Model_double_exps

March 11, 2017

1 Simple model using double exponentials

```
In [52]: from sympy import *

In [53]: from IPython.display import display, Markdown

In [54]: init_printing()

In [55]: 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 \\

In [56]: SymbolDict = {t: "Time (ms)", P: "Proportion of $g_i/g_e$", e_r: "Excitatory Rise (ms)"

In [57]: estimateDict = { P: (1.9,2.1), e_r: (1.5,5), e_d: (8,20), delta_e: (0,0), rho_e: (2,7),

In [58]: averageEstimateDict = {key: pow(value[0]*value[1],0.5) for key,value in estimateDict.it

In [59]: #averageEstimateDict = {key: ((value[0]+value[1])/2.) for key,value in estimateDict.ite

In [60]: print "| Variable  |  Meaning |  Range |"
         print "|---|---|---|"
         print "|$t$|Time (ms)|0-100|"
         for i in [P, e_r, e_d, delta_e, rho_e, g_e, i_r, i_d, delta_i, rho_i, g_i, b]:
             print "|${}$|{}|{}-{}|".format(i, SymbolDict[i], estimateDict[i][0], estimateDict[i

| Variable  |  Meaning |  Range |
|---|---|---|
|$t$|Time (ms)|0-100|
|$P$|Proportion of $g_i/g_e$|1.9-2.1|
|$\tau_{er}$|Excitatory Rise (ms)|1.5-5|
|$\tau_{ed}$|Excitatory Fall (ms)|8-20|
|$\delta_e$|Excitatory onset time (ms)|0-0|
|$\rho_e$|Excitatory $tau$ ratio (fall/rise)|2-7|
|$\bar{g}_e$|Excitatory max conductance|0.02-0.25|
|$\tau_{ir}$|Inhibitory Rise (ms)|1.5-5|
|$\tau_{id}$|Inhibitory Fall(ms)|14-60|
|$\delta_i$|Inhibitory onset time(ms)|3-8|
|$\rho_i$|Inhibitory $tau$ ratio (fall/rise)|5-20|
|$\bar{g}_i$|Inhibitory max conductance|0.04-0.5|
|$\beta$|Inhibitory/Excitatory $tau$ rise ratio|0.5-5|
```

Variable	Meaning	Range
t	Time (ms)	0-100
P	Proportion of g_i/g_e	1.9-2.1
τ_{er}	Excitatory Rise (ms)	1.5-5
τ_{ed}	Excitatory Fall (ms)	8-20
δ_e	Excitatory onset time (ms)	0-0
ρ_e	Excitatory *tau* ratio (fall/rise)	2-7
$\bar{g}_e$	Excitatory max conductance	0.02-0.25
τ_{ir}	Inhibitory Rise (ms)	1.5-5
τ_{id}	Inhibitory Fall(ms)	14-60
δ_i	Inhibitory onset time(ms)	3-15
ρ_i	Inhibitory *tau* ratio (fall/rise)	5-20
$\bar{g}_i$	Inhibitory max conductance	0.04-0.5
β	Inhibitory/Excitatory *tau* rise ratio	0.5-5

1.0.1 Double exponential to explain the net synaptic conductance.

```
In [61]: alpha = exp(-(t-delta_e)/e_d) - exp(-(t-delta_e)/e_r)

In [62]: alpha
```

Out[62]:

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

```
In [63]: #alpha = alpha.subs(e_d, (rho_e*e_r)).doit()

In [64]: alpha_prime = alpha.diff(t)

In [65]: alpha_prime
```

Out[65]:

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

```
In [66]: theta_e = solve(alpha_prime,t) # Time to peak

In [67]: theta_e = logcombine(theta_e[0])

In [68]: theta_e
```

Out[68]:

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

```
In [69]: N(theta_e.subs(averageEstimateDict))
```

Out[69]:

$$5.34841395444272$$

```
In [70]: alpha_star = simplify(alpha.subs(t, theta_e).doit())
```

```
In [71]: alpha_star
```

Out[71]:

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

```
In [72]: #alpha_star = simplify(alpha) # Replacing e_d/e_r with tau_e
```

