

Cambridge University Press
0521843219 - The Neuron Book
Ted Carnevale and Michael Hines
Frontmatter
[More information](#)

The NEURON Book

CARNEVALE and HINES

The authoritative reference on NEURON, the simulation environment for modeling biological neurons and neural networks that enjoys wide use in the experimental and computational neuroscience communities. This book will show you how to use NEURON to construct and apply empirically based models. Written primarily for neuroscience investigators, teachers, and students, it assumes no previous knowledge of computer programming or numerical methods. Readers with a background in the physical sciences or mathematics, who have some knowledge about brain cells and circuits and are interested in computational modeling, will also find it helpful. The NEURON Book covers material that ranges from the inner workings of this program to practical considerations involved in specifying the anatomical and biophysical properties that are to be represented in models. It uses a problem-solving approach, with many working examples that readers can try for themselves.

Nicholas T. Carnevale is a Senior Research Scientist in the Department of Psychology at Yale University. He directs the NEURON courses at the annual meetings of the Society for Neuroscience, and the NEURON Summer Courses at the University of California, San Diego, and University of Minnesota, Minneapolis.

Michael L. Hines is a Research Scientist in the Department of Computer Science at Yale University. He created NEURON in collaboration with John W. Moore at Duke University, Durham NC, and is the principal investigator and chief software architect on the project that continues to support and extend it.

Cambridge University Press
0521843219 - The Neuron Book
Ted Carnevale and Michael Hines
Frontmatter
[More information](#)

The NEURON Book

TED CARNEVALE
Department of Psychology
Yale University, New Haven, CT, USA
ted.carnevale@yale.edu

MICHAEL HINES
Department of Computer Science
Yale University, New Haven, CT, USA
michael.hines@yale.edu



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE, NEW YORK, MELBOURNE,
MADRID, CAPE TOWN, SINGAPORE,
SÃO PAULO

Cambridge University Press
0521843219 - The Neuron Book
Ted Carnevale and Michael Hines
Frontmatter
[More information](#)

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK
www.cambridge.org

Information on this title: www.cambridge.org/9780521843218
© Cambridge University Press 2005

This book is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2006
Printed in the United Kingdom at the University Press, Cambridge

*A catalog record for this book is available from the British Library
Library of Congress Cataloging in Publication data
ISBN HB 0 521 84321 9 ISBN PB ISBN OT*

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party Internet web sites referred to in this book, and does not guarantee that any content on such web sites is, or will remain, accurate or appropriate.

Table of contents

It is some systematized exhibition of the whale in his broad genera, that I would now fain put before you. Yet is it no easy task. The classification of the constituents of a chaos, nothing less is here essayed.

<i>Preface</i>	<i>xvii</i>
<i>Acknowledgments</i>	<i>xix</i>
1 A tour of the NEURON simulation environment	1
1.1 Modeling and understanding	1
1.2 Introducing NEURON	1
1.3 State the question	2
1.4 Formulate a conceptual model	3
1.5 Implement the model in NEURON	5
1.5.1 Starting and stopping NEURON	6
1.5.2 Bringing up a CellBuilder	6
1.5.3 Entering the specifications of the model cell	8
1.5.3.1 Topology	8
1.5.3.2 Subsets	10
1.5.3.3 Geometry	12
1.5.3.4 Biophysics	13
1.5.4 Saving the model cell	16
1.5.5 Executing the model specification	16
1.6 Instrument the model	18
1.6.1 Signal sources	18
1.6.2 Signal monitors	20
1.7 Set up controls for running the simulation	21
1.8 Save model with instrumentation and run control	21
1.9 Run the simulation experiment	23
1.10 Analyze results	24
References	30
2 The modeling perspective	32
2.1 Why model?	32
2.2 From physical system to computational model	33
2.2.1 Conceptual model: a simplified representation of a physical system	33

