Intelligent Control: Automation Via the Emulation of Biological Intelligence

Kevin M. Passino

Dept. Electrical Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210-1272
(614) 292-5716, k.passino@osu.edu
http://eewww.eng.ohio-state.edu/~passino

Outline

- What is "Intelligent Control"?
- Intelligent vs. "Conventional Control"
- Intelligent Control: Basic Techniques and Applications
  - Fuzzy control, direct and adaptive, tanker ship heading regulation
  - Neural networks, gradient training, jet engine parameter estimation problem
  - Genetic algorithms, evolution of coffee growers in Colombia, genetic adaptive control for brake systems
- Outlook on Intelligent Control

What is "Intelligent Control"?

- Do "intelligent controllers" exist? Yes.
- If you were to control something, you would call yourself an intelligent controller!
- Example: "Control" of a child's behavior (an uncontrollable, unobservable, stochastic, nonlinear, sometimes unstable system!)

- Also, intelligent controllers for technological problems:
  - Process operators at, for instance, a nuclear power plant.
  - Missile guidance via B.F. Skinner's pigeons
- Problems with the use of biological intelligent systems:
  - Expensive
  - Are they reliable? How do we guarantee this?
  - We do not understand the dynamical behavior of biological systems very well!

- Can we replace the above biological systems with computers?
- If successful, many people call such systems (artificial) intelligent controllers (but there are degrees of intelligence)
- In this sense, conventional controllers can be called "intelligent" also since they can replace biological intelligent systems
- Sometimes we do not seek to replace a biological system, but to emulate successful functions of biological intelligent systems to help us solve control problems.
- Our early attempts at emulating/implementing intelligence:
  - Fuzzy Systems (or more general rule-based expert systems or planning systems)
  - Neural networks
  - Genetic algorithms
- But both biological system replacement and biological

intelligent function emulation is quite difficult:

- The brain is not well-understood.
- Intelligence is not well-understood.

- Hence, currently we only use crude models of biological systems.
- Best to view "intelligent control" as a goal rather than a reality (how we should view "optimal control").

Intelligent vs. "Conventional Control"

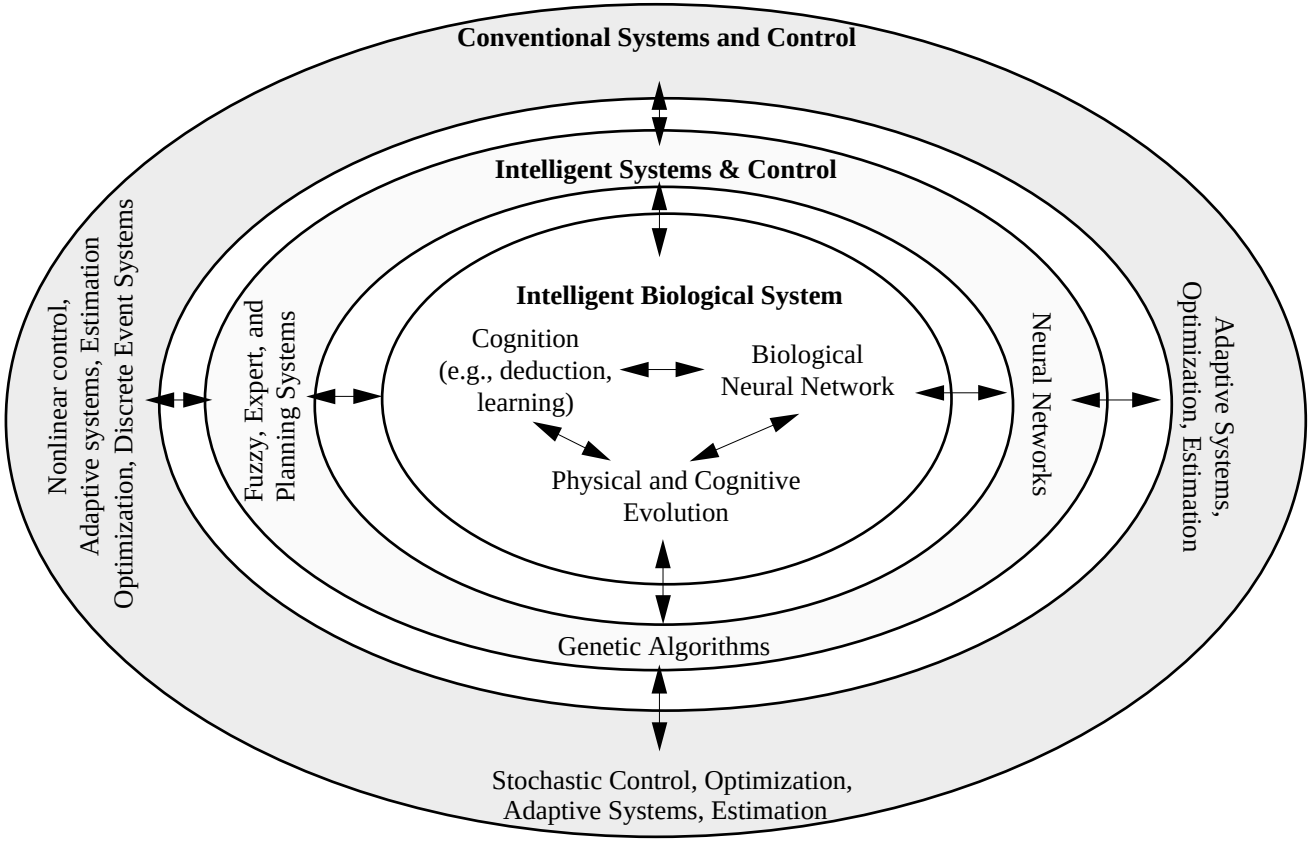

Figure 1: Intelligent biological systems and intelligent and conventional systems and control.

- Researchers beware - do not want to work hard to achieve something that is considered child's play by another group!
- Similarities and differences, but practically speaking often found to be a useful tool in your toolbox!
- Below, will point out some of the main advantages...
- Ultimately, the primary issue is whether you are intelligent in constructing control systems, and not whether or not you use intelligent control techniques...

Intelligent Control: Basic Techniques and Applications

- Fuzzy control, neural networks, genetic algorithms
- Focus:
  - For estimation/control, not pattern recognition or classification.
  - For reliable implementation, not science of artificial intelligence (don't care if the biological system behaves this way)

Fuzzy Control

Direct (Nonadaptive) Fuzzy Control

- Fuzzy control: a methodology to represent and implement a (smart) human's knowledge about how to control a system.

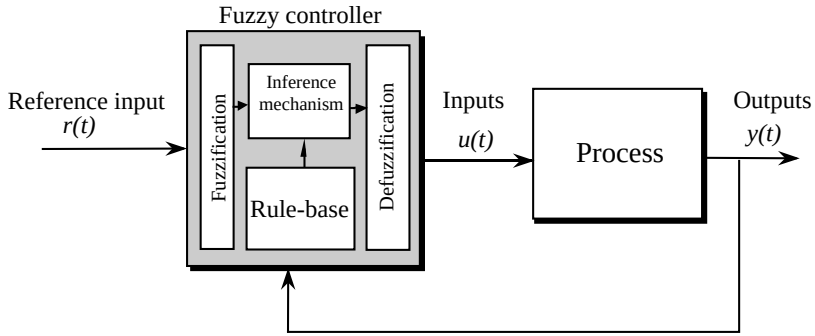

Figure 2: Fuzzy controller.

- Rule-base is a set of rules about how to control
- Inference mechanism processes knowledge, and coupled with fuzzification and defuzzification, it applies it.
- Consider the tanker ship steering application in Figure 3

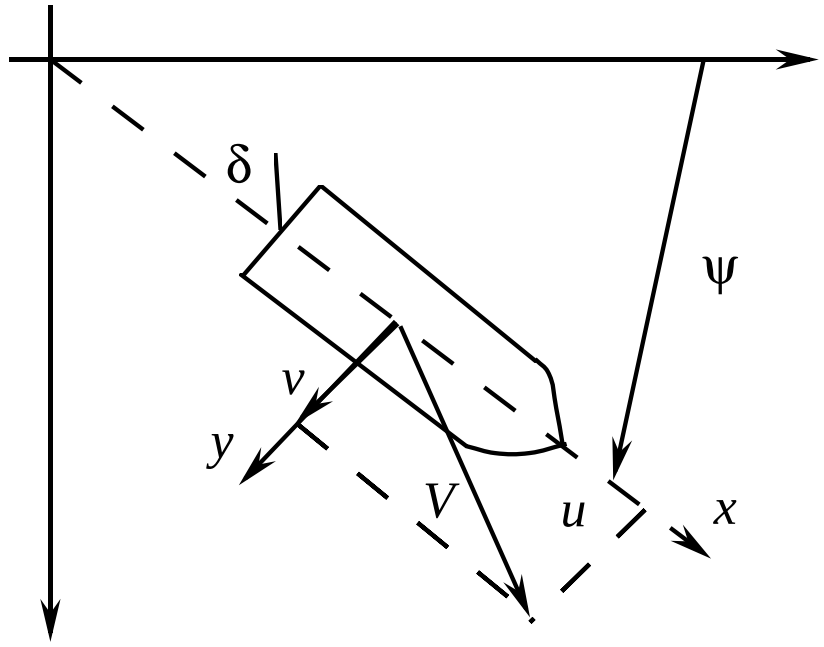

Figure 3: Tanker ship steering problem.

- Use the control system in Figure 4 and the following rules:

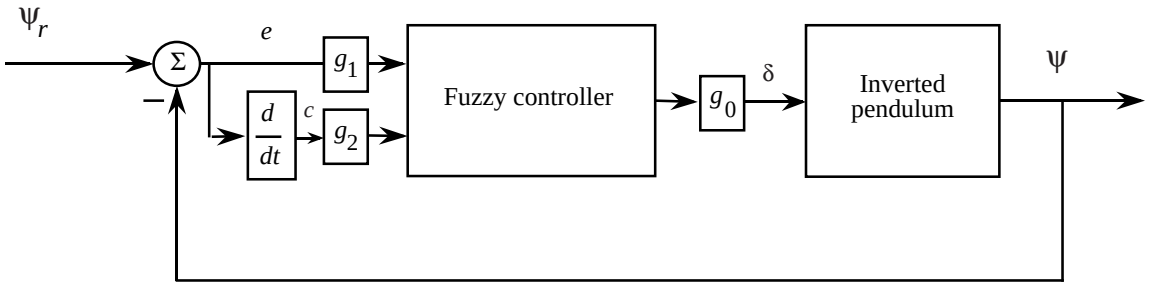

Figure 4: Control system for tanker.

1. **If** e is neg **and** c is neg **Then** δ is poslarge
2. **If** e is neg **and** c is zero **Then** δ is possmall
3. **If** e is neg **and** c is pos **Then** δ is zero
4. **If** e is zero **and** c is neg **Then** δ is possmall
5. **If** e is zero **and** c is zero **Then** δ is zero
6. **If** e is zero **and** c is pos **Then** δ is negsmall
7. **If** e is pos **and** c is neg **Then** δ is zero

8. **If** e is pos **and** c is zero **Then** δ is negsmall
9. **If** e is pos **and** c is pos **Then** δ is neglarge

- Rule 5: Heading is good so let the rudder input be zero.
- Rule 1:
  - "e is neg" means that ψ is higher than ψ_r.
  - "c is neg" means that ψ is moving away from ψ_r (if ψ_r is fixed).
  - In this case we need a large positive rudder angle to get the ship heading in the direction of ψ_r.