1.0.2 Finding maximum of the curve and substituting ratio of taus

```
In [73]: alpha_star
```

Out[73]:

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

```
In [74]: E = Piecewise((0, t < delta_e), (g_e * (alpha/alpha_star), True))
```

1.0.3 Final equation for Excitation normalized to be maximum at g_e

```
In [75]: E
```

Out[75]:

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

1.0.4 Verifying that E Behaves

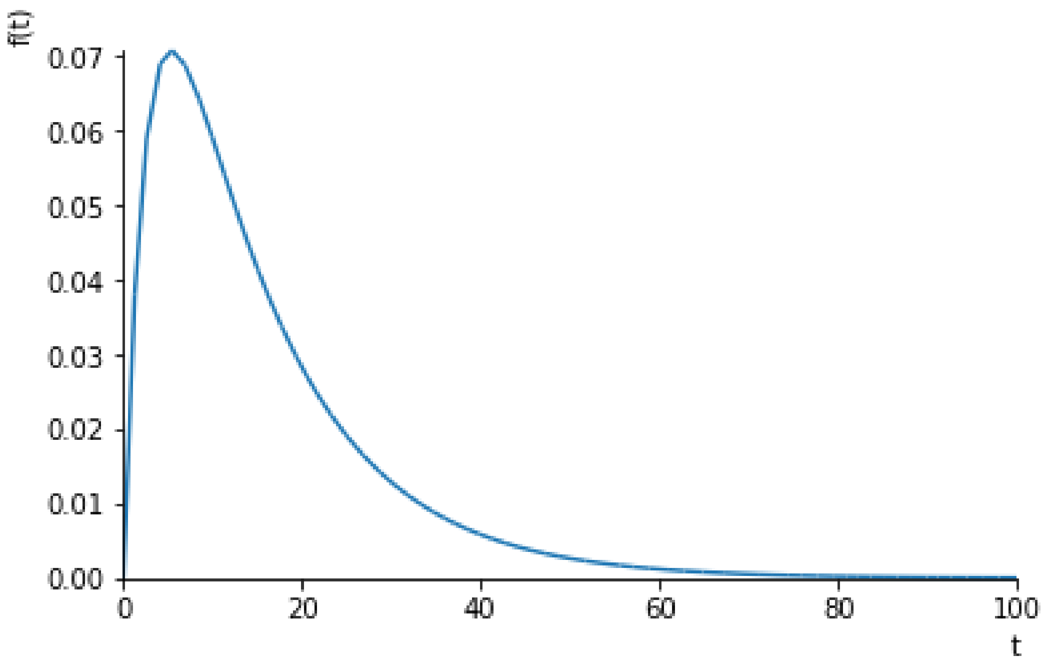
```
In [76]: E_check = N(E.subs(averageEstimateDict))
```

```
In [77]: E_check.free_symbols
```

Out[77]:

$$\{t\}$$

```
In [78]: plot(E_check,(t,0,100))
```



Out[78]: <sympy.plotting.plot.Plot at 0x7f46217d7d10>

1.0.5 Doing the same with inhibition

In [79]:
```
I = E.xreplace({g_e: g_i, rho_e: rho_i, e_r:i_r, e_d: i_d, delta_e: delta_i})
```