2.2.2	Computational model: an accurate representation of a conceptual model	33
2.2.3	An example	34
3	Expressing conceptual models in mathematical terms	36
3.1	Chemical reactions	36
3.1.1	Flux and conservation in kinetic schemes	37
3.1.2	Stoichiometry, flux, and mole equivalents	39
3.1.3	Compartment size	41
3.1.3.1	Scale factors	43
3.2	Electrical circuits	44
3.3	Cables	50
	References	54
4	Essentials of numerical methods for neural modeling	55
4.1	Spatial and temporal error in discretized cable equations	56
4.1.1	Analytic solutions: continuous in time and space	56
4.1.2	Spatial discretization	57
4.1.3	Adding temporal discretization	60
4.2	Numerical integration methods	62
4.2.1	Forward Euler: simple, inaccurate and unstable	62
4.2.1.1	Numerical instability	64
4.2.2	Backward Euler: inaccurate but stable	66
4.2.3	Crank–Nicholson: stable and more accurate	68
4.2.3.1	Efficient handling of nonlinearity	70
4.2.4	Adaptive integration: fast or accurate, occasionally both	72
4.2.4.1	Implementational considerations	73
4.2.4.2	The user’s perspective	75
4.2.4.3	Local variable time step method	80
4.2.4.4	Discrete event simulations	83
4.3	Error	83
4.4	Summary of NEURON’s integration methods	86
4.4.1	Fixed time step integrators	86
4.4.1.1	Default: backward Euler	86
4.4.1.2	Crank–Nicholson	86
4.4.2	Adaptive integrators	87

Table of contents

vii

	4.4.2.1	CVODE	88
	4.4.2.2	DASPK	88
References			88
5	Representing neurons with a digital computer		90
5.1	Discretization		90
5.2	How NEURON separates anatomy and biophysics from purely numerical issues		92
	5.2.1	Sections and section variables	92
	5.2.2	Range and range variables	93
	5.2.3	Segments	95
	5.2.4	Implications and applications of this strategy	96
		5.2.4.1 Spatial accuracy	96
		5.2.4.2 A practical test of spatial accuracy	97
5.3	How to specify model properties		98
	5.3.1	Which section do we mean?	98
		5.3.1.1 Dot notation	99
		5.3.1.2 Section stack	99
		5.3.1.3 Default section	100
5.4	How to set up model topology		101
	5.4.1	Loops of sections	101
	5.4.2	A section may have only one parent	101
	5.4.3	The root section	101
	5.4.4	Attach sections at 0 or 1 for accuracy	102
	5.4.5	Checking the tree structure with <code>topology()</code>	102
	5.4.6	Viewing topology with a Shape plot	103
5.5	How to specify geometry		103
	5.5.1	Stylized specification	104
	5.5.2	3-D specification	105
	5.5.3	Avoiding artifacts	107
		5.5.3.1 Beware of zero diameter	107
		5.5.3.2 Stylized specification may be reinterpreted as 3-D specification	108
5.6	How to specify biophysical properties		111
	5.6.1	Distributed mechanisms	111
	5.6.2	Point processes	112
	5.6.3	User-defined mechanisms	113
	5.6.4	Working with range variables	114
		5.6.4.1 Iterating over nodes	114

	5.6.4.2 Linear taper	115
	5.6.4.3 How changing nseg affects range variables	115
5.7	Choosing a spatial grid	117
	5.7.1 A consideration of intent and judgment	118
	5.7.2 Discretization guidelines	121
	5.7.2.1 The d_lambda rule	122
References		126
6	How to build and use models of individual cells	128
6.1	Graphical user interface vs. hoc code: which to use, and when?	128
6.2	Hidden secrets of the GUI	129
6.3	Implementing a model with hoc	130
	6.3.1 Topology	130
	6.3.2 Geometry	132
	6.3.3 Biophysics	133
	6.3.4 Testing the model implementation	133
	6.3.5 An aside: how does our model implementation in hoc compare with the output of the CellBuilder?	135
6.4	Instrumenting a model with hoc	139
6.5	Setting up simulation control with hoc	139
	6.5.1 Testing simulation control	141
6.6	Evaluating and using the model	141
6.7	Combining hoc and the GUI	141
	6.7.1 No NEURON Main Menu toolbar?	142
	6.7.2 Default section? We ain't got no default section!	142
	6.7.3 Strange Shapes?	144
	6.7.3.1 The barbed wire model	144
	6.7.3.2 The case of the disappearing section	148
	6.7.4 Graphs don't work?	151
	6.7.5 Conflicts between hoc code and GUI tools	152
6.8	Elementary project management	154
	6.8.1 Iterative program development	155
Reference		156
7	How to control simulations	157
7.1	Simulation control with the graphical user interface	157
7.2	The standard run system	159

