Model_double_exps

March 8, 2017

1 Simple model using double exponentials

```
In [1]: from sympy import *

In [2]: from IPython.display import display

In [3]: init_printing()

In [4]: t, P, e_r, e_d, delta_e, rho_e, g_e, i_r, i_d, delta_i, rho_i, g_i, b = symbols('t P \\t

In [5]: SymbolDict = {t: "Time", P: "Proportion of $g_i/g_e$", e_r: "Excitatory Rise", e_d: "Exc
```

Variable	Meaning	Range
t	Time	0 - 100 ms
τ_{ed}	Excitatory Fall	8-20 ms
τ_{er}	Excitatory Rise	2-8 ms
$\bar{g}_e$	Excitatory max conductance	--
δ_e	Excitatory onset time	0 ms
τ_{id}	Inhibitory Fall	14-60 ms
τ_{ir}	Inhibitory Rise	1.5-5 ms
$\bar{g}_i$	Inhibitory max conductance	--
δ_i	Inhibitory onset time	3-15 ms
P	Proportion of g_i/g_e	~2
ρ_e	τ_{ed}/τ_{er}	2-7
ρ_i	τ_{id}/τ_{ir}	5-20
β	τ_{ir}/τ_{er}	--

```
In [60]: #for i,value in SymbolDict.items():
         #    print "|${}$|{}|--|".format(i, value)

|$\tau_{ir}$|Inhibitory Rise|--|
|$\bar{g}_i$|Inhibitory max conductance|--|
|$\bar{g}_e$|Excitatory max conductance|--|
|$\delta_e$|Excitatory onset time|--|
|$P$|Proportion of $g_i/g_e$|--|
|$\tau_{er}$|Excitatory Rise|--|
```

```
|$t$|Time|--|
|$\tau_{id}$|Inhibitory Fall|--|
|$\tau_{ed}$|Excitatory Fall|--|
|$\rho_e$|Excitatory $tau$ ratio (fall/rise)|--|
|$\delta_i$|Inhibitory onset time|--|
|$\rho_i$|Inhibitory $tau$ ratio (fall/rise)|--|
|$\beta$|Inhibitory/Excitatory $tau$ rise ratio|--|
```

1.0.1 Double exponential to explain the net synaptic conductance.

```
In [7]: alpha = exp(-(t-delta_e)/e_d) - exp(-(t-delta_e)/e_r)
```

```
In [8]: alpha
```

Out[8]:

$$e^{\frac{1}{\tau_{ed}}(\delta_e - t)} - e^{\frac{1}{\tau_{er}}(\delta_e - t)}$$

```
In [9]: alpha_prime = alpha.diff(t)
```

```
In [10]: alpha_prime
```

Out[10]:

$$\frac{1}{\tau_{er}} e^{\frac{1}{\tau_{er}}(\delta_e - t)} - \frac{1}{\tau_{ed}} e^{\frac{1}{\tau_{ed}}(\delta_e - t)}$$

```
In [11]: theta_e = solve(alpha_prime,t) # Time to peak
```

```
In [12]: theta_e = logcombine(theta_e[0])
```

```
In [13]: theta_e
```

Out[13]:

$$\frac{1}{\tau_{ed} - \tau_{er}} \left(\delta_e \left(\tau_{ed} - \tau_{er} \right) - \log \left(\left(\frac{\tau_{er}}{\tau_{ed}} \right)^{\tau_{ed} \tau_{er}} \right) \right)$$

```
In [14]: N(theta_e.subs({e_d:20, e_r: 5, delta_e:0}))
```

Out[14]:

$$9.24196240746594$$

```
In [15]: E_star = alpha.subs(t, theta_e)
```

```
In [16]: E_star = simplify(E_star.subs(e_d/e_r, rho_e)) # Replacing e_d/e_r with tau_e
```

1.0.2 Finding maximum of the curve and substituting ratio of taus

In [17]:
```
E_star
```

Out[17]:

$$-\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed}-\tau_{er}}} + \left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed}-\tau_{er}}}$$

