

Index

Symbols

- !(logical NOT operator) – see hoc syntax: expressions: operators
- != (inequality operator) – see hoc syntax: expressions: operators
- \$ – see funcs and procs: arguments: positional syntax
- \$1 – see funcs and procs: arguments: positional syntax
- \$i – see funcs and procs: arguments: symbolic positional syntax
- \$o – see funcs and procs: arguments: positional syntax
- \$s – see funcs and procs: arguments: positional syntax
- % (remainder operator) – see hoc syntax: expressions: operators
- & (pointer operator) – see hoc syntax: pointer operator
- && (logical AND operator) – see hoc syntax: expressions: operators
- ' (apostrophe) – see DERIVATIVE block: ' (apostrophe)
- () (paired left and right parentheses) – see hoc syntax: expressions: operators
- (1) – see units: dimensionless
- * (multiplication operator) – see hoc syntax: expressions: operators
- /* – see hoc syntax: comments
- + (addition operator) – see hoc syntax: expressions: operators
- (subtraction operator) – see hoc syntax: expressions: operators
- (unaryminus operator) – see hoc syntax: expressions: operators
- > (sink reaction indicator) – see KINETIC block: –> (sink reaction indicator)
- / (division operator) – see hoc syntax: expressions: operators
- /* – see hoc syntax: comments
- // – see hoc syntax: comments
- :
- : (colon) – see NMODL: comments, range variable: linear taper
- < (less than operator) – see hoc syntax: expressions: operators
- <-> (reaction indicator) – see KINETIC block: <-> (reaction indicator)
- << (explicit flux) – see KINETIC block: << (explicit flux)
- <= (less than or equal to operator) – see hoc syntax: expressions: operators
- = (assignment operator) – see hoc syntax: expressions: operators
- == (equality operator) – see hoc syntax: expressions: operators
- > (greater than operator) – see hoc syntax: expressions: operators
- >= (greater than or equal to operator) – see hoc syntax: expressions: operators
- [] (index operator) – see hoc syntax: variables: double, Vector class, object: array, section: array, NMODL: arrays, STATE variable: array in NMODL
- \ (backslash) – see hoc syntax: continuation character
- ^ (exponentiation operator) – see hoc syntax: expressions: operators
- ^C (Control C) – see hoc: interrupting execution
- ^D (Control D) – see hoc: starting and exiting
- _ (underscore) – see hoc syntax: names

`_ion` – see `ion` mechanism: `_ion` suffix
`x_ion` – see `ion` mechanism:
 automatically created
`xi0_x_ion` – see `ion` mechanism: default
 concentration: specification in `hoc`
`xo0_x_ion` – see `ion` mechanism: default
 concentration: specification in `hoc`
`{ }` – see `hoc` syntax: statements: compound
 statement
`||` (logical OR operator) – see `hoc` syntax:
 expressions: operators
`~` (tilde) – see `KINETIC` block: `~` (tilde)

3

3 – see `nseg`: why triple `nseg`?
 3-D model – see model: 3-D, 3-D
 specification of geometry
 3-D specification of geometry 103, 105, 144
 3-D information 105, 109
 `arc3d()` 109
 calculation of `L`, `diam`, `area()`, and `ri()`
 105, 108
 coordinates
 absolute vs. relative 146–147
 data – see 3-D specification of geometry:
 3-D information
 also see quantitative morphometric data
 `diam3d()` 105
 checking 108
 diameter 105
 problems 108
 also see 3-D specification of geometry:
 calculation of `L`, `diam`, `area()`,
 and `ri()`
 `n3d()` 105, 109
 number of 3-D points
 effect on computational efficiency 106
 vs. `nseg` 106
 also see `n3d()`
 origin of cell – see section: root section: is
 3-D origin of cell
 points – see 3-D specification of geometry:
 3-D information
 `pt3dadd()` 105
 also see stylized specification of geometry

A

abrupt change – see variable: abrupt change
 of, `NET_RECEIVE` block: handling
 abrupt changes and discontinuities
 absolute error 63, 71–72
 local 75

tolerance 75, 79, 251
 also see `STATE` block: specifying local
 absolute error tolerance
 also see relative error, numerical error
 absolute tolerance—see absolute error: local:
 tolerance
 abstraction 33–34
 AC length constant – see length constant: AC
 access 100, 143
 also see section: currently accessed: default
 section
 accumulation – see `ion` accumulation
 accuracy 33, 64, 91
 effect of boundary conditions 54
 of ionic currents – see membrane current:
 ionic: accuracy, standard run system:
 `fcurrent()`
 order – see numeric integration: order of
 accuracy
 physiological 79, 85
 quantitative 67
 spatial – see spatial accuracy
 vs. speed 73, 117
 temporal – see temporal accuracy
 also see judgment, discretization, numerical
 error, parameters: sensitivity to
`acell_home_` – see `NetWork Builder`:
 exporting reusable code: `acell_home_`
 active current – see channel: voltage-gated
 active transport 41, 199
 electrically silent 256
 initialization 203–205
 pump transient 204
 kinetic scheme 200
 also see Example 9.9: a calcium pump
 pump current 255–258
 countering with a `NONSPECIFIC_`
 `CURRENT` 256
 initialization 258
 also see buffer, diffusion, `ion` accumulation
 adaptive integration – see numeric integration:
 adaptive
 adaptive time step – see numeric integration:
 adaptive
 adding items to `NEURON` Main Menu – see
 `NEURONMainMenu` class:
 `miscellaneous_add()`
 adding new functions to `NEURON` – see
 `NMODL`
 adding new mechanisms – see mechanisms:
 user-defined, `nrniv`: adding new
 mechanisms, `NMODL`

- addition operator – see hoc syntax:
 - expressions: operators
- addplot () – see standard run system:
 - addplot ()
- advance microstep – see numeric integration:
 - adaptive: advance microstep
 - also see standard run system: fadvance ()
 - local time step integration
- advance () – see standard run system:
 - advance ()
- afferent event – see event: external
- aggregation of events to time step boundaries
 - see numeric integration: fixed time step: event aggregation
- algebra
 - linear – see linear algebra
- algebraic equation – see equation: algebraic
- algebraic difference equation – see equation:
 - difference
- AlphaSynapse 19
 - graphical interface – see
 - PointProcessManager GUI
 - parameters 20
 - preserving spatial accuracy – see point
 - process: preserving spatial accuracy
 - also see Example 10.4: alpha function synapse
- alpha function synapse – see AlphaSynapse,
 - Example 10.4: alpha function synapse
- Alt key – see Graph class: menu_tool ()
- amount of material – see material: amount
- ampa . mod – see Example 10.6: saturating synapses
- AMPAergic synapse – see Example 10.3:
 - synapse with exponential decay, Example 10.6: saturating synapses
- amplification factor – see amplifier: gain
- amplifier 45
 - feedback – see feedback: amplifier
 - gain 45, 49
 - headstage 49
 - also see circuit: element: amplifier
- analysis 24
 - initialization – see initialization: analysis
- analytic solution
 - and discrete event simulation – see discrete
 - event simulation: conditions for
 - cable equation – see equation: cable:
 - analytic solution
- anatomical data – see quantitative
 - morphometric data
- anatomical distance – see distance
- anatomical properties
 - separating biology from numerical issues 92
 - also see section, range, range variable
 - specifying – see 3-D specification of
 - geometry, stylized specification of
 - geometry, topology: specifying
 - also see geometry, topology
- AND operator – see hoc syntax: expressions:
 - operators
- apostrophe – see DERIVATIVE block:
 - ' (apostrophe)
- approximate Jacobian – see Jacobian:
 - approximate
- approximation 25, 34, 52–53
 - of a continuous system by a discrete system
 - 60, 90
 - piecewise linear – see continuous variable:
 - piecewise linear approximation
 - also see Jacobian: approximate
- arc length – see range
- arc3d () – see 3-D specification of
 - geometry: arc3d ()
- architecture – see topology
- area
 - membrane – see surface area, membrane area
 - surface – see surface area, membrane area
 - zero area nodes – see section: nodes: zero area
- area
 - as an ASSIGNED variable – see ASSIGNED
 - variable: v, celsius, t, dt, diam, and area
 - also see area ()
- area () 104
 - calculation of – see stylized specification of
 - geometry: calculation of area () and ri (), 3-D specification of geometry: calculation of L, diam, area (), and ri ()
 - effect of creating a Shape – see Shape
 - object: creating: effect on diam, area (), and ri ()
 - effect of define_shape () – see
 - define_shape (): effect on diam, area (), and ri ()
 - stylized vs. 3-D surface integral 110
- arguments – see funcs and procs: arguments
- section as – see SectionRef class
- trapping – see good programming style:
 - bulletproofing against nonsense arguments
- also see FUNCTION block, PROCEDURE block, NET_RECEIVE block

- arithmetic operators – see hoc syntax:
 - expressions: operators
- arrays
 - in NMODL – see NMODL: arrays, STATE
 - variable: array in NMODL
 - of numbers – see hoc syntax: variables:
 - double, Vector class
 - of objects – see object: array
 - of sections – see section: array
- ArtCellGUI**
 - bringing up an ArtCellGUI 311
 - cell names – see NetWork Builder: cells:
 - names
 - changing – see ArtCellGUI: specifying
 - cell types
 - changing cell parameters – see NetWork Builder: caveats
 - also see PointProcessGroupManager
 - specifying cell types 311
- Artificial Cell – see ArtCellGUI
- artificial cell – see artificial spiking cell
- artificial neuron – see artificial spiking cell
- artificial spiking cell 82–83, 289
 - advantages and uses 265
 - computational efficiency 290, 292
 - differences from other point processes 290–291
 - implemented as point processes 290
 - IntFire1 – see IntFire1 class,
 - Example 10.7: IntFire1, a basic integrate and fire model
 - IntFire2 – see IntFire2 class,
 - Example 10.8: IntFire2, firing rate proportional to input
 - IntFire4 – see IntFire4 class,
 - Example 10.9: IntFire4, different synaptic time constants
 - under CVODE 88
 - also see biophysical neuron model,
 - integrate and fire, IntFire1 class, IntFire2 class, IntFire4 class, NEURON block: ARTIFICIAL_CELL
- ARTIFICIAL_CELL – see NEURON block: ARTIFICIAL_CELL, artificial spiking cell
- ASCII file – see plain text file
- ASSIGNED block 212, 222, 231, 250
 - also see PARAMETER block, STATE block
- ASSIGNED variable 184, 195, 212
 - abrupt change of – see variable: abrupt
 - change of, NetCon class: event (), NET_RECEIVE block: handling
 - abrupt changes and discontinuities
- accuracy 169, 172
- GLOBAL
 - spatial variation 232
 - vs. RANGE 215, 230
 - also see NEURON block: GLOBAL
- initialization 187
 - also see initialization
- is a range variable by default 213
 - also see NEURON block: RANGE, range variable
- v, celsius, t, dt, diam, and area 213
- visibility at the hoc level 213, 218
- when to use for an equilibrium potential 223
- also see PARAMETER variable, STATE variable, state variable: as an ASSIGNED variable, ion_style ()
- assignment operator – see hoc syntax:
 - expressions: operators
- assumptions 1, 41–42, 46, 51, 53
- at_time () – see CVODE: and model
 - descriptions: at_time (), variable:
 - abrupt change of, NET_RECEIVE block: handling abrupt changes and discontinuities
- atol 79
 - also see absolute error: local: tolerance
- attaching a section – see connect
- attenuation
 - at high frequencies 122
 - also see membrane time constant: and
 - attenuation of fast signals, spatial decay of fast signals
- Avogadro's number – see units: mole
- axial current 51–54, 56
 - positive current convention 51
- axial resistance 52–53
 - infinite 107, 115
 - also see diameter: zero or narrow diameter
 - also see ri (), Ra, cytoplasmic resistivity
- B**
- b_flux – see KINETIC block: b_flux
- backward Euler method 61, 66–68, 165
 - and LONGITUDINAL_DIFFUSION 253
 - iteration coefficient 62
 - local error 67, 83
 - stability 67
 - summary 86
 - also see modified Euler method, numeric
 - integration: fixed time step
- backward flux – see flux: backward,
 - KINETIC block: b_flux

- backslash (\) – see hoc syntax: continuation character
 - balance
 - charge – see charge: conservation
 - mass – see material: conservation
 - ball and stick 34
 - bandpass 122
 - barbed wire shape – see stylized specification of geometry: strange shapes
 - base class – see class: base class, object-oriented programming: polymorphism, object-oriented programming: inheritance
 - biological properties vs. purely computational issues 92
 - biophysical mechanism – see distributed mechanism, point process
 - biophysical neuron model 82, 265
 - also see artificial spiking cell
 - biophysical properties
 - separating biology from numerical issues 92
 - also see section, range, range variable
 - specifying 111, 133
 - also see distributed mechanism, point process
 - blocks – see NMODL: named blocks
 - boundary conditions 51
 - effect on accuracy – see accuracy: effect of boundary conditions
 - sealed end 54, 56
 - branch
 - cell 92
 - also see neurite, section
 - circuit – see circuit: edge
 - branched architecture 51, 53, 91, 98
 - also see topology
 - branched cable – see cable: branched
 - break – see hoc syntax: flow control: break
 - BREAKPOINT block 213, 223
 - abrupt change of a variable – see variable: abrupt change of, NetCon class: event (), NET_RECEIVE block: handling abrupt changes and discontinuities
 - and computations that must be performed only once per time step 224
 - and counts, flags, and random numbers 223
 - and PROCEDURES 224
 - and rate functions 224
 - and variables that depend on the number of executions 223
 - at_time() – see CVODE: and model descriptions: at_time(), variable: abrupt change of, NET_RECEIVE block: handling abrupt changes and discontinuities
 - currents assigned at end of 223
 - METHOD – see BREAKPOINT block: SOLVE
 - SOLVE 168–170, 223, 225
 - cnexp 225
 - derivimplicit 226
 - is not a function call 224
 - sparse 87, 243
 - also see STATE variable
 - state_discontinuity() – see CVODE: and model descriptions: state_discontinuity(), variable: abrupt change of, NET_RECEIVE block: handling abrupt changes and discontinuities
 - time-dependent PARAMETER – see PARAMETER variable: time-dependent, Vector class: play() translation of 168, 170
 - browser – see directory browser, variable browser, Plot what? GUI
 - buffer 37
 - also see ion accumulation, diffusion, Example 9.8: calcium diffusion with buffering
 - built-in constants – see hoc syntax: variables: built-in constants
 - built-in editor – see em, emacs
 - bulletproofing against nonsense arguments – see good programming style: bulletproofing against nonsense arguments
- C**
- C code
 - embedding – see NMODL: VERBATIM . . . ENDVERBATIM
 - ca_ion – see ion mechanism: automatically created
 - cable 50
 - branched 53
 - equation – see equation: cable
 - passive cylindrical 56–58, 60
 - unbranched 50, 53–54, 92
 - CABLE 164
 - cadif.mod – see Example 9.8: calcium diffusion with buffering
 - cagk.mod – see Example 9.5: a calcium-activated, voltage-gated current
 - cai – see calcium, ion mechanism

- `cai0_ca_ion` – see `calcium, ion`
 - mechanism: default concentration:
 - specification in `hoc`
- `calcium`
 - amount of 41
 - buffering – see Example 9.8: `calcium`
 - diffusion with buffering
 - concentration 42
 - free 80
 - current 199
 - effect on concentration 200
 - diffusion – see Example 9.8: `calcium`
 - diffusion with buffering
 - pump 41, 79, 199
 - also see Example 9.9: a `calcium` pump,
 - active transport
 - also see `ion` mechanism
 - `calcium-activated current` – see Example 9.5: a
 - `calcium-activated, voltage-gated current`
- call by reference vs. call by value – see `funcs` and `procs`: arguments: call by reference vs. call by value, `NET_RECEIVE` block: arguments are call by reference, `FUNCTION` block: arguments are call by value, `PROCEDURE` block: arguments are call by value
 - also see `funcs` and `procs`: arguments:
 - pointer, `hoc` syntax: pointer operator
- `cao` – see `calcium, ion` mechanism
- `cao0_ca_ion` – see `calcium, ion`
 - mechanism: default concentration:
 - specification in `hoc`
- capacitance
 - electrode – see `electrode: capacitance`
 - compensation – see `electrode: capacitance: compensation`
 - membrane – see `cm, membrane capacitance, specific membrane capacitance`
- capacitive current – see `membrane current: capacitive`
- capacitor
 - feedback – see `feedback: capacitor`
 - also see `circuit: element: capacitor`
- Cell Map – see `NetWork Builder: cells: Cell Map`
- cell time queue – see `standard run system: event delivery system: cell time queue`
- CellBuilder 5
 - bringing up 6
 - `hoc` output
 - exported cell 135
 - root section 8, 21
 - also see `section: default section`
 - spatial grid – see `CellBuilder GUI: Geometry` page: `d_lambda`
- CellBuilder GUI
 - Biophysics page 13
 - assigning values 16
 - specifying strategy 15
 - Continuous Create 17, 152–153
 - Geometry page 12
 - assigning values 14
 - `d_lambda` 14
 - also see `spatial accuracy, spatial grid, d_lambda` rule
 - specifying strategy 13
 - Management page 16
 - Export 16, 135
 - Subsets page 8, 10
 - all subset 11
 - making a new subset 12
 - Topology page 8
 - base name 10
 - Basename 10
 - changing the name of a section 11
 - making a new section 9
- `celsius` 212–213, 231
 - as an `ASSIGNED` variable – see
 - `ASSIGNED` variable: `v, celsius, t, dt, diam, and area`
- central difference method – see `Crank–Nicholson` method
- changing a variable in mid-run – see
 - `PARAMETER` variable: time-dependent, variable: abrupt change of
- changing point process location with `hoc`
 - code – see `point process: loc()`
- changing standard functions and procedures – see `standard GUI library: redefining functions and procedures, standard run library: redefining functions and procedures`
- channel 53
 - analytic integration of states – see `numeric integration: analytic integration of channel states`
 - `calcium-activated` – see Example 9.5: a
 - `calcium-activated, voltage-gated current`
 - conductance 53
 - current
 - accuracy – see `membrane current: ionic: accuracy, standard run system: fcurrent()`
 - density 84, 92