- Other rules explained similarly.
- What, precisely, do we (or the captain) mean by, for example, "e is pos", or "c is zero", or "δ is poslarge" ?
- Quantify the meaning with "fuzzy sets" ("membership functions"), as shown in Figure 5

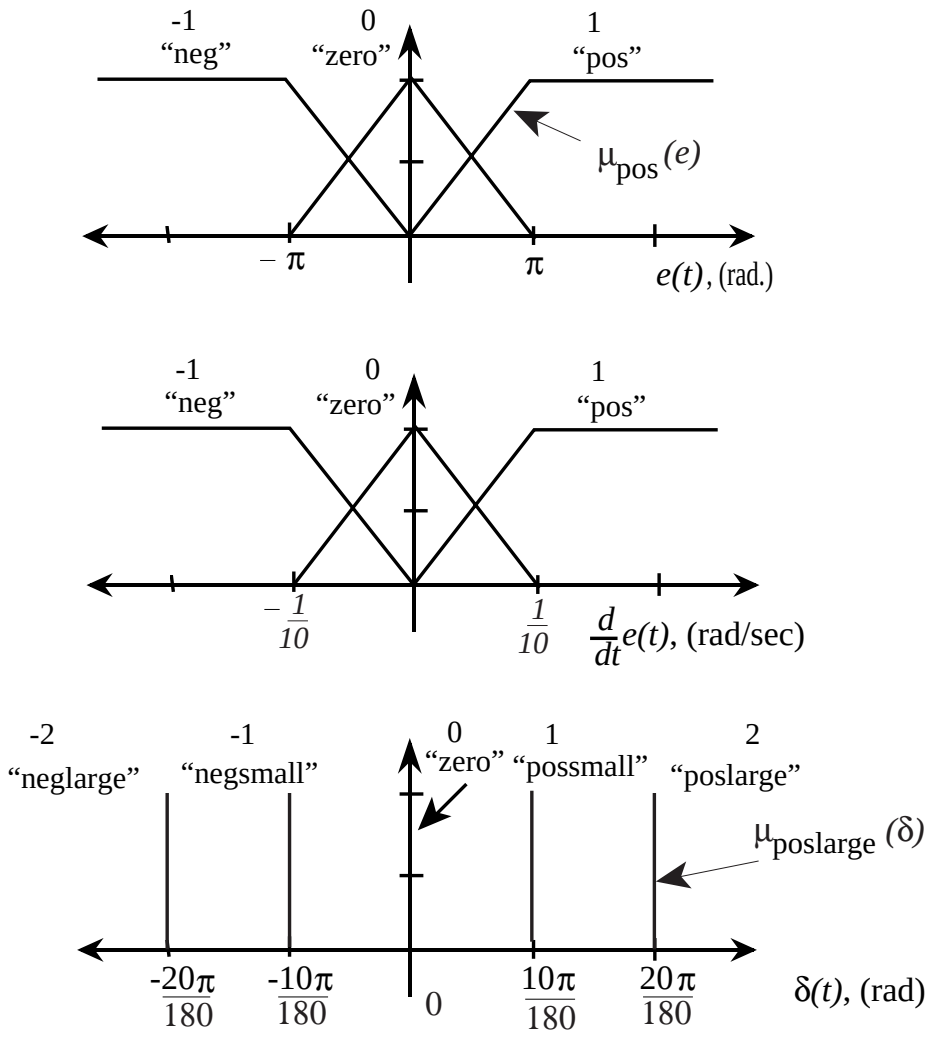

Figure 5: Membership functions for inputs and output.

- Many other membership function types are possible.
- Fuzzification is simply the act of finding, e.g., $\mu_{pos}(e)$ for a specific value of e.
- We use "fuzzy logic" to quantify the conjunctions in the premises of the rules.
- For instance, the premise of Rule 2 is "e is neg **and** c is zero"
  - Let $\mu_{neg}(e)$ and $\mu_{zero}(c)$ denote their respective membership functions.
  - Then, the premise certainty for Rule 2 can be defined by

$$\mu_{premise(2)} = \min\{\mu_{neg}(e), \mu_{zero}(c)\}$$

  - Why? Think about the conjunction of two uncertain statements. The certainty of the assertion of two things is the certainty of the least certain statement.
- In general, more than one $\mu_{premise(i)}$ will be nonzero at a time

so more than one rule is "on" (applicable) at every time.

- Recommendation of Rule 2: "Implied fuzzy set" $\mu_{(2)}$ where

$$\mu_{(2)}(\delta) = \min\left\{\mu_{premise(2)}, \mu_{possmall}(\delta)\right\}$$

- For example, if $\mu_{premise(2)} = 0.8$, then $\mu_{(2)}(\delta)$ is a singleton of height 0.8 centered at $\frac{10\pi}{180}$.

- Defuzzification: Combining the conclusions (implied fuzzy sets) of all the rules. "Center-average" defuzzification uses

$$\delta = \frac{\sum_{i=1}^{9} b_i \mu_{premise(i)}}{\sum_{i=1}^{9} \mu_{premise(i)}}$$

where b_i is the position of the center of the output membership function for the i^{th} rule (i.e., the position of the singleton).

- This is simply a weighted average of the conclusions (if a rule is on more, i.e., we are more certain about it, then it will more

strongly influence the controller output).

- A mathematical description of the fuzzy controller.
- This completes the description of a simple fuzzy controller (and notice that we did not use a mathematical model in its construction).
- Extensions: MISO, quantification of "and" with fuzzy logic, other inference approaches, "Takagi-Sugeno" fuzzy systems
- Using a nonlinear model for a tanker ship we get the response in Figures 6 and 7 (tuned) and the controller surface in Figure 8.

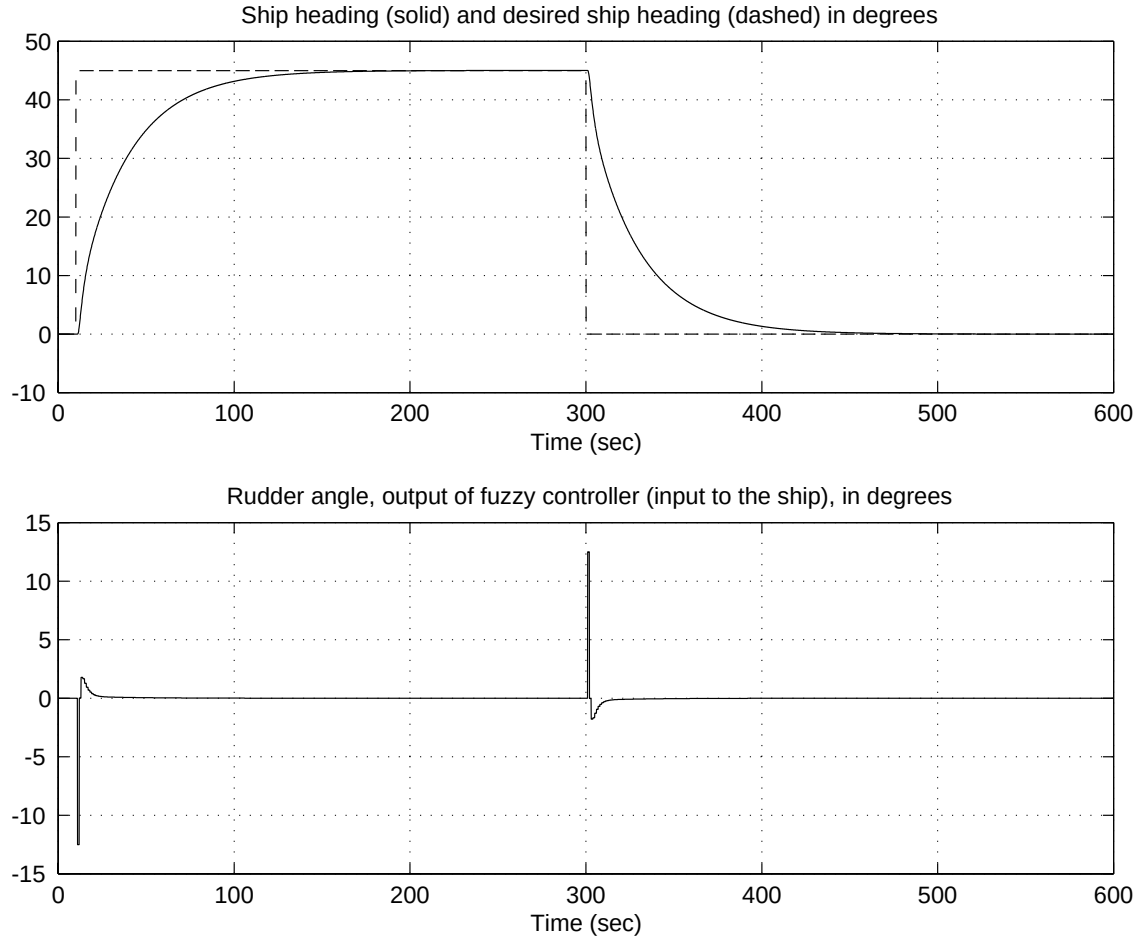

Figure 6: Response of fuzzy control system for tanker heading regulation, first guess (`g1=1/pi;,g2=10;,g0=2*pi/18;`).

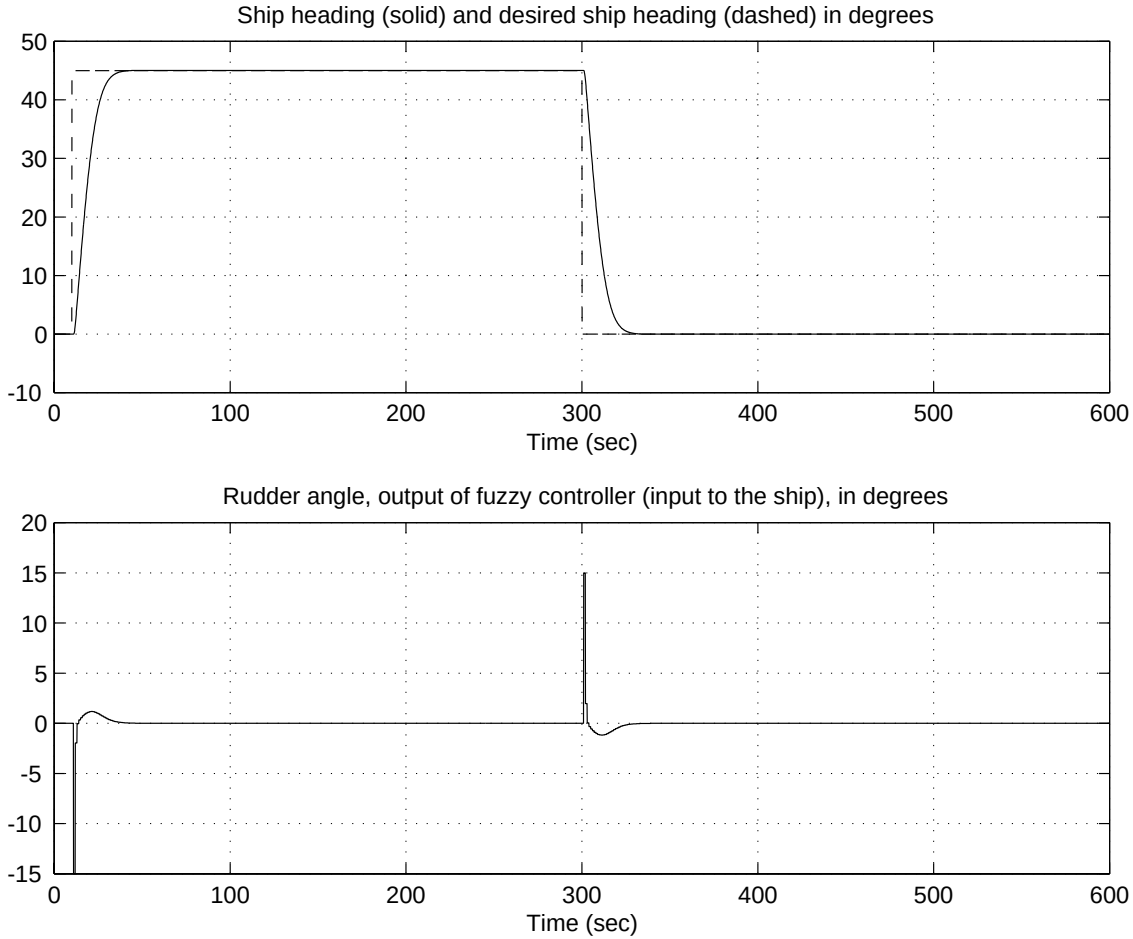

Figure 7: Response of fuzzy control system for tanker heading regulation, tuned parameters (`g1=2/pi;,g2=7;,g0=2*pi/18`).