In [18]:
```
E = Piecewise((0, t < delta_e), (g_e * (alpha/E_star), True))
```

1.0.3 Final equation for Excitation normalized to be maximum at g_e

In [19]:
```
E
```

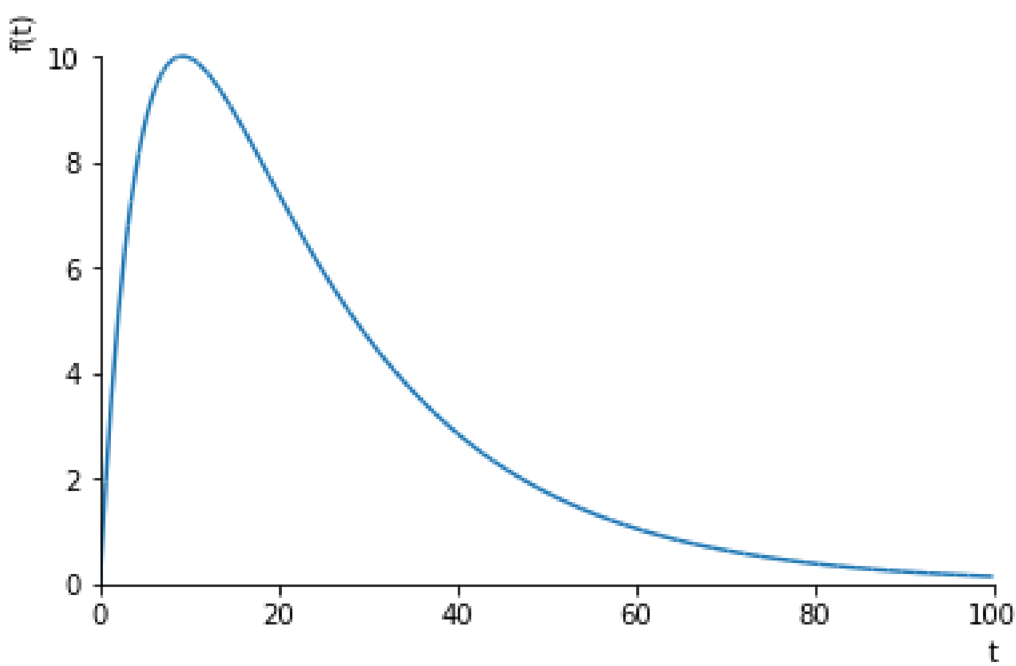
Out[19]:

$$\begin{cases} 0 & \text{for } t < \delta_e \\ \frac{\bar{g}_e \left(e^{\frac{1}{\tau_{ed}}(\delta_e - t)} - e^{\frac{1}{\tau_{er}}(\delta_e - t)}\right)}{-\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed}-\tau_{er}}} + \left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed}-\tau_{er}}}} & \text{otherwise} \end{cases}$$

1.0.4 Verifying that E Behaves

In [20]:
```
E_check = N(E.subs({e_d:20, e_r: 5, delta_e:0, g_e:10}))
```

In [21]:
```
plot(E_check,(t,0,100))
```



Out[21]:
```
<sympy.plotting.plot.Plot at 0x7fc301d03090>
```

1.0.5 Doing the same with inhibition

```
In [22]: I = E.xreplace({g_e: g_i, rho_e: rho_i, e_r:i_r, e_d: i_d, delta_e: delta_i})
```

```
In [23]: I_star = E_star.xreplace({g_e: g_i, rho_e: rho_i, e_r:i_r, e_d: i_d, delta_e: delta_i})
```

1.0.6 Similar equation for Inhibition

```
In [24]: I
```

Out[24]:

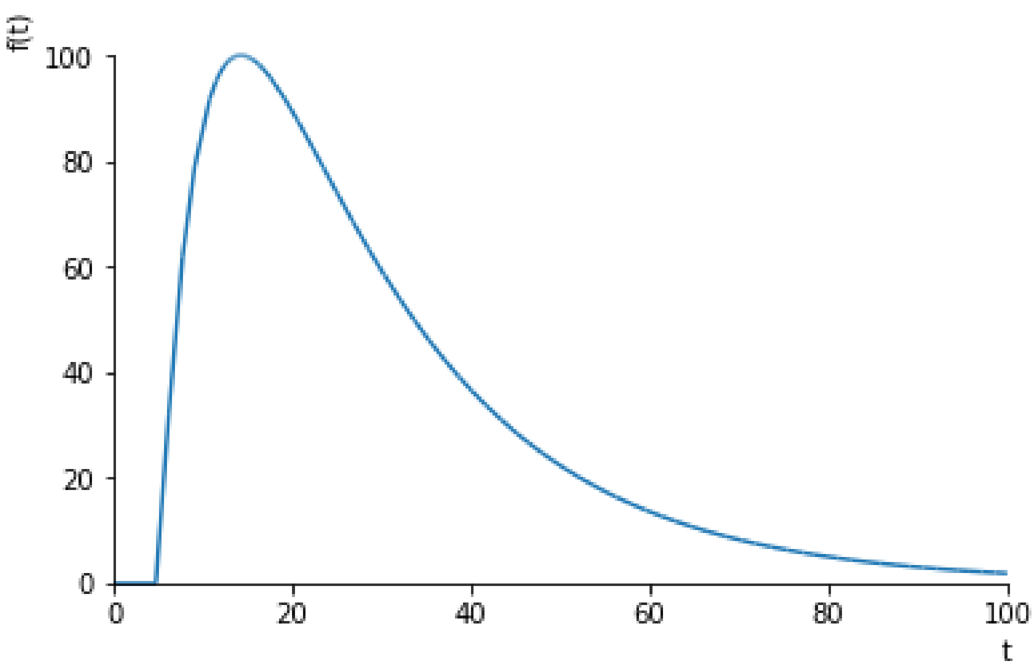
$$\begin{cases} 0 & \text{for } t < \delta_i \\ \frac{\bar{g}_i \left(e^{\frac{1}{\tau_{id}}(\delta_i - t)} - e^{\frac{1}{\tau_{ir}}(\delta_i - t)}\right)}{-\left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{id}}{\tau_{id}-\tau_{ir}}} + \left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{ir}}{\tau_{id}-\tau_{ir}}}} & \text{otherwise} \end{cases}$$

1.0.7 Verifying that I Behaves

```
In [41]: delay = 5
```

```
In [42]: I_check = N(I.subs({i_d:20, i_r: 5, delta_i:delay, g_i:100}))
```

```
In [43]: plot(I_check,(t,0,100))
```




```
Out[43]: <sympy.plotting.plot.Plot at 0x7fc300acca90>
```

1.0.8 Now finding the control peak using difference of these double-exponentials

```
In [44]: C =  E - I
```

```
In [45]: C
```

Out[45]:

$$\begin{cases} 0 & \text{for } t < \delta_e \\ \frac{\bar{g}_e \left(e^{\frac{1}{\tau_{ed}}(\delta_e - t)} - e^{\frac{1}{\tau_{er}}(\delta_e - t)}\right)}{-\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed}-\tau_{er}}} + \left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed}-\tau_{er}}}} & \text{otherwise} \end{cases} - \begin{cases} 0 & \text{for } t < \delta_i \\ \frac{\bar{g}_i \left(e^{\frac{1}{\tau_{id}}(\delta_i - t)} - e^{\frac{1}{\tau_{ir}}(\delta_i - t)}\right)}{-\left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{id}}{\tau_{id}-\tau_{ir}}} + \left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{ir}}{\tau_{id}-\tau_{ir}}}} & \text{otherwise} \end{cases}$$

```
In [46]: delay = 3
```

```
In [47]: C_check = N(C.subs({e_d:10, e_r: 3, delta_e:0, g_e:10, i_d:20, i_r: 4, delta_i:delay, g
```

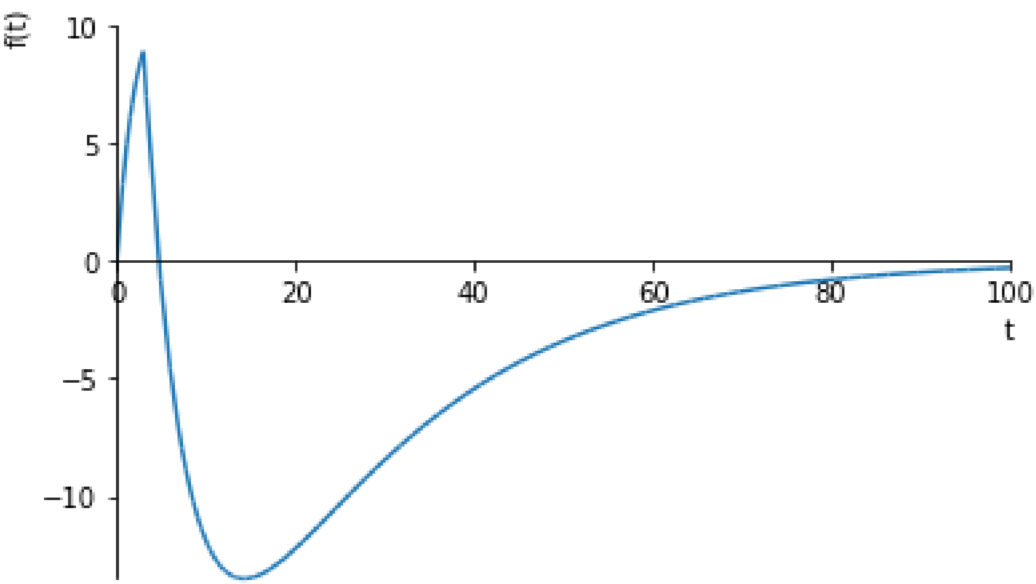
```
In [48]: C_check
```

Out[48]:

$$- \begin{cases} 0 \\ -37.383719530530513547797474853522834884086493940367659 e^{-\frac{t}{4}+\frac{3}{4}} + 37.38371953053051354779747485352 \end{cases}$$