- gating model 87
 - HH type 70, 225
 - also see channel: linear
 - nonlinear 226
 - under CVODE 88
- initialization – see initialization: channel model
- ligand-gated 36
 - also see Example 10.1: graded synaptic transmission, Example 10.5: use-dependent synaptic plasticity, Example 10.6: saturating synapses
- linear 87
 - also see channel: gating model: HH-type
- model 36
- nonlinear 87
 - under CVODE 88
- ohmic – see channel: linear
- voltage-gated 36
 - also see Example 9.4: a voltage-gated current, Example 9.5: a calcium-activated, voltage-gated current, Example 9.7: kinetic scheme for a voltage-gated current
- chaos and numerical error – see numerical error: chaotic system
- chaotic system – see initialization: categories: to a desired state
- charge 50
 - conservation 51–52
 - also see Kirchhoff's current law, gap junction: conservation of charge
 - electronic – see e: electronic charge vs. units conversion factor
- chemical
 - notation – see kinetic scheme
 - reaction – see kinetic scheme
 - signal – see signal: chemical
- child section – see section: child
- chord conductance – see conductance: chord
- circuit 44
 - analysis 44
 - branch 44
 - edge 44
 - element 45
 - amplifier 45
 - capacitor 45
 - current source 45
 - ground 45
 - resistor 45
 - voltage source 45
 - wire 45
 - equivalent 48, 53
 - also see section: equivalent circuit
 - linear – see linear circuit
 - node 44
 - parallel RC 45
 - positive current convention 45
- clamp – see current clamp, voltage clamp
- class 363
 - base class 376
 - also see object-oriented programming: polymorphism, object-oriented programming: inheritance
 - defining a new class – see template: writing a template
 - subclass 376
 - also see object-oriented programming: polymorphism, object-oriented programming: inheritance
 - vs. object 363
 - also see object, object-oriented programming, template
- class definition – see template, class
- Close button – see GUI: tools: Close button
- closed end – see boundary conditions: sealed end
- closed system 36, 38, 40
- cm 13, 15–16, 94
 - default value 103
 - also see specific membrane capacitance, membrane capacitance
- Cm – see cm
- cnexp – see DERIVATIVE block, BREAKPOINT block: SOLVE: cnexp
- collection
 - of numbers – see hoc syntax: variables: double, Vector class
 - of objects – see object: array, List class
 - of sections – see SectionList class, CellBuilder GUI: Subsets page
- command history – see hoc: history function
- COMMENT . . . ENDCOMMENT – see NMODL: comments
- comments – see hoc syntax: comments, NMODL: comments
- comparison of real values – see hoc syntax: float_epsilon
- COMPARTMENT – see KINETIC block: COMPARTMENT
- compartment 37, 52, 60
 - adjacent 53, 56
 - size 37, 41–42, 52, 92
 - vs. biologically relevant structures 92, 96
 - vs. conceptual clarity 96
 - also see segment

- compartmental model – see model: compartmental
- compartmentalization 8, 14
 - also see discretization
- complexity 32–34, 91
- compound statement – see hoc syntax: statements: compound statement
- computational efficiency 68–74, 82, 87–88, 91, 98, 126, 163, 181, 207, 210, 225, 239, 243, 253, 261
 - and STATES 83
 - rate tables – see numeric integration: analytic integration of channel states
- staggered time steps – see Crank–Nicholson method: staggered time steps
- tree topology 165, 167, 169
- why is NEURON fast? 344
- also see d_lambda rule, function table, IntFire2 class: firing time: efficient computation, IntFire4 class: firing time: efficient computation, synaptic transmission: spike-triggered: computational efficiency in NEURON, discrete event simulation: computational efficiency
- computational model
 - implementing with hoc 130
 - also see model: computational
- computational solution – see numeric solution
- computer display – see GUI: screen
- computing
 - purpose – see insight
- concentration 36, 41, 42
 - and accuracy 87
 - default – see ion mechanism: default concentration
 - specification in hoc – see ion mechanism: default concentration: specification in hoc
 - gradient 50
 - initialization – see ion mechanism: default concentration, initialization: ion
 - specification – see ion mechanism: default concentration, initialization: ion
 - also see ion accumulation, ion mechanism, initialization: ion
- conceptual clarity 129, 144, 207–208, 239, 261
- conceptual model – see model: conceptual
- conductance
 - absolute 112
 - density 111
 - also see channel: density
- ion channel – see channel: conductance
- membrane – see specific membrane conductance, membrane resistance
- slope 70
- conflicts between hoc and GUI tools – see hoc: conflicts with GUI
- connect 101, 130
 - preserving spatial accuracy 102
 - preventing confusion – see section: child: connect 0 end to parent
 - also see section
- connectivity – see NetCon class, List object: managing network connections with
- conservation law 50
 - also see charge: conservation, material: conservation, kinetic scheme: conservation rules
- CONSERVE – see KINETIC block: CONSERVE
- consistency of units – see units: consistency
- CONSTANT
 - vs. PARAMETER or LOCAL variable 257
- CONSTANT block 257
- constant
 - built-in constant – see hoc syntax: variables: built-in constants
 - length – see length constant
 - membrane time – see membrane time constant
 - rate – see rate constant
 - time – see time constant
- constant current mechanism 196
 - also see initialization: strategies: injecting a constant current
- continuation character (\) 348
- continue – see hoc syntax: flow control: continue
- continuerun() – see standard run system: continuerun()
- continuous function – see function: continuous
- continuous system – see system: continuous
- continuous variable 90–91
 - piecewise linear approximation 96–97
 - also see range variable: estimating by linear interpolation between nodes
- Control C – see hoc: interrupting execution
- Control D – see hoc: starting and exiting
- Control key – see Graph class: menu_tool()
- control – see simulation control
- convergence 275–276
 - also see synaptic transmission: spike-triggered

- conversion factor – see scale factor, NMODL:
 - units conversion factor, UNITS block:
 - units scaling
- correspondence between conceptual and computational model – see model:
 - correspondence between conceptual and computational
- Crank–Nicholson method 61, 68–72, 165
 - hybrid of backward and forward Euler 68
 - iteration coefficient 62
 - local error 68, 84, 166
 - kinetic scheme 87
 - second order correct plots 180
 - stability 70
 - staggered time steps 70–72, 165, 169
 - summary 86
 - unstaggered time steps 70–71
 - also see second order
- create 101, 130
 - also see section
- creating an object – see object: creating
- creating an object reference – see object
 - reference: declaring
- criterion for proper initialization – see
 - initialization: criterion for proper initialization
- Ctrl key – see Graph class: menu_tool ()
- current 44
 - absolute 112
 - also see current: density
 - axial – see axial current
 - balance – see equation: current balance, Kirchhoff's current law
 - calcium-activated – see Example 9.5: a calcium-activated, voltage-gated current
 - capacitive 122
 - also see membrane current: capacitive
 - density 51, 54, 111
 - also see current: absolute
 - electrode 49, 51
 - also see ELECTRODE_CURRENT
 - ionic – see membrane current: ionic
 - membrane – see membrane current
 - sign convention – see membrane current:
 - positive current convention, axial current: positive current convention, circuit: positive current convention
 - source 51, 53
 - also see circuit: element: current source
 - transmembrane – see membrane current
 - voltage-gated – see channel: voltage-gated
- current clamp 48
 - preserving spatial accuracy – see point
 - process: preserving spatial accuracy
 - also see IClamp class, Example 9.3: an intracellular stimulating electrode
- currently accessed section – see section:
 - currently accessed
- cursor – see focus: cursor
- custom GUI – see user interface: custom GUI
- custom initialization – see initialization:
 - categories, initialization: strategies
- CVODE 73–82, 171
 - abrupt changes and discontinuities – see
 - CVODE: and model descriptions, variable: abrupt change of,
 - NET_RECEIVE block: handling abrupt changes and discontinuities
 - and LONGITUDINAL_DIFFUSION 253
 - and model descriptions 88
 - at_time () 171, 218, 260–262
 - state_discontinuity () 260–262
 - also see NET_RECEIVE block: state_discontinuity ()
 - as generic term for adaptive integration 172
 - default error criteria 80
 - interpolation formulas – see numeric
 - integration: adaptive: interpolation formulas
 - local error 75, 79
 - step () under – see standard run system:
 - step () : under CVODE
 - summary 88
 - also see numeric integration: adaptive, standard run system: CVODE
- CVode class 172
 - re_init () 161, 187–188, 196, 263
 - also see initialization: strategies:
 - changing a state variable
 - record () 83, 166, 179, 187
- CVODES 172
 - also see CVODE, numeric integration:
 - adaptive
- cytoplasmic resistivity 15, 54, 56, 94
 - also see Ra
- D**
- d_lambda rule 8, 122–126
 - also see discretization, spatial grid
- d_X rule 123
 - also see discretization, spatial grid

- DASPK 73–74, 172
 - summary 88
 - also see numeric integration: adaptive, standard run system: DASPK, IDA
- daughter section – see section: child
- DC length constant – see length constant: DC
- declaring an object reference – see object reference: declaring
- declaring variables – see NMODL: declaring variables
- default section – see access, section: currently accessed: default section
- deferred computation – see standard run system: event delivery system: implementing deferred computation
- DEFINE – see NMODL: DEFINE
- define_shape ()
 - effect on diam, area (), and ri () 108
- defining a new class – see template: writing a template
- defining a new template – see template: writing a template
- delayed action – see standard run system: event delivery system: implementing deferred computation
- delta function – see function: delta
- demonstration program – see neurondemo
- density 36, 42
 - also see channel: density, conductance: density, current: density, material: density, state: as density
- density mechanism – see distributed mechanism
- DERIVATIVE block 225, 232, 237
 - ' (apostrophe) 225
 - and CVODE 88
 - dependent variable
 - is a STATE variable 184
 - also see initialization: channel model: Hodgkin–Huxley style
- derivimplicit – see DERIVATIVE block, BREAKPOINT block: SOLVE: derivimplicit
- detaching a section – see disconnect ()
- detail 33–34
 - how much 1, 27
 - also see judgment
- developing new GUI tools – see GUI tool development
- diam 13, 94, 132
 - as an ASSIGNED variable – see ASSIGNED variable: v, celsius, t, dt, diam, and area
 - calculation from 3-D data – see 3-D specification of geometry: calculation of L, diam, area (), and ri ()
- checking 108
 - also see 3-D specification of geometry: diam3d (): checking
- default value 103
- effect of creating a Shape – see Shape object: creating: effect on diam, area (), and ri ()
- effect of define_shape () – see define_shape (): effect on diam, area (), and ri ()
- specifying
 - 3-D specification – see 3-D specification of geometry: calculation of L, diam, area (), and ri ()
 - stylized specification 104
- tapering 115
- updating from 3-D data 109
- diam3d () – see 3-D specification of geometry: diam3d ()
- diameter 3, 94
 - abrupt change 110
 - change flag 168
 - checking – see diam: checking, 3-D specification of geometry: diam3d (): checking
 - problems – see 3-D specification of geometry: diameter: problems
 - zero or narrow diameter 107
 - also see axial resistance: infinite, diam: checking, 3-D specification of geometry: diam3d (): checking
 - also see diam, stylized specification of geometry, 3-D specification of geometry: diam3d ()
- difference equation – see equation: difference
- differential algebraic solver – see DASPK
- differential equation – see equation: differential
- diffusion 41, 199
 - kinetic scheme 200, 238
 - longitudinal 253
 - model geometry – see ion accumulation: initialization: of model geometry
 - radial 246, 254
 - restricted – see Example 9.6: extracellular potassium accumulation
 - under CVODE 88
 - with buffering – see Example 9.8: calcium diffusion with buffering

- also see active transport, buffer, ion
 - accumulation, Example 9.8: calcium
 - diffusion with buffering
- dimensionless variable – see units: dimensionless
- dimensions – see units
- directory browser 17–18
- `disconnect()` 101
- discontinuity – see variable: abrupt change of,
 - diameter: abrupt change
 - also see function: discrete
- discrepancy
 - between conceptual model and
 - computational model 34
 - also see model: correspondence between
 - conceptual and computational
 - between physical system and conceptual
 - model 28
 - between prediction and simulation 27, 29
- discrete event simulation 83, 88
 - computational efficiency 290
 - conditions for 83, 88, 290
 - also see standard run system:
 - `fadvance()`: global time step
 - integration, standard run system:
 - `fadvance()`: local time step
 - integration
- discrete event system – see standard run
 - system: event delivery system
- discrete events and adaptive integration – see
 - standard run system: event delivery
 - system: adaptive integration and, standard
 - run system: `fadvance()`: global time
 - step integration, standard run system:
 - `fadvance()`: local time step integration
- discrete input event – see event: external
- discrete simulation – see discrete event
 - simulation
- discretization 14, 25, 51, 56
 - guidelines 121
 - intent and judgment 91, 118
 - parameter – see `nseg`
 - rule – see `d_lambda` rule, `d_X` rule,
 - discretization: guidelines
 - spatial 51, 53–54, 91–92
 - also see λ , length constant, `d_lambda`
 - rule, spatial grid, spatial accuracy
 - temporal 51, 91, 118
 - also see `dt`, Δt
 - testing – see spatial accuracy, temporal
 - accuracy
- discretization interval – see Δx , spatial grid
- display – see GUI: screen
- distance
 - normalized distance along a section – see
 - range
 - physical distance along a section 114
- distributed mechanism 13, 18, 111, 113, 133,
 - 208, 220
 - adding new – see mechanisms: user-
 - defined, NMODL
 - `insert` – see `insert`
 - vs. point process 113
 - also see NEURON block: SUFFIX
- Distributed Mechanism GUI
 - Manager
 - Inserter 214
- Distributed Mechanism Manager – see
 - Distributed Mechanism GUI: Manager
- distributed physical system – see system:
 - continuous
- divergence 275
 - also see synaptic transmission: spike-
 - triggered
- divide and conquer – see good programming
 - style: divide and conquer
- division operator – see hoc syntax:
 - expressions: operators
- `doEvents()` – see standard run system:
 - `doEvents()`
- dot notation
 - accessing object members – see object:
 - public members: dot notation
 - specifying section properties – see section:
 - currently accessed: dot notation
- double – see hoc syntax: variables: double
- `dt` 24
 - as an ASSIGNED variable – see
 - ASSIGNED variable: `v`, `celsius`, `t`,
 - `dt`, `diam`, and `area`
 - also see `dt`: use in NMODL
 - fixed – see numeric integration: fixed time
 - step
 - use in NMODL 210, 213
 - variable – see numeric integration: adaptive
 - also see standard run system: `setdt()`,
 - standard run system: `fadvance()`,
 - RunControl GUI: `dt`, RunControl GUI:
 - Points plotted/ms
- Δt 61–72, 91
 - fixed – see numeric integration: fixed time
 - step
 - also see discretization
- Δx 59–61
 - also see spatial grid, discretization

E**e**

electronic charge vs. units conversion factor 231
 also see NMODL: units conversion factor
 also see units: e