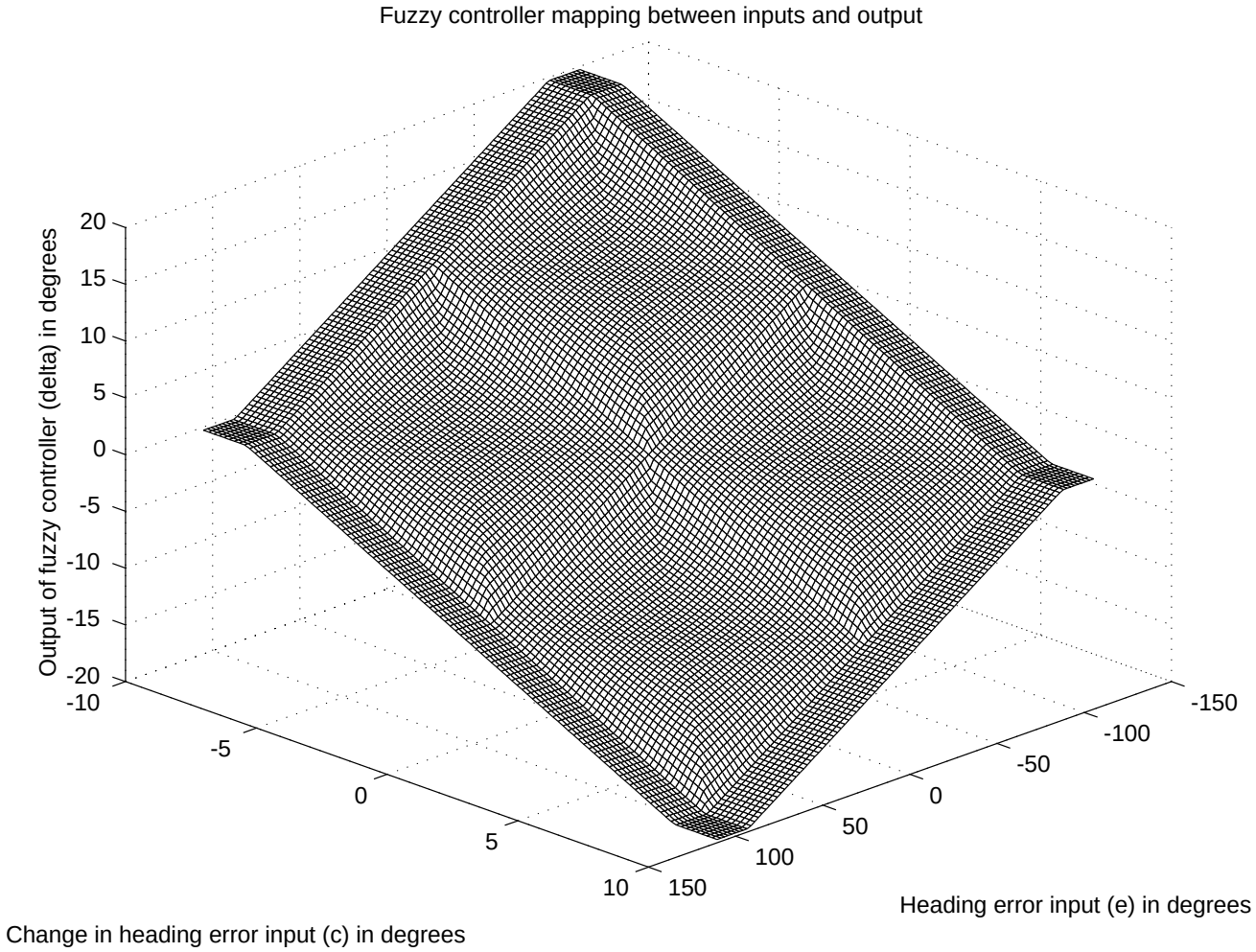

Figure 8: Fuzzy controller surface.

- Nothing mystical! A static (i.e., memoryless) nonlinear map. Can do stability analysis.
- For real-world applications most often the surface will have been shaped by the rules to have interesting nonlinear shapes
- How good is this fuzzy controller?
- Between trips, let there be a change from "ballast" to "full" conditions on the ship (a weight change) and suppose that on this trip there is a wind disturbance.
- In this case we get the response in Figure 9.

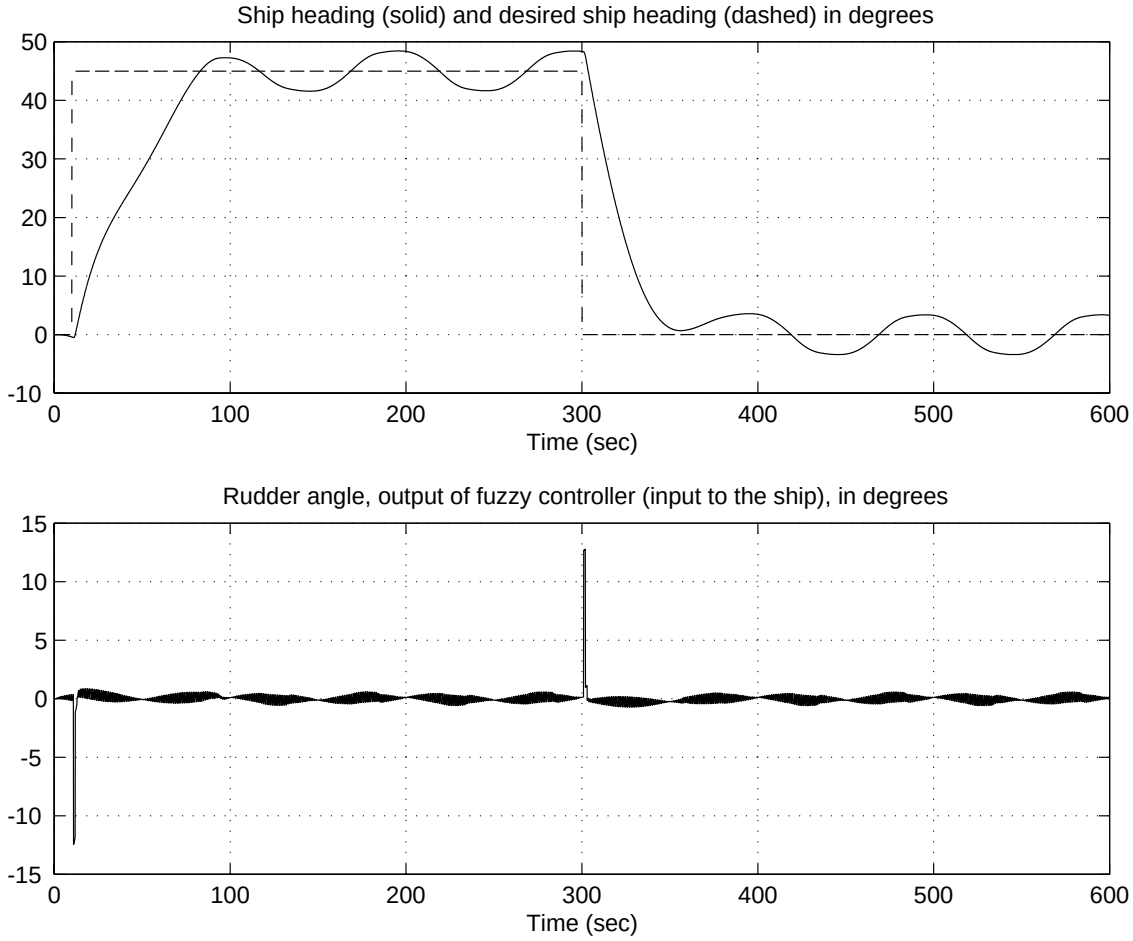

Figure 9: Response of fuzzy control system for tanker heading regulation, tuned parameters (`g1=2/pi;,g2=7;,g0=2*pi/18`), weight change and wind disturbance.

Adaptive Fuzzy Control

- Motivation: Can be difficult to design a rule-base (i.e., to shape the nonlinearity). Later, the plant may change, requiring re-tuning to maintain performance.
- Can we automatically synthesize/tune fuzzy controllers?
- Fuzzy model reference learning control (FMRLC) is one approach and it is shown in Figure 10

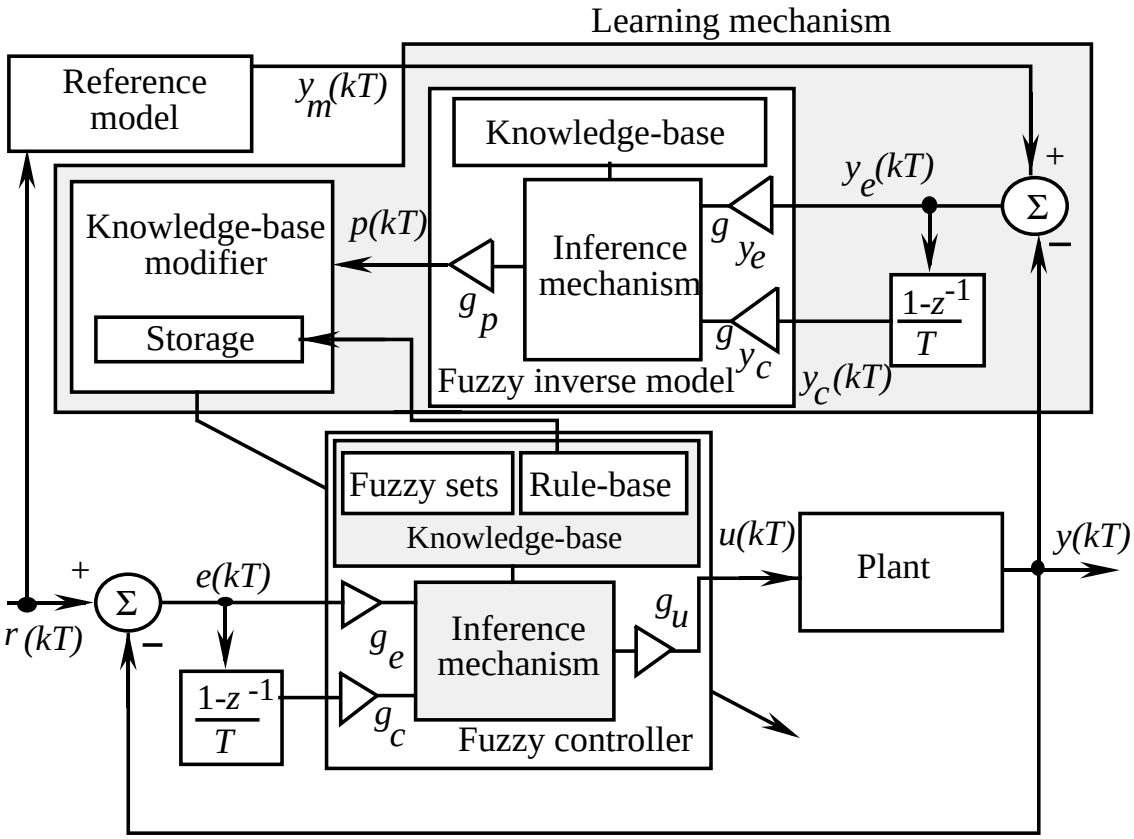

Figure 10: Fuzzy model reference learning controller.

