Three-compartment neuron model

Martin Nilsson and Henrik Jörntell, 2020-12-16. This program is written for Mathematica v. 11.

Neuronal units: ms/kHz/mV/pA/pF/GΩ/nS

In[ ]:=
```
{time = 10,
  r_dp = 0.0066,  (* from [Llano et al 1991; p 188] *)
  c_p = 87,       (* from [Llano et al 1991; p 188] *)
  r_p = 0.0074,   (* from [Llano et al 1991; p 188] *)
  τ_K = 0.28,     (* from [Raman and Bean 1999; p 1667] *)
  i_K0 = 6000,
  α = 1 / 2.2 ,   (* logarithmic voltage sensitivity from [Hille 2001; p 610] *)
  i_Na0 = -100 ,
  v_d0 = 0,
  v_p0 = 0,
  v_a0 = 0,
  i_d = 0,
  r_pa = r_dp,     (* Assumption r_pa ≈ r_dp *)
  c_d  = 30 c_p,   (* Assumption c_d>>c_p *)
  c_a  = 0.03 c_p, (* Assumption c_a<<c_p *)
  r_a = 30 r_p ,   (* Assumption r_a>>r_p *)
  r_d  = 0.030,    (* Henrik Jörntell's measurements *)
  τ = 5.5,         (* Computed in this paper *)
  i_ch0 = 50,
  channelnoise};
SeedRandom[9]; (* 9,11 are OK för ich0=100, 50*)
channelnoise =
  RandomFunction[OrnsteinUhlenbeckProcess[0, 1, 1 / τ, 0], {0, time, 0.01}];
solution = NDSolve[{
    (* ----- First compartment (distal dendrites; integrate) *)
    v_d[0] == v_d0,
    c_d v_d'[t] ==  -i_dp[t] - i_rd[t] + i_d,
    i_dp[t] == (v_d[t] - v_p[t]) / r_dp,
    i_rd[t] == v_d[t] / r_d,
    (* ----- Second compartment (proximal dendrites and soma; generate ramp) *)
    v_p[0] == v_p0,
    c_p v_p'[t] ==
     -i_rp[t] + i_dp[t] - i_p[t] - i_pa[t] - i_ch0 channelnoise["SliceData", t][[1]],
    i_rp[t] == v_p[t] / r_p,
    i_p[t] == i_K0 Exp[-t / τ_K],  (* K-channels, [Hille 2001; p 47] *)
    i_pa[t] == (v_p[t] - v_a[t]) / r_pa,
    (* ----- Third compartment (axon initial segment; detect level, "fire") *)
    v_a[0] == v_a0,
```

```
    ca va'[t] == ipa[t] - ira[t] - ia[t],
    ira[t] == va[t] / ra,
    ia[t] == Max[iNa0 Exp[α va[t]], -10^5] (* NaV-channels, [Hille 2001; p 610] *)
  }, {vd, vp, va}, {t, 0, time}];

Plot[{vd[t] /. solution, vp[t] /. solution, va[t] /. solution}, {t, 0, time},
 PlotLegends → Placed[{Vd, Vp, Va}, {0.15, 0.8}],
 PlotStyle → {DotDashed, _, Dashed},
 PlotLabel → Style["Resting potential offset", Black],
 PlotRange → {{0, time}, {-10, 10}}, AxesStyle → Black,
 AxesLabel → {"t (ms)", "(mV)"}]
```

Out[∘]=