`e_pas` – see `pas` mechanism

edge – see circuit: edge

efficiency – see computational efficiency

eigenfunction 65–66

eigenvalue 48, 65

elapsed simulation time 24
 also see `t`

electrical synapse – see gap junction, synapse: ephaptic

electrode
 capacitance 48
 compensation 48
 current – see current: electrode, `ELECTRODE_CURRENT`
 intracellular stimulating – see Example 9.3:
 an intracellular stimulating electrode
 resistance 48
 shunting effect of sharp microelectrode –
 see Example 9.2: a localized shunt
`ELECTRODE_CURRENT` 186

electronic charge – see `e`: electronic charge vs.
 units conversion factor

electronic instrumentation – see linear circuit

electrotonic architecture
 spurious effect of changing `nseg` 102

`else` – see hoc syntax: flow control: `else`

`em` 362
 also see `emacs`

`emacs`
 blocks of text
 copying to the kill buffer 408
 cutting to the kill buffer 408
 marking 408
 pasting from the kill buffer 408

buffers 406, 409

cursor movement 407

entering 407

exiting 407

files 409

modes 406, 408

repeating commands
 kill current command 407
 macros 411
 repeating commands 411

search and replace 409

send command to OS 409

text

deleting 408

formatting 409

inserting 408

windows 410

embedding C code – see NMODL:
`VERBATIM . . . ENDVERBATIM`

empty temporal resolution – see temporal
 accuracy: empty

encapsulating code – see object-oriented
 programming: encapsulating code

`ENDCOMMENT` – see NMODL: comments

`ENDVERBATIM` – see NMODL:
`VERBATIM . . . ENDVERBATIM`

enhancing NEURON – see NMODL

`EOT (^D)` – see hoc: starting and exiting

ephapse – see synapse: ephaptic

ephaptic synapse – see synapse: ephaptic

equality operator – see hoc syntax:
 expressions: operators

equation
 algebraic 36, 74, 86–87, 114
 cable 50, 53
 analytic solution 56

characteristic 45

conservation 211

current balance 44, 46, 167–169, 211
 also see Kirchhoff's current law

difference 51

differential 36–39, 42, 46, 48, 50, 63, 65, 114

coupled vs. independent 66, 71

ordinary 46, 53

partial 51

stiff – see system: stiff

iteration – see iteration: equation

node – see equation: current balance,
 Kirchhoff's current law

sacred runes 85

system – see system equations

equilibrium potential
`ASSIGNED` vs. `PARAMETER` – see
`ASSIGNED` variable: when to use for
 an equilibrium potential

computation 186, 195
 also see `ion_style()`, NEURON block:
`NONSPECIFIC_CURRENT`:
 equilibrium potential

initialization – see initialization: ion
 also see `ion_style()`

equivalent circuit – see circuit: equivalent,
 section: equivalent circuit

Erase – see Graph GUI: primary menu: Erase

- erasing traces – see Graph GUI: primary menu: Erase
- error – see numerical error, hoc: error handling, error message
- error message
 - diam = 0 107
 - no accessed section 143
 - no message for pt3dadd with zero diameter 108
 - also see hoc: error handling
- Euler method – see backward Euler method, forward Euler method
- event 81, 87
 - afferent – see event: external
 - aggregation – see numeric integration: fixed time step: event aggregation
 - delivery 80–81
 - delivery system – see standard run system: event delivery system
 - discrete – see discrete event simulation
 - external 288
 - distinguishing from a self-event 288
 - flag 288
 - also see event: external: distinguishing from a self-event
 - handler – see NetCon class: event ()
 - input 80–81
 - also see event: external
 - logical 80
 - mouse – see mouse: events
 - net_send 186
 - queue – see standard run system: event delivery system: cell time queue, standard run system: event delivery system: event time queue
 - self-event 261, 288
 - distinguishing from an external event – see event: external: distinguishing from a self-event
 - implementing absolute refractory period – see IntFire1 class: refractory period
 - implementing deferred computation – see standard run system: event delivery system: implementing deferred computation
 - implementing transmitter release duration – see Example 10.6: saturating synapses
 - also see event: external, NET_RECEIVE block: net_move (), NET_RECEIVE block:
 - net_send (), IntFire2 class: firing time: role of self-events, IntFire4 class: firing time: role of self-events
- spike – see event: external
- times
 - notification – see CVOID: and model descriptions: at_time (), NetCon class: event ()
 - with adaptive integration 218, 263
 - also see NetCon class: event ()
 - with fixed time step integration – see numeric integration: fixed time step: event aggregation
 - also see standard run system: event delivery system: event time queue
- event aggregation to time step boundaries – see numeric integration: fixed time step: event aggregation
- event handler – see NetCon class: event ()
- event queue – see standard run system: event delivery system: event time queue
- event times – see event: times
- event-driven simulation – see discrete event simulation
- event () – see NetCon class: event ()
- Example 9.1: a passive “leak” current 208
- Example 9.2: a localized shunt 214
- Example 9.3: an intracellular stimulating electrode 217
- Example 9.4: a voltage-gated current 220
- Example 9.5: a calcium-activated, voltage-gated current 228
- Example 9.6: extracellular potassium accumulation 233
- Example 9.7: kinetic scheme for a voltage-gated current 240
- Example 9.8: calcium diffusion with buffering 245
- Example 9.9: a calcium pump 255
- Example 10.1: graded synaptic transmission 266
- Example 10.2: a gap junction 271
- Example 10.3: synapse with exponential decay 277
- Example 10.4: alpha function synapse 280
- Example 10.5: use-dependent synaptic plasticity 281
- Example 10.6: saturating synapses 284
- Example 10.7: IntFire1, a basic integrate and fire model 290

- Example 10.8: `IntFire2`, firing rate proportional to input 297
- Example 10.9: `IntFire4`, different synaptic time constants 301
- `execute()` 385
- exiting `hoc` – see `hoc`: starting and exiting
- exiting NEURON – see NEURON: starting and exiting
- `Exp2Syn`
 computational efficiency 280
 also see Example 10.4: alpha function synapse
- explicit Euler method – see forward Euler method
- explicit flux – see KINETIC block:
 `<<(explicit flux)`
- exploiting reusable code – see good programming style: exploiting reusable code
- exploratory simulations – see GUI: vs. `hoc`
- exponentiation operator – see `hoc` syntax: expressions: operators
- `ExpSyn` – see Example 10.3: synapse with exponential decay
- `expsyn1.mod` – see Example 10.3: synapse with exponential decay
- extensive variable – see variable: extensive
- external event – see event: external
- extracellular field – see extracellular mechanism, system equations: matrix form: extracellular field
- extracellular mechanism 56, 74, 87
 computational efficiency 167
 effect of `ELECTRODE_CURRENT` – see NEURON block:
 `ELECTRODE_CURRENT`: effect on extracellular mechanism
 initialization – see initialization: extracellular mechanism
 `vext` 186
 also see NEURON block:
 `ELECTRODE_CURRENT`: effect on extracellular mechanism
- extracellular potassium accumulation – see Example 9.6: extracellular potassium accumulation
- F**
- `f_flux` – see KINETIC block: `f_flux`
- F-H space – see Example 9.6: extracellular potassium accumulation
- factor
 amplification – see amplifier: gain
 conversion or scale – see scale factor, NMODL: units conversion factor, UNITS block: units scaling
- `fadvance.c` 164, 185
 also see initialization: `finitialize()`
- `fadvance()` – see standard run system: `fadvance()`
- FALSE – see `hoc` syntax: expressions: logical expressions
- fan-in – see convergence
 also see synaptic transmission: spike-triggered
- fan-out – see divergence
 also see synaptic transmission: spike-triggered
- faraday – see units: faraday
- fast signals
 spatial decay – see spatial decay of fast signals
- `fast_flushPlot()` – see standard run system: `fast_flushPlot()`
- `fast_flush_list` – see standard run system: plotting system:
 `fast_flush_list`
- `fcurrent()` – see standard run system: `fcurrent()`
- feedback
 amplifier 49
 capacitor 49
 positive 49
- field – see extracellular mechanism
- file
 `hoc` file – see `hoc`
 mod file – see mod file, NMODL
 reading and writing – see `hoc` syntax: basic input and output
 `ses` file – see session file
 session file – see session file
- `finitialize()` – see initialization: `finitialize()`
- fixed
 `dt` – see numeric integration: fixed time step
 `Δt` – see numeric integration: fixed time step
 time step – see numeric integration: fixed time step
- `float_epsilon` – see `hoc` syntax: `float_epsilon`
- flow control – see `hoc` syntax: flow control
- `flushPlot()` – see standard run system: `flushPlot()`
- `flush_list` – see standard run system: plotting system: `flush_list`
- flux 37–39, 42

- backward 39
 - also see KINETIC block: `b_flux`
- forward 39
 - also see KINETIC block: `f_flux`
 - also see KINETIC block: `<<` (explicit flux)
- FOCAL 164
- focus
 - cursor 10
- for – see hoc syntax: flow control: for
- for (x) 114
- forall 100
- forsec 100
- forward Euler method 61
 - iteration coefficient 61
 - local error 64, 83
 - stability 62, 64, 243
 - also see numerical integration: fixed time step
- forward flux – see flux: forward, KINETIC block: `f_flux`
- Fourier theory 56
- fprint() – see hoc syntax: basic input and output: fprint()
- FROM . . . TO . . . – see NMODL: FROM . . . TO . . . (loop statement)
- Frankenhaeuser-Hodgkin space – see Example 9.6: extracellular potassium accumulation
- frecord_init() – see initialization: frecord_init()
- frequency 66
 - spatial 57, 59–61, 122
 - also see spatial accuracy, spatial grid, numerical error: spatial
 - temporal 122
- fscan() – see hoc syntax: basic input and output: fscan()
- funcs and procs
 - \$ – see funcs and procs: arguments: positional syntax
 - \$i – see funcs and procs: arguments: symbolic positional syntax
 - \$o – see funcs and procs: arguments: objects and objrefs
 - \$s – see funcs and procs: arguments: strdup
 - & – see hoc syntax: pointer operator
- arguments
 - call by reference vs. call by value 359, 366
 - numarg() 358
 - objects and objrefs 358–359, 366
 - pointer 358
 - also see hoc syntax: pointer operator
 - positional syntax 358
 - strdef 358–359
 - symbolic positional syntax 358
- defining 357
- local variable 360
 - also see hoc syntax: names
- recursion 360
- return 357
 - also see hoc syntax: flow control: return
- returned value – see funcs and procs: return
- FUNCTION
 - calling from hoc – see hoc: calling an NMODL FUNCTION or PROCEDURE
- function
 - continuous
 - of space 51
 - also see range variable
 - of time 51
 - delta 51
 - discrete 77
 - piecewise linear 78, 80
 - also see continuous variable: piecewise linear approximation
 - rate – see BREAKPOINT block: and rate functions, KINETIC block: reaction rates: voltage sensitive
 - table – see function table
 - also see funcs and procs, FUNCTION
- FUNCTION block 226, 232
 - arguments are call by value 283
- function call operator – see hoc syntax: expressions: operators
- function table 114, 170
 - also see NMODL: FUNCTION_TABLE
- FUNCTION_TABLE – see NMODL: FUNCTION_TABLE
- G**
 - g_pas – see pas mechanism
 - gain – see amplifier: gain
 - gap junction 265–266, 271
 - computational efficiency 167
 - conservation of charge 271
 - also see charge: conservation, equation: current balance, Kirchhoff's current law
 - spurious oscillations 271
 - under CVODE 88
 - also see Example 10.2: a gap junction, synapse: ephaptic, LinearCircuitBuilder: for gap junctions

- `gap.mod` – see Example 10.2: a gap junction
- gating model – see channel: gating model
- gating state
 - initialization – see initialization: channel model
 - custom initialization – see initialization: strategies: changing a state variable
- gating variable – see gating state
- Gaussian elimination 74, 167–169
- generic starting point for GUI tool
 - development – see GUI tool
 - development: generic starting point
- GENESIS 207, 210
- geometry 98
 - artifacts
 - stylized specification reinterpreted as 3-D specification 108
 - zero diameter 107
 - specifying – see 3-D specification of geometry, stylized specification of geometry
 - also see CellBuilder GUI: Geometry page
 - also see topology, anatomical properties
- `getstr()` – see hoc syntax: basic input and output: `getstr()`
- GLOBAL – see NEURON block: GLOBAL, ASSIGNED variable: GLOBAL, PARAMETER variable: is GLOBAL by default
 - also see LOCAL variable: declared outside an equation block: scope and persistence
- global – see GLOBAL, global variable
- global error – see numerical error: global
- global time step – see numeric integration: adaptive: global time step
- global variable – see variable: local vs. nonlocal, hoc syntax: names, template: writing a template: external
- GMODL 210
- good programming style
 - bulletproofing against nonsense arguments 330
 - divide and conquer 5
 - exploiting reusable code 307, 324, 328
 - iterative development 155, 329, 378, 381
 - modular programming 5, 142, 328
 - control – see simulation control
 - instrumentation – see instrumentation
 - model specification – see model specification
 - program organization 98, 154
 - separating model specification from user interface 155, 328
 - also see hoc: idiom
- graded synapse – see synaptic transmission: graded
- gradient – see concentration: gradient, voltage: gradient
- `gradsyn.mod` – see Example 10.1: graded synaptic transmission
- Graph
 - automatically updating plots and x axis – see standard run system: `addplot()`
 - creating
 - Shape plot 22
 - Space Plot 22
 - Voltage axis 21
 - incorporating into plotting system – see standard run system: plotting system: incorporating Graphs and objects, standard run system: `addplot()`
- Graph class
 - `addexpr()` 180
 - `addvar()` 180
 - `begin()` 180–181
 - also see initialization: `initPlot()`, standard run system: `Plot()`
 - `beginline()` 390
 - `erase_all()` 337, 389
 - `exec_menu()` 392
 - `flush()` 180–181, 393
 - `mark()` 393
 - `menu_tool()` 380
 - `plot()` 180–181
 - also see standard run system: `Plot()`
 - `save_name()` 385
 - `size()` 181, 337, 391
 - `view_count()` 180–181
 - `view_info()` 394
 - `view()` 335
 - also see standard run system: `addplot()`
- Graph GUI
 - primary menu
 - Erase 29
 - Keep Lines 28
 - Plot what? – see Plot what? GUI
- graph
 - raster – see spike trains: recording and plotting, NetWork Builder: buttons: SpikePlot
 - also see Graph
- graph lists – see standard run system: plotting system: `graphLists`

- graph theory 44, 317
- graphical interface – see user interface:
 - custom GUI, NEURON Main Menu
- graphical tools – see NEURON Main Menu,
 - CellBuilder, Graph, PFWM, PointProcessManager, RunControl, Shape plot, Shape Plot, Space Plot, Plot what?, GUI tool development
- graphics terminology – see GUI: graphics terminology
- greater than operator – see hoc syntax:
 - expressions: operators
- greater than or equal to operator – see hoc syntax:
 - expressions: operators
- ground – see circuit: element: ground
- gsyn.mod – see Example 10.5: use-dependent synaptic plasticity
- GUI
 - combining with hoc 141, 324
 - computer display – see GUI: screen
 - conflicts with hoc or other GUI tools 152
 - focus – see focus: cursor
 - graphics terminology 378
 - implementing a computational model – see
 - CellBuilder, NetWork Builder
 - mapping to the screen – see GUI: screen:
 - mapping to the screen
 - model 378
 - model coordinates 378
 - also see GUI: scene coordinates
 - scene 378
 - scene coordinates 378
 - making scene and view coordinates equivalent 395
 - vs. screen coordinates 394
 - screen 378
 - mapping to the screen 378
 - also see GUI tool development:
 - mapping to the screen
 - screen coordinates 378
 - vs. scene coordinates – see GUI:
 - scene coordinates: vs. screen coordinates
 - tools
 - are implemented in hoc 129, 344
 - Close button 381, 383
 - developing new tools – see GUI tool development
 - work by constructing hoc programs 129, 344
 - also see NEURON Main Menu GUI
 - view 378
 - which view contains the mouse – see
 - Graph class: view_info()
 - vs. hoc 128
 - also see user interface: custom GUI, NEURON Main Menu, standard GUI library
 - GUI tool development
 - basic pattern – see GUI tool development:
 - generic starting point
 - box – see VBox, HBox
 - Close button – see GUI: tools: Close button
 - general issues
 - allowing multiple instances 380, 383–384
 - destroying 381, 383–384
 - encapsulating 380
 - saving and retrieving 380, 385, 395
 - also see session file, session file:
 - object_push(), session file: object_pop(), GUI: scene coordinates: making scene and view coordinates equivalent
 - generic starting point 387
 - graphical model – see GUI: model
 - mapping to the screen 386–387
 - window title 384
 - also see GUI: screen: mapping to the screen, VBox class: map()
 - mouse events – see mouse: events
 - panel – see xpanel()
 - which view contains the mouse – see
 - Graph class: view_info()
- H**
 - Hamming, R.W. 86
 - HBox 383
 - also see VBox
 - hh mechanism 16
 - hiding information – see object-oriented programming: information hiding
 - Hinton plot 180
 - history function – see hoc: history function
 - hoc 2, 17, 128, 343
 - adding new mechanisms – see nrniv:
 - adding new mechanisms, mechanisms: user-defined, NMODL
 - calling an NMODL FUNCTION or PROCEDURE 226
 - specifying proper instance with setdata_ 226
 - can do anything that a GUI tool can 130
 - combining with GUI 141, 324