- How does it work for the tanker ship?
  - Let all output membership function centers be placed at zero (initially it knows little about how to control the ship)
  - To see that it can synthesize a rule-base see the response in Figure 11 (compare to direct fuzzy controller)
  - To see that it can re-tune the controller for plant changes ("ballast to "full") and a wind disturbance see Figure 12 (compare to direct fuzzy controller)
  - Fuzzy controller surface synthesized by the FMRLC is in Figure 13 (but it is still being tuned).

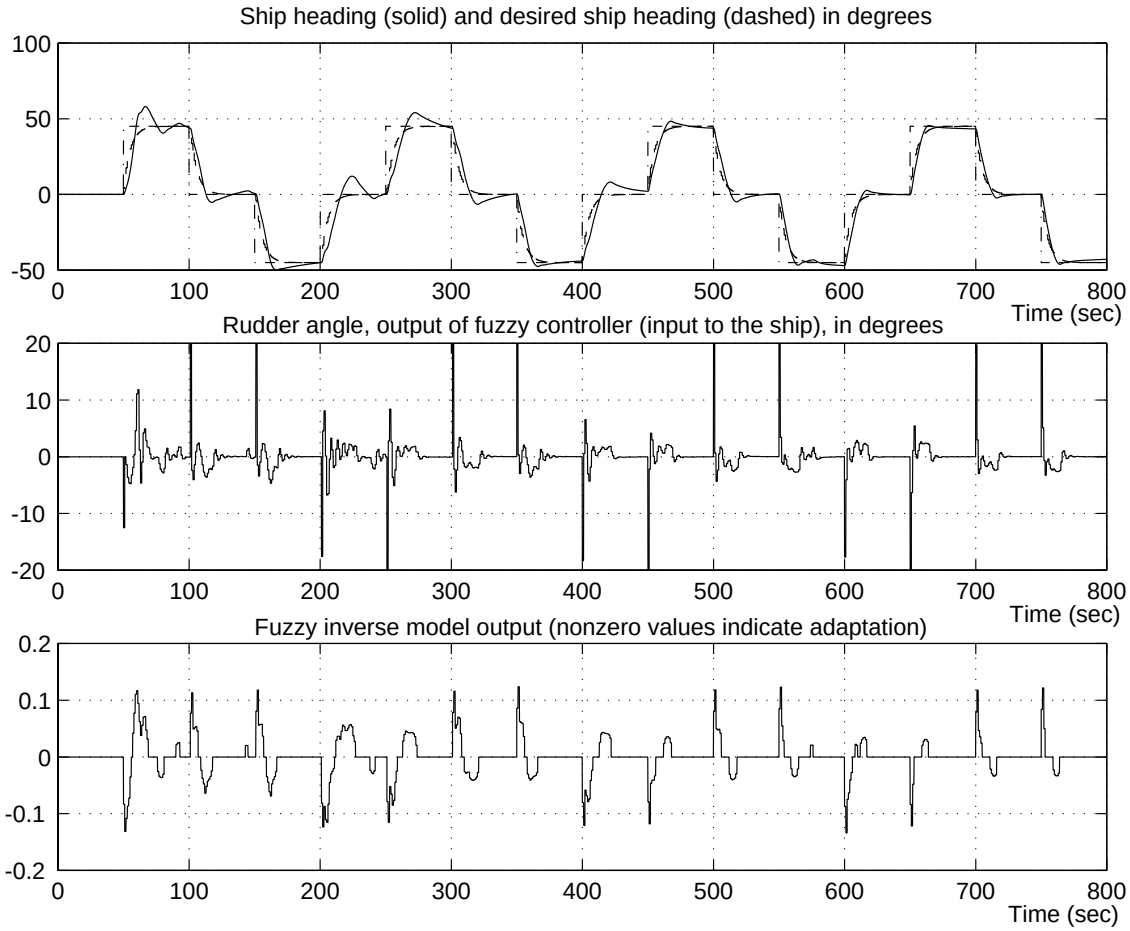

Figure 11: FMRLC response, rule-base synthesis for tanker.

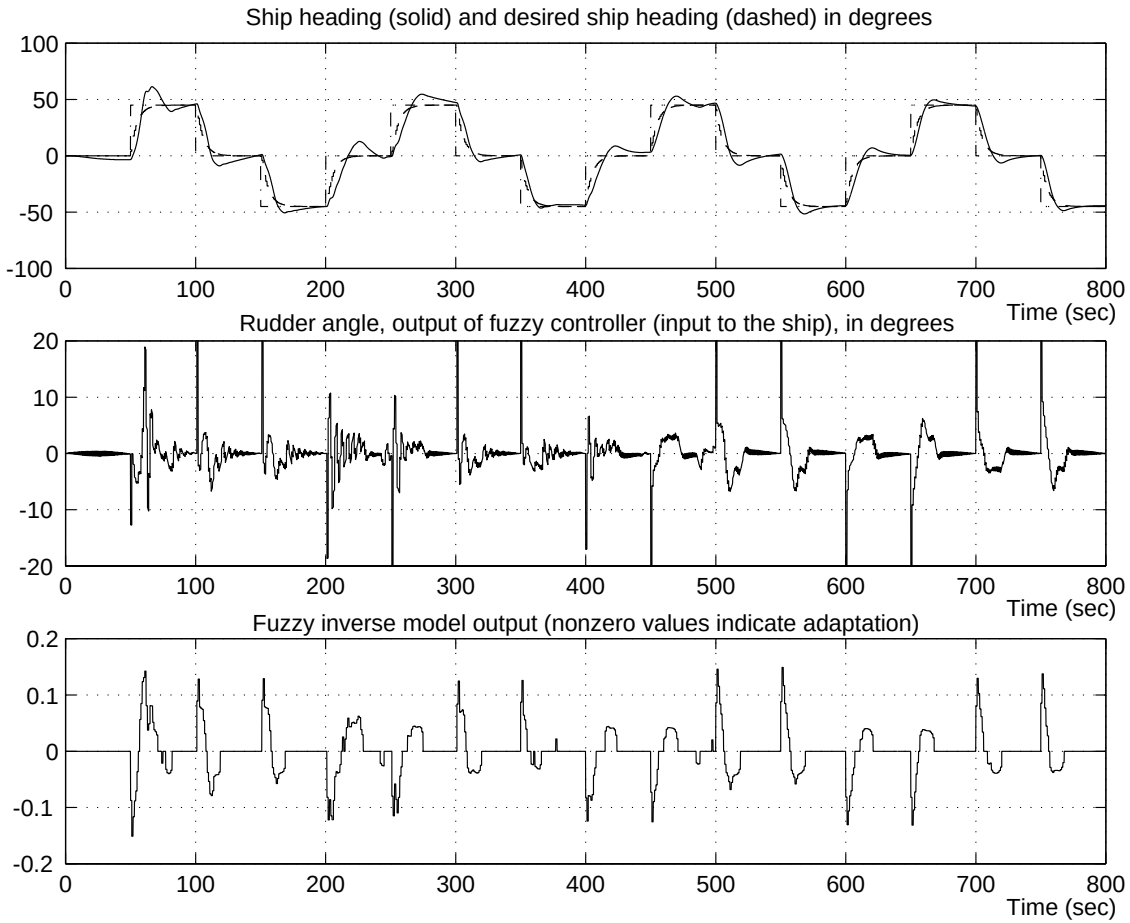

Figure 12: FMRLC response, rule-base synthesis/tuning for tanker (ship weight change and wind disturbance).

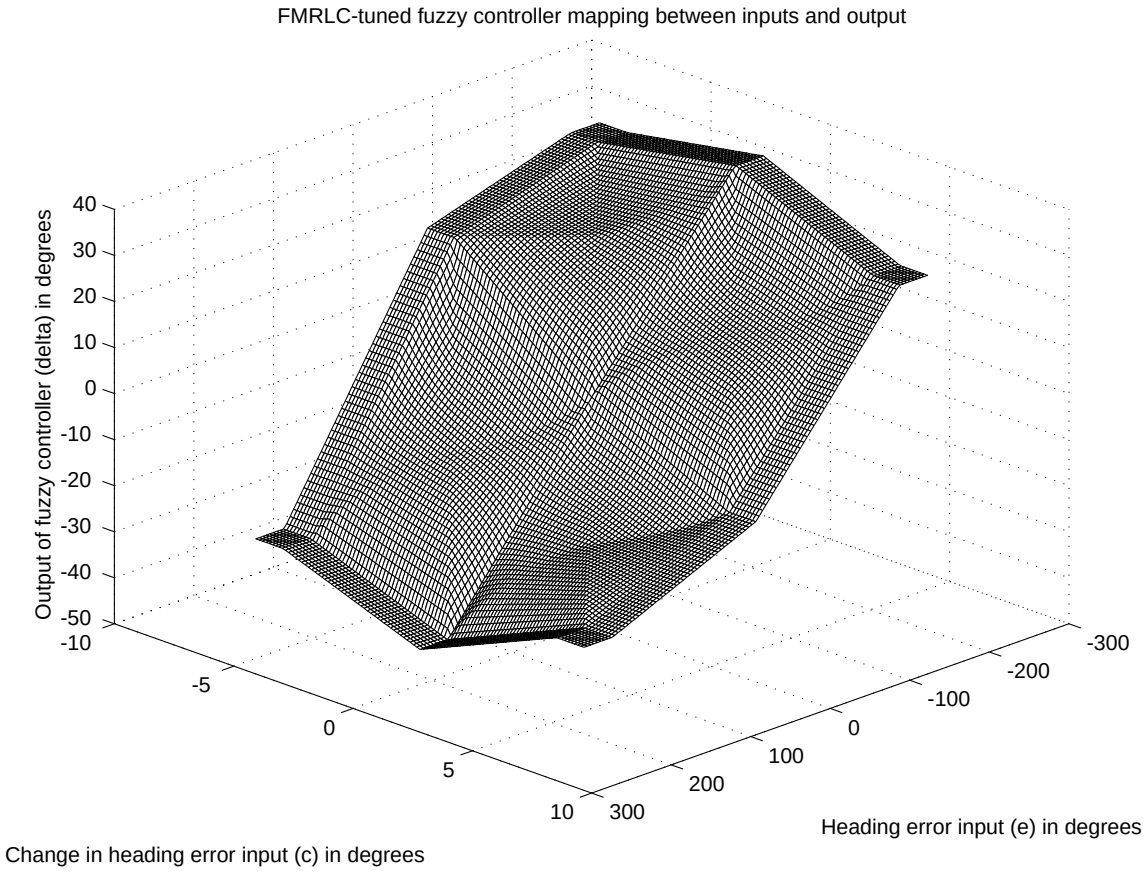

Figure 13: Synthesized fuzzy controller surface.

Remarks

- Design Concerns: Stability, robustness, comparative analysis
- Current Research: Stable robust adaptive strategies (several strategies already shown to be stable for uncertain nonlinear systems), complex learning strategies
- Fuzzy control, the main advantage: Heuristic (not necessarily model-based) approach to nonlinear controller construction.

Neural Networks

- Artificial neural networks are circuits, computer algorithms, or mathematical representations of the massively connected set of neurons that form biological neural networks.
- An alternative computing technology
- Proven useful in a variety of pattern recognition, signal processing, estimation, and control problems.
- Focus here: neural networks found useful in control

Multilayer Perceptrons

- Another popular neural network: radial basis function neural network (one form is identical to a fuzzy system).

