

INTELLIGENT MOTION CONTROL WITH AN ARTIFICIAL CEREBELLUM

Russell L. Smith
July 1998

COPYRIGHT ©1998 RUSSELL L. SMITH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN ENGINEERING.



THE DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
UNIVERSITY OF AUCKLAND,
NEW ZEALAND

*The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say.*

—J.R.R. Tolkien

TO MY FRIENDS AND LAB-MATES —
AARON, BRIAN, CHRIS, DAVE,
DOMINIC, GAVIN, GEOFF, LEE,
MELISSA, RUSS M., STEVE, SYLVIA,
TONY AND WOEI . . .
. . . IT'S BEEN A LONG JOURNEY,
BUT THANKS TO YOU, A FUN ONE.

TO MY FAMILY, MUM, DAD AND RAEWYN.

TO MY SUPERVISOR, GEORGE.

Abstract

This thesis describes a novel approach for adaptive optimal control and demonstrates its application to a variety of systems, including motion control learning for legged robots. The new controller, called “FOX”, uses a modified form of Albus’s CMAC neural network. It is trained to generate control signals that minimize a system’s performance error. A theoretical consideration of the adaptive control problem is used to show that FOX must assign each CMAC weight an “eligibility” value which controls how that weight is updated. FOX thus implements a kind of reinforcement learning which makes it functionally similar to the cerebellum (a part of the brain that modulates movement). A highly efficient implementation is described which makes FOX suitable for on-line control.

FOX requires a small amount of dynamical information about the system being controlled: the system’s impulse response is used to choose the rules that update the eligibility values. A FOX-based controller design methodology is developed, and FOX is tested on four control problems: controlling a simulated linear system, controlling a model gantry crane, balancing an inverted pendulum on a cart, and making a wheeled robot follow a path. In each case FOX is effective: it associates sensor values with (and anticipates) the correct control actions, it compensates for system nonlinearities, and it provides robust control as long as the training is comprehensive enough.

FOX is also applied to the control of a simulated hopping monopod, and a walking biped. FOX learns parameters that fine tune the movements of pre-programmed controllers, in a manner analogous to the cerebellar modulation of spinal cord reflexes in human movement. The robots are successfully taught how to move with a steady gait along flat ground, in any direction, and how to climb and descend slopes.

Contents

Title	i
Contents	xiv
Nomenclature	xv
Contents of the CDROM	xix
1 Introduction	1
1.1 Intelligent motion control	1
1.2 Thesis overview	2
1.3 Thesis contribution	3
1.4 Background on robots and learning	3
1.4.1 Why is intelligent control useful?	3
1.4.2 Modular behavior and the break with classical AI	4
1.4.3 Neural network control	5
1.4.4 Reinforcement learning (RL)	5
1.4.5 Eligibility-based RL techniques, and others	6
1.5 Summary	6
2 Biological motor control	7
2.1 Introduction	7
2.2 Overall structure	8
2.3 Neuron biology	9
2.3.1 Neuron models	12
2.3.2 Models of learning	12
2.4 Muscles	13
2.4.1 Muscle contraction	14
2.4.2 Muscle spindles	14
2.4.3 Golgi tendon organs	15
2.5 Spinal cord	15
2.5.1 Stretch reflex	17
2.5.2 Reciprocal inhibition reflex	17
2.5.3 Tendon reflexes	18
2.5.4 More complex reflexes, and central pattern generators	18
2.5.5 Postural and locomotion reflexes	18
2.5.6 Other spinal reflexes	19

2.5.7	Spinal motor model	19
2.6	Brain Stem	20
2.6.1	Posture and equilibrium	20
2.7	Cerebellum	21
2.7.1	Cerebellar cortex: Anatomy	21
2.7.2	Cerebellar cortex: Operation	22
2.7.3	Cerebellar cortex: Training	25
2.8	Motor cortex	26
2.9	Biological realism	26
2.10	Conclusion	27
3	The CMAC neural network	29
3.1	Introduction	29
3.2	The CMAC	29
3.2.1	The CMAC as a neural network	29
3.2.2	The CMAC as a lookup table	31
3.3	Training	34
3.4	Hashing	34
3.4.1	The number of CMAC weights	34
3.4.2	What is hashing?	35
3.4.3	How the CMAC uses hashing	36
3.4.4	CMAC hashing algorithms	36
3.5	Properties of the CMAC	38
3.5.1	Limited input space	38
3.5.2	Piecewise constant	38
3.5.3	Local generalization	38
3.5.4	Training sparsity	38
3.5.5	Training interference	41
3.5.6	Multidimensional inflexibility	43
3.5.7	Comparison with the MLP	43
3.6	Design decisions	44
3.6.1	Input issues	44
3.6.2	Overlay displacements	44
3.6.3	Hashing performance	45
3.6.4	Number of AUs	45
3.6.5	Weight smoothing	48
3.7	Extensions	48
3.8	Conclusion	50
4	Feedback-error control	51
4.1	Introduction	51
4.2	The basics	51
4.3	The feed-forward path	52
4.4	The feedback path	52
4.5	Feed-forward and feedback together	53
4.6	Feedback-error training	53
4.7	Why it works	54