- hoc (*contd*)
 - conflicts with GUI 152
 - efficient simulation – see computational efficiency: why is NEURON fast?
 - enhancements and extensions 343
 - error handling 348
 - functions – see funcs and procs
 - history function 347
 - idiom
 - forall nseg*=3 97
 - forall psection() 134
 - load_file("nrngui.hoc") 142, 159
 - also see List class: iteration
 - immediate mode 347
 - implementing a computational model 130
 - interrupting execution 349
 - also see hoc: starting and exiting
 - Kernighan and Pike 343
 - keywords – see hoc syntax: keywords
 - libraries 344
 - also see standard GUI library, standard run library
 - oc> prompt 347
 - parse errors – see hoc: error handling
 - procedures – see funcs and procs
 - run-time errors – see hoc: error handling
 - scalar – see hoc syntax: variables: scalars
 - scope – see hoc syntax: names, funcs and procs: local variable, variable: local vs. nonlocal, LOCAL variable, ASSIGNED variable: GLOBAL, NEURON block: GLOBAL, PARAMETER variable: is GLOBAL by default
 - also see object: public members, template: writing a template: public, template: writing a template: external
 - speed – see computational efficiency: why is NEURON fast?
 - starting and exiting 346
 - stopping – see hoc: interrupting execution
 - also see hoc: starting and exiting
 - syntax – see hoc syntax
 - top level of the interpreter 385
 - also see execute()
 - user-defined variable – see hoc syntax: names
 - vs. GUI 128
 - also see hoc syntax, Programmer's Reference, good programming style
- hoc file
 - executing – see NEURON: starting with a specific hoc file, nrngui, nrniv
 - also see load_file(), hoc syntax: basic input and output: xopen()
- hoc syntax
 - basic input and output
 - fprint() 361
 - fscan() 362
 - getstr() 362
 - print 361
 - printf() 361
 - read() 361
 - ropen() 362
 - sprint() 361
 - wopen() 361
 - xopen() 362
 - also see load_file()
 - xred() 362
 - comments 130, 355
 - continuation character 348
 - expressions 354
 - logical comparison of real values – see hoc syntax: float_epsilon
 - logical expressions 355
 - operators 354
 - precedence – see hoc syntax: expressions: operators
- float_epsilon 355
- flow control 356
 - break 100, 357
 - continue 100, 357
 - else 356
 - for 356
 - if 356
 - iterator 356
 - iterator_statement 356
 - quit() 357
 - return 100, 357
 - also see funcs and procs: return
 - stop 357
 - while 356
- functions – see funcs and procs
- keywords 350
- names 350
- operators – see hoc syntax: expressions: operators, hoc syntax: pointer operator
- pointer operator 358
- precedence of operations – see hoc syntax: expressions: operators

- procedures – see `funcs` and `procs`
 - statements 355
 - compound statement 355
 - variables 353
 - arrays – see `hoc` syntax: variables:
 - double
 - built-in constants 353
 - cannot change type 156
 - cannot redefine type 354
 - double 353
 - also see `Vector` class
 - `objref` – see object reference: `objref`
 - scalars 353
 - `strdef` 354
 - strings – see `hoc` syntax: variables:
 - `strdef`
 - also see `hoc`: idiom, Programmer's Reference
 - Hodgkin–Huxley delayed rectifier – see
 - Example 9.4: a voltage-gated current
 - Hodgkin–Huxley mechanism – see `hh`
 - mechanism
 - hypothesis 1, 24
 - testing 32–33
 - also see model: conceptual
- I**
- `IClamp` class 112
 - preserving spatial accuracy – see point
 - process: preserving spatial accuracy
 - `iclamp1.mod` – see Example 9.3: an
 - intracellular stimulating electrode
 - IDA 172
 - initialization 187
 - also see numeric integration: adaptive,
 - DASPK, SUNDIALS
 - idiom – see `hoc`: idiom
 - `if` – see `hoc` syntax: flow control: `if`
 - immediate mode – see `hoc`: immediate mode
 - implicit Euler method – see backward Euler
 - method
 - inequality operator – see `hoc` syntax:
 - expressions: operators
 - information hiding – see object-oriented
 - programming: information hiding
 - inheritance – see object-oriented
 - programming: inheritance
 - INITIAL block 186, 188, 190, 224
 - initializing STATE variables – see STATE
 - variable: initialization
 - inside `NET_RECEIVE` block – see
 - `NET_RECEIVE` block: INITIAL block
 - sequence-dependent 186
 - SOLVE
 - STEADYSTATE `sparse` 189, 243, 258
 - also see initialization: channel model:
 - kinetic scheme
 - also see initialization, mechanisms:
 - initialization sequence, mechanisms:
 - user-defined: call order
 - `init()` – see initialization: `init()`
 - initial value problem 48
 - initialization 24, 28, 140
 - adaptive integration – see numeric integration:
 - adaptive: initialization, `CVode` class:
 - `re_init()`, IDA: initialization
 - active transport – see active transport:
 - initialization
 - analysis 183
 - ASSIGNED variable – see ASSIGNED
 - variable: initialization
 - basic 185
 - also see initialization:
 - `finalize()`, initialization:
 - default
 - categories 225
 - of a chaotic system – see initialization:
 - categories: to a desired state
 - of an oscillating system – see
 - initialization: categories: to a
 - desired state
 - overview of custom initialization 185, 188
 - also see initialization: `init()`: custom
 - to a desired state 198
 - to a particular resting potential 195
 - to steady state 197
 - also see initialization: basic, initialization:
 - default
 - channel model 188
 - Hodgkin–Huxley style 188
 - kinetic scheme 189
 - also see INITIAL block: SOLVE:
 - STEADYSTATE `sparse`, initialization: strategies: changing a
 - state variable, initialization:
 - strategies: changing model
 - parameters, initialization: strate-
 - gies: steady state initialization of
 - complex kinetic schemes
 - criterion for proper initialization 183
 - custom 152
 - also see initialization: categories,
 - initialization: strategies,
 - initialization: `init()`: custom

- initialization (*contd*)
 - default 187
 - also see initialization: `init()`,
 - initialization: `stdinit()`,
 - initialization: basic
 - diffusion – see ion accumulation:
 - initialization
 - `dt` – see standard run system: `setdt()`
 - event delivery system – see standard run
 - system: event delivery system:
 - initialization, initialization:
 - `finitialize()`
 - extracellular mechanism 186
 - `fcurrent()` – see standard run system:
 - `fcurrent()`: in initialization
 - `finitialize()` 160, 164, 175, 185–187, 219, 224
 - fixed time step integration – see numeric
 - integration: fixed time step:
 - initialization
 - `frecord_init()` 188
 - `init()` 160, 164, 187–188
 - custom 195–198, 204
 - also see initialization: strategies:
 - changing a state variable
 - `initPlot()` 164, 180, 187
 - internal data structures dependent on
 - topology and geometry 185
 - also see ion accumulation: initialization:
 - of model geometry
 - ion 186, 191–194
 - accumulation – see ion accumulation:
 - initialization
 - also see ion mechanism: default
 - concentration, `ion_style()`,
 - initialization: strategies: changing a state variable
 - kinetic scheme 183
 - also see INITIAL block: SOLVE:
 - STEADYSTATE `sparse`,
 - KINETIC block: CONSERVE: when is it required for initialization?
 - linear circuit 183, 186
 - membrane potential – see initialization:
 - `v_init`, membrane potential:
 - initialization
 - NetCon object – see NET_RECEIVE
 - block: INITIAL block
 - network 183, 186
 - non-steady state 166
 - object – see template: variable initialization
 - random number generator 183
 - `Random.play()` 185
 - recording 183
 - also see initialization:
 - `frecord_init()`, Vector class:
 - `record()`: initialization
 - `startsw()` 187
 - also see run time, standard run system:
 - realtime
 - STATE variable – see STATE variable:
 - initialization
 - `state0` – see STATE variable:
 - initialization
 - `stdinit()` 160, 164, 180, 187
 - steady state – see initialization: categories: to
 - steady state, initialization: categories: to a particular resting potential
 - strategies 225
 - changing a state variable 187–188
 - also see STATE variable: initialization,
 - ion mechanism: initialization,
 - ion mechanism: default concentration: specification in hoc
 - changing an equilibrium potential 195
 - changing model parameters 199
 - groundhog day 199
 - injecting a constant current 196
 - jumping back to move forward 197
 - steady state initialization of complex
 - kinetic schemes 258
 - also see initialization: `init()`: custom
 - synaptic weight vector – see NET_RECEIVE
 - block: INITIAL block
 - `t` 185
 - template – see template: variable initialization
 - `v_init` 24, 186–188
 - also see membrane potential: initialization
 - `Vector.play()` 186
 - `Vector.record()` – see initialization:
 - `frecord_init()`, Vector class:
 - `record()`: initialization
 - weight vector – see NET_RECEIVE block:
 - INITIAL block
 - also see RunControl GUI: Init, RunControl
 - GUI: Init & Run, INITIAL block,
 - NET_RECEIVE block: INITIAL
 - block, Graph class: `begin()`,
 - mechanisms: initialization sequence,
 - mechanisms: user-defined: call order
 - initialize microstep – see numeric integration:
 - adaptive: initialize microstep
 - also see standard run system: `fadvance()`:
 - local time step integration

- `initPlot()` – see initialization:
 - `initPlot()`
- input and output – see hoc syntax: basic input and output
- input event – see event: external
- insert 111, 133
 - also see `uninsert`
- insight 32, 34, 86
- instability – see numeric integration:
 - instability, KINETIC block: reaction rates: STATE-dependent, and instability
- instrumentation 5, 18, 154
- integrate and fire 83, 88
 - also see artificial spiking cell
- integration – see numeric integration
- intensive variable – see variable: intensive
- intent – see user's intent
- interpolate microstep – see numeric integration:
 - adaptive: interpolate microstep
 - also see standard run system: `fadvance()`: local time step integration
- interpolation – see `Vector` class: `play()`: with interpolation
- interpreter – see hoc
- `IntFire1` class 290
 - effect of an input event 292
 - `m` – see `IntFire1` class: membrane state variable
 - `M` – see `IntFire1` class: membrane state variable: visualizing
 - membrane state variable 290
 - time constant 290
 - visualizing 292, 296
 - `refrac` – see `IntFire1` class: refractory period
 - refractory period 294
 - `tau` – see `IntFire1` class: membrane state variable: time constant
 - also see artificial spiking cell, Example 10.7: `IntFire1`, a basic integrate and fire model
- `IntFire2` class 297
 - approximate firing rate 298
 - bias – see `IntFire2` class: synaptic current state variable: bias
 - constraint on time constants 298
 - effect of an external event 297, 299
 - firing time
 - efficient computation 299
 - role of self-events 299
 - `i` – see `IntFire2` class: synaptic current state variable
 - `ib` – see `IntFire2` class: synaptic current state variable: bias
 - `m` – see `IntFire2` class: membrane state variable
 - membrane state variable 297
 - time constant 297
 - also see `IntFire2` class: constraint on time constants
 - synaptic current state variable 297
 - bias 297
 - time constant 297
 - also see `IntFire2` class: constraint on time constants
- `IntFire4` class 301
 - constraint on time constants 301, 399
 - convergence tolerance 304
 - `e` – see `IntFire4` class: synaptic current state variables: excitatory
 - effect of an external event 301
 - `eps` – see `IntFire4` class: convergence tolerance
 - firing time
 - efficient computation 302
 - role of self-events 303
 - `i1` – see `IntFire4` class: synaptic current state variables: inhibitory
 - `i2` – see `IntFire4` class: synaptic current state variables: inhibitory
 - `m` – see `IntFire4` class: membrane state variable
 - membrane state variable 301
 - time constant 301, 399
 - synaptic current state variables
 - excitatory 302
 - inhibitory 302
 - time constants 301, 399
 - also see `IntFire4` class: constraint on time constants
 - `taue` – see `IntFire4` class: synaptic current state variables: time constants
 - `taui1` – see `IntFire4` class: synaptic current state variables: time constants
 - `taui2` – see `IntFire4` class: synaptic current state variables: time constants

`IntFire4` class (*contd*)
 `taum` – see `IntFire4` class: synaptic
 current state variables: time
 constants
 also see artificial spiking cell, Example
 10.9: `IntFire4`, different synaptic
 time constants
 intuition 32–33
 ion accumulation 199
 initialization 200
 of model geometry 251
 also see ion mechanism: default
 concentration, `ion_style()`,
 initialization: strategies: changing a
 state variable, initialization:
 strategies: changing model
 parameters
 kinetic scheme 200
 also see Example 9.8: calcium diffusion
 with buffering, Example 9.9: a
 calcium pump
 under CVODE 88
 also see active transport, buffer, diffusion,
 ion mechanism, `ion_style()`,
 equilibrium potential, `STATE` variable:
 ion concentration as, Example 9.6:
 extracellular potassium accumulation
 ion channel 13
 ion concentration – see concentration, ion
 accumulation, ion mechanism
 ion mechanism
 `_ion` suffix 190
 automatically created 190, 235
 default concentration
 for user-created ion names 194
 name 194
 specification in `hoc` 194–195
 equilibrium potential – see equilibrium
 potential: computation
 initialization 194, 236
 also see initialization: ion
`ion_style()` 194
 also see ASSIGNED variable, PARAMETER
 variable, `STATE` variable, concentration,
 initialization: ion
 ionic conductance 18
 also see channel: density
 isopotential – see membrane potential:
 isopotential
 iteration
 coefficient 61
 equation 61

 of nonlinear equations – see numeric
 integration: iteration of nonlinear
 equation
 over a `List` – see `List` class: iteration
 over nodes – see range variable: iterating
 over nodes
 over sections – see section: iterating over
 sections
 using `iterator` – see `hoc` syntax: flow
 control: `iterator`
 iterative development – see good
 programming style: iterative development
`iterator` – see `hoc` syntax: flow control:
 `iterator`
`iterator_statement` – see `hoc` syntax:
 flow control: `iterator_statement`

J

Jacobian 239, 243
 analytic 114
 approximate 74, 226, 253, 271
 computing di/dv elements 168, 189, 262
 nearly singular 244
 user-supplied 226
 judgment 1, 30, 34, 85
 also see accuracy, detail, qualitative results

K

`k-mole` – see units: `k-mole`
`k_ion` – see ion mechanism: automatically
 created
`k3st.mod` – see Example 9.7: kinetic
 scheme for a voltage-gated current
`kd.mod` – see Example 9.4: a voltage-gated
 current
 Keep Lines – see Graph GUI: primary menu:
 Keep Lines
 Kernighan, B.W. – see `hoc`: Kernighan and
 Pike
`kext.mod` – see Example 9.6: extracellular
 potassium accumulation
 keyboard
 Alt key – see Graph class: `menu_tool()`
 Control C – see `hoc`: interrupting execution
 Control D – see `hoc`: starting and exiting
 Control key – see Graph class:
 `menu_tool()`
 Shift key – see Graph class:
 `menu_tool()`
 keywords – see `hoc` syntax: keywords,
 NMODL
`ki` – see potassium, ion mechanism

