```
COMMENT
A "simple" implementation of the Izhikevich neuron with AMPA, NMDA,
GABA_A, and GABA_B receptor dynamics. Equations and parameter values are taken from
  Izhikevich EM (2007).
  "Dynamical systems in neuroscience"
  MIT Press

Equation for synaptic inputs taken from
  Izhikevich EM, Edelman GM (2008).
  "Large-scale model of mammalian thalamocortical systems."
  PNAS 105(9) 3593-3598.

Example usage (in Python):
  from neuron import h
  dummycell = h.Section() # Since Izhi is a point process, it needs to be in a section
  izhl = [h.Izhi2007(0.5) for i in range(2)] # Create two new Izhikevich cells
  connection = h.NetCon(izhl[0], izhl[1]) # Connect them
  izhl[0].Iin = 70  # activate 1 cell

Cell types available are based on Izhikevich, 2007 book:
    1. RS - Layer 5 regular spiking pyramidal cell (fig 8.12 from 2007 book)
    2. IB - Layer 5 intrinsically bursting cell (fig 8.19 from 2007 book)
    3. CH - Cat primary visual cortex chattering cell (fig8.23 from 2007 book)
    4. LTS - Rat barrel cortex Low-threshold  spiking interneuron (fig 8.25 from 2007
book)
    5. FS - Rat visual cortex layer 5 fast-spiking interneuron (fig 8.27 from 2007
book)
    6. TC - Cat dorsal LGN thalamocortical (TC) cell (fig 8.31 from 2007 book)
    7. RTN - Rat reticular thalamic nucleus (RTN) cell  (fig 8.32 from 2007 book)
ENDCOMMENT

: Declare name of object and variables
NEURON {
  POINT_PROCESS Izhi2007a
  RANGE C, k, vr, vt, vpeak, a, b, c, d, Iin, tauAMPA, tauNMDA, tauGABAA, tauGABAB,
tauOpsin, celltype, alive, cellid, verbose
  RANGE V, u, gAMPA, gNMDA, gGABAA, gGABAB, gOpsin, I
  RANGE factor, eventflag, delta, t0
}

: Specify units that have physiological interpretations (NB: ms is already declared)
UNITS {
  (mV) = (millivolt)
  (uM) = (micrometer)
}

: Parameters from Izhikevich 2007, MIT Press for regular spiking pyramidal cell
PARAMETER {
  C = 1 : Capacitance
  k = 0.7
  vr = -60 (mV) : Resting membrane potential
  vt = -40 (mV) : Membrane threhsold
  vpeak = 35 (mV) : Peak voltage
  a = 0.03
  b = -2
  c = -50
  d = 100
  Iin = 0
  Vpre = 0
  tauAMPA = 5 (ms) : Receptor time constant, AMPA
  tauNMDA = 150 (ms) : Receptor time constant, NMDA
  tauGABAA = 6 (ms) : Receptor time constant, GABAA
  tauGABAB = 150 (ms) : Receptor time constant, GABAB
  tauOpsin = 50 (ms) : Receptor time constant, opsin, from Mattis et al. (2011)
  celltype = 1 : A flag for indicating what kind of cell it is,  used for changing the
dynamics slightly (see list of cell types in initial comment).
  alive = 1 : A flag for deciding whether or not the cell is alive -- if it's dead,
acts normally except it doesn't fire spikes
  cellid = -1 : A parameter for storing the cell ID, if required (useful for
diagnostic information)
  verbose = 0 : Whether or not to print diagnostic information to file -- WARNING, do
not modify this manually -- it's set by useverbose()
}
: Variables used for internal calculations
ASSIGNED {
  factor : Voltage factor used for calculating the current
  eventflag : For diagnostic information
  V (mV) : Membrane voltage
  u (mV) : Slow current/recovery variable
  gAMPA : AMPA conductance
  gNMDA : NMDA conductance
  gGABAA : GABAA conductance
  gGABAB : GABAB conductance
  gOpsin : Opsin conductance
  I : Total current
  delta : Time step
  t0 : Previous time
}
: Initial conditions
INITIAL {
  V = vr
  u = 0.0
  t0 = t
  gAMPA = 0
  gNMDA = 0
  gGABAA = 0
  gGABAB = 0
  gOpsin = 0
  I = 0
  delta = 0
  net_send(0,1) : Required for the WATCH statement to be active
}
: Function for printing diagnostic information to a file -- usage example:
cell.useverbose(2,"logfile.txt")
VERBATIM
char filename[1000]; // Allocate some memory for the filename
ENDVERBATIM
PROCEDURE useverbose() { : Create user-accessible function
  VERBATIM
  #include<stdio.h> // Basic input-output
  verbose = (float) *getarg(1); // Set verbosity -- 0 = none, 1 = events, 2 = events +
timesteps
  strcpy(filename, gargstr(2)); // Copy input filename into memory
  ENDVERBATIM
}
: Define neuron dynamics
BREAKPOINT {
  delta = t-t0 : Find time difference
  : Receptor dynamics -- the correct form is gAMPA = gAMPA*exp(-delta/tauAMPA), but
this is 30% slower and, in the end, not really any more physiologically realistic
  gAMPA = gAMPA - delta*gAMPA/tauAMPA : "Exponential" decays -- fast excitatory (AMPA)
  gNMDA = gNMDA - delta*gNMDA/tauNMDA : Slow excitatory (NMDA)
  gGABAA = gGABAA - delta*gGABAA/tauGABAA : Fast inhibitory (GABA_A)