In [80]:
```
I
```

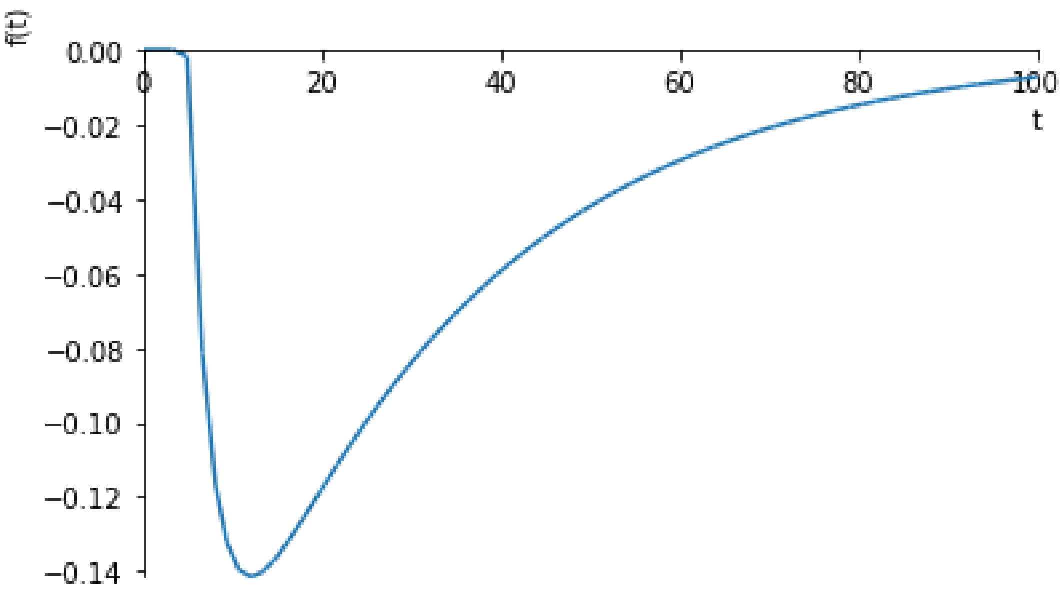
Out[80]:

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

1.0.6 Verifying that I Behaves

In [81]:
```
I_check = N(I.subs(averageEstimateDict))
```

In [82]:
```
plot(-I_check,(t,0,100))
```



Out[82]: <sympy.plotting.plot.Plot at 0x7f46206e5350>

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

```
In [83]: C =  E - I
```

```
In [84]: C
```

Out[84]:

$$\begin{cases} 0 & \text{for}\: t < \delta_{e} \\ \frac{\bar{g}_{e} \left(e^{\frac{1}{\tau_{ed}} (\delta_{e} - t)} - e^{\frac{1}{\tau_{er}} (\delta_{e} - t)}\right)}{- \left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed} - \tau_{er}}} + \left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed} - \tau_{er}}}} & \text{otherwise} \end{cases} - \begin{cases} 0 & \text{for}\: t < \delta_{i} \\ \frac{\bar{g}_{i} \left(e^{\frac{1}{\tau_{id}} (\delta_{i} - t)} - e^{\frac{1}{\tau_{ir}} (\delta_{i} - t)}\right)}{- \left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{id}}{\tau_{id} - \tau_{ir}}} + \left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{ir}}{\tau_{id} - \tau_{ir}}}} & \text{otherwise} \end{cases}$$

1.0.8 Substituting excitatory and inhibitory ratios and putting δ_e to zero.

```
In [85]: #C = C.subs({g_i: g_e*P, i_r : e_r*b}) # Replacing g_i with P*ge
         C = C.subs({delta_e:0})
```

```
In [86]: C_check = N(C.subs(averageEstimateDict))
```