`ki0_k_ion` – see potassium, ion
 mechanism: default concentration:
 specification in `hoc`
 KINETIC block 243, 252, 259
 $\rightarrow$ (sink reaction indicator) 261
 $\sim$ (tilde) 238
 $\leftrightarrow$ (reaction indicator) 238
 $\ll$ (explicit flux) 253
 and CVOID 88
 `b_flux` 238
 COMPARTMENT 247, 253–254
 CONSERVE 243–244
 when is it required for initialization?
 244, 252, 258
 dependent variable
 is a STATE variable 184
 also see KINETIC block: reactants:
 ASSIGNED or PARAMETER
 variables as
 `f_flux` 238
 initialization – see initialization: channel
 model: kinetic scheme, INITIAL
 block: SOLVE: STEADYSTATE
 sparse
 LONGITUDINAL DIFFUSION 253
 radial diffusion – see Example 9.8: calcium
 diffusion with buffering
 reactants 238
 ASSIGNED or PARAMETER variables as
 242
 also see KINETIC block: dependent
 variable: is a STATE variable
 reaction rates 238
 STATE-dependent, and instability 239
 voltage-sensitive 239
 reaction statement 238
 also see Crank–Nicholson method: local
 error: kinetic scheme, BREAKPOINT
 block: SOLVE: sparse
 kinetic scheme 36–37
 compartment size 41
 also see scale factor, NMODL: units
 conversion factor, KINETIC block:
 COMPARTMENT
 conservation rules 39
 also see scale factor, NMODL: units
 conversion factor, KINETIC block:
 COMPARTMENT
 equivalent differential equations 38
 initialization – see initialization: channel
 model: kinetic scheme, initialization:
 strategies: changing a state variable,

initialization: strategies: changing
 model parameters, KINETIC block:
 CONSERVE: when is it required for
 initialization?
 reactants – see KINETIC block: reactants
 reaction rates – see KINETIC block:
 reaction rates
 also see KINETIC block
 Kirchhoff's current law 44
 also see charge: conservation, equation:
 current balance
`ko` – see potassium, ion mechanism
`ko0_k_ion` – see potassium, ion
 mechanism: default concentration:
 specification in `hoc`
L
 λ – see length constant
 L 13, 94, 132
 calculation from 3-D data – see 3-D
 specification of geometry: calculation
 of `L`, `diam`, `area()`, and `ri()`
 change flag 168
 default value 103
 specifying
 stylized specification 104
 3-D specification – see 3-D specification
 of geometry: calculation of `L`,
 `diam`, `area()`, and `ri()`
 updating from 3-D data 109
 also see length
`lambda_f()` 123
 leak current – see `pas` mechanism, `leak.mod`
`leak.mod` – see Example 9.1: a passive
 “leak” current
 legacy code – see troubleshooting: legacy
 code
 length 3, 94
 arc – see range
 also see L
 length constant 13–14
 AC 122
 DC 121
 also see `d_lambda` rule
 less than operator – see `hoc` syntax:
 expressions: operators
 less than or equal to operator – see `hoc`
 syntax: expressions: operators
 linear algebra 65
 LINEAR block 223
 dependent variable
 is a STATE variable 184

linear circuit 56, 74, 87
 computational efficiency 167
 initialization – see initialization: linear circuit
 also see system equations: matrix form:
 linear circuit
 linear taper – see range variable: linear taper
 LinearCircuitBuilder
 for gap junctions 272
 linearity – see channel: linear, channel: gating
 model: HH-type
 LinearMechanism class 101
 list
 of objects – see List class
 of sections – see SectionList class
 List class 373
 append() 335, 373
 count() 325, 330, 373
 iteration 373
 object stack 374
 object() 325, 330, 373
 remove_all() 330
 remove() 375
 List object
 managing network cells with 327
 managing network connections with 279, 327
 load_file() 124, 362
 also see hoc syntax: basic input and output:
 xopen(), hoc: idiom:
 load_file("nrngui.hoc")
 loc() – see point process: loc()
 local – see hoc syntax: names, funcs and
 procs: local variable, LOCAL variable
 also see object: public members: vs. private
 members, template: writing a template:
 public
 local absolute error – see absolute error: local
 local error – see numerical error: local
 local relative error – see relative error: local
 local time step – see numeric integration:
 adaptive: local time step
 LOCAL variable
 declared outside an equation block
 initial value 251
 scope and persistence 251
 also see NEURON block: GLOBAL
 declared within an equation block
 scope and persistence 227
 local variable – see hoc syntax: names, funcs
 and procs: local variable, LOCAL variable
 also see object: public members: vs. private
 members, template: writing a template:
 public

logical comparison of real values – see hoc
 syntax: float_epsilon
 logical comparison operators – see hoc
 syntax: expressions: operators
 logical expressions – see hoc syntax:
 expressions: logical expressions
 logical operators – see hoc syntax:
 expressions: operators
 LONGITUDINAL_DIFFUSION – see
 KINETIC block:
 LONGITUDINAL_DIFFUSION
 longitudinal diffusion – see diffusion: kinetic
 scheme: longitudinal, KINETIC block:
 LONGITUDINAL_DIFFUSION
 loop statement – see NMODL: FROM...
 TO... (loop statement)
 also see hoc syntax: flow control: for,
 hoc syntax: flow control: while

M

mapping to the screen – see GUI: screen:
 mapping to the screen
 Markov process
 kinetic scheme 238
 mass 50
 also see material
 mass balance – see material: conservation
 material 36
 amount 36, 42
 concentration 36
 conservation 36, 238, 242–244, 259
 density 36
 mechanisms
 adding new – see mechanisms: user-defined
 initialization sequence 186
 involving delay – see standard run system:
 event delivery system: implementing
 deferred computation
 user-defined 113
 call order 186
 also see nrniv: adding new
 mechanisms, NMODL
 also see distributed mechanism, point process
 membrane area 51
 also see area(), surface area
 membrane capacitance 15, 53, 91
 also see cm, specific membrane capacitance
 membrane conductance – see specific
 membrane conductance
 membrane current
 capacitive 53, 122
 ionic 53, 122

- accuracy 169
- positive current convention 51
- membrane potential 20–21, 49, 54, 94
- initialization 186–188
 - also see initialization: `v_init`,
 - initialization: categories,
 - initialization: strategies
- isopotential 53
- also see `v`, membrane state variable
- membrane resistance 16, 121
- membrane state variable – see `IntFire1`
 - class: membrane state variable,
 - `IntFire2` class: membrane state
 - variable, `IntFire4` class: membrane
 - state variable
- membrane time constant 122
 - and attenuation of fast signals 123
- `METHOD` – see `BREAKPOINT` block: `SOLVE`
- microelectrode – see electrode
- MicroEMACS 362
 - also see `em`, `emacs`
- microstep – see numeric integration: adaptive:
 - advance microstep, numeric integration:
 - adaptive: initialize microstep, numeric
 - integration: adaptive: interpolate
 - microstep
- `miscellaneous_add()` – see
 - `NEURONMainMenu` class:
 - `miscellaneous_add()`
- mistake
 - programming – see `hoc`: error handling
- `mknrndll` – see `NMODL`: translator:
 - `mknrndll`
- `mod` file 207, 345
 - compiling
 - under `MSWindows` – see `NMODL`:
 - translator: `mknrndll`
 - under `UNIX/Linux` – see `NMODL`:
 - translator: `nrnivmodl`
 - also see `NMODL`
- model
 - 3-D 110
 - also see 3-D specification of geometry
 - ball and stick 34
 - compartmental 92, 96
 - computational 1, 5, 33
 - analysis 24
 - essential steps 128
 - implementation 34, 98
 - also see computational model:
 - implementing with `hoc`
 - model specification 16
 - conceptual 1, 3, 28, 33, 36, 118
 - correspondence between conceptual and
 - computational 128, 130
 - also see discrepancy: between conceptual
 - model and compartmental model
 - network – see network model
 - purpose – see judgment, user's intent
 - simulation control – see simulation control
 - stylized 104
 - also see stylized specification of geometry
 - testing 141
 - also see topology: checking
- Model Description Language – see `MODL`
- model coordinates – see `GUI`: model
 - coordinates
 - vs. screen coordinates – see `GUI`: scene
 - coordinates: vs. screen coordinates
- model description language – see `NMODL`
- model properties
 - specifying 98
 - also see anatomical properties:
 - specifying, biophysical properties:
 - specifying
- model specification 5, 16, 154
 - as virtual experimental preparation 154
 - vs. user interface – see good programming
 - style: separating model specification
 - from user interface
 - also see model properties: specifying
- modeling 73, 79
 - empirically-based 32, 61
 - rationale 33
- modified Euler method 271
 - also see backward Euler method
- modifying standard functions and procedures
 - see standard run library: redefining
 - functions and procedures, standard `GUI`
 - library: redefining functions and
 - procedures
- `MODL` 208
 - vs. `NMODL` 208, 210
- `modlunit` 216, 231
 - also see units: checking
- modular programming – see good
 - programming style: modular
 - programming
- modulus operator – see `hoc` syntax:
 - expressions: operators
- `mole` – see units: `mole`
- mole equivalents 39
- morphometric data – see quantitative
 - morphometric data

- mouse
 - button – see mouse: events
 - cursor – see focus: cursor
 - events 379
 - Alt, Control, or Shift key – see Graph class: menu_tool()
 - cursor coordinates 380, 392
 - handling 380, 392
 - also see Graph class: menu_tool()
 - which view contains the mouse – see Graph class: view_info()
- movie – see Vector: movie
- multiplication operator – see hoc syntax: expressions: operators
- N**
- n3d() – see 3-D specification of geometry: n3d()
- na_ion – see ion mechanism: automatically created
- nai – see ion mechanism
- nai0_na_ion – see ion mechanism: default concentration: specification in hoc
- nao – see ion mechanism
- nao0_na_ion – see ion mechanism: default concentration: specification in hoc
- National Biomedical Simulation Resource project 208
- Nernst equation – see initialization: ion, ion_style()
- Nernst potential – see equilibrium potential, initialization: ion, ion_style()
- net_event() – see NET_RECEIVE block: net_event()
- net_move() – see NET_RECEIVE block: net_move()
- NET_RECEIVE block 275, 278
 - arguments are call by reference 283
 - flag – see event: flag
 - handling abrupt changes and discontinuities 262
 - INITIAL block 186, 224, 283
 - also see NetCon class: weight vector
 - net_event() 292, 299, 303
 - net_move() 289, 299
 - net_send() 289, 295
 - state_discontinuity() 262
 - also see Examples 10.3–10.9
- net_send() – see NET_RECEIVE block: net_send()
- NetCon
 - and standard run system 166
 - NetCon class 272, 274
 - delay 274
 - event() 263
 - record() 335
 - source variable 274
 - states – see NetCon class: weight vector
 - stream-specificity 274, 279
 - target 275
 - threshold 274
 - weight 274
 - weight vector 275
 - initialization 283
 - NetCon object
 - as a channel for a stream of events 289
 - initialization – see NET_RECEIVE block: INITIAL block
- netcvcde.cpp 166
- NetGUI class – see NetWork Builder
- NetReadyCellGUI
 - bringing up a NetReadyCellGUI 322
 - cell names – see NetWork Builder: cells: names
 - changing cell parameters – see NetWork Builder: caveats
- NetWork Builder
 - adjusting model parameters 318
 - bringing up a NetWork Builder 313
 - buttons
 - Create 318, 323
 - Delays – see NetWork Builder: specifying delays and weights
 - Hoc File – see NetWork Builder: exporting reusable code
 - Locate – see NetWork Builder: cells: creating
 - Show Cell Map – see NetWork Builder: cells: Cell Map
 - SpikePlot 318
 - Src→ Tar – see NetWork Builder: setting up network architecture
 - Weights – see NetWork Builder: specifying delays and weights
- canvas 314
 - dragging 315
- caveats 322
- cells
 - Cell Map 324
 - connecting – see NetWork Builder: setting up network architecture, NetWork Builder: specifying delays and weights
 - creating 314

- names 314, 323
 - also see **NetWork Builder: cells: Cell Map**
- types – see **NetWork Builder: palette of cell types**, **ArtCellGUI**, **NetReadyCellGUI**
- changing a network – see **NetWork Builder: caveats**
- changing cell parameters – see **NetWork Builder: caveats**
- creating an instance of a network – see **NetWork Builder: buttons: Create**
- default section – see **NetWork Builder: exporting reusable code: acell_home_**
- delays and weights – see **NetWork Builder: specifying delays and weights**
- exploiting reusable code 328
- exporting reusable code 324
 - acell_home_** 326
 - network cell templates 326
 - network instantiation 328
 - network specification interface 327
- hints 314
- network architecture – see **NetWork Builder: setting up network architecture**
- palette of cell types 314
- plotting spikes – see **NetWork Builder: buttons: SpikePlot**
- reusable code – see **NetWork Builder: exploiting reusable code**
- saving a hoc file – see **NetWork Builder: exporting reusable code**
- setting up network architecture 315
- specifying delays and weights 316
- network model
 - architecture – see **convergence, divergence**
 - also see **NetCon**, **List** object:
 - managing network connections with
 - building with GUI – see **ArtCellGUI**, **NetReadyCellGUI**, **NetWork Builder**
 - connectivity – see **NetCon**, **List** object:
 - managing network connections with
 - creating algorithmically 329
 - initialization – see **initialization: network**, **NET_RECEIVE** block: **INITIAL** block
- neurite 51, 92, 105
 - also see section
- NEURON
 - adding new mechanisms – see **mechanisms: user-defined, nrniv: adding new mechanisms, NMODL**
 - last change date – see **NEURON: startup banner**
 - starting and exiting 6
 - starting with a specific hoc file 133
 - startup banner 347
 - version number – see **NEURON: startup banner**
 - also see **hoc: starting and exiting**
- NEURON block 210
- ARTIFICIAL_CELL 291
 - also see **artificial spiking cell**, **IntFire1** class, **IntFire2** class, **IntFire4** class
- ELECTRODE_CURRENT 218
 - effect on extracellular mechanism 218
- GLOBAL 190, 211
 - also see **ASSIGNED** variable: **GLOBAL**, **PARAMETER** variable: is **GLOBAL** by default
- NONSPECIFIC_CURRENT 211
 - equilibrium potential 223
 - also see **active transport: pump current: countering with a NONSPECIFIC_CURRENT**
- POINT_PROCESS 215
 - also see **point process**
- POINTER 268
 - also see **POINTER** variable
- RANGE 190, 211, 218
 - also see **range, range variable**, **ASSIGNED** variable: **GLOBAL**: vs. **RANGE**, **ASSIGNED** variable: is a **range** variable by default, **PARAMETER** variable: **GLOBAL** vs. **RANGE**, **STATE** variable: is automatically **RANGE**
- SUFFIX 211
 - also see **distributed mechanism**
- USEION 222, 230
 - effect on initialization sequence 186
- READ ex (reading an equilibrium potential) 222
- READ ix (reading an ionic current) 235, 250
- READ xi (reading an intracellular concentration) 230, 250
- READ xo (reading an extracellular concentration) 255
- WRITE ix (writing an ionic current) 222, 230, 233, 255, 259
- WRITE xi (writing an intracellular concentration) 193, 250
- WRITE xo (writing an extracellular concentration) 193, 235

- NEURON demonstration program – see *neurondemo*
- NEURON Main Menu 6
 - adding items – see *NEURONMainMenu* class: *miscellaneous_add()*
 - creating 142, 155
- NEURON Main Menu GUI
 - Build
 - CellBuilder 6
 - NetWork Builder 313
 - File
 - load session 18
 - Quit 347
 - save session 17, 23
 - Graph
 - Current axis 180
 - Phase Plane 180
 - Shape plot 22
 - State axis 180
 - Voltage axis 21, 180
 - Tools
 - Miscellaneous 384
 - Point Processes 19
 - also see *PointProcessGroupManager*, *PointProcessManager*
 - RunControl 24
 - VariableStepControl 165, 319
 - also see *VariableTimeStep* GUI
- NEURON MODEL Description Language – see *NMODL*
- NEURON program group 6
- neuron.exe* 345
- neurondemo* 345
- NEURONMainMenu* class
 - miscellaneous_add()* 384
- NEURONMainMenu* object
 - is always *NEURONMainMenu[0]* 384
- NEURON's interpreter – see *hoc*
- new* – see object: *new*
- NMODL 113
 - abrupt change of a variable – see variable:
 - abrupt change of, *NET_RECEIVE* block: handling abrupt changes and discontinuities
 - arrays
 - are not dynamic 250, 255
 - index starts at 0 250
 - STATE variable – see STATE variable:
 - array in NMODL
 - COMMENT . . . ENDCOMMENT – see NMODL: comments
 - comments 209
 - declaring variables 211
 - specifying units 212
 - also see NMODL: named blocks:
 - variable declaration, ASSIGNED block, CONSTANT block, PARAMETER block, STATE block, LOCAL variable, NMODL: DEFINE
 - DEFINE 250
 - dt* – see *dt*: use in NMODL
 - FROM . . . TO . . . (loop statement) 252
 - function – see FUNCTION block
 - FUNCTION_TABLE 244
 - loop statement – see NMODL: FROM . . . TO . . . (loop statement)
 - mknrndll* – see NMODL: translator:
 - mknrndll*
 - named blocks 208
 - ASSIGNED – see ASSIGNED block
 - BREAKPOINT – see BREAKPOINT block
 - CONSTANT – see CONSTANT block
 - DERIVATIVE – see DERIVATIVE block
 - equation definition 208
 - also see BREAKPOINT block, DERIVATIVE block, FUNCTION block, INITIAL block, KINETIC block, PROCEDURE block
 - FUNCTION – see FUNCTION block
 - general form 210
 - INITIAL – see INITIAL block
 - KINETIC – see KINETIC block
 - NEURON – see NEURON block
 - NONLINEAR – see NONLINEAR block
 - PARAMETER – see PARAMETER block
 - PROCEDURE – see PROCEDURE block
 - variable declaration 208
 - also see ASSIGNED block, PARAMETER block, STATE block, LOCAL variable
 - nocmodl* – see NMODL: translator:
 - nocmodl*
 - nrnivmodl* – see NMODL: translator:
 - nrnivmodl*
 - procedure – see PROCEDURE block
 - t* – see *t*: use in NMODL
 - translator 207, 209, 239–240, 244
 - mknrndll* 186, 191, 346
 - also see *nrnmecch.dll*
 - nmodl* 211
 - nocmodl* 192, 211, 346
 - nocmodl.exe* 211
 - nrnivmodl* 186, 191, 345