The Neuron

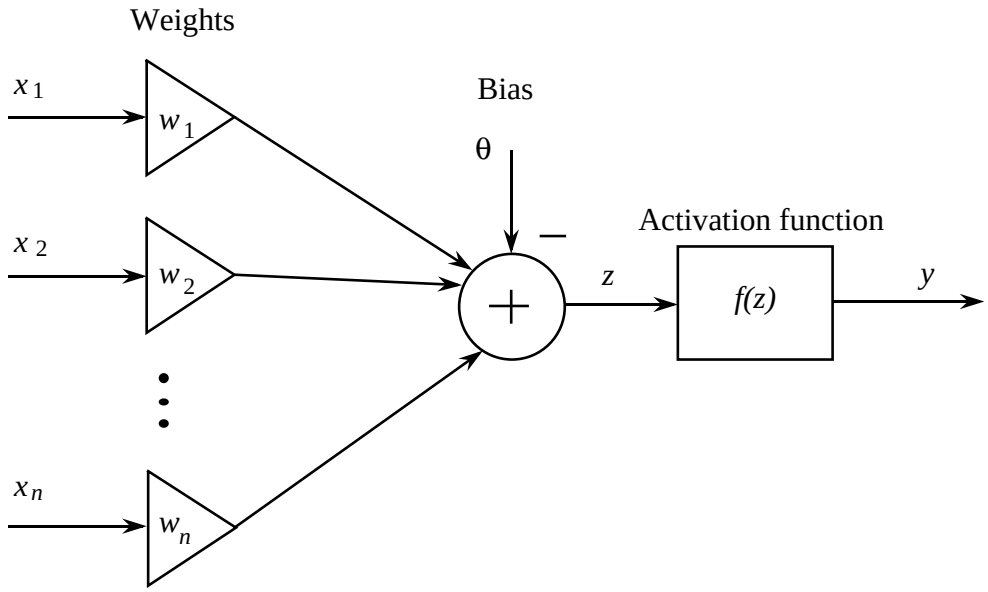

Figure 14: Single neuron model.

- Here,

$$z = \sum_{i=1}^{n} w_i x_i - \theta$$

  - w_i are the interconnection "weights"
  - θ is the "bias" for the neuron (these parameters model the interconnections between the cell bodies in the neurons of a biological neural network).

- The signal z represents a signal in the biological neuron and the processing that the neuron performs on this signal is represented with an "activation function" f

$$y = f(z) = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right). \tag{1}$$

- The neuron model represents the biological neuron that "fires" (turns on) when its inputs are significantly excited (i.e., z is big enough).
- "Firing" is defined by an activation function f (many possibilities):
  - Threshold Function:

$$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

  - Sigmoid Function:

$$f(z) = \frac{1}{1 + exp(-bz)} \tag{2}$$

Network of Neurons

- Equation (1), with one of the above activation functions, represents the computations made by one neuron - next, we interconnect them.
- Let circles represent the neurons (weights, bias, and activation function) lines represent the connections between the inputs and neurons, and the neurons in one layer and the next layer.
- Figure 15: A three "layer" perceptron since there are three stages of neural processing between the inputs and outputs.

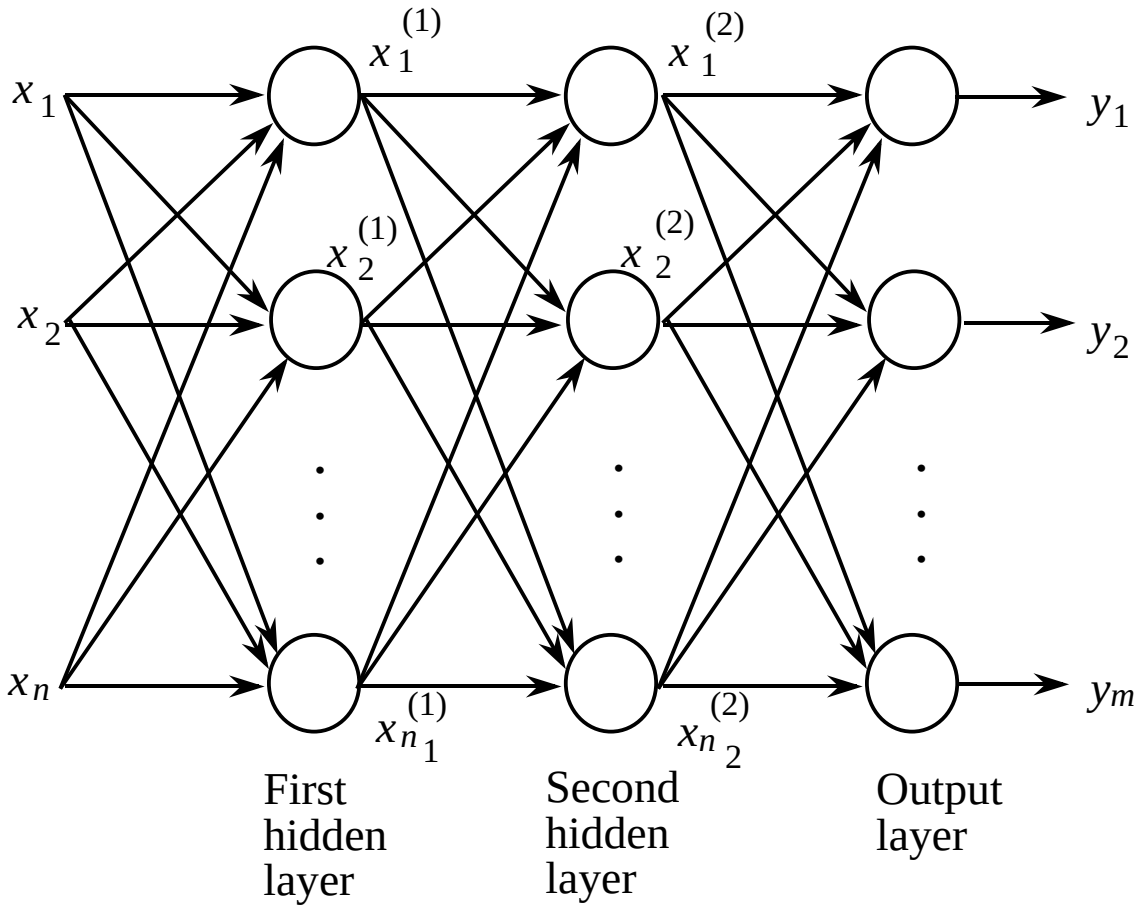

Figure 15: Multilayer perceptron model.

- We have

$$x_j^1 = f_j^1\left(\sum_{i=1}^{n} w_{ij}^1 x_i - \theta_j^1\right)$$

with $j = 1, 2, \ldots, n_1$.

- We have

$$x_j^2 = f_j^2\left(\sum_{i=1}^{n_1} w_{ij}^2 x_i^1 - \theta_j^2\right)$$

with $j = 1, 2, \ldots, n_2$.

- We have

$$y_j = f_j\left(\sum_{i=1}^{n_2} w_{ij} x_i^2 - \theta_j\right)$$

with $j = 1, 2, \ldots, m$.

Training Neural Networks

- How do we construct a neural network? We train it with examples.
- Regardless of the type of network, we will refer to it as $y = \mathcal{F}(x, A)$ where A is the vector of parameters that we tune to shape the nonlinearity it implements ($\mathcal{F}$ could be a fuzzy system too).
- Suppose that we gather input-output training data from a function $y = g(x)$ that we do not have an analytical expression for (e.g., it could be a physical process).
- Suppose that y is a scalar but that $x = [x_1, \ldots, x_n]^\top$.
- Suppose that $x^i = [x_1^i, \ldots, x_n^i]^\top$ is the i^{th} input vector to g and that $y^i = g(x^i)$.

- Let the training data set be

$$G = \{(x^i, y^i) : i = 1, \ldots, M\}$$

- Estimation/control problems:
  - For system identification the x^i are composed of past system inputs and outputs (a regression vector) and the y^i are the resulting outputs.
  - For parameter estimation the x^i can be regression vectors but the y^i are parameters that you want to estimate.

- Consider the situation in which it is desired to cause a neural network $\mathcal{F}(x, A)$ to match the function $g(x)$ at only a single point $\bar{x}$ where $\bar{y} = g(\bar{x})$.
- Given an input $\bar{x}$ we would like to adjust A so that the difference between the desired output and neural network output

$$e = \bar{y} - \mathcal{F}(\bar{x}, A) \tag{3}$$

  is reduced, where $\bar{y}$ may be either vector or scalar valued.
- In terms of an optimization problem, we want to minimize the cost function

$$J(A) = e^{\top} e. \tag{4}$$

- Taking infinitesimal steps along the gradient of $J(A)$ with respect to A will ensure that $J(A)$ is nonincreasing.
- That is, choose

$$\dot{A} = -\bar{\eta} \bigtriangledown J(A), \tag{5}$$

where $\bar{\eta} > 0$ is a constant and if $A = [a_1, \ldots, a_p]^\top$,

$$\bigtriangledown J(A) = \frac{\partial J(A)}{\partial A} = \begin{bmatrix} \frac{\partial J(A)}{\partial a_1} \\ \vdots \\ \frac{\partial J(A)}{\partial a_p} \end{bmatrix} \tag{6}$$

- Using the definition for $J(A)$ we get

$$\dot{A} = -\bar{\eta}\frac{\partial e^\top e}{\partial A}$$

or

$$\dot{A} = -\bar{\eta}\frac{\partial}{\partial A}(\bar{y} - \mathcal{F}(\bar{x}, A))^\top(\bar{y} - \mathcal{F}(\bar{x}, A))$$

so that

$$\dot{A} = -\bar{\eta}\frac{\partial}{\partial A}(\bar{y}^\top\bar{y} - 2\mathcal{F}(\bar{x}, A)^\top\bar{y} + \mathcal{F}(\bar{x}, A)^\top\mathcal{F}(\bar{x}, A))$$

- Now, taking the partial we get

$$\dot{A} = -\bar{\eta}(-2\frac{\partial \mathcal{F}(\bar{x}, A)^\top}{\partial A}\bar{y} + 2\frac{\partial \mathcal{F}(\bar{x}, A)^\top}{\partial A}\mathcal{F}(\bar{x}, A))$$

- If we let $\eta = 2\bar{\eta}$ we get

$$\dot{A} = \eta \left.\frac{\partial \mathcal{F}(\bar{x}, z)}{\partial z}\right|_{z=A}^\top (\bar{y} - \mathcal{F}(\bar{x}, A))$$

so

$$\dot{A} = \eta \zeta(\bar{x}, A) e \tag{7}$$

where $\eta > 0$, and

$$\zeta(\bar{x}, A) = \left.\frac{\partial \mathcal{F}(\bar{x}, z)}{\partial z}\right|_{z=A}^\top, \tag{8}$$

- Discrete-time updates: series updates
- Series updating is accomplished by selecting the pair (x^i, y^i) where $i \in \{1, \ldots, M\}$ is a random integer chosen at each iteration and then use Euler's first order approximation of so that the parameter update is defined by

$$A(k+1) = A(k) + \eta \zeta^{i\top}(k) e(k). \tag{9}$$

  where k is the iteration step, $e(k) = y^i - \mathcal{F}(x^i, A(k))$ and

$$\zeta^i(k) = \left. \frac{\partial \mathcal{F}(x^i, z)}{\partial z} \right|^{\top}_{z=A(k)}. \tag{10}$$

- For the problem when M input-output pairs, or patterns, (x^i, y^i) where $y^i = g(x^i)$ are to be matched (i.e., a "batch update") let

$$e^i = y^i - \mathcal{F}(x^i, A), \tag{11}$$

and let the cost function be

$$J(A) = \sum_{j=1}^{q} {e^i}^\top e^i. \tag{12}$$

and like above you can derive the update formulas.