4.8	Some other FBE features	54
4.8.1	Step changes in y_d	54
4.8.2	Unskilled feedback	54
4.9	An example	55
4.10	Arbitrary reference trajectory	56
4.11	Output limiting	57
4.12	Conclusion	60
5	Theory of the FOX controller	61
5.1	Introduction	61
5.2	FBE and weight eligibility	62
5.3	Introduction to FOX	63
5.3.1	The critic function	66
5.3.2	The sensor vector and CMAC inputs	67
5.4	Derivation of the training algorithm	67
5.5	Discussion	69
5.5.1	Algorithm summary	69
5.5.2	Approximation: $\partial \mathbf{x} / \partial \mathbf{y} = 0$	70
5.5.3	Approximation: instantaneous weight update	70
5.5.4	The synapse mechanism	71
5.5.5	Eligibility profile	71
5.6	FOX and FBE	72
5.7	Implementation	74
5.7.1	Assumptions	74
5.7.2	Operation	74
5.7.3	Correcting inactive weights	76
5.7.4	Summary of the algorithm	79
5.8	More flexible error functions	82
5.8.1	Adding \mathbf{x} constraints (output limiting)	82
5.8.2	Adding \mathbf{x} derivative constraints	83
5.8.3	Adding extra \mathbf{y} constraints	83
5.9	Special training situations	84
5.9.1	Disturbances	84
5.9.2	Switched CMACs	84
5.9.3	Timer control	85
5.10	The continuous version	85
5.11	Related methods	86
5.12	Conclusion	86
6	Testing the FOX controller	87
6.1	Introduction	87
6.2	Design technique	87
6.2.1	Step 1: Feedback control	87
6.2.2	Step 2: Select error function	88
6.2.3	Step 3: Measure eligibility profiles	89
6.2.4	Step 4: Synthesize A, B, C	89
6.2.5	Step 5: Select σ	89

6.2.6	Step 6: Select the CMAC inputs	90
6.2.7	Step 7: Testing	90
6.3	System approximation requirements	90
6.4	Testing: Simulation	91
6.4.1	Standard error function with the fourth order model	92
6.4.2	Reduced order models	95
6.4.3	Training methods	97
6.4.4	Eligibility profile accuracy	99
6.4.5	History buffer size selection	100
6.4.6	CMAC input configuration	100
6.4.7	Concluding notes	106
6.5	Testing: Gantry crane	106
6.5.1	Eligibility profiles	108
6.5.2	Experiments	110
6.5.3	Concluding notes	112
6.6	Testing: Inverted pendulum	114
6.6.1	Finding the eligibility profile	115
6.6.2	Learning feedback control experiment	117
6.6.3	Learning feed-forward control experiment	117
6.6.4	Concluding notes	118
6.7	Testing: Mobile robot	120
6.8	Conclusions	124
7	Autonomous Robot Motion Control	127
7.1	Introduction	127
7.2	Specifying behavior	127
7.2.1	Hard-wired and generic controllers	127
7.2.2	Generic controllers using scalar error signals	128
7.2.3	Analytical motion	129
7.3	General design principles	129
7.4	The simulation and virtual environment	129
7.5	Hopping robot	131
7.5.1	Introduction	131
7.5.2	The basic controller	131
7.5.3	The learning controller	133
7.5.4	Results	133
7.6	System components	136
7.7	Making a biped robot walk	138
7.7.1	A short review of biped locomotion	138
7.7.2	The problem	139
7.7.3	Feed-forward controller structure	139
7.7.4	Walking performance	148
7.8	Conclusion	157
8	Future work and conclusions	159
8.1	Suggestions for future work	159
8.2	Conclusions	160

