platforms 4

- Interesting: lower level industrial approach

References: 1) Rudas and Fodor, 2008; 2) Palit and Popovic, 2005; 3) Valera et al., 2012; 4) Larzabal et al., 2013
- MOGA for optimization of control actions: NSGA-II
  - Good for complex, non-linear, non-convex problems
  - Computational cost = $O(MN^2)G$

- Neural Network for Fitness evaluation
- Modelling of nonlinear multivariable systems

References: 5) Deb, 2002
Implementation

- ARM-based redundant 180MHz 32bit μC
- Flexible software structure
  - Coded in C and speed-optimized
  - NSGA2-II Deb’s Code adaptation
  - NN: Simulink™ code-generation
- Additional HiL setup
  - For future algorithm validations
• Tested NSGA-II setups

<table>
<thead>
<tr>
<th>Population</th>
<th>Generations</th>
<th>Chromosome</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>«Small»</td>
<td>15</td>
<td>20 / 24</td>
<td>6 / 10</td>
</tr>
<tr>
<td>«Big»</td>
<td>60</td>
<td>60</td>
<td>6</td>
</tr>
</tbody>
</table>

• Used Fitness functions
  • Algebraic functions (simple and heavy)
  • Nonlinear differential model (horizon=5)
  • NARX Neural Network (horizon=5)
    • Main layer: 8 neurons, hidden layer: 12 neurons
• Small MOGA < 28ms, Big MOGA < 1s
• < 100ms ← Pop.=15, Gener.=24, NN (Horizon=5)
• Code size: ~79KB (Algebraic), ~105KB (NN)
• RAM: ~12/85KB (Small/Big MOGA), ~14KB (NN)
• NN execution ≈ 34.7µs x 5
• Maximum population = 113 (256KB-RAM limit)

• Extended expression for computation cost proposed:

\[ O(MN^2)G \rightarrow T = C'(MN^2)G + C'(NF)G = C'[N[MN + F]]G \rightarrow C' \approx 0.0022 \]

Platform computation constant Fitness evaluation Cost
### Results

- Small MOGA < 28ms, Big MOGA < 1s
- < 100ms $\iff$ Pop.=15, Gener.=24, NN (Horizon=5)
- Code size: ~79KB (Algebraic), ~105KB (NN)
- RAM: ~12/85KB (Small/Big MOGA), ~14KB (NN)
- NN execution $\approx 34.7\mu s \times 5$
- Maximum population = 113 (256KB-RAM limit)

<table>
<thead>
<tr>
<th>Fitness function</th>
<th>Big MOGA test</th>
<th></th>
<th>Small MOGA test</th>
<th></th>
<th>Twin Rotor test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple (1)</td>
<td>Heavy (2)</td>
<td>Simple (1)</td>
<td>heavy (2)</td>
<td>Nonlinear Model</td>
</tr>
<tr>
<td>Objectives</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Population</td>
<td>60</td>
<td>60</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Generations</td>
<td>60</td>
<td>60</td>
<td>20</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>Chromosome size</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Prediction horizon</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Constraints</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Code size</td>
<td>79.0 KB</td>
<td>79.7KB</td>
<td>78.4 KB</td>
<td>79.2 KB</td>
<td>101.8 KB</td>
</tr>
<tr>
<td>RAM occupation</td>
<td>85.5 KB</td>
<td>85.5 KB</td>
<td>11.6 KB</td>
<td>11.6 KB</td>
<td>14.9 KB</td>
</tr>
<tr>
<td>$T_{cycle}$ Avg./Worst [ms]</td>
<td>922/970</td>
<td>998/1085</td>
<td>21.5/24.2</td>
<td>28.1/32.3</td>
<td>74.6/82.1</td>
</tr>
<tr>
<td>$C$ factor (see section 2)</td>
<td>0.002134</td>
<td>0.002310</td>
<td>0.002389</td>
<td>0.003122</td>
<td>0.006907</td>
</tr>
<tr>
<td>$teval$ Average</td>
<td>0.335 $\mu$s</td>
<td>21.8 $\mu$s</td>
<td>0.335 $\mu$s</td>
<td>21.8 $\mu$s</td>
<td>136.2 $\mu$s</td>
</tr>
<tr>
<td>$C'$ factor (see eq. 3)</td>
<td>0.002130</td>
<td>0.002130</td>
<td>0.002378</td>
<td>0.002378</td>
<td>0.002371</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

- Feasibility of MOGA + NN on reasonable μC
- Relatively low cycle times possible
- Great potential for embedded intelligent control
- Degrees of freedom for future research
  - Algorithm development
  - Optimization and use of other platforms (DSP, FPGA, ...)
- Future work
  - Algorithm development with HiL
  - Hybrid hardware implementation: Processor + FPGA
Thank you for your attention

Questions?

Contact for further questions: martin@dendaluze.com
NSGA-II Diagram

1. Initialize Population
2. Evaluate Fitness Functions
3. Rank Population
4. Selection
5. Crossover
6. Mutation
7. Combine Parents and Offspring, Rank Classification
8. Individuals Selection
9. Elitism
10. Stop Criteria?
11. Yes, Final Population; No, go back to Evaluate Fitness Functions
HiL testing setup

ADC → scaling & calibr. → Current Value → INTELLIGENT CONTROLLER → U → scaling → PWM

Pulse Capture → [Duty Cycle] → scaling & calibr. → U → Current Value → scaling → PWM

Can Analyzer/Logger

Main GUI (control & visualiz.)
Controller Data
Parameter modification
Reset/Run
Tuning/Override/Calibration...

Secondary GUI (alteration & supervisory)
Scaled Visual Model
[LED displays]