- This is actually the "backpropagation method" (except we have not noted the fact that due to the structure of the layered neural networks certain computational savings are possible).
- Extensions: Levenberg-Marquardt and Conjugate-Gradient methods

Design Concerns

- Training sets
- Approximator structure choice (e.g., universal approximation, size)
- Convergence in training? Local and global minima
- When to stop training? Validation set
- Test set
- Interpolation and "generalization," over-matching

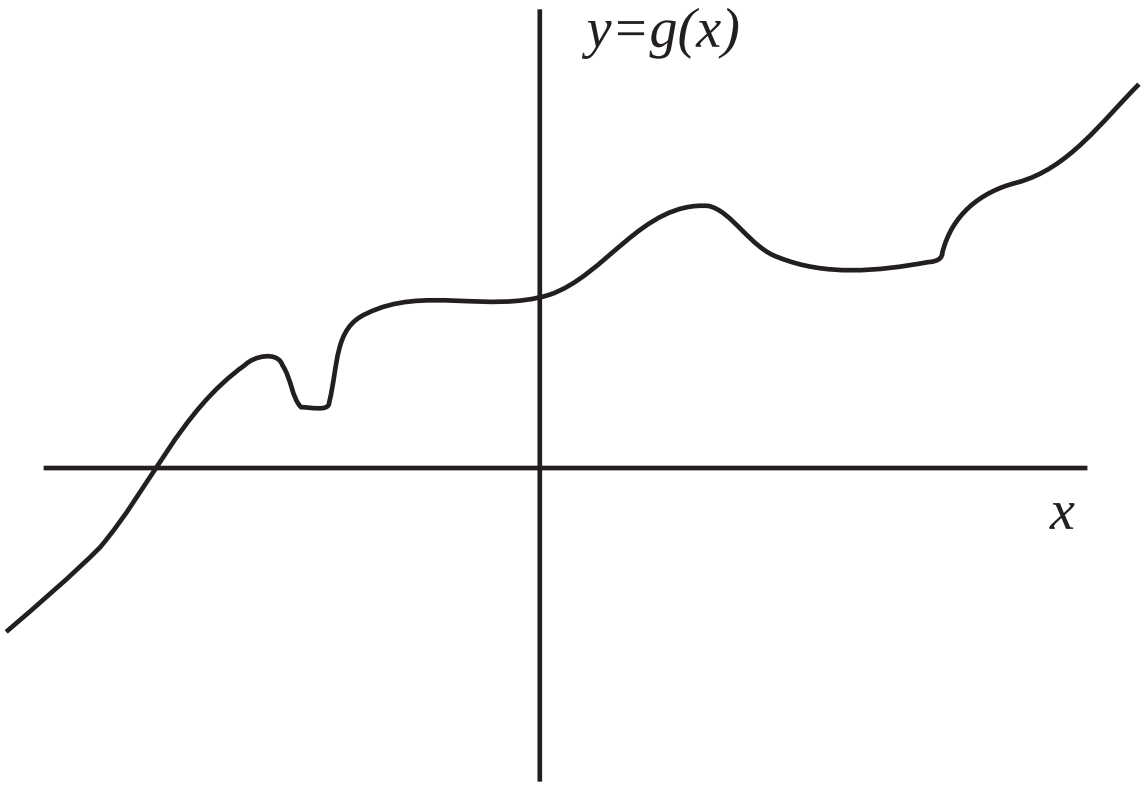

Figure 16: Example function we would like to approximate.

Application: Thrust Estimation for a Jet Engine

- Jet engine application
- Why is estimation needed?
  - For more accurate thrust regulation (and stall margin regulation)
  - For failure diagnosis
- Approach:
  - Use engine model to generate engine input-output data for different parameter values and flight conditions
  - Train neural network to approximate the mapping inherent in the data
  - Test under engine quality and deterioration differences, flight conditions

Neural Control

- Most common approach shown in Figure 17 where we use a neural network as the identifier model and, for example, a gradient method for updating it.

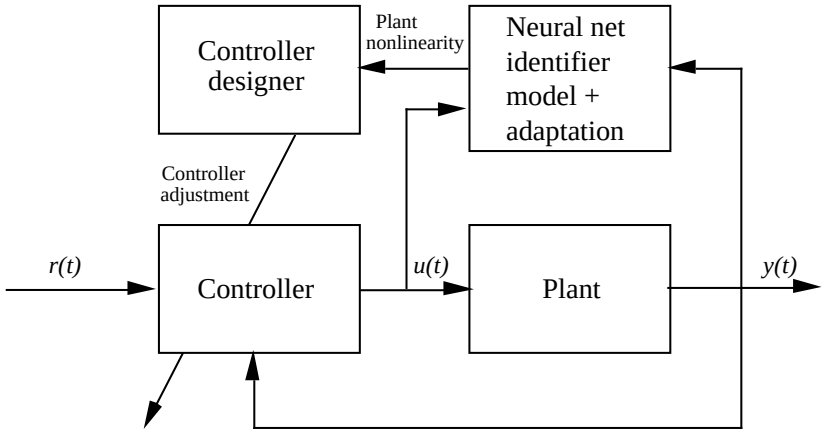

Figure 17: Indirect adaptive control via a neural network.

- Many heuristic approaches

Remarks

- Design Concerns: Stability, robustness, comparative analysis
- Current Research: Stable robust adaptive strategies (several strategies already shown to be stable for uncertain nonlinear systems), complex learning strategies
- Neural networks, the main advantage: Can achieve good approximation accuracy with a reasonable number of parameters by training with data (hence, a lack of dependence on models)

Genetic Algorithms

- A genetic algorithm (GA) uses the principles of evolution, natural selection (Darwin), and genetics (Mendel) from natural biological systems in a computer algorithm to simulate evolution.
- It is a stochastic optimization technique that performs a parallel, stochastic, but directed search to evolve the most fit population.
- If it "gets stuck" at a local optimum it tries to simultaneously find other parts of the search space that will allow it to "jump out" of the local optimum to a global one.
- Does not need "gradient information" (but could use it)

Representation and the Population of Individuals

- The "fitness function" measures the ability to survive.
- The GA seeks to maximize the fitness function $J(\theta)$ by selecting the individuals that we represent with θ.
- To represent the GA in a computer we make θ a string.

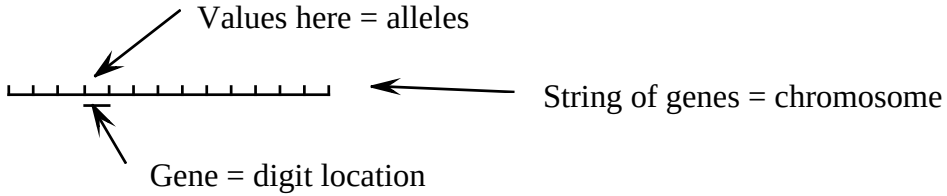

Figure 18: String for representing an individual.

- In base-2: alleles are $\{0, 1\}$
- In base-10: alleles are $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Binary chromosome: 1011110001010
- Base-10 chromosome: 8219345127066

- We add a gene for the sign of the number and fix a position for the decimal point.
- Chromosome encodes a fuzzy system, neural network, or a controller's parameters
- For example, to tune the fuzzy controller discussed earlier for the tanker ship you could use the chromosome:

$$b_1 b_2 \cdots b_p$$

  (these are the output membership function centers).
- How do we represent a set of individuals (i.e., a population)?
- Let $\theta_i^j(k)$ be a single parameter at time k (a fixed length string with sign digit) and suppose that chromosome j is composed of N of these parameters that are sometimes called "traits."

- Let
$$\theta^j(k) = \left[\theta_1^j(k), \theta_2^j(k), \ldots, \theta_N^j(k)\right]^\top.$$
be the j^{th} chromosome.
- The population at time k ("generation") is
$$P(k) = \left\{\theta^j(k) : j = 1, 2, \ldots, S\right\} \tag{13}$$
- Pick S to be big enough, but not too big
- Evolution occurs as we go from a generation at time k to the next generation at time $k+1$ via the use of genetic operations:

Selection:

- Darwin: the most qualified individuals survive to mate.
- Quantify "most qualified" via an individual's fitness $J(\theta^j(k))$.
- Create a "mating pool" at time k:

$$M(k) = \left\{ m^j(k) : j = 1, 2, \ldots, S \right\}. \tag{14}$$

- Select an individual for mating by letting each $m^j(k)$ be equal to $\theta^i(k) \in P(k)$ with probability

$$p_i = \frac{J(\theta^i(k))}{\sum_{j=1}^{S} J(\theta^j(k))}. \tag{15}$$

- More fit individuals will tend to end up mating more often, thereby providing more off-spring

Reproduction Phase, Crossover:

- Crossover = mating in biological terms (the process of

combining chromosomes), for individuals in $M(k)$.

- Procedure (use "crossover probability" p_c): Randomly pair off the individuals in the mating pool $M(k)$, then see Figure 19

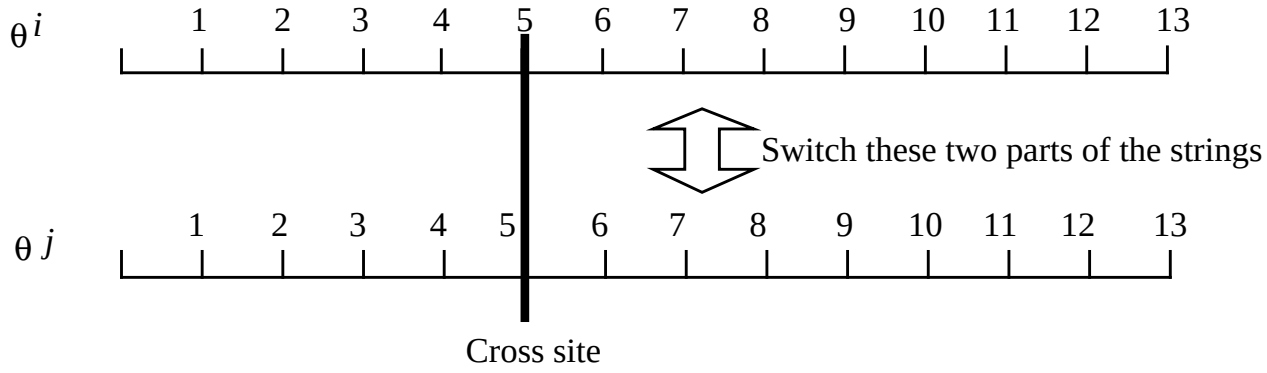

Figure 19: Crossover operation example.

- Crossover perturbs the parameters near good positions to try to find better solutions to the optimization problem.

Reproduction Phase, Mutation:

- The biological analog of our mutation operation is the random

mutation of genetic material.

- With probability p_m change (mutate) each gene location on each chromosome randomly to a member of the number system being used.
- Mutation tries to make sure that we do not get stuck at a local maxima and that we seek to explore other areas of the search space to help find a global maximum for $J(\theta)$.
- Next generation:

$$P(k+1) = M(k)$$

- Evolution = repitition of the above process

Design Concerns

- Understand the optimization problem
  - Know what you want to optimize (fitness function choice)
  - Parameters? Constraints? How good must you do??
- Representation (accuracy, complexity)
- Choice of genetic operators
- Stability, convergence, and robustness
  - Local and global optima
  - How long can you wait?