A	Definitions and basic concepts	163
A.1	Notation	163
A.2	Parameterized mappings and generalization	163
A.3	Some notes on training and generalization	164
A.4	Adaptive control	167
B	Chain-rule propagation algorithms	169
B.1	Introduction ¹	169
B.2	Derivation of FP and BP	170
B.2.1	Notation	170
B.2.2	Forward propagation algorithm (FP)	171
B.2.3	Backward propagation algorithm (BP)	171
B.3	Efficiency of FP and BP	172
B.3.1	Space and time requirements	172
B.3.2	Why is FP slower than BP?	173
B.4	Control system extension to the prototype	174
C	The multi-layer perceptron	177
D	Dynamic programming	181
D.1	The problem	181
D.2	The solution	182
D.3	A linear controller example	182
E	Eligibility profile cookbook	185
F	Trace precision	187
F.1	The problem	187
F.2	The scalar case	187
F.3	The matrix case	188
F.4	Trace buffer size	189
G	Eligibility order reduction	191
H	Software systems	193
H.1	Introduction	193
H.2	CyberSim	195
H.3	CyberView and DSVIEW	195
H.4	R3D	195
H.4.1	The multi-resolution Z-buffer algorithm	198
H.5	FoxN	198
H.6	RSDF	198
H.7	Druid	199

¹This appendix is derived from the author's paper in the Proceedings of the 1995 ANNES conference [114].

I	RoboDyn	201
I.1	Introduction	201
I.2	Links, joints and the AB tree	201
I.3	Widgets	202
I.4	RoboDyn configuration file	202
I.5	RoboDyn class hierarchy	203
J	Dyson	205
J.1	Introduction	205
J.2	dnd2exe	205
J.2.1	Introduction	205
J.2.2	Simulation external C functions	206
J.2.3	Command line arguments	206
J.3	Dynamic network description (DND) language	207
J.3.1	Introduction	207
J.3.2	Grammar (in pseudo-BNF)	207
J.3.3	Expressions	210
J.3.4	Statements	210
J.3.5	Modules	212
J.3.6	Expressions and variables	213
J.3.7	Groups	213
J.3.8	Constant folding	214
K	Real time control system	215
K.1	Introduction	215
K.2	Supervisor application	215
K.3	Kernel modification: mempool	216
K.4	Kernel modules	218
	Bibliography	220
	Last word	231

Nomenclature

Acronyms

CMAC	Cerebellar m odel a rticulation controller.
FBE	F eedback e rror.
FOX	F airly o bvious e xtension (to the CMAC).
IFC	I nternal f eedback controller.
WIB	W eight i ndex b uffer.

CMAC symbols (introduced in Chapter 3)

y_i	Inputs.
x_i	Outputs.
z_{ij}	Feature detecting neurons (2-layer example).
a_{ij}	Association neurons (2-layer example).
n_y	Number of inputs y_i .
n_x	Number of outputs x_i .
n_a	Number of association neurons activated per input, or the number of AU (association unit) tables.
n_v	Number of association neurons a_{ij} .
n_w	Total number of physical weights per output.
w_{ijk}	Weights (neural network model).
d_j^i	Displacement for AU i along input y_j .
q_i	Quantized version of input y_i .
p_j^i	Table index of AU i along axis y_j .
α	Learning rate.
e_i	Error signal for output x_i .
f_v	Virtual address generating function.
f_h	Hashing function.

n_p	Number of physical weights.
\min_i, \max_i	Minimum and maximum values for y_i .
res_i	Input resolution; the number of quantization steps for input y_i .
μ_i	Index into the weight tables for AU i .
$W_j[i]$	Weight i in the weight table for output x_j .

FBE symbols (introduced in Chapter 4)

$y(t)$	System state.
$x(t)$	Control force (controller output, system input).
$e(t)$	Error signal.
$y_d(t)$	Desired system state over time.
$F(y, x)$	Controlled system function.
A	Trainable controller.
Q	Fixed feedback controller.
x_a	Output of A .
x_q	Output of Q .
w	Weights which parameterize A .
p	Position of the simple 2nd order example system.
p_d	Desired value of p .

FOX symbols (introduced in Chapter 5 and Chapter 6)

$F(\mathbf{y}_i, \mathbf{x}_i)$	The system function, which takes the current state and control input and generates the next time step's state.
F^*	A linear approximation to F .
$C(\mathbf{y}_i)$	The "critic" function, which generates a scalar error value at each time step.
\mathbf{y}_i	The system state vector for time i . Size $n_y \times 1$.
\mathbf{x}_i	The system control input vector for time i (the output of the CMAC controller). Size $n_x \times 1$.
\mathbf{s}_i	A vector of CMAC association-unit "sensors" that encodes the current system state. Size $n_s \times 1$.
W	The matrix of CMAC weights. Note that $\mathbf{x}_i = W \mathbf{s}_i$. Size $n_x \times n_s$.
e_i	A scalar error value generated by the critic function for time i .
y_i^d	A scalar position reference (for time i) used in the calculation of e .