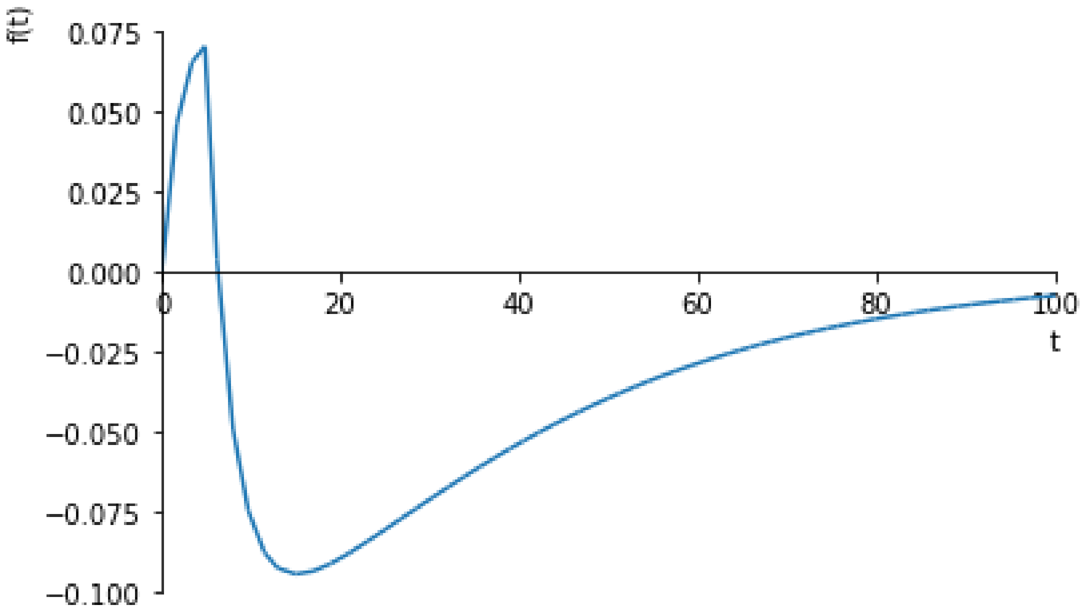
```
In [87]: C_check
```

Out[87]:

$$- \begin{cases} 0 & \text{for}\: t < 4.89897948556636 \\ - \frac{1.19517432161688}{e^{0.365148371670111 t}} + \frac{0.236565186151897}{e^{0.0345032779671177 t}} & \text{otherwise} \end{cases} - \frac{0.137746930949975}{e^{0.365148371670111 t}} + \frac{0.137746930949975}{e^{0.0790569415042095 t}}$$

1.0.9 Verifying that C behaves

In [88]:
```
plot(C_check,(t,0,100))
```



Out[88]: <sympy.plotting.plot.Plot at 0x7f4620b8e4d0>

In [89]:
```
#C_check = N(C.subs({rho_e:7, rho_i: 15}))
```

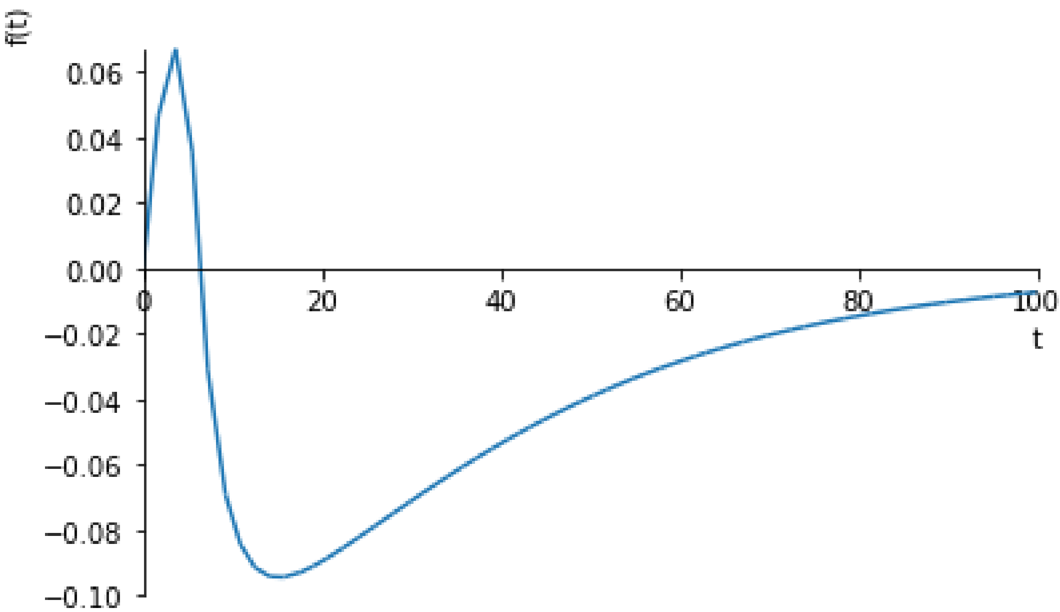
In [90]:
```
C_check
```

Out[90]:

$$-\begin{cases} 0 & \text{for } t < 4.89897948556636 \\ -\frac{1.19517432161688}{e^{0.365148371670111t}} + \frac{0.236565186151897}{e^{0.0345032779671177t}} & \text{otherwise} \end{cases} - \frac{0.137746930949975}{e^{0.365148371670111t}} + \frac{0.137746930949975}{e^{0.0790569415042095t}}$$