Table of contents

ix

7.2.1	An outline of the standard run system	160
7.2.1.1	<code>fadvance()</code>	160
7.2.1.2	<code>advance()</code>	161
7.2.1.3	<code>step()</code>	161
7.2.1.4	<code>steprun()</code> and <code>continuerun()</code>	162
7.2.1.5	<code>run()</code>	163
7.3	Details of <code>fadvance()</code>	164
7.3.1	The fixed step methods: backward Euler and Crank–Nicholson	165
7.3.2	Adaptive integrators	171
7.3.2.1	Local time step integration with discrete events	173
7.3.2.2	Global time step integration with discrete events	179
7.4	Incorporating Graphs and new objects into the plotting system	179
	References	181
8	How to initialize simulations	183
8.1	State variables and <code>STATE</code> variables	183
8.2	Basic initialization in NEURON: <code>finitialize()</code>	185
8.3	Default initialization in the standard run system: <code>stdinit()</code> and <code>init()</code>	187
8.3.1	<code>INITIAL</code> blocks in NMODL	188
8.3.1.1	Default vs. explicit initialization of <code>STATES</code>	190
8.3.1.2	Ion concentrations and equilibrium potentials	190
8.4	Examples of custom initializations	195
8.4.1	Initializing to a particular resting potential	195
8.4.2	Initializing to steady state	197
8.4.3	Initializing to a desired state	198
8.4.4	Initializing by changing model parameters	199
8.4.4.1	Details of the mechanism	200
8.4.4.2	Initializing the mechanism	202
	Reference	206
9	How to expand NEURON’s library of mechanisms	207
9.1	Overview of NMODL	207
9.2	Example 9.1: A passive “leak” current	208
9.2.1	The NEURON block	210

9.2.2	Variable declaration blocks	211
9.2.2.1	The <code>PARAMETER</code> block	212
9.2.2.2	The <code>ASSIGNED</code> block	212
9.2.3	Equation definition blocks	213
9.2.3.1	The <code>BREAKPOINT</code> block	213
9.2.4	Usage	214
9.3	Example 9.2: A localized shunt	214
9.3.1	The <code>NEURON</code> block	215
9.3.2	Variable declaration blocks	215
9.3.3	Equation definition blocks	216
9.3.3.1	The <code>BREAKPOINT</code> block	216
9.3.4	Usage	217
9.4	Example 9.3: An intracellular stimulating electrode	217
9.4.1	The <code>NEURON</code> block	218
9.4.2	Equation definition blocks	218
9.4.2.1	The <code>BREAKPOINT</code> block	218
9.4.2.2	The <code>INITIAL</code> block	219
9.4.3	Usage	219
9.5	Example 9.4: A voltage-gated current	220
9.5.1	The <code>NEURON</code> block	222
9.5.2	The <code>UNITS</code> block	222
9.5.3	Variable declaration blocks	222
9.5.3.1	The <code>ASSIGNED</code> block	222
9.5.3.2	The <code>STATE</code> block	223
9.5.4	Equation definition blocks	223
9.5.4.1	The <code>BREAKPOINT</code> block	223
9.5.4.2	The <code>INITIAL</code> block	224
9.5.4.3	The <code>DERIVATIVE</code> block	225
9.5.4.4	The <code>FUNCTION</code> block	226
9.5.5	Usage	227
9.6	Example 9.5: A calcium-activated, voltage-gated current	228
9.6.1	The <code>NEURON</code> block	230
9.6.2	The <code>UNITS</code> block	231
9.6.3	Variable declaration blocks	231
9.6.3.1	The <code>ASSIGNED</code> block	231
9.6.3.2	The <code>STATE</code> block	232
9.6.4	Equation definition blocks	232
9.6.4.1	The <code>BREAKPOINT</code> block	232

Table of contents

xi

9.6.4.2	The DERIVATIVE block	232
9.6.4.3	The FUNCTION and PROCEDURE blocks	232
9.6.5	Usage	233
9.7	Example 9.6: Extracellular potassium accumulation	233
9.7.1	The NEURON block	235
9.7.2	Variable declaration blocks	236
9.7.2.1	The PARAMETER block	236
9.7.2.2	The STATE block	236
9.7.3	Equation definition blocks	236
9.7.3.1	The BREAKPOINT block	236
9.7.3.2	The INITIAL block	236
9.7.3.3	The DERIVATIVE block	237
9.7.4	Usage	237
9.8	General comments about kinetic schemes	238
9.9	Example 9.7: Kinetic scheme for a voltage-gated current	240
9.9.1	The NEURON block	242
9.9.2	Variable declaration blocks	242
9.9.2.1	The STATE block	242
9.9.3	Equation definition blocks	243
9.9.3.1	The BREAKPOINT block	243
9.9.3.2	The INITIAL block	243
9.9.3.3	The KINETIC block	243
9.9.3.4	The FUNCTION_TABLES	244
9.9.4	Usage	245
9.10	Example 9.8: Calcium diffusion with buffering	245
9.10.1	Modeling diffusion with kinetic schemes	246
9.10.2	The NEURON block	250
9.10.3	The UNITS block	250
9.10.4	Variable declaration blocks	250
9.10.4.1	The ASSIGNED block	250
9.10.4.2	The STATE block	250
9.10.4.3	LOCAL variables declared outside of equation definition blocks	251
9.10.5	Equation definition blocks	251
9.10.5.1	The INITIAL block	251
9.10.5.2	PROCEDURE factors ()	252