E	The total scalar error of the entire system.
w_{pq}	Element (p,q) of the matrix W .
ξ_i^{pq}	The eligibility vector for weight w_{pq} . Size $n_y \times 1$.
$C\xi$	The eligibility profile (scalar).
\hat{s}_i^{pq}	A synonym for dx_i/dw_{pq} . \hat{s}_i^{pq} is all zero except for its p 'th element which is equal to the q 'th element of s_i . Size $n_x \times 1$.
n_w	Total number of weights, $n_w = n_s n_x$.
n_a	Number of association units in the CMAC, i.e. the number of nonzero elements in s_i .
A	A system matrix for the linear system F^* . Size $n_y \times n_y$.
B	A system matrix for the linear system F^* . Size $n_y \times n_x$.
C	The matrix for the simple linear critic function. Size $1 \times n_y$.

FOX learning rate symbols

α	The main FOX learning rate (scalar).
β	The output limiting FOX learning rate (scalar).
γ	The output derivative limiting FOX learning rate (scalar).
α_1, α_2	The overshoot error function FOX learning rates (scalars).

Eligibility profile cookbook symbols

k_a, k_b	Parameters for the selection of over-damped second order eligibility profiles (see Appendix E).
t_{\max}	Parameter for selecting critically damped second order eligibility profiles (see Appendix E).

FOX algorithm symbols

σ	Cut-off (decay-to-zero) point for the eligibility profile.
δ	WIB buffer size, $\delta = \sigma + 2$
λ_i	The WIB buffer, an array of δ vectors each of size $1 \times n_y$.
T_t	The current time step.
g	The position of the current time step in the WIB.
T_0	The time step at λ_0 .
Λ	The accumulated value of: $e CA^i$ (size $1 \times n_y$).
Γ	The current value of: CA^g (size $1 \times n_y$).

Contents of the CDROM

This thesis is accompanied by a CDROM containing several movies in MPEG and AVI format. The subdirectory `thesis` on the CDROM also contains postscript files for this thesis (individual chapters as well as the entire document).

Inverted Pendulum Movie (AVI)

- `inverted.avi` : This shows various stages of the inverted pendulum experiment:
 - PD controller with a zero reference.
 - Trained FOX controller with a zero reference.
 - Trained FOX controller with a square wave reference.
 - PD controller with reference = $\sin(x) + \cos(2x)$.
 - Trained FOX controller with reference = $\sin(x) + \cos(2x)$.

Hopping Robot Movies (MPEG)

- `hopper1.mpg` : This shows four out-takes from the training of the hopping robot. Watch it fall over in various ways as it learns.
- `hopper2.mpg` : This shows the fully trained hopping robot following a path through its environment. It can go up and down a ramp without falling over.
- `hopper3.mpg` : This shows one of the hopping robot's failed attempts to climb the ramp. A foot trail is rendered, which can help diagnose the problem.

Biped Walking Movies (MPEG)

- `walk1.mpg` : Only the motion sequencer is used in the controller, no other controller modules are active. The biped makes stereotyped stepping movements and falls down immediately.
- `walk2.mpg` : The effect of hip side compensation is demonstrated. Before hip side compensation training the biped sways from side to side as it walks. After training the biped stays more upright.
- `walk3.mpg` : The combined effect of hip twist compensation and arm swing is demonstrated. Before these things are trained the biped twists from left to right as it walks. After training the biped is steadier.

- `walk4.mpg` : The effect of global side drift compensation is demonstrated. Before compensation training the biped becomes unstable when it leaves its desired path.
- `walk5.mpg` : A tripping event is demonstrated — the biped's foot touches the ground prematurely during a walking cycle.
- `walk6.mpg` : The effect of leg placement is demonstrated. Without it the biped makes no attempt to correct a falling motion. With correct leg placement the biped will place the feet to try and keep the body upright.
- `walk8.mpg` : Some out-takes from training are shown — the biped falls over in various ways as it learns to walk.
- `walkt1.mpg`, `walkt2.mpg`, `walkt3.mpg` : Successful walking — walking with a steady gait along flat ground, walking at various speeds, changing direction, and walking up and down slopes. Three copies of the movie are provided which use different MPEG bit rates (there are differences in movie quality).
- `explode.mpg` : This shows what can happen when the simulation goes astray: an incorrectly configured joint controller produces a large force that sends the robot flying into the air.