In [91]:
```
C_check = C_check.subs(averageEstimateDict)
```

In [92]:
```
plot(C_check,(t,0,100))
```

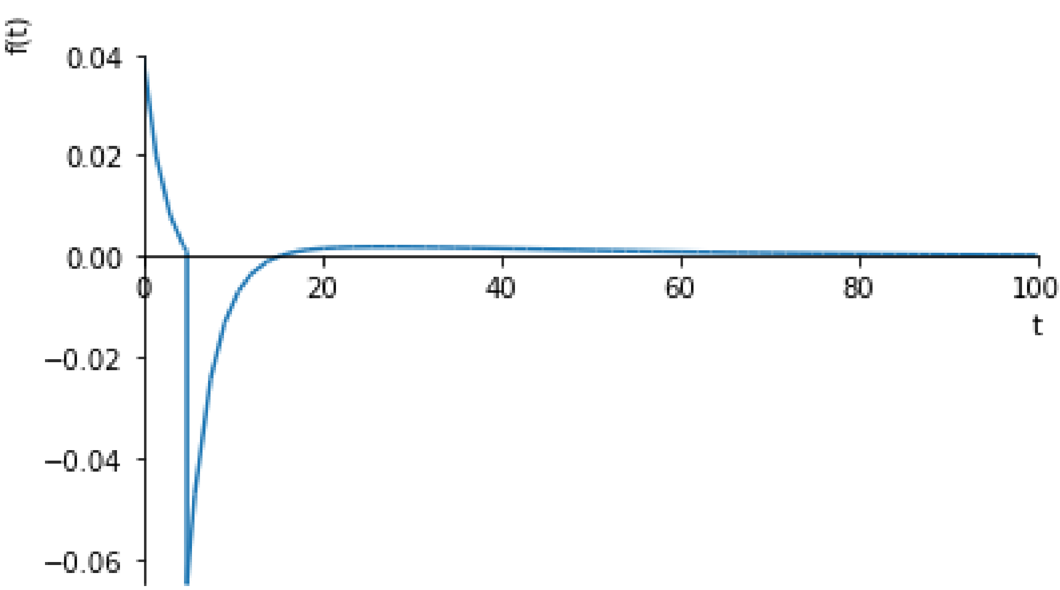


Out[92]: <sympy.plotting.plot.Plot at 0x7f4620ddb290>

In [93]:
```
C_prime = diff(C,t)
```

In [94]:
```
C_prime_check = N(C_prime.subs(averageEstimateDict))
```

In [95]:
```
plot(C_prime_check,(t,0,100))
```

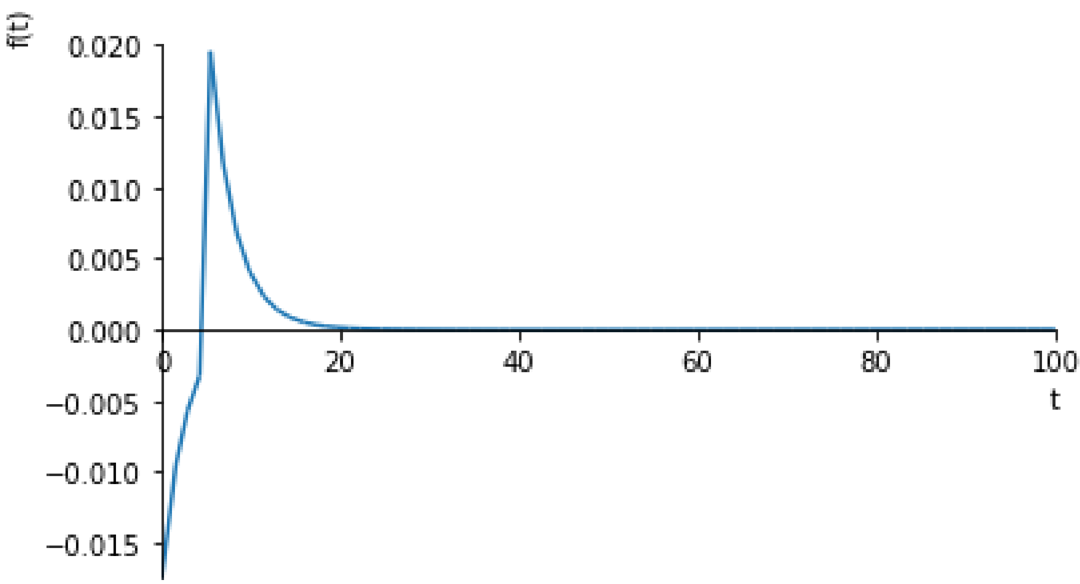


Out[95]: <sympy.plotting.plot.Plot at 0x7f46216e6d10>

```
In [96]: C_prime_prime = diff(C_prime,t)

In [97]: C_prime_prime_check = N(C_prime_prime.subs(averageEstimateDict))

In [98]: plot(C_prime_prime_check,(t,0,100))
```



Out[98]: <sympy.plotting.plot.Plot at 0x7f46216f7d50>

```
In [99]: #simplify(C.subs(t,log(T)))
```

```
    ---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call last)

    <ipython-input-99-9e24bd7593e4> in <module>()
----> 1 simplify(C.subs(t,log(T)))


    NameError: name 'T' is not defined
```

```
In [ ]: #C.subs(delta_i, 1/g_e)
```

```
In [ ]: #x, denominator = cse(simplify(C_prime.as_numer_denom()))
```

```
In [ ]: #T = symbols('T')
```

```
In [105]: simplify(C_prime)
```

Out[105]:

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

Explicit solving this equation doesn't work

1.0.10 Trying to use lambert function

```
In [120]: a,b,c,d = -t/e_r, -t/e_d, -(t - delta_i)/i_r, -(t - delta_i)/i_d
```

```
In [133]: alpha_star
```

Out[133]:

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

```
In [121]: W = lambda z: z*exp(z)
```

```
In [151]: LambertW(W(a))
```

Out[151]:

$$\operatorname{LambertW}\left(-\frac{t}{\tau_{er}e^{\frac{t}{\tau_{er}}}}\right)$$

```
In [158]: C = g_e*(exp(LambertW(W(a)) - exp(LambertW(W(b))))/alpha_star - g_i*(exp(LambertW(W(c
```

```
In [159]: C
```

Out[159]:

$$\frac{\bar{g}_e\left(-e^{\operatorname{LambertW}\left(-\frac{t}{\tau_{ed}e^{\frac{t}{\tau_{ed}}}}\right)}+e^{\operatorname{LambertW}\left(-\frac{t}{\tau_{er}e^{\frac{t}{\tau_{er}}}}\right)}\right)}{-\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed}-\tau_{er}}}+\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed}-\tau_{er}}}}-\frac{\bar{g}_i}{-\left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{id}}{\tau_{id}-\tau_{ir}}}+\left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{ir}}{\tau_{id}-\tau_{ir}}}}\left(-e^{\operatorname{LambertW}\left(\frac{1}{\tau_{id}}(\delta_i-t)e^{\frac{1}{\tau_{id}}(\delta_i-t)}\right)}+e^{\text{La}}\right.$$

```
In [163]: C.diff(t)
```

Out[163]:

$$\frac{\bar{g}_e}{-\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed}-\tau_{er}}}+\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed}-\tau_{er}}}}\left(\frac{\tau_{ed}e^{\frac{t}{\tau_{ed}}}e^{\operatorname{LambertW}\left(-\frac{t}{\tau_{ed}e^{\frac{t}{\tau_{ed}}}}\right)}\operatorname{LambertW}\left(-\frac{t}{\tau_{ed}e^{\frac{t}{\tau_{ed}}}}\right)}{t\left(\operatorname{LambertW}\left(-\frac{t}{\tau_{ed}e^{\frac{t}{\tau_{ed}}}}\right)+1\right)}\left(-\frac{1}{\tau_{ed}e^{\frac{t}{\tau_{ed}}}}+\frac{t}{\tau_{ed}^2e^{\frac{t}{\tau_{ed}}}}\right)-\frac{\tau_{er}e^{\frac{t}{\tau_{er}}}e^{\operatorname{Lam}}}{t\,(}\right.$$