  gGABAB = gGABAB - delta*gGABAB/tauGABAB : Slow inhibitory (GABA_B)
  gOpsin = gOpsin - delta*gOpsin/tauOpsin : Optogenetic (opsin)

  : Calculate current
  factor = ((V+80)/60)*((V+80)/60)
  I = gAMPA*(V-0) + gNMDA*factor/(1+factor)*(V-0) + gGABAA*(V+70) + gGABAB*(V+90) +
gOpsin*(V-0) : Treat the opsin channel like an AMPA channel

  : Calculate neuronal dynamics; -I since I = -I_{syn}, which is really what I is as
I've defined it above
  Vpre = V
  V = V + delta*(k*(V-vr)*(V-vt) - u - I + Iin)/C  : Calculate voltage

  if (Vpre<=c && V>vpeak) {V=c+1} : if just spiked, wait at least 1 timestep before
increasing V>vpeak again, so V reset value takes effect; WATCH statement requires V to
cross the vpeak threshod)

  : Cell-type specific dynamics
  if (celltype<5) {
    u = u + delta*a*(b*(V-vr)-u) : Calculate recovery variable
  }
  else {
     : For FS neurons, include nonlinear U(v): U(v) = 0 when v<vb ; U(v) = 0.025(v-vb)
when v>=vb (d=vb=-55)
     if (celltype==5) {
       if (V<d) {
        u = u + delta*a*(0-u)
       }
       else {
        u = u + delta*a*((0.025*(V-d)^3)-u)
       }
     }

     : For TC neurons, reset b
     if (celltype==6) {
       if (V>-65) {b=0}
       else {b=15}
       u = u + delta*a*(b*(V-vr)-u) : Calculate recovery variable
     }

     : For TRN neurons, reset b
     if (celltype==7) {
       if (V>-65) {b=2}
       else {b=10}
       u = u + delta*a*(b*(V-vr)-u) : Calculate recovery variable
     }
  }
  t0=t : Reset last time so delta can be calculated in the next time step

  : Print diagnostic inormation to a file
  if (verbose>1) { : Verbose turned on?
    VERBATIM
    FILE *outfile; // Declare file object
    outfile=fopen(filename,"a"); // Open file for appending
    fprintf(outfile,"%8.2f   cell=%6.0f   delta=%8.2f   gAMPA=%8.2f   gNMDA=%8.2f
gGABAA=%8.2f   gGABAB=%8.2f   gOpsin=%8.2f   factor=%8.2f   I=%8.2f   V=%8.2f
u=%8.2f (timestep)\n",t,cellid,delta,gAMPA,gNMDA,gGABAA,gGABAB,gOpsin,factor,I,V,u);
    fclose(outfile); // Close file
    ENDVERBATIM
  }
}

: Input received
NET_RECEIVE (wAMPA, wNMDA, wGABAA, wGABAB, wOpsin) {
  INITIAL { wAMPA=wAMPA wNMDA=wNMDA wGABAA=wGABAA wGABAB=wGABAB wOpsin=wOpsin} :
Insanely stupid but required, otherwise reset to 0,

  : Check if spike occurred
  if (flag == 1) { : Fake event from INITIAL block
    if (celltype < 4 || celltype == 5 || celltype == 7) { : default
      WATCH (V>vpeak) 2 : Check if threshold has been crossed, and if so, set flag=2
    }
    else if (celltype == 4) { : LTS cell
      WATCH (V>(vpeak-0.1*u)) 2 : Check if threshold has been crossed, and if so, set
flag=2
    }
    else if (celltype == 6) { : TC cell
      WATCH (V>(vpeak+0.1*u)) 2 : Check if threshold has been crossed, and if so, set
flag=2
    }
  }

  : Event created by WATCH statement -- i.e. threshold crossed
  else if (flag == 2) {
    if (alive) {net_event(t)} : Send spike event if the cell is alive

    : For RS, IB and CH neurons, and RTN
    if (celltype < 4 || celltype == 7) {
      V = c : Reset voltage
      u = u+d : Reset recovery variable
    }
    : For LTS neurons
    else if (celltype == 4) {
      V = c+0.04*u : Reset voltage
      if ((u+d)<670) {u=u+d} : Reset recovery variable
      else {u=670}
     }
    : For FS neurons (only update v)
    else if (celltype == 5) {
      V = c : Reset voltage
     }
    : For TC neurons (only update v)
    else if (celltype == 6) {
      V = c-0.1*u : Reset voltage
      u = u+d : Reset recovery variable
     }

    gAMPA = 0 : Reset conductances -- not mentioned in Izhikevich's paper but
necessary to stop things from exploding!
    gNMDA = 0
    gGABAA = 0
    gGABAB = 0
    gOpsin = 0
  }

  : Actual input, calculate receptor dynamics
  else {
    gAMPA = gAMPA + wAMPA
    gNMDA = gNMDA + wNMDA
    gGABAA = gGABAA + wGABAA
    gGABAB = gGABAB + wGABAB
    gOpsin = gOpsin + wOpsin
  }

  : Print diagnostic information to a file
  if (verbose>0) { : Verbose turned on?
    eventflag = flag
    VERBATIM
    FILE *outfile; // Declare file object
//if(cellid>=0 && cellid < 300) {
    outfile=fopen(filename,"a"); // Open file for appending
    fprintf(outfile,"t=%8.2f   cell=%6.0f   flag=%1.0f   gAMPA=%8.2f   gNMDA=%8.2f
gGABAA=%8.2f   gGABAB=%8.2f   gOpsin=%8.2f   V=%8.2f   u=%8.2f (event)\n",t,
cellid,eventflag,gAMPA,gNMDA,gGABAA,gGABAB,gOpsin,V,u);
    fclose(outfile); // Close file
//}
    ENDVERBATIM
  }


}
```