1.0.9 Verifying that C behaves

```
In [49]: plot(C_check,(t,0,100))
```

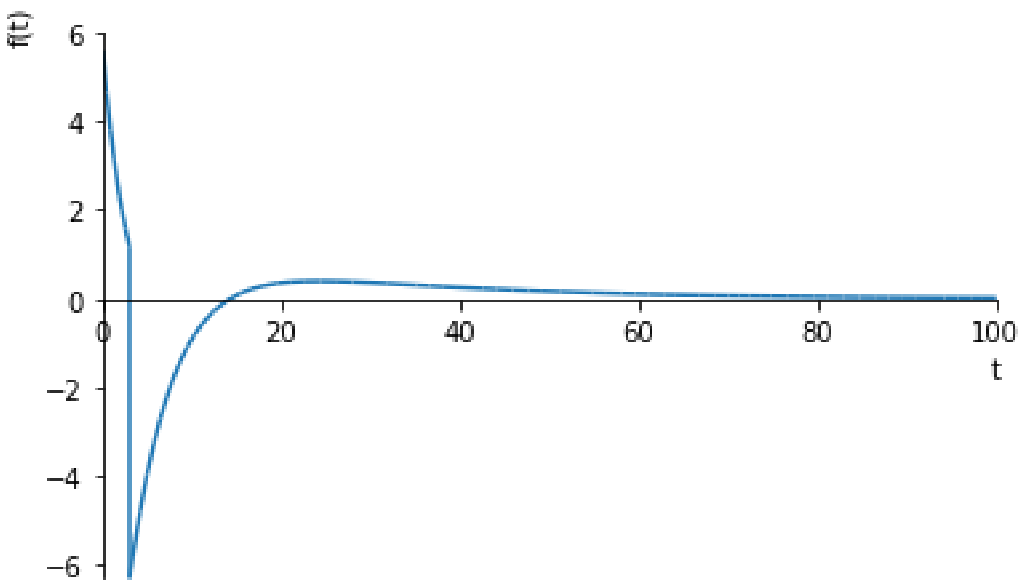


```
Out[49]: <sympy.plotting.plot.Plot at 0x7fc301c083d0>

In [50]: C_prime = diff(C,t)

In [51]: C_prime_check = N(C_prime.subs({e_d:10, e_r: 3, delta_e:0, g_e:10, i_d:20, i_r: 4, delt

In [52]: plot(C_prime_check,(t,0,100))
```