	9.10.5.3 The KINETIC block	252
	9.10.6 Usage	254
9.11	Example 9.9: A calcium pump	255
	9.11.1 The NEURON block	255
	9.11.2 The UNITS block	256
	9.11.3 Variable declaration blocks	256
	9.11.3.1 The PARAMETER block	256
	9.11.3.2 The ASSIGNED block	257
	9.11.3.3 The CONSTANT block	257
	9.11.3.4 The STATE block	257
	9.11.4 Equation definition blocks	257
	9.11.4.1 The BREAKPOINT block	257
	9.11.4.2 The INITIAL block	258
	9.11.4.3 The KINETIC block	259
	9.11.5 Usage	260
9.12	Models with discontinuities	260
	9.12.1 Discontinuities in PARAMETERS and ASSIGNED variables	260
	9.12.2 Discontinuities in STATES	261
	9.12.3 Event handlers	263
9.13	Time-dependent PARAMETER changes	263
	References	264
10	Synaptic transmission and artificial spiking cells	265
10.1	Modeling communication between cells	266
	10.1.1 Example 10.1: Graded synaptic transmission	266
	10.1.1.1 The NEURON block	268
	10.1.1.2 The BREAKPOINT block	269
	10.1.1.3 Usage	269
	10.1.2 Example 10.2: A gap junction	271
	10.1.2.1 Usage	272
	10.1.3 Modeling spike-triggered synaptic transmission: an event-based strategy	272
	10.1.3.1 Conceptual model	273
	10.1.3.2 The NetCon class	274
	10.1.4 Example 10.3: Synapse with exponential decay	277
	10.1.4.1 The BREAKPOINT block	278
	10.1.4.2 The DERIVATIVE block	278
	10.1.4.3 The NET_RECEIVE block	278
	10.1.4.4 Usage	278

Table of contents

xiii

10.1.5	Example 10.4: Alpha function synapse	280
10.1.6	Example 10.5: Use-dependent synaptic plasticity	281
10.1.6.1	The NET_RECEIVE block	283
10.1.7	Example 10.6: Saturating synapses	284
10.1.7.1	The PARAMETER block	287
10.1.7.2	The STATE block	287
10.1.7.3	The INITIAL block	287
10.1.7.4	The BREAKPOINT and DERIVATIVE blocks	288
10.1.7.5	The NET_RECEIVE block	288
10.2	Artificial spiking cells	289
10.2.1	Example 10.7: IntFire1, a basic integrate and fire model	290
10.2.1.1	The NEURON block	291
10.2.1.2	The NET_RECEIVE block	292
10.2.1.3	Enhancements to the basic mechanism	292
10.2.2	Example 10.8: IntFire2, firing rate proportional to input	297
10.2.2.1	Implementation in NMODL	298
10.2.3	Example 10.9: IntFire4, different synaptic time constants	301
10.2.4	Other comments regarding artificial spiking cells	304
References		305
11	Modeling networks	306
11.1	Building a simple network with the GUI	307
11.2	Conceptual model	308
11.3	Adding a new artificial spiking cell to NEURON	309
11.4	Creating a prototype net with the GUI	311
11.4.1	Define the types of cells	311
11.4.2	Create each cell in the network	312
11.4.3	Connect the cells	315
11.4.3.1	Setting up network architecture	315
11.4.3.2	Specifying delays and weights	316
11.4.4	Set up instrumentation	318
11.4.5	Set up controls for running simulations	319
11.4.6	Run a simulation	322