Evolution of Coffee Growers and Coffee Plantations in Colombia

- Arabica coffee bean grows best at elevations of about 1000 to 1800 meters (so there is a region of optimum positions to put a coffee plantation and for coffee growers to live)
- Topographical data for Colombia
  - National Geophysical Data Base
  - 5 minute data for entire region
  - Use linear interpolation for points in between available data

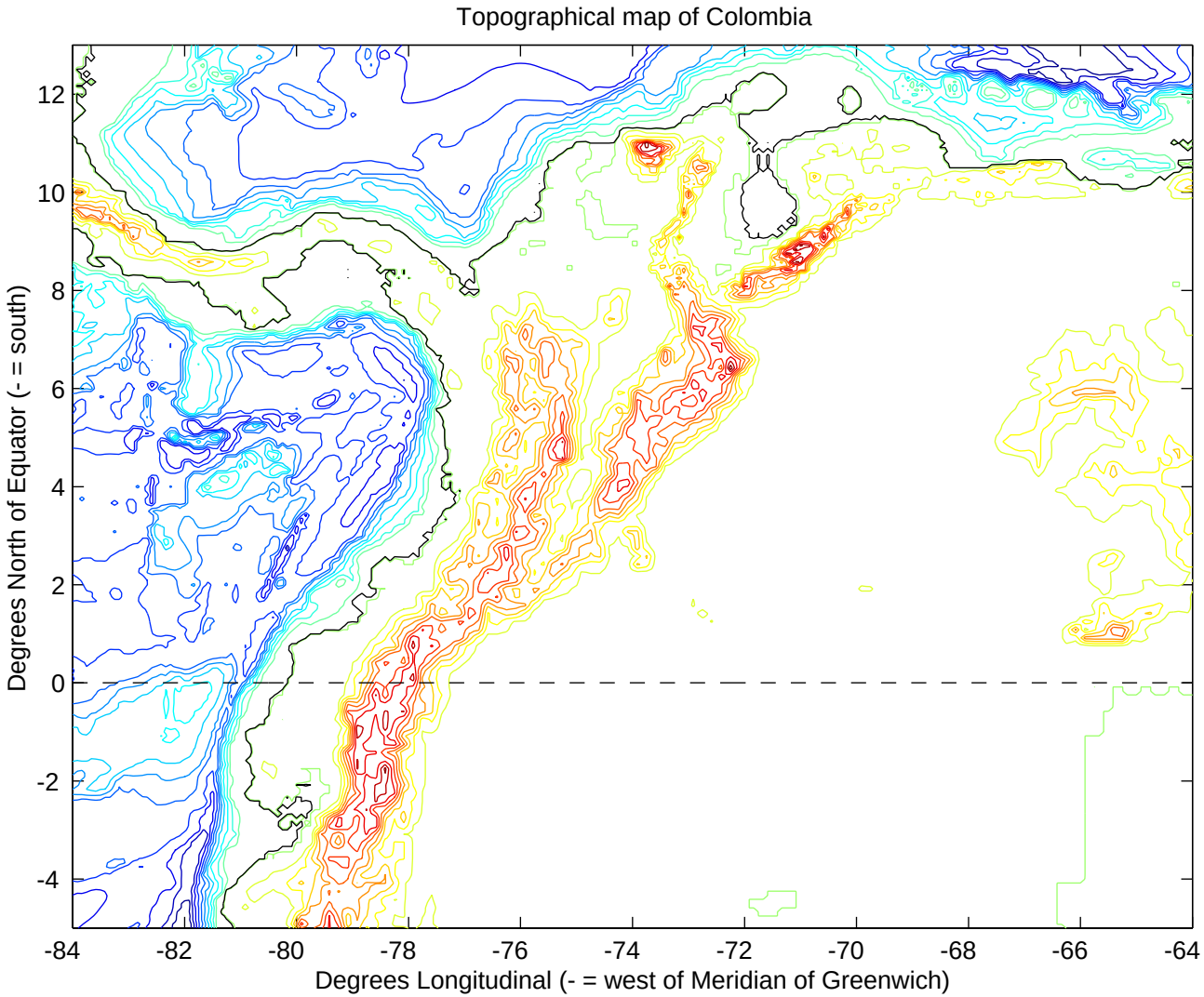

Figure 20: Topographical map of Colombia and surrounding regions.

- GA for showing evolution of coffee growers and plantations
  - Fitness: Gaussian function of elevation, centered at 1400 meters
  - 1000 individuals try to start coffee farms (keep population size constant), run for 10 generations
  - Randomly distribute initially (but not in the water)
  - Mutation probability: 0.05; Crossover probability: 0.9
  - If an odd or mutant child jumps into the water, fix them and re-settle them at a random point on land
  - Locations of attempts by individuals are indicated by a dot
  - Individuals who are near the correct elevation are more likely to propagate to have children in the next generation (so we give them a plantation, indicated by a small circle)

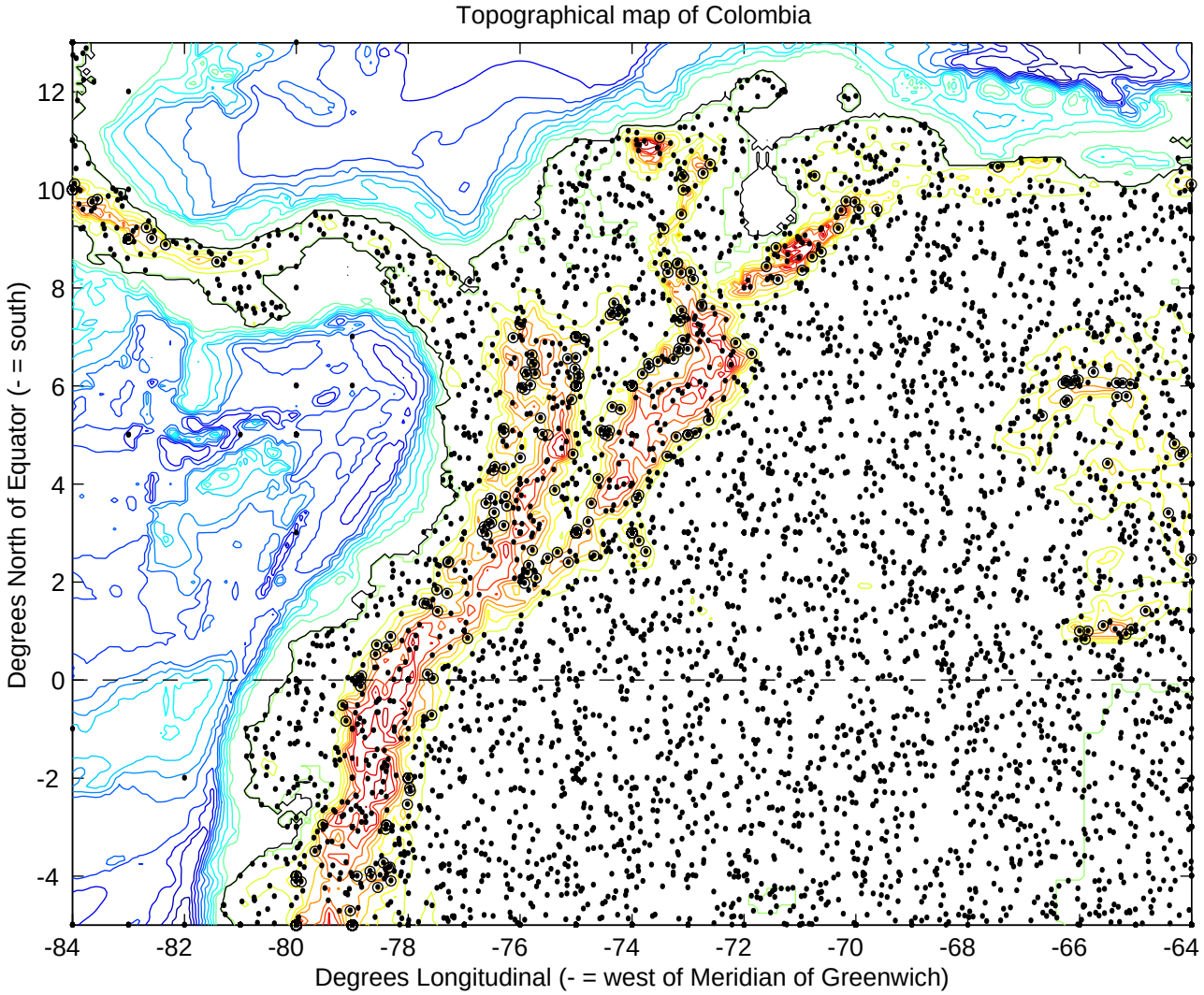

Figure 21: Evolved coffee growers and plantations.

- Some just luckily started in the right place and hence had children who stayed in the area and also started plantations
- Coffee growers near the correct regions had more children who stay in that region
- Some fit individuals mated with less fit ones and moved closer (or to) a good coffee growing regions
- The trend towards optimizing their placement of coffee plantations is shown in Figure 22

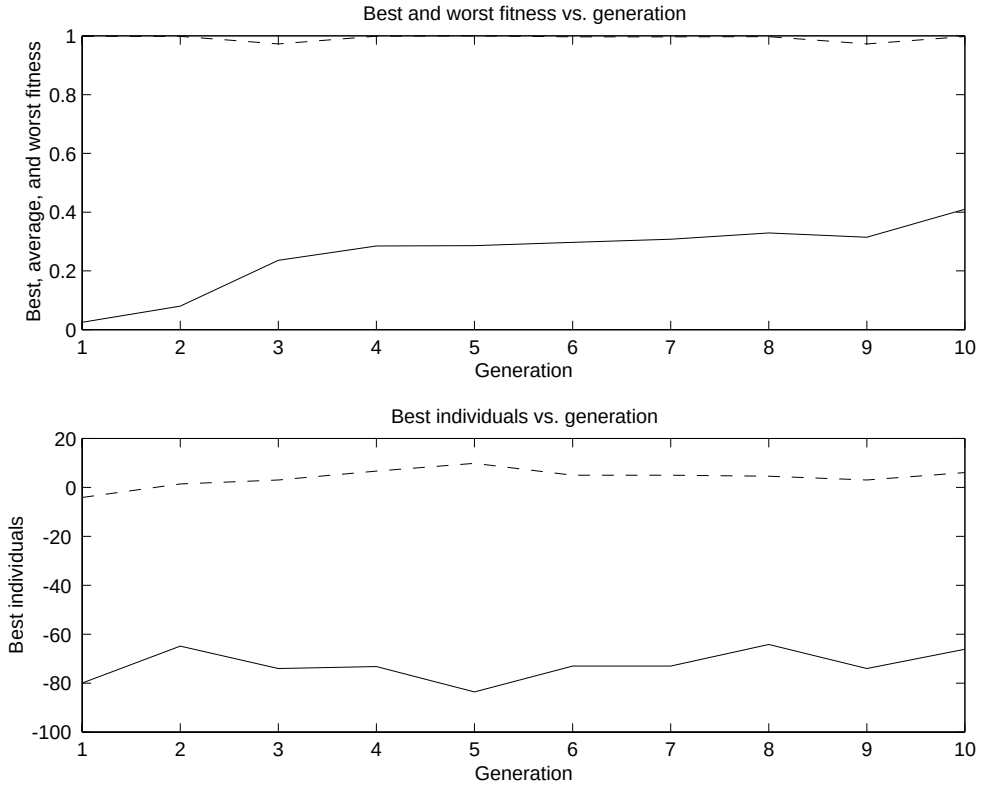

Figure 22: Fitness and best individuals.

Applications

- Aircraft design, manufacturing system design
- Computer-aided control system design (optimize closed-loop system performance by evolving the controller off-line)
- Design evolution of physical hardware
- Genetic adaptive control...