- units conversion factor 216, 231, 237, 250–251, 259
 - parentheses 216–217
 - also see UNITS block: units scaling, scale factor
- UNITSOFF . . . UNITSON 227
- user-defined variable 210
- VERBATIM . . . ENDVERBATIM 209
- vs. MODL – see MODL: vs. NMODL
 - also see mod file
- nmodl – see NMODL: translator: nmodl
- nocmodl – see NMODL: translator:
 - nocmodl
- nocmodl.exe – see NMODL: translator:
 - nocmodl.exe
- node
 - circuit – see circuit: node
 - equation – see equation: current balance, Kirchhoff's current law
 - section – see section: nodes
- “nonbiological” model – see artificial spiking cell, integrate and fire
- NONLINEAR block 223
 - dependent variable
 - is a STATE variable 184
- nonlinearity – see system: nonlinear, channel: nonlinear
- noninteractive simulations – see GUI: vs. hoc
- NONSPECIFIC_CURRENT – see NEURON block: NONSPECIFIC_CURRENT
- normalized distance along a section – see range
- NOT operator – see hoc syntax: expressions: operators
- notation
 - chemical reaction – see kinetic scheme
 - circuit – see circuit
- nrngui 6, 133
 - loads GUI and standard run library 142
- nrniv 133, 345
 - adding new mechanisms 345
 - also see mechanisms: user-defined, NMODL
 - also see nrngui
- nrniv.exe 345
 - also see nrniv, neuron.exe
- nrnivmodl – see NMODL: translator:
 - nrnivmodl
- nrnmec.dll 346
 - also see mknrndll
- nrnunits.lib 212, 231
 - also see units: database
- nseg 95, 132
 - effect on spatial accuracy and resolution 96
 - effect on range variable – see range
 - variable: effect of changing nseg,
 - range variable: inhomogeneous:
 - reassert after changing nseg
 - may reposition internally attached sections
 - and point processes 102, 112
 - vs. number of 3-D points 106
 - why triple nseg? 97
 - why use odd values? 97
 - also see spatial grid, discretization, segment
- nstep_steprun – see standard run system:
 - setdt(), RunControl GUI: Points plotted/ms
- NULLObject – see object: NULLObject
- numarg() – see funcs and procs:
 - arguments: numarg()
- numeric integration 28, 56, 62, 86
 - accuracy – see accuracy, numeric
 - integration: order of accuracy
- adaptive 72, 87, 224, 226, 236, 243, 260–261, 263
 - advance microstep 173
 - and events – see event: times: with
 - adaptive integration, standard run system: event delivery system:
 - adaptive integration and, standard run system: fadvance(): global time step integration, standard run system: fadvance(): local time step integration
 - and the standard run system – see
 - standard run system: event delivery system: adaptive integration and, standard run system: fadvance(): global time step integration, standard run system: fadvance(): local time step integration
- error control – see numerical error: control
- global time step 81–82, 88
 - also see standard run system: fadvance(): global time step integration, VariableTimeStep GUI–global vs. local time steps
- global variable time step – see numeric
 - integration: adaptive: global time step
- GUI control – see VariableTimeStep GUI
- initialization 187
 - also see CNode class: re_init(), initialization
- initialize microstep 173
- interpolate microstep 173

- numeric integration (*contd*)
 - interpolation formulas 174
 - local time step 74, 80–82, 88, 166
 - also see standard run system: `fadvance()`: local time step integration, standard run system:
 - event time queue, standard run system: cell time queue, CVode class: `record()`, `VariableTimeStep` GUI–global vs. local time steps
 - local variable time step – see numeric integration: adaptive: local time step
 - microstep – see numeric integration:
 - adaptive: advance microstep, numeric integration: adaptive: initialize microstep, numeric integration: adaptive: interpolate microstep
 - switching to fixed time step 88
 - toggling on and off – see `VariableTimeStep` GUI: toggling adaptive integration on and off
 - with discrete events – see standard run system: event delivery system:
 - adaptive integration and, standard run system: `fadvance()`: global time step integration, standard run system: `fadvance()`: local time step integration, event: times: with adaptive integration
 - also see CVODE, CVODES, DASP, IDA, SUNDIALS, standard run system: `fadvance()`: global time step integration, standard run system: `fadvance()`: local time step integration
- analytic integration of channel states 72, 165, 170
- central difference method – see Crank–Nicholson method
- error – see accuracy, numerical error, numeric integration: order of accuracy
- explicit 63, 86, 243
 - also see forward Euler method
- fixed time step 75, 86, 225, 236, 263
 - event aggregation 81, 166, 171, 218, 260
 - also see event: times: with adaptive integration
- initialization 186
 - also see initialization
- switching to adaptive 88
 - also see backward Euler method, forward Euler method, Crank–Nicholson method, standard run system: `fadvance()`: fixed time step
- implicit 63, 86–87, 243
 - also see backward Euler method, Crank–Nicholson method
- initialization – see numeric integration:
 - adaptive: initialization, numeric integration: fixed time step: initialization, initialization
- instability 62, 64
 - also see numeric integration: stability
- iteration of nonlinear equations 70, 87
- NEURON's default method – see backward Euler method
- order of accuracy 71, 75, 83
 - first 226, 243
 - also see backward Euler method, forward Euler method
 - second 225, 246
 - also see Crank–Nicholson method, spatial accuracy: second order
- variable 226
 - also see numeric integration: adaptive, CVODE, DASP
- precision – see numeric integration: order of accuracy
- stability 74, 86, 91
 - effect of signal sources 66, 86
 - also see system equations: effect of signal sources
 - also see numeric integration: instability, system: stiff, backward Euler method, forward Euler method, Crank–Nicholson method
- summary 86
- numeric solution – see numeric integration
- numerical error 83
 - absolute – see absolute error
 - caused by changing `nseg` – see `nseg`: may reposition internally attached sections and point processes
- chaotic system 84
- control 62, 79
 - also see absolute error: local: tolerance, relative error: local: tolerance
- criterion – see absolute error: local: tolerance, relative error: local: tolerance
- cumulative – see numerical error: global
- global 76, 83

integrated 166
 also see numerical error: global
 local 64, 68, 75, 83, 171
 also see absolute error: local, relative
 error: local
 message – see error message, hoc: error
 handling
 oscillations 67, 70
 programming – see hoc: error handling
 relative – see relative error
 roundoff 65, 97
 spatial 56, 60, 96
 also see spatial accuracy
 temporal 56
 effect of spatial discretization 60, 119
 also see temporal accuracy
 tolerance – see absolute error: local:
 tolerance, relative error: local: tolerance
 total – see numerical error: global
 also see parameters: sensitivity to, accuracy:
 physiological, accuracy: qualitative
 numerical oscillations – see numerical error:
 oscillations, numeric integration:
 instability
 Nyquist sampling theorem 59

O

object 363
 array 372
 as an argument – see funcs and procs:
 arguments: objects and objrefs
 calling a method – see object: public
 members: accessing from hoc
 collection – see object: array, List class
 creating 365
 destroying 365
 dot notation – see object: public members:
 dot notation
 incorporating into plotting system – see
 standard run system: plotting system:
 incorporating Graphs and objects
 initializing variables – see template:
 variable initialization
 list – see List class
 methods 366, 368
 calling – see object: public members:
 accessing from hoc
 name
 how generated 370
 vs. object reference 370
 new 365
 notifying at every step – see standard run
 system: plotting system: notifying
 Graphs and objects

NULLObject 365, 371, 372
 using the NULLObject 372
 objref – see object reference: objref
 passing to a func or proc – see funcs
 and procs: arguments: objects and
 objrefs
 private vs. public – see object: public
 members: vs. private members,
 template: writing a template: public
 public members
 accessing from hoc 366
 dot notation 366
 vs. private members 366
 also see template: writing a template:
 public
 reference – see object reference, object:
 reference count
 reference count 366, 370, 383
 set – see object: array, List class
 specifying attributes – see template: writing,
 object: public members: dot notation
 stack – see List class: object stack
 state 363, 366
 this – see object reference: this
 vs. class 363
 vs. object reference 364
 also see object reference, class, template,
 object-oriented programming
 object name – see object: name
 object name vs. object reference – see object
 reference: vs. object name
 object reference 112, 364
 as an argument – see funcs and procs:
 arguments: objects and objrefs
 cannot be redefined as scalar, double, or
 string 365
 count – see object: reference count
 declaring 364
 objectvar 364
 also see object reference: cannot redefine
 as scalar, double, or string
 objref 112, 365
 also see object reference: cannot redefine
 as scalar, double, or string
 passing to a func or proc – see funcs
 and procs: arguments: objects and
 objrefs
 points to an object 364, 366
 this 383
 vs. object 364
 vs. object name 370
 object variable – see object reference:
 objectvar

- `object_pop()` – see session file: `object_pop()`
- `object_push()` – see session file: `object_push()`
- object-oriented programming
 - encapsulating code 375
 - information hiding 363
 - inheritance 327, 363, 376
 - also see class: base class, class: subclass
 - polymorphism 181, 363, 376
 - also see class: base class, class: subclass
 - also see object, object reference, class, template
- `objectvar` – see object reference: `objectvar`
- `objref` – see object reference: `objref`
- `oc` 346
 - also see `nrniv`
- `ocbox_` – see session file: `ocbox_`
- Ohm's law 52
- online Programmer's Reference – see Programmer's Reference
- operators – see hoc syntax: expressions: operators
- operator precedence – see hoc syntax: expressions: operators
- optimization 30
- OR operator – see hoc syntax: expressions: operators
- ordinary differential equation – see equation: differential: ordinary
- oscillating system – see initialization: categories: to a desired state
- oscillations – see numerical error: oscillations
- oscilloscope 23
- oxymoron 306
- P**
- `panel` – see `xpanel()`
- PARAMETER block 212
 - assigning default PARAMETER values 212
 - default value of `state0` 190
 - also see STATE variable: `state0`, STATE block: START
 - specifying minimum and maximum limits 212
- PARAMETER variable 184, 212
 - abrupt change of – see variable: abrupt change of, NET_RECEIVE block: handling abrupt changes and discontinuities
 - default value – see PARAMETER block: assigning default PARAMETER values
 - GLOBAL vs. RANGE 212, 235, 256
 - also see NEURON block: RANGE
 - is GLOBAL by default 254
 - also see NEURON block: GLOBAL
 - RANGE 218
 - specifying minimum and maximum limits – see PARAMETER block: specifying minimum and maximum limits
 - time-dependent 161, 263
 - visibility at the hoc level 212
 - when to use for an equilibrium potential 223
 - also see ASSIGNED variable, STATE variable, `ion_style()`
- parameters
 - biophysical 15–16
 - geometric 13–14
 - sensitivity to 84
 - also see PARAMETER variable
- parent section – see section: parent
- parentheses – see hoc syntax: expressions: operators, NMODL: units conversion factor: parentheses, UNITS block: units scaling
- parse errors – see hoc: error handling
- partial differential equation – see equation: differential: partial
- `pas` mechanism 15
 - `e_pas` 16
 - `g_pas` 16
- passive cylindrical cable – see cable: passive cylindrical
- passive leak current – see Example 9.1: a passive “leak” current
- PFWM 16
 - is implemented in C 344
 - saving and retrieving session files – see session file: saving: from PFWM, session file: loading: from PFWM
- physical distance – see distance
- physical system 33
 - representing by a model 33–34
- physiological accuracy – see accuracy: physiological
- piecewise linear approximation – see range variable: estimating by linear interpolation between nodes
- piecewise linear function – see function: piecewise linear
- Pike, R. – see hoc: Kernighan and Pike
- plain text file 130

- plasticity
 - synaptic – see synaptic plasticity
- Plot what? GUI 228, 233
 - also see variable browser
- Plot () – see standard run system: Plot ()
- plot lists – see standard run system: plotting
 - system: fast_flush_list, standard run system: plotting system: flush_list, standard run system: plotting system: graphLists
- plotting spike trains – see spike trains: recording and plotting, NetWork Builder: buttons: SpikePlot
- plotting system – see standard run system: plotting system
- point process 18, 112, 214, 217
 - adding new – see mechanisms: user-defined, NMODL
 - attaching – see point process: inserting
 - changing location with hoc code – see point process: loc ()
 - creating 112
 - destroying 112
 - effect of nseg on location 112
 - inserting 112
 - loc () 113
 - preserving spatial accuracy 97
 - specifying attributes 112
 - vs. distributed mechanism 113
 - also see NEURON block:
 - POINT_PROCESS, artificial spiking cell, NEURON block: ARTIFICIAL_CELL
- Point Process Viewer GUI 217
- POINTER – see POINTER variable, NEURON block: POINTER, setpointer
- pointer – see hoc syntax: pointer operator, NEURON block: POINTER, POINTER variable
- pointer operator – see hoc syntax: pointer operator
- Pointer class 359
- POINTER variable 268
 - effect on Jacobian – see Jacobian: approximate
 - also see NEURON block: POINTER, setpointer, variable: local vs. nonlocal
- PointProcessGroupManager
 - bringing up a PointProcessGroupManager 319
- PointProcessManager 18
 - configuring as
 - AlphaSynapse 19
 - creating 19, 148
 - location 20, 27
 - changing 27
 - parameters 20
- PointProcessManager GUI
 - SelectPointProcess 19
 - Show
 - Shape 27
- Points plotted/ms – see RunControl GUI: Points plotted/ms
- polymorphism – see object-oriented programming: polymorphism
- potassium
 - concentration – see Example 9.6: extracellular potassium accumulation
 - current – see Example 9.4: a voltage-gated current, Example 9.5: a calcium-activated, voltage-gated current
 - also see ion mechanism
- potential
 - reversal – see equilibrium potential
 - also see membrane potential, voltage
- precedence of operations – see hoc syntax: expressions: operators
- prediction 33
- print – see hoc syntax: basic input and output: print
- Print & File Window Manager – see PFWM
- printf () – see hoc syntax: basic input and output: printf ()
- private vs. public – see object: public members: vs. private members, template: writing a template: public
- proc – see funcs and procs
- PROCEDURE
 - calling from hoc – see hoc: calling an NMODL FUNCTION or PROCEDURE
 - also see PROCEDURE block, BREAKPOINT block: and PROCEDURES
- procedure – see funcs and procs, PROCEDURE
- PROCEDURE block 232
 - arguments are call by value 283
- program organization – see good programming style: program organization
- Programmer's Reference 343
- programming – see hoc
 - good practices – see good programming style
 - mistake – see hoc: error handling
- project management 154
 - also see good programming style

proper initialization – see initialization:
 criterion for proper initialization
`psection()` 103
`pt3d` – see 3-D specification of geometry
`pt3d` data – see 3-D specification of geometry:
 3-D information
`pt3d` list – see 3-D specification of geometry:
 3-D information
`pt3dadd()` – see 3-D specification of
 geometry: `pt3dadd()`
`public` – see object: public members,
 template: writing a template: `public`
 pump
 calcium – see calcium: pump
 also see active transport
 purpose of computing – see insight
 purpose of the model – see user's intent
`push_section()` 100

Q

qualitative results 67
 also see accuracy, judgment
 quantitative morphometric data 98, 105, 126,
 144
 bad diameter values – see diameter: zero or
 narrow diameter
 also see 3-D specification of geometry
 queue
 cell time – see standard run system: event
 delivery system: cell time queue
 event time – see standard run system: event
 delivery system: event time queue
 Quiet – see RunControl GUI: Quiet
`quit()` – see hoc syntax: flow control:
 `quit()`
 quitting NEURON – see NEURON: exiting,
 hoc syntax: flow control: `quit()`

R

`Ra` 13, 15, 94
 default value 16, 103
 also see cytoplasmic resistivity
 radial diffusion – see diffusion: radial
 ramp clamp – see voltage clamp: ramp clamp
 Random class
 `play()`
 initialization – see initialization:
 `Random.play()`
 random number generator
 initialization – see initialization: random
 number generator