11.4.7	Caveats and other comments	322
11.4.7.1	Changing the properties of an existing network	322
11.4.7.2	A word about cell names	323
11.5	Combining the GUI and programming	324
11.5.1	Creating a hoc file from the NetWork Builder	324
11.5.1.1	NetGUI default section	326
11.5.1.2	Network cell templates	326
11.5.1.3	Network specification interface	327
11.5.1.4	Network instantiation	328
11.5.2	Exploiting the reusable code	328
	References	341
12	hoc, NEURON's interpreter	343
12.1	The interpreter	344
12.2	Adding new mechanisms to the interpreter	345
12.3	The stand-alone interpreter	346
12.3.1	Starting and exiting the interpreter	346
12.3.2	Error handling	348
12.4	Syntax	350
12.4.1	Names	350
12.4.2	Keywords	350
12.4.3	Variables	353
12.4.4	Expressions	354
12.4.5	Statements	355
12.4.6	Comments	355
12.4.7	Flow control	356
12.4.8	Functions and procedures	357
12.4.8.1	Arguments	358
12.4.8.2	Call by reference vs. call by value	359
12.4.8.3	Local variables	360
12.4.8.4	Recursive functions	360
12.4.9	Input and output	361
12.4.10	Editing	362
	Reference	362
13	Object-oriented programming	363
13.1	Object vs. class	363
13.2	The object model in hoc	364
13.3	Objects and object references	364
13.3.1	Declaring an object reference	364

Table of contents

xv

13.3.2	Creating and destroying an object	365
13.3.3	Using an object reference	366
13.3.3.1	Passing obj refs (and objects) to functions	366
13.3.4	Defining an object class	367
13.3.4.1	Direct commands	368
13.3.4.2	Initializing variables in an object	368
13.3.4.3	Keyword names	369
13.3.5	Object references vs. object names	370
13.3.5.1	An example of the didactic use of object names	371
13.4	Using objects to solve programming problems	372
13.4.1	Dealing with collections or sets	372
13.4.1.1	Array of objects	372
13.4.1.2	List of objects	373
13.4.2	Encapsulating code	375
13.5	Polymorphism and inheritance	376
Reference		377
14	How to modify NEURON itself	378
14.1	A word about graphics terminology	378
14.2	Graphical interface programming	378
14.2.1	General issues	380
14.2.1.1	A pattern for defining a GUI tool template	381
14.2.1.2	Enclosing the GUI tool in a single window	383
14.2.1.3	Saving the window to a session	385
14.2.2	Tool-specific development	389
14.2.2.1	Plotting	389
14.2.2.2	Handling events	392
14.2.2.3	Finishing up	395
Appendix A1	Mathematical analysis of IntFire4	399
A1.1	Proof that the estimate is never later than the true firing time	401
A1.1.1	Part 1: If $m'_0 \leq 0$, then $m(t)$ remains < 1	402
A1.1.2	Part 2: If $m' > 0$, $(1 - m)/m'$ underestimates the firing time	404

Appendix A2	NEURON's built-in editor	406
A2.1	Starting and stopping	407
	A2.1.1 Switching from hoc to emacs	407
	A2.1.2 Returning from emacs to hoc	407
	A2.1.3 Killing the current command	407
A2.2	Moving the cursor	407
A2.3	Modes	408
A2.4	Deleting and inserting	408
A2.5	Blocks of text: marking, cutting, and pasting	408
A2.6	Searching and replacing	409
A2.7	Text formatting and other tricks	409
A2.8	Buffers and file I/O	409
A2.9	Windows	410
A2.10	Macros and repeating commands	411
References		411
<i>Epilogue</i>		412
<i>Index</i>		413

Preface

I promise nothing complete; because any human thing supposed to be complete, must for that very reason infallibly be faulty.

Who should read this book?

This book is about how to use the NEURON simulation environment to construct and apply empirically based models of neurons and neural networks. It is written primarily for neuroscience investigators, teachers, and students, but readers with a background in the physical sciences or mathematics who have some knowledge about brain cells and circuits and are interested in computational modeling will also find it helpful. The emphasis is on the most productive use of NEURON as a means for testing hypotheses that are founded on experimental observations, and for exploring ideas that may lead to the design of new experiments. Therefore the book uses a problem-solving approach, with many working examples that readers can try for themselves.

What this book is, and is not, about

Formulating a *conceptual model* is an attempt to capture the essential features that underlie some particular function. This necessarily involves simplification and abstraction of real-world complexities. Even so, one may not necessarily understand all implications of the conceptual model. To evaluate a conceptual model it is often necessary to devise a hypothesis or test in which the behavior of the model is compared against a prediction. *Computational models* are useful for performing such tests. The conceptual model and the hypothesis should determine what is included in a computational model and what is left out. This book is not about how to come up with conceptual models or hypotheses, but instead focuses on how to use NEURON to create and use computational models as a means for evaluating conceptual models.