Genetic Adaptive Control for Brake Systems

- In the base-braking control problem we seek to develop a controller that can ensure that the braking torque commanded by the driver will be achieved (can help anti-skid brake system (ABS) also)
- We need adaptive control to reduce the effects of variations in the process due to temperature.
- Work supported by Delphi Chassis Division of General Motors (significant previous work on implementation of conventional controllers, both fixed and adaptive)

The Base Braking Control Problem

- Figure 23 shows the diagram of the base braking system

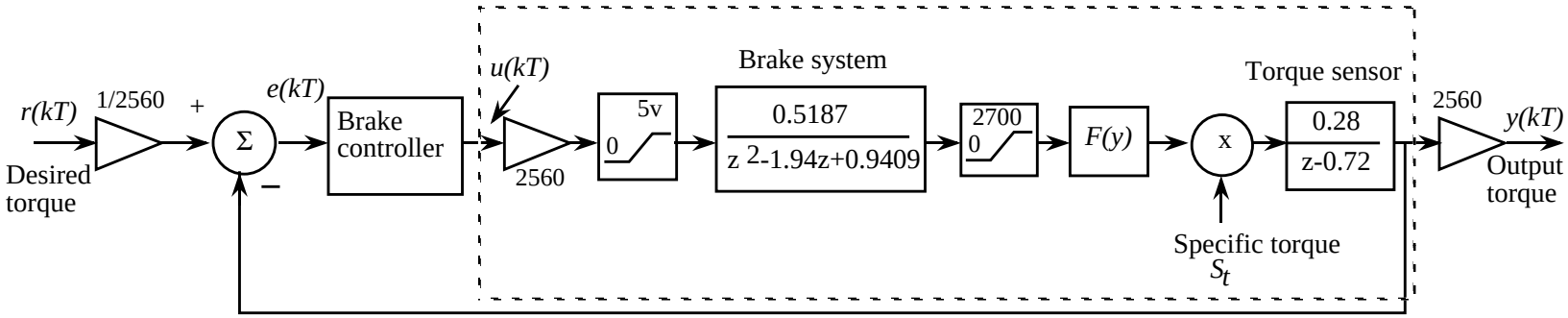

Figure 23: Base braking control system.

- $r(kT)$ is the braking torque requested by the driver.
- $y(kT)$ is the output of a torque sensor
- Also,

$$F(y) = \frac{y}{2502.4419} + 0.0139878$$

- The function $F(y)$ was experimentally determined and

represents the relationship between brake fluid pressure and the stopping force on the car.

- Next, $F(y)$ is multiplied by the specific torque S_t.
- The specific torque (S_t) in the braking process reflects the variations in the stopping force of the brakes as the brake pads increase in temperature and

$$0.85 \leq S_t \leq 1.70.$$

- The lower value corresponds to cold brakes (125 degrees F) and the higher value corresponds to hot brakes (250 degrees F).

- All the simulations we conducted use a repeating 4 second input reference signal and a varying specific torque to represent the brakes heating up as shown in Figure 24.

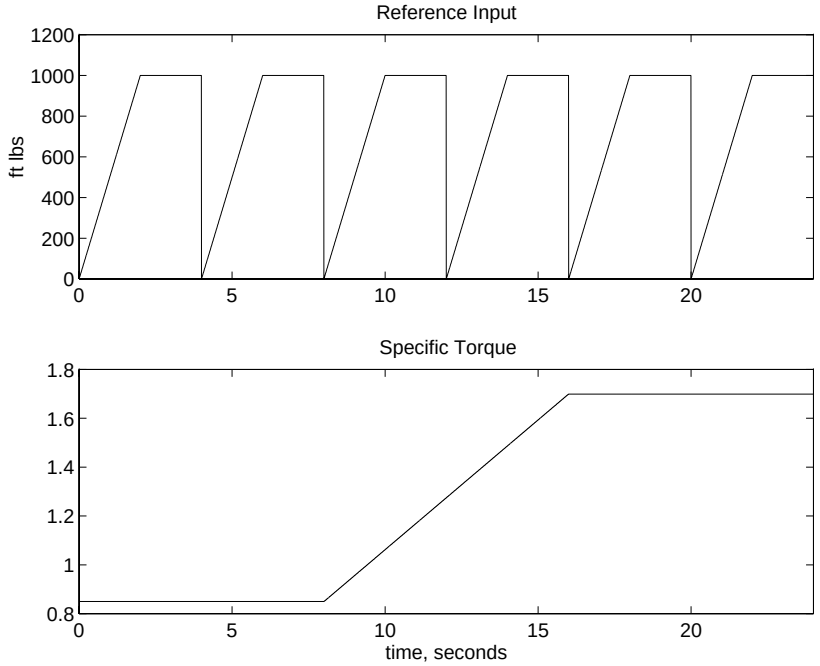

Figure 24: Reference input and specific torque.

Genetic Model Reference Adaptive Control

- The GMRAC is shown in Figure 25 (it uses a base-10 GA with elitism)

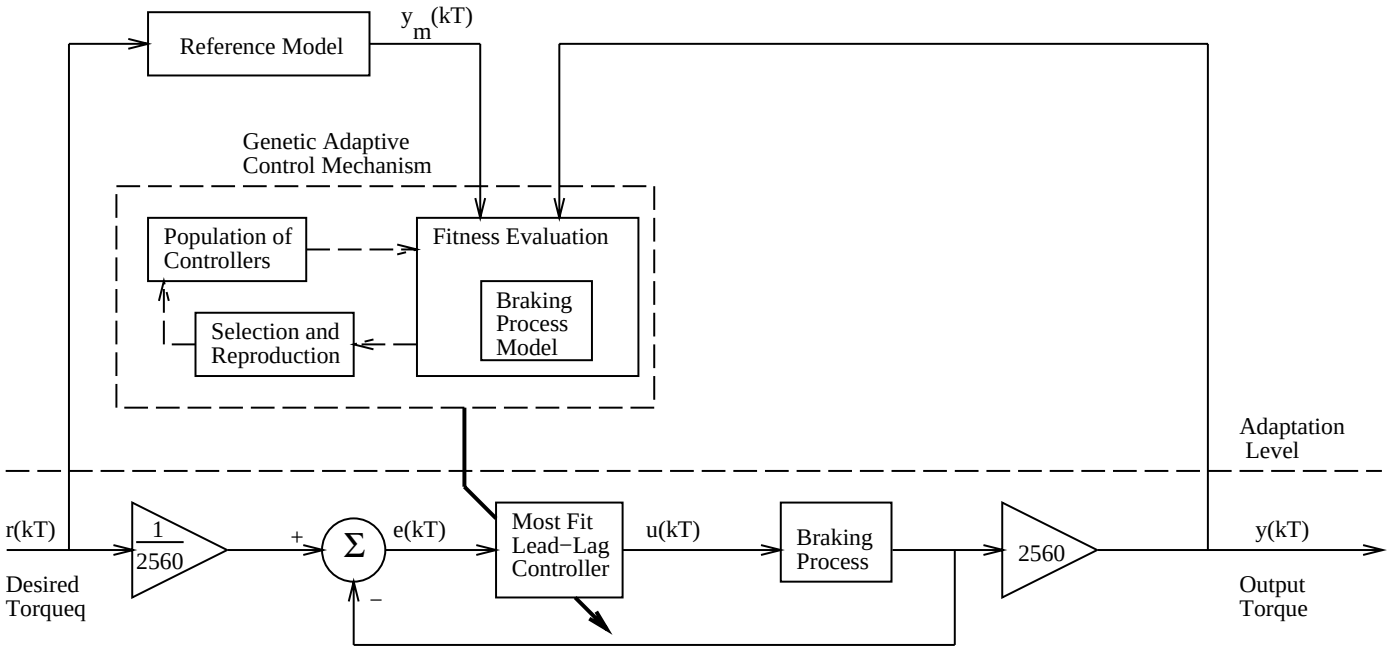

Figure 25: GMRAC for base braking.

- It uses a simplified model of the braking process to evaluate a population (set) of braking controllers at each step (by simulating the performance of each candidate controller ahead in time at each step)
- It selects the most fit controller from the population to control the process at each step
- In this way it "evolves" a good controller for the braking process.
- Figure 26 shows the results of the braking simulation using the GMRAC.

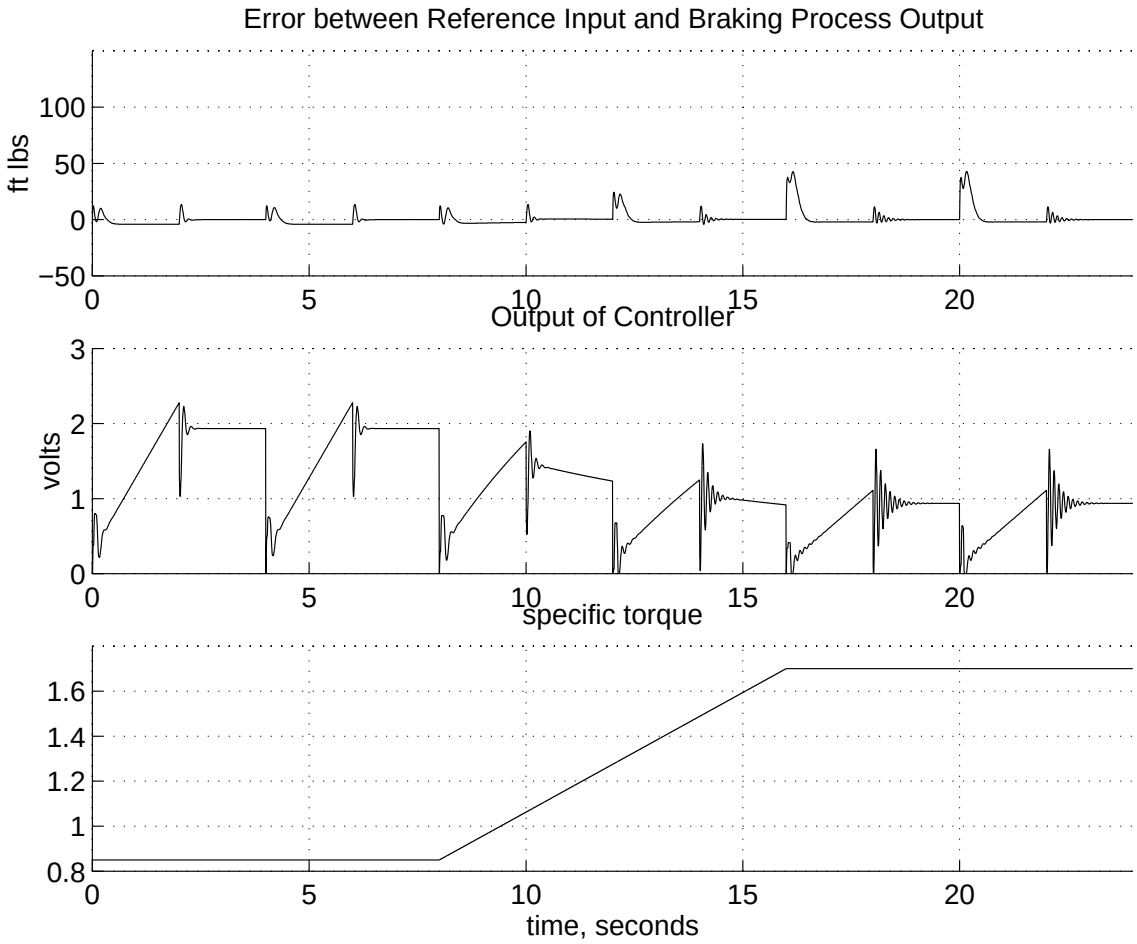

Figure 26: Results using GMRAC.

- The GMRAC does well, but its performance does degrade somewhat as specific torque increases (this problem can be solved with a seeding of fixed controllers in the population).
- Another even more successful approach: Use another GA to evolve the model used in the fitness function for the GMRAC (and can seed population with guesses of the model)

Remarks

- Design Concerns: Stability, robustness, comparative analysis
- Current Research: Complex stochastic learning strategies
- Genetic algorithms, the main advantage: Evolution-based stochastic search that can be useful in finding good solutions to practical complex optimization problems (sometimes models are not needed)

Outlook on Intelligent Control

- Current theoretical research:
  - Mathematical stability/convergence/robustness analysis for learning systems
  - Mathematical comparative analysis with nonlinear adaptive methods
- Current research on the development of new techniques:
  - Complex heuristic learning strategies
  - Memory and computational savings
  - Coping with "hybrid" discrete event / differential equation models

- Applications/Implementations:
  - Definite need for experimental research (especially in comparative analysis and new non-traditional applications)
  - Definite successes in industry (examples)
  - Working with industry is challenging and important.
- Intelligent control not only tries to borrow ideas from the sciences of physics and mathematics to help develop control systems, but also biology, neuroscience, psychology, and others... It is refreshing and fun! :-)