RANGE – see NEURON block: RANGE, range,
 range variable
 range 93
 RANGE variable – see range variable
 range variable 93
 dot notation – see section: currently
 accessed: dot notation
 effect of changing `nseg` 115–117
 estimating by linear interpolation between
 nodes 97
 inhomogeneous
 reassert after changing `nseg` 116
 in which section? – see section: currently
 accessed
 iterating over nodes 114
 linear taper 115
 linear variation along a section – see range
 variable: linear taper
 `rangevar(x)` returns value at nearest
 internal node 94
 also see range, NEURON block: RANGE,
 ASSIGNED variable: is a range
 variable by default, PARAMETER
 variable: RANGE, STATE variable: is
 automatically RANGE
 raster plot – see spike trains: recording and
 plotting, NetWork Builder: buttons:
 SpikePlot
 rate constant 37
 rate functions – see BREAKPOINT block: and
 rate functions, KINETIC block: reaction
 rates: voltage-sensitive
`re_init()` – see CCode class: `re_init()`
 reactants – see KINETIC block: reactants
 reaction – see kinetic scheme
 chemical – see kinetic scheme
 rates – see KINETIC block: reaction rates
 reactants – see KINETIC block: reactants
 scheme – see kinetic scheme, KINETIC
 block
 statement – see KINETIC block: reaction
 statement
 also see KINETIC block: `<->` (reaction
 indicator)
 READ – see NEURON block: USEION
`read()` – see hoc syntax: basic input and
 output: `read()`
 reading an ion concentration – see NEURON
 block: USEION
 reading and writing files – see hoc syntax:
 basic input and output

- real model neuron – see oxymoron
- realtime – see standard run system:
 - realtime
- recording – see CNode class: `record()`,
 NetCon class: `record()`, Vector
 class: `record()`
- recording spike trains – see NetCon class:
 - `record()`, spike trains: recording and
 plotting, NetWork Builder: buttons:
 SpikePlot
- recursion – see `funcs` and `procs`: recursion
 redefining standard functions and
 procedures – see standard run library:
 redefining functions and procedures,
 standard GUI library: redefining
 functions and procedures
- reference count – see object: reference
 count
- refractory period – see IntFire1 class:
 - refractory period
- relative error
 - local 75, 79
 - tolerance 75
 - also see absolute error: local, numerical
 error: local
 - also see absolute error
- relative local error – see relative error: local
- relative tolerance – see relative error: local:
 tolerance
- remainder operator – see hoc syntax:
 - expressions: operators
- resistance
 - axial – see axial resistance
 - also see `ri()`, R_a , cytoplasmic resistivity
 - electrode – see electrode: resistance
 - membrane – see specific membrane
 resistance
- resistivity – see cytoplasmic resistivity, R_a
- resistor – see circuit: element: resistor
- restore() – see SaveState class:
 - restore()
- restricted diffusion – see Example 9.6:
 - extracellular potassium accumulation
- return – see hoc syntax: flow control:
 - return
 - also see `funcs` and `procs`: return
- returned value – see `funcs` and `procs`:
 - return
- reusable code – see good programming style:
 - exploiting reusable code
- reversal potential – see equilibrium potential
- `ri()`
 - calculation of – see stylized specification of
 geometry: calculation of `area()` and
`ri()`, 3-D specification of geometry:
 calculation of L , $diam$, `area()`, and
`ri()`
 - effect of creating a Shape – see Shape
 object: creating: effect on $diam$,
`area()`, and `ri()`
 - effect of `define_shape()` – see
`define_shape()`: effect on $diam$,
`area()`, and `ri()`
 - infinite 107, 115
 - also see diameter: zero or narrow
 diameter
 - also see axial resistance
- rise time 122
- R_m – see specific membrane resistance
- root section – see section: root section
- `ropen()` – see hoc syntax: basic input and
 output: `ropen()`
- roundoff error – see numerical error:
 - roundoff
- run control – see RunControl GUI, simulation
 control, standard run system
- run time 98, 158, 171
 - also see RunControl GUI: Real Time,
 standard run system: realtime,
 initialization: `startsw()`
- run time system – see standard run system
- run-time errors – see hoc: error handling
- `run()` – see standard run system: `run()`
- RunControl 21
 - creating 24, 151, 157
- RunControl GUI
 - Continue for 158–159, 162
 - also see standard run system:
 continuerun()
 - Continue til 158–159, 162
 - also see standard run system:
 continuerun()
- `dt` 24, 158
 - also see RunControl GUI: Points
 plotted/ms, standard run system:
`setdt()`
- Init 24, 158–159
- Init & Run 24, 152, 158–159, 163
 - also see standard run system: `run()`
- Points plotted/ms 24, 158–159, 162
 - also see standard run system: `setdt()`,
 standard run system: `step()`

- RunControl GUI (*contd*)
 Quiet 158–159, 162
 also see standard run system:
 stdrun_quiet
 Real Time 158, 163
 also see standard run system:
 continuerun(), standard run
 system: realtime
 Single Step 158–159, 162
 also see standard run system: step()
 Stop 158, 162–163
 also see standard run system: stoprun
 t 24, 158
 Tstop 24, 152, 158, 163
 also see standard run system: tstop
 Runge-Kutta method
 stability 243
 running a simulation 26
 runtime
 error – see hoc: error handling
 system – see standard run system
 also see run time
- S**
 sacred runes – see equation: sacred runes
 sampling theorem – see Nyquist sampling
 theorem
 SaveState class
 fread() 198
 fwrite() 198
 restore() 199
 save() 198
 also see initialization: categories: to a
 desired state
 scalar – see hoc syntax: variables: scalars
 scale factor 43
 also see NMODL: units conversion factor,
 UNITS block: units scaling
 scene – see GUI: scene
 scene coordinates – see GUI: scene coordinates
 scene coordinates vs. screen coordinates – see
 GUI: scene coordinates: vs. screen
 coordinates
 screen – see GUI: screen
 screen coordinates – see GUI: screen
 coordinates
 SCoP 208, 213
 scope – see hoc syntax: names, funcs and
 procs: local variable, variable: local vs.
 nonlocal, LOCAL variable, ASSIGNED
 variable: GLOBAL, NEURON block:
 GLOBAL, PARAMETER variable: is
 GLOBAL by default
 also see object: public members, template:
 writing a template: public, template:
 writing a template: external
 sealed end – see boundary conditions: sealed
 end
 SEClamp
 preserving spatial accuracy – see point
 process: preserving spatial accuracy
 also see voltage clamp
 secname() 108
 second messenger – see Example 10.5: use-
 dependent synaptic plasticity
 also see calcium
 secondorder 86, 169, 180
 also see Crank–Nicholson method,
 backward Euler method
 section 8, 92
 access – see access, section: currently
 accessed: default section
 arc length – see range
 area – see segment: surface area
 array 103
 as argument or variable – see
 SectionRef class
 attaching – see connect
 child 101
 connect 0 end to parent 105, 148
 connecting – see connect
 creating – see create
 currently accessed
 default section 21, 23, 100, 143–144
 dot notation 94–95, 99
 section stack 99
 daughter – see section: child
 default parameter values – see cm, diam,
 L, Ra
 default section – see section: currently
 accessed: default section
 vs. root section – see section: root
 section: vs. default section
 detaching – see disconnect()
 diam – see diam
 diameter – see diameter, diam, 3-D
 specification of geometry: diam3d()
 disconnecting – see disconnect()
 equivalent circuit 106–107
 iterating over nodes – see range variable:
 iterating over nodes
 iterating over sections 108, 123
 L – see L
 length – see L
 nodes 96

- internal vs. terminal 95
- iterating over – see range variable:
 - iterating over nodes
- locations 95
 - also see nseg: why use odd values?
 - zero area 74, 97
- normalized distance along a section – see range
- nseg – see nseg
- orientation 138, 145, 147
- parent 101
- properties
 - specifying – see section: currently accessed, 3-D specification of geometry, stylized
 - specification of geometry, biophysical
 - properties: specifying
- Ra – see Ra
- root section 8, 101, 134
 - is 3-D origin of cell 146, 149
 - vs. default section 102, 134
- sets – see SectionList class
- stack – see section: currently accessed:
 - section stack
- variable – see section variable
- section variable 93
 - also see L, nseg, Ra
- section.h 169
- SectionList class 138
- SectionRef class 100
- segment 95
 - diameter – see diam
 - surface area – see area, area ()
 - also see compartment, nseg, section: nodes
- self-event – see event: self-event
- sensitivity – see parameters: sensitivity to
- separating biology from numerical issues 92
 - also see section, range, range variable
- ses file – see session file
- session file 16
 - loading
 - from NEURON Main Menu 18
 - from PFWM 16
 - object_pop () 388
 - object_push () 388
 - ocbox_ 388
 - also see VBox class: map ()
 - saving
 - from NEURON Main Menu 17
 - from PFWM 16
- set
 - of numbers – see hoc syntax: variables:
 - double, Vector class
 - of objects – see object: array, List class
 - of sections – see SectionList class,
 - CellBuilder GUI: Subsets page
 - setdata_ – see hoc: calling an NMODL FUNCTION or PROCEDURE: specifying proper instance with setdata_
 - setdt () – see standard run system:
 - setdt ()
 - setpointer 269
 - also see POINTER variable
- Shape object
 - creating
 - effect on diam, area (), and ri () 108
- Shape plot 9, 103
 - creating 22, 145
 - effect on diam, area (), and ri () 108
- Shape Plot 180
- Shape plot GUI
 - primary menu
 - Space Plot 22
 - also see Space Plot
 - Shape Style
 - Show Diam 145
- shared object – see NMODL: translator:
 - nrnivmodl, nrniv
- shell 41
- Shift key – see Graph class: menu_tool ()
- shunt.mod – see Example 9.2: a localized shunt
- sign convention – see membrane current:
 - positive current convention, axial current:
 - positive current convention, circuit:
 - positive current convention
- signal
 - chemical 50, 119
 - electrical 50, 119
- signal monitors 20
 - vs. signal sources 20
- signal sources 18
 - effect on system equations – see system equations: effect of signal sources, numeric integration: stability: effect of signal sources
 - load – see system equations: effect of signal sources, numeric integration: stability: effect of signal sources
 - also see distributed mechanism, point process
- simplification 33–35
- simulation
 - control – see simulation control
 - event-driven – see discrete event simulation

- simulation (*contd*)
 - interactive vs. noninteractive – see GUI: vs. hoc
 - time 24
 - also see *t*, elapsed simulation time
 - simulation control 5, 21, 139, 154
 - running 26
 - starting 26
 - stopping – see standard run system:
 - tstop*, RunControl GUI: *Tstop*, RunControl GUI: *Stop*, hoc: interrupting execution
 - also see standard run system, good programming style: modular programming, good programming style: program organization
- Simulation Control Program – see SCoP
- simulation environment
 - utility of 32, 34
- sink reaction – see KINETIC block: $\rightarrow$ (sink reaction indicator)
- size – see compartment: size, discretization, nseg
- slope conductance – see conductance: slope
- sodium – see ion mechanism
- solution
 - analytic – see analytic solution
 - computational – see numeric integration
 - numeric – see numeric integration
- SOLVE – see BREAKPOINT block: SOLVE, INITIAL block: SOLVE: STEADYSTATE sparse
- solve.c* 169
- source
 - current – see current: source, circuit:
 - element: current source
 - NetCon class: source variable
 - signal – see signal sources
 - voltage – see circuit: element: voltage source
- space 21–22, 25, 32
- space constant – see λ , length constant, *d_lambda* rule
- space plot 20
- Space Plot 180
 - creating 22
- sparse – see KINETIC block, BREAKPOINT block: SOLVE: sparse, INITIAL block: SOLVE: STEADYSTATE sparse
- spatial accuracy 96
 - checking 97
 - second order 57, 96
 - preserving 97, 102
 - also see discretization: spatial, nseg: effect on spatial accuracy and resolution
- spatial decay of fast signals 122
- spatial discretization – see discretization: spatial, spatial grid
- spatial error – see numerical error: spatial, spatial accuracy
- spatial frequency – see frequency: spatial
- spatial grid 57
 - checking 97
 - choosing – see discretization: guidelines, *d_lambda* rule, *d_X* rule, CellBuilder GUI: Geometry page: specifying strategy
 - also see discretization: spatial
- spatial resolution – see discretization: spatial, numerical error: spatial, spatial accuracy, spatial grid
- specific membrane capacitance 53, 56, 91–92, 94
 - also see *cm*
- specific membrane conductance 56
- specific membrane resistance 122
 - also see specific membrane conductance
- specifying geometry – see 3-D specification of geometry, stylized specification of geometry
 - also see geometry, anatomical properties
- specifying model properties – see model properties: specifying
- specifying object attributes – see template: writing, object: public members: dot notation
- specifying section properties – see section: currently accessed, 3-D specification of geometry, stylized specification of geometry, biophysical properties: specifying
- specifying the current section – see section: currently accessed
- specifying the spatial grid – see discretization: spatial grid, nseg, spatial grid
- specifying topology – see topology: specifying
- speed – see computational efficiency
- speed vs. accuracy – see accuracy: vs. speed
- spike event – see event: external
- spike trains
 - recording and plotting 335
 - also see NetCon class: *record()*, NetWork Builder: buttons: SpikePlot
- SpikePlot – see NetWork Builder: buttons: SpikePlot

- sprint () – see hoc syntax: basic input and output: sprint ()
- squid axon 28
 - also see hh mechanism
- stability – see numeric integration: stability
- stack of objects – see List class: object stack
- staggered time steps – see Crank–Nicholson
 - method: staggered time steps
- standard GUI library
 - changing functions and procedures – see standard GUI library: redefining functions and procedures
 - hoc source accompanies NEURON 344
 - loading – see nrngui, hoc: idiom: load_file ("nrngui.hoc")
 - not loading – see nrniv
 - redefining functions and procedures 344
- standard run library
 - changing functions and procedures – see standard run library: redefining functions and procedures
 - hoc source accompanies NEURON 344
 - redefining functions and procedures 344
- standard run system 152
 - addplot () 152, 181
 - advance () 160, 181
 - continuerun () 160, 162–163, 181
 - CVODE 164
 - DASPK 165
 - discrete events – see standard run system:
 - fadvance () : global time step integration, standard run system: fadvance () : local time step integration
 - doEvents () 163
 - event delivery system 165
 - adaptive integration and 171, 179
 - and models with discontinuities – see variable: abrupt change of, event: times
 - cell time queue 173
 - also see numeric integration: adaptive:
 - local time step, standard run system: fadvance () : global time step integration, standard run system: fadvance () : local time step integration
 - event time queue 173, 292
 - also see numeric integration: adaptive:
 - local time step, standard run system: fadvance () : global time step integration, standard run system: fadvance () : local time step integration
 - event times – see event: times, standard run system: event time queue
 - external event – see event: external
 - implementing deferred computation 289, 296, 302
 - also see event: self-event
 - initialization 183, 185–186
 - input event – see event: external
 - self event – see event: self-event
 - also see event, standard run system:
 - fadvance () : global time step integration, standard run system: fadvance () : local time step integration
- fadvance () 81, 158, 160, 164, 184, 243, 263
 - discrete events – see standard run system:
 - fadvance () : global time step integration, standard run system: fadvance () : local time step integration
 - fixed time step 165
 - global time step integration 179
 - local time step integration 166, 173
- fast_flushPlot () 181
- fcurrent () 164, 170, 186
 - in initialization 188–189, 196
- finitialize () – see initialization:
 - finitialize ()
- flushPlot () 162, 180–181
- graph lists – see standard run system:
 - plotting system: graphLists
- Graphs and objects
 - incorporating – see standard run system:
 - plotting system: incorporating Graphs and objects
 - notifying – see standard run system:
 - plotting system: notifying Graphs and objects
- init () – see initialization: init ()
- initialization – see initialization, numeric
 - integration: adaptive: initialize microstep
- initPlot () – see initialization:
 - initPlot ()
- is implemented in hoc 344
- NetCon – see NetCon: and standard run system
- nstep_steprun – see standard run system: setdt (), RunControl GUI: Points plotted/ms