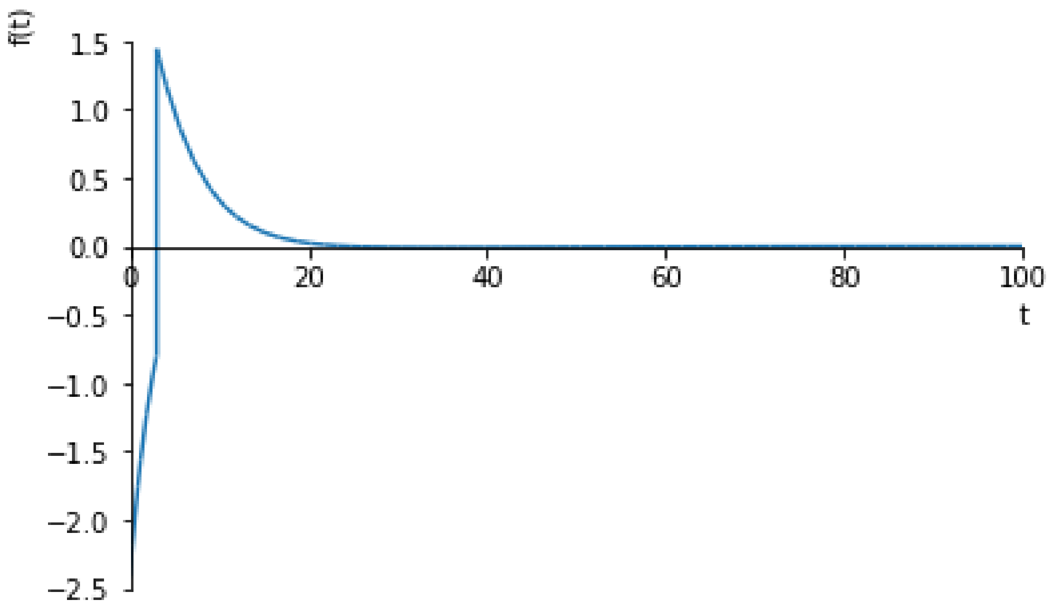



```
Out[52]: <sympy.plotting.plot.Plot at 0x7fc300984b10>

In [53]: C_prime_prime = diff(C_prime,t)

In [54]: C_prime_prime_check = N(C_prime_prime.subs({e_d:10, e_r: 3, delta_e:0, g_e:10, i_d:20,

In [55]: plot(C_prime_prime_check,(t,0,100))
```




```
Out[55]: <sympy.plotting.plot.Plot at 0x7fc3007f9d50>
```

```
In [ ]: T = Symbol('T',positive=True)
```

```
In [ ]: C = C.subs({i_d: (i_r*rho_i)+i_r, e_d: (e_r*rho_e)+e_r, g_i: g_e*P}) # Replacing e_d/e_r
```

```
In [ ]: C = simplify(C)
```

```
In [ ]: C = simplify(C.subs({delta_e:0}))
```

```
In [ ]: simplify(C)
```

```
In [ ]: C_prime = diff(C,t)
```

```
In [ ]: C_prime
```

```
In [ ]: C_prime = expand(cancel(C_prime).collect(g_e))
```

```
In [ ]: C_prime.as_expr()
```

```
In [ ]: C_prime_check
```

1.0.10 Since the τ_{decay} will not contribute to the first peak, we can eliminate them.

```
In [ ]: C = (g_e* (- exp((t-delta_e)/e_r)))/E_star - g_i*(- exp((t-delta_i)/i_r))/I_star
```

```
In [ ]: C
```

```
In [ ]: C_prime = diff(C,t)
```

```
In [ ]: C_prime
```

```
In [ ]: theta_C = solve(C_prime, t)
```

```
In [ ]: theta_C
```

```
In [ ]: C_star = C.subs(t, theta_C[0])
```

1.0.11 Substituting rise and fall ratios and putting δ_e to zero.

```
In [ ]: C_star = C_star.subs(delta_e,0.) # Putting excitatory delay to zero
```

1.0.12 Assuming that certain ratios are more than one and substituting

```
In [ ]: C_star = C_star.subs({i_d: (i_r*tau_i)+i_r, e_d: (e_r*tau_e)+e_r, g_i: g_e*P, i_r:e_r*b}
```

```
In [ ]: C_star = cancel(powsimp(factor(C_star), deep=True))
```

```
In [ ]: C_star = C_star.collect([g_e, delta_i, P])
```

```
In [ ]: #C_star1 = limit(limit(C_star, (1/tau_e), 0), (1/tau_i),0)
```

```
In [ ]: #simplify(C_star.subs({e_r: 4., i_r: 2.73, g_i:P*g_e, e_d: g_e*b, i_d : g_e*g, delta_i:
```

```
In [ ]: #cancel(C_star1.together(deep=True))
```

```
In [ ]: C_star.free_symbols
```

```
In [ ]: #tau_e1, tau_i1 = symbols('\\tau_{e1} \\tau_{i1}', real=True, positive=True)
```

```
In [ ]: #simplify(C_star.subs({tau_e:tau_e1+1, tau_i:tau_i1+1}))
```

```
In [ ]: C_star = simplify(C_star)
```

```
In [ ]: C_star
```

```
In [ ]: cse(C_star)
```

```
In [ ]: cse(simplify(diff(C_star,g_e)))
```

```
In [ ]: x = Symbol('x')
```

```
In [ ]: y = x**(1/x)
```

```
In [ ]: y.subs(x,40).evalf()
```

```
In [ ]: theta_C_nice = simplify(theta_C[0].subs({i_d: (i_r*tau_i)+i_r, e_d: (e_r*tau_e)+e_r, g_i
```

```
In [ ]: cse(cancel(expand(theta_C_nice)))
```

```
In [ ]: theta_C_nice
```

```
In [ ]: limit(x/(x-1),x,5)
```