What to read, and why?

Chapter 1 conveys a basic idea of NEURON's primary domain of application by guiding the reader through the construction and use of a model neuron. This exercise is based entirely on NEURON's graphical user interface (GUI), and requires no programming ability or prior experience with NEURON whatsoever.

Chapter 2 considers the role of computational modeling in neuroscience research from a general perspective. Chapters 3 and 4 focus on aspects of applied mathematics and numerical methods that are particularly relevant to computational neuroscience. Chapter 5 discusses the concepts and strategies that are used in NEURON to simplify the task of representing neurons, which (at least at the level of synapses and cells) are distributed and continuous in space and time, in a digital computer, where neither time nor numeric values are continuous. Chapter 6 returns to the topic of model construction, emphasizing the use of programming.

Chapters 7 and 8 provide “inside information” about NEURON’s standard run and initialization systems, so that readers can make best use of their features and customize them to meet special modeling needs. Chapter 9 shows how to use the NMODL programming language to add new biophysical mechanisms to NEURON. This theme continues in Chapter 10, which starts with mechanisms of communication between cells (gap junctions, graded and spike-triggered synaptic transmission), and moves on to models of artificial spiking neurons (e.g. integrate and fire cells). The first half of Chapter 11 is a tutorial on NEURON’s GUI tools for creating simple network models, and the second half shows how to use the combined strength of the GUI and hoc programming to create more complex networks.

Chapter 12 discusses the elementary features of the hoc programming language itself. Chapter 13 describes the object-oriented extensions that have been added to hoc. These extensions have greatly facilitated construction of NEURON’s GUI tools, and they can also be very helpful in many other complex programming tasks such as creating and managing network models. Chapter 14 presents an example of how to use object-oriented programming to increase the functionality of NEURON.

Appendix 1 presents a mathematical analysis of the IntFire4 artificial spiking cell mechanism, proving a result that is used to achieve computational efficiency. Appendix 2 summarizes the commands for NEURON’s built-in text editor.

Typeface conventions

This book uses special typefaces to distinguish two different sets of keywords – code and GUI – from each other and from words with the same spelling that are not being used as keywords. Ordinary program code is printed with a *courier* typeface. Optional code, or items that are generic placeholders that the reader should substitute with his or her own specific entries, are indicated by *slanted courier* (not really italic, is it?). Samples of command-line usage employ **courier** to signify user input. GUI keywords, which are the labels that appear in NEURON’s graphical interface, are presented with a sans-serif typeface.

Acknowledgments

First and foremost, we want to thank our mentor and colleague John W. Moore for his vision, support, encouragement, and active participation in the development of NEURON, without which neither it nor this book would exist. Through his research and teaching, he was introducing students to “computational neuroscience” long before that glorious term was invented. NEURON had its beginnings in John’s laboratory at Duke University almost three decades ago, when he and one of the authors (MLH) started their collaboration to develop simulation software for neuroscience research. Users of NEURON on the Macintosh owe John a particular debt. He continues to participate in the development and dissemination of NEURON, concentrating most recently on educational applications in collaboration with Ann Stuart (Moore and Stuart 2004).

The list of others who have added in one way or another to the development of NEURON is far too long for this short preface. Zach Mainen, Alain Destexhe, Bill Lytton, Terry Sejnowski, and Gordon Shepherd deserve special mention for many contributions, both direct and indirect, that range from specific enhancements to the program, to fostering the wider acceptance of computational approaches in general, and NEURON in particular, by the neuroscience community at large. We also thank the countless NEURON users whose questions and suggestions continue to help guide the evolution of this software and its documentation. The development of NEURON and this book has been made possible by support from the National Institutes of Health and the National Science Foundation. We are sure that our readers will recognize the epigrams from Herman Melville’s *Moby Dick* that are scattered throughout this book, as well as the (mis)quotation from *The Treasure of the Sierra Madre* (book by B. Traven, screenplay by John Huston). We hope that everyone else will forgive any omission and remind us, gently, in time for the second edition.

Finally, we thank our wives and children for their encouragement and patience while we completed this book.

REFERENCE

Moore, J.W. and Stuart, A.E. *Neurons in Action: Computer Simulations with NeuroLab*. Sunderland, MA: Sinauer Associates, 2004.

N.T. Carnevale and M.L. Hines
Yale University, New Haven, CT, USA