- standard run system (*contd*)
 - Plot() 162, 180
 - also see standard run system: step
 - plot lists – see standard run system: plotting
 - system: fast_flush_list,
 - standard run system: plotting system: flush_list, standard run system: plotting system: graphLists
 - plotting system 179
 - fast_flush_list 180
 - flush_list 180
 - graphLists 179
 - incorporating Graphs and objects 181
 - also see standard run system: plotting
 - system: graphLists, standard run system: addplot()
 - notifying Graphs and objects 179
 - also see standard run system: plotting
 - system: graphLists
 - special uses 162
 - realtime 163–164, 187
 - run() 159–160, 163, 179
 - setdt() 161, 164, 187
 - stdinit() – see initialization: stdinit()
 - stdrun_quiet 163, 180
 - step 158, 162
 - step() 160–162
 - under CVOICE 162
 - steprun() 160, 162
 - steps_per_ms – see standard run system: setdt(), standard run system: step, RunControl GUI: Points plotted/ms
 - stoprun 162–163
 - tstop 152, 163, 181
 - also see RunControl GUI, standard run library
- START – see STATE block: START, STATE variable: initialization
- starting hoc – see hoc: starting and exiting, NEURON: starting and exiting
- starting NEURON – see NEURON: starting and exiting, hoc: starting and exiting
- startsw() – see initialization: startsw()
- state 36, 38, 41
 - as amount of material 36
 - as concentration 36
 - as density 36
 - as probability 36
 - restoring – see SaveState class: restore(), initialization: categories: to a desired state
 - saving – see SaveState class: save()
- STATE block 223
 - specifying local absolute error tolerance 251
- START 190
 - also see STATE variable: initialization, PARAMETER block: default value of state0
 - also see ASSIGNED block, PARAMETER block
- state variable 67, 70, 73
 - as an ASSIGNED variable 185
 - changing after finitalize() – see initialization: strategies: changing a state variable
 - custom initialization – see initialization: strategies: changing a state variable
 - initialization – see STATE variable: initialization, initialization
 - of a mechanism vs. state variable of a model 213
 - vs. STATE variable – see STATE variable: vs. state variable
- STATE variable 168, 184, 223
 - abrupt change of – see variable: abrupt change of, NetCon class: event(), NET_RECEIVE block: handling abrupt changes and discontinuities
 - and COMPARTMENT statement – see KINETIC block: COMPARTMENT
 - array in NMODL 244
 - conservation – see KINETIC block: CONSERVE
 - initialization 224
 - default vs. explicit 190
 - state0 190, 237
 - also see PARAMETER block: default value of state0, STATE block: START
 - also see initialization, KINETIC block: CONSERVE
 - ion concentration as 236
 - is automatically RANGE 223
 - also see NEURON block: RANGE
 - specifying local absolute error tolerance – see STATE block: specifying local absolute error tolerance
 - vs. state variable 184, 242
 - also see ASSIGNED variable, PARAMETER variable, ion_style(), state variable

- `state_discontinuity()` – see
 CVODE: and model descriptions:
 `state_discontinuity()`, variable:
 abrupt change of, NET_RECEIVE block:
 `state_discontinuity()`
- `state0` – see STATE variable: initialization,
 INITIAL block
- `stdgui.hoc` – see standard GUI library
- `stdinit()` – see initialization:
 `stdinit()`
- `stdlib.hoc` 123, 357, 373
 also see standard run system
- `stdrun.hoc` 159
 also see standard run system, standard run
 library
- `stdrun_quiet` – see standard run system:
 `stdrun_quiet`
- steady state
 initialization – see initialization: steady
 state, initialization: non-steady state
 of complex kinetic schemes – see
 initialization: strategies: steady state
 initialization of complex kinetic
 schemes, INITIAL block: SOLVE:
 STEADYSTATE sparse
- STEADYSTATE – see INITIAL block: SOLVE:
 STEADYSTATE sparse, initialization
- step – see standard run system: step
- `step()` – see standard run system: `step()`
- `steprun()` – see standard run system:
 `steprun()`
- stiffness – see system: stiff
- stoichiometry 39
- stop – see hoc syntax: flow control: stop
- stopping hoc – see hoc: interrupting
 execution, hoc: starting and exiting
- stopping NEURON – see NEURON: starting
 and exiting, hoc: interrupting execution
- `stoprun` – see standard run system: `stoprun`
- storage oscilloscope 23
- strange shapes – see stylized specification of
 geometry: strange shapes
- `strdef` – see hoc syntax: variables:
 `strdef`
- `string` – see hoc syntax: variables: `strdef`
- String class 373
- stylized model – see model: stylized, stylized
 specification of geometry
- stylized specification of geometry 103–104,
 132
 calculation of `area()` and `ri()` 104
 `diam` – see `diam`
- L – see L
 reinterpretation as 3-D specification 108
 strange shapes 144
 also see 3-D specification of geometry
- subclass – see class: subclass, object-oriented
 programming: polymorphism, object-
 oriented programming: inheritance
- subtraction operator – see hoc syntax:
 expressions: operators
- SUFFIX – see NEURON block: SUFFIX
- SUNDIALS 172
 also see CVODES, IDA
- surface area 52
 also see `area()`, membrane area
- synapse
 alpha function – see AlphaSynapse,
 Exp2Syn, Example 10.4: alpha
 function synapse
 AlphaSynapse 19
 AMPAergic – see Example 10.3: synapse
 with exponential decay
 also see Example 10.6: saturating
 synapses
- as instrumentation 139
- conductance change 19
 alpha function – see AlphaSynapse,
 Exp2Syn, Example 10.4: alpha
 function synapse
 exponentially decaying – see ExpSyn,
 Example 10.3: synapse with
 exponential decay
 also see Example 10.5: use-dependent
 synaptic plasticity, Example 10.6:
 saturating synapses
- electrical – see gap junction, synapse:
 ephaptic
- ephaptic 265
- graded – see synaptic transmission: graded
 plasticity – see synaptic plasticity
- saturating – see Example 10.6: saturating
 synapses
- strength – see NetCon class: weight
- weight – see NetCon class: weight
 also see AlphaSynapse, ExpSyn,
 Exp2Syn
- synaptic connection – see NetCon class
- synaptic convergence – see convergence
 also see synaptic transmission: spike-
 triggered
- synaptic delay – see NetCon class: delay
- synaptic divergence – see divergence
 also see synaptic transmission: spike-triggered

- synaptic latency – see *NetCon* class: delay
 - synaptic plasticity 83
 - stream-specific – see Example 10.5: use-dependent synaptic plasticity, Example 10.6: saturating synapses
 - synaptic strength – see *NetCon* class: weight
 - synaptic transmission
 - delay – see *NetCon* class: delay
 - graded 265
 - conceptual model 266
 - implementation in NMODL 268
 - also see Example 10.1: graded synaptic transmission
 - latency – see *NetCon* class: delay
 - saturating – see Example 10.6: saturating synapses
 - spike-triggered
 - computational efficiency in NEURON 275
 - conceptual model 273
 - event-based implementation 273
 - also see *NetCon* class,
 - NET_RECEIVE block, event: external, event: self-event, Example 10.3: synapse with exponential decay, Example 10.4: alpha function synapse, Example 10.5: use-dependent synaptic plasticity, Example 10.6: saturating synapses
 - weight – see *NetCon* class: weight
 - also see gap junction
 - synaptic weight – see *NetCon* class: weight
 - syntax – see hoc syntax, NMODL
 - also see Programmer's Reference
 - syntax error
 - example 100
 - also see hoc: error handling
 - system
 - chaotic – see initialization: categories: to a desired state
 - continuous 55–58, 60, 96
 - piecewise linear approximation – see range variable: estimating by linear interpolation between nodes
 - discretized 57, 60
 - also see discretization
 - linear 65, 71, 86
 - nonlinear 71, 87
 - oscillating – see initialization: categories: to a desired state
 - stiff 66, 73, 86–87, 169
 - also see numeric integration: stability
 - system equations
 - effect of signal sources 20
 - also see numeric integration: stability: effect of signal sources
 - matrix form 73
 - extracellular field 74
 - linear circuit 74
 - also see equation: algebraic
 - stiff – see system: stiff
 - also see eigenfunction, eigenvalue
- T**
- t 24, 26
 - as an ASSIGNED variable – see ASSIGNED variable: v, celsius, t, dt, diam, and area
 - also see t: use in NMODL
 - initialization – see initialization: t
 - the independent variable in NEURON 213
 - use in NMODL 213
 - also see time
- τ – see time constant
 - τ_m – see membrane time constant
 - table – see function table
 - table_ – see NMODL: FUNCTION_TABLE
 - tapering – see diam: tapering, range variable: linear taper
 - Taylor's series 66, 83
 - temperature – see celsius
 - template 363
 - cannot be redefined 156, 367
 - direct commands 368
 - names cannot redefine hoc keywords 369
 - this – see object reference: this
 - variable initialization
 - default initialization 368
 - init() procedure 368
 - visibility – see template: writing a template: public, object: public members
 - writing a template 367
 - begintemplate 367
 - endtemplate 368
 - external 368
 - public 367
 - also see object: public members
 - also see class, object, object-oriented programming
 - temporal accuracy
 - empty 87
 - also see discretization: temporal
 - temporal discretization – see discretization: temporal

temporal frequency – see frequency: temporal
 temporal resolution – see temporal accuracy,
 discretization: temporal
 this – see object reference: this
 three dimensional specification – see 3-D
 specification of geometry
 tilde – see KINETIC block: ~ (tilde)
 time 21, 24–26, 32
 rise – see rise time
 also see t
 time constant 48
 also see membrane time constant
 time-dependent variable – see PARAMETER
 variable: time-dependent, variable: time-
 dependent, Vector class: play()
 time step
 and stability – see numeric integration:
 stability, backward Euler method,
 forward Euler method, Crank–
 Nicholson method
 choosing – see discretization: temporal
 fixed – see numeric integration: fixed time
 step
 also see dt, Δt , discretization: temporal,
 numeric integration: adaptive
 time stream – see numeric integration: local
 time step
 tolerance
 error – see absolute error: local: tolerance,
 relative error: local: tolerance
 top level of the interpreter – see hoc: top level
 of the interpreter
 topology 8, 35, 98
 checking 102–103, 134, 150
 loops of sections 101
 specifying 101, 130
 viewing 103
 also see branched architecture
 topology, subsets, geometry, biophysics 138
 topology() 102, 150
 total error – see numerical error: global
 total ionic current – see membrane current:
 ionic
 transient signals
 spatial decay – see spatial decay of fast signals
 transmembrane current – see membrane current
 treeset.c 167
 troubleshooting
 conflicts between hoc and GUI – see GUI:
 conflicts with hoc or other GUI tools
 disappearing section 148
 Graphs don't work 151

legacy code 142
 no default section 142
 no NEURON Main Menu toolbar 142
 strange shapes – see stylized specification
 of geometry: strange shapes
 TRUE – see hoc syntax: expressions: logical
 expressions
 tstop – see standard run system: tstop
 Tstop – see RunControl GUI: Tstop, standard
 run system: tstop

U

unaryminus operator – see hoc syntax:
 expressions: operators
 understanding 32, 33
 uninsert 153
 units 36
 checking 212, 216, 254
 also see modlunit
 consistency 40, 42–43, 247, 254
 conversion factor – see scale factor,
 NMODL: units conversion factor,
 UNITS block: units scaling
 database 212
 also see nrnunits.lib
 defining new names – see UNITS block:
 defining new names
 dimensionless
 (1) 250
 by default 212
 e 231
 also see e: electronic charge vs. units
 conversion factor
 faraday 231
 k-mole 231
 mole 256
 scaling – see UNITS block: units scaling
 specifying 227
 also see NMODL: declaring variables:
 specifying units
 UNITS block 222
 defining new names 222
 units scaling 231, 250
 units scaling – see UNITS block: units scaling
 UNITSOFF . . . UNITSON – see NMODL:
 UNITSOFF . . . UNITSON
 UNITSON – see NMODL: UNITSOFF . . .
 UNITSON
 use-dependent synaptic plasticity – see
 Example 10.5: use-dependent synaptic
 plasticity, Example 10.6: saturating
 synapses

- USEION – see NEURON block: USEION
- user-defined mechanisms – see mechanisms:
 - user-defined
- user interface 154
 - as virtual experimental rig 23, 154
 - custom GUI 22, 29
 - vs. model specification – see good
 - programming style: separating model specification from user interface
- user's intent 81
- V**
- v 21, 94, 168
 - as an ASSIGNED variable – see
 - ASSIGNED variable: v, celsius, t, dt, diam, and area
 - is a RANGE variable 211
 - also see membrane potential
- v_init – see initialization: v_init
- variable
 - abrupt change of 161, 171, 260–262, 277–278, 280, 291–292, 294, 297–298, 301–302
 - also see CVOICE: and model
 - descriptions: at_time(), CVOICE: and model descriptions:
 - state_discontinuity(),
 - NetCon class: event(),
 - NET_RECEIVE block: handling
 - abrupt changes and discontinuities
- ASSIGNED – see ASSIGNED variable
- changing in mid-run – see variable: abrupt change of, PARAMETER variable: time-dependent
- CONSTANT – see CONSTANT
- continuous – see continuous variable
- declaring in NMODL – see NMODL:
 - declaring variables, NMODL: named blocks: variable declaration,
 - ASSIGNED block, CONSTANT block, PARAMETER block, STATE block,
 - LOCAL variable, NMODL: DEFINE
- dimensionless – see units: dimensionless
- extensive 247
- global – see variable: local vs. nonlocal,
 - hoc syntax: names, template: writing a template: external, NEURON block: GLOBAL, ASSIGNED variable: GLOBAL, PARAMETER variable: is GLOBAL by default
- initializing – see initialization, template:
 - variable initialization
- intensive 247
- LOCAL – see LOCAL variable
- local vs. nonlocal 266
 - also see POINTER variable
- names – see hoc syntax: names
- object – see object, object reference
- PARAMETER – see PARAMETER variable
- POINTER – see POINTER variable
- range – see range variable
- scope – see hoc syntax: names, funcs and procs: local variable, variable: local vs. nonlocal, LOCAL variable, ASSIGNED variable: GLOBAL, NEURON block: GLOBAL, PARAMETER variable: is GLOBAL by default
- also see object: public members,
 - template: writing a template: public, template: writing a template: external
- section – see section variable
- state – see state variable
- state vs. STATE – see STATE variable: vs. state variable
- STATE – see STATE variable
- string – see hoc syntax: variables: strdef
- time-dependent – see PARAMETER
 - variable: time-dependent, Vector class: play()
- user-defined
 - in hoc – see hoc syntax: names
 - in NMODL – see NMODL: user-defined variable
- variable browser 29
 - also see Plot what? GUI
- variable order integration – see numeric
 - integration: adaptive
- variable order, variable time step
 - integration – see numeric integration: adaptive
- variable time step method – see numeric
 - integration: adaptive
- VariableTimeStep GUI
 - bringing up – see NEURON Main Menu: Tools: VariableStepControl
- Details
 - Local step – see VariableTimeStep GUI:
 - global vs. local time steps
- global vs. local time steps 319
 - also see numeric integration: adaptive:
 - global time step, numeric integration: adaptive: local time step

toggling adaptive integration on and off 320
 Use variable dt checkbox – see
 VariableTimeStep GUI: toggling
 adaptive integration on and off
 VBox 383
 VBox class
 intercept() 383
 map() 383–384, 386
 also see session file: ocbox_
 mapping to the screen
 window title 384
 ref() 383
 save() 385
 Vector
 movie 180
 Vector class
 c() 337
 fill() 337
 mark() 337
 play() 77
 at specific times 167
 initialization – see initialization:
 Vector.play()
 under adaptive integration 78
 under fixed time step integration 78
 with interpolation 78, 161, 167, 263
 record() 171, 187
 at specific times 167
 initialization 187
 also see initialization:
 frecord_init()
 also see CNode class: record()
 VERBATIM – see NMODL: VERBATIM . . .
 ENDVERBATIM
 verification 28
 version number – see NEURON: startup banner
 vext – see extracellular mechanism:
 vext
 view – see GUI: view
 virtual experimental preparation – see model
 specification: as “virtual experimental
 preparation”
 virtual experimental rig – see user interface: as
 virtual experimental rig
 visibility – see template: writing a template:
 public, object: public members
 Vm 20
 also see membrane potential, v
 voltage 44

gradient 50
 membrane – see membrane potential, v
 voltage clamp
 and stability – see numeric integration:
 stability: effect of signal sources,
 system equations: effect of signal
 sources
 current
 accuracy 169
 preserving spatial accuracy – see point
 process: preserving spatial accuracy
 ramp clamp 77
 voltage-gated current – see channel: voltage-
 gated
 volume – see compartment: size

W

weight – see NetCon class: weight
 weight vector – see NetCon class: weight
 vector
 what are the names of things? 29
 which view contains the mouse? – see Graph
 class: view_info()
 while – see hoc syntax: flow control: while
 why is NEURON fast? – see computational
 efficiency: why is NEURON fast?
 window title – see GUI tool development:
 mapping to the screen: window title
 wopen() – see hoc syntax: basic input and
 output: wopen()
 WRITE – see NEURON block: USEION
 writing an ion concentration – see NEURON
 block: USEION
 writing files – see hoc syntax: basic input and
 output

X

x-expression – see standard run system:
 plotting system: graphLists
 xopen() – see hoc syntax: basic input and
 output: xopen()
 xpanel() 334, 395
 xpvalue() 395
 xred() – see hoc syntax: basic input and
 output: xred()
 xvalue() 334, 395

Z

zero area node – see section: nodes: zero area