In [164]: t_star = solve(expand(C.diff(t)),t)

```
---------------------------------------------------------------------------

KeyboardInterrupt                         Traceback (most recent call last)

<ipython-input-164-50cdcd1e9cd8> in <module>()
----> 1 t_star = solve(expand(C.diff(t)),t)


/usr/local/lib/python2.7/dist-packages/sympy/solvers/solvers.pyc in solve(f, *symbols, *
   1051     ###########################################################################
   1052     if bare_f:
-> 1053         solution = _solve(f[0], *symbols, **flags)
   1054     else:
   1055         solution = _solve_system(f, symbols, **flags)


/usr/local/lib/python2.7/dist-packages/sympy/solvers/solvers.pyc in _solve(f, *symbols,
   1609         flags.pop('tsolve', None)  # allow tsolve to be used on next pass
   1610         try:
-> 1611             soln = _tsolve(f_num, symbol, **flags)
   1612             if soln is not None:
   1613                 result = soln


/usr/local/lib/python2.7/dist-packages/sympy/solvers/solvers.pyc in _tsolve(eq, sym, **f
   2522             # it's time to try factoring; powdenest is used
   2523             # to try get powers in standard form for better factoring
-> 2524             f = factor(powdenest(lhs - rhs))
   2525             if f.is_Mul:
   2526                 return _solve(f, sym, **flags)


/usr/local/lib/python2.7/dist-packages/sympy/polys/polytools.pyc in factor(f, *gens, **a
   6061
   6062     try:
-> 6063         return _generic_factor(f, gens, args, method='factor')
```

```
   6064     except PolynomialError as msg:
   6065         if not f.is_commutative:


 /usr/local/lib/python2.7/dist-packages/sympy/polys/polytools.pyc in _generic_factor(expr
   5753     options.allowed_flags(args, [])
   5754     opt = options.build_options(gens, args)
-> 5755     return _symbolic_factor(sympify(expr), opt, method)
   5756
   5757


 /usr/local/lib/python2.7/dist-packages/sympy/polys/polytools.pyc in _symbolic_factor(exp
   5698         if hasattr(expr,'_eval_factor'):
   5699             return expr._eval_factor()
-> 5700         coeff, factors = _symbolic_factor_list(together(expr), opt, method)
   5701         return _keep_coeff(coeff, _factors_product(factors))
   5702     elif hasattr(expr, 'args'):


 /usr/local/lib/python2.7/dist-packages/sympy/polys/polytools.pyc in _symbolic_factor_lis
   5666             func = getattr(poly, method + '_list')
   5667
-> 5668             _coeff, _factors = func()
   5669             if _coeff is not S.One:
   5670                 if exp.is_Integer:


 /usr/local/lib/python2.7/dist-packages/sympy/polys/polytools.pyc in factor_list(f)
   3097         if hasattr(f.rep, 'factor_list'):
   3098             try:
-> 3099                 coeff, factors = f.rep.factor_list()
   3100             except DomainError:
   3101                 return S.One, [(f, 1)]


 /usr/local/lib/python2.7/dist-packages/sympy/polys/polyclasses.pyc in factor_list(f)
    757     def factor_list(f):
    758         """Returns a list of irreducible factors of ``f``. """
--> 759         coeff, factors = dmp_factor_list(f.rep, f.lev, f.dom)
    760         return coeff, [ (f.per(g), k) for g, k in factors ]
    761


 /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_factor_list(f,
   1277         if K.is_ZZ:
   1278             levels, f, v = dmp_exclude(f, u, K)
-> 1279             coeff, factors = dmp_zz_factor(f, v, K)
```

```
   1280
   1281             for i, (f, k) in enumerate(factors):


/usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_factor(f, u
   1089     if dmp_degree(g, u) > 0:
   1090         g = dmp_sqf_part(g, u, K)
-> 1091         H = dmp_zz_wang(g, u, K)
   1092         factors = dmp_trial_division(f, H, u, K)
   1093


/usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_wang(f, u,
   1012     try:
   1013         f, H, LC = dmp_zz_wang_lead_coeffs(f, T, cs, E, H, A, u, K)
-> 1014         factors = dmp_zz_wang_hensel_lifting(f, H, LC, A, p, u, K)
   1015     except ExtraneousFactors:  # pragma: no cover
   1016         if query('EEZ_RESTART_IF_NEEDED'):


/usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_wang_hensel
    876             if not dmp_zero_p(C, w - 1):
    877                 C = dmp_quo_ground(C, K.factorial(k + 1), w - 1, K)
--> 878                 T = dmp_zz_diophantine(G, C, I, d, p, w - 1, K)
    879
    880                 for i, (h, t) in enumerate(zip(H, T)):


/usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


/usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


/usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    821             if not dmp_zero_p(C, v):
    822                 C = dmp_quo_ground(C, K.factorial(k + 1), v, K)
--> 823                 T = dmp_zz_diophantine(G, C, A, d, p, v, K)
```

```
    824
    825                 for i, t in enumerate(T):


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    821             if not dmp_zero_p(C, v):
    822                 C = dmp_quo_ground(C, K.factorial(k + 1), v, K)
--> 823                 T = dmp_zz_diophantine(G, C, A, d, p, v, K)
    824
    825                 for i, t in enumerate(T):


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    821             if not dmp_zero_p(C, v):
    822                 C = dmp_quo_ground(C, K.factorial(k + 1), v, K)
--> 823                 T = dmp_zz_diophantine(G, C, A, d, p, v, K)
    824
    825                 for i, t in enumerate(T):


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
```

```
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    821             if not dmp_zero_p(C, v):
    822                 C = dmp_quo_ground(C, K.factorial(k + 1), v, K)
--> 823                 T = dmp_zz_diophantine(G, C, A, d, p, v, K)
    824
    825                 for i, t in enumerate(T):


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    801         v = u - 1
    802
--> 803         S = dmp_zz_diophantine(G, C, A, d, p, v, K)
    804         S = [ dmp_raise(s, 1, v, K) for s in S ]
    805


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    821             if not dmp_zero_p(C, v):
    822                 C = dmp_quo_ground(C, K.factorial(k + 1), v, K)
--> 823                 T = dmp_zz_diophantine(G, C, A, d, p, v, K)
    824
    825                 for i, t in enumerate(T):


    /usr/local/lib/python2.7/dist-packages/sympy/polys/factortools.pyc in dmp_zz_diophantine
    794
    795         for f in F:
--> 796             B.append(dmp_quo(e, f, u, K))
    797             G.append(dmp_eval_in(f, a, n, u, K))
    798


    /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_quo(f, g, u, K)
   1666
   1667     """
-> 1668     return dmp_div(f, g, u, K)[0]
```

```
   1669
   1670


 /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_div(f, g, u, K)
   1624         return dmp_ff_div(f, g, u, K)
   1625     else:
-> 1626         return dmp_rr_div(f, g, u, K)
   1627
   1628


 /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_rr_div(f, g, u,
   1390     while True:
   1391         lc_r = dmp_LC(r, K)
-> 1392         c, R = dmp_rr_div(lc_r, lc_g, v, K)
   1393
   1394         if not dmp_zero_p(R, v):


 /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_rr_div(f, g, u,
   1398
   1399         q = dmp_add_term(q, c, j, u, K)
-> 1400         h = dmp_mul_term(g, c, j, u, K)
   1401         r = dmp_sub(r, h, u, K)
   1402


 /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_mul_term(f, c,
    185         return dmp_zero(u)
    186     else:
--> 187         return [ dmp_mul(cf, c, v, K) for cf in f ] + dmp_zeros(i, v, K)
    188
    189


 /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_mul(f, g, u, K)
    829
    830         for j in range(max(0, i - dg), min(df, i) + 1):
--> 831             coeff = dmp_add(coeff, dmp_mul(f[j], g[i - j], v, K), v, K)
    832
    833         h.append(coeff)


 /usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_mul(f, g, u, K)
    829
    830         for j in range(max(0, i - dg), min(df, i) + 1):
--> 831             coeff = dmp_add(coeff, dmp_mul(f[j], g[i - j], v, K), v, K)
```

```
832
833         h.append(coeff)


/usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_mul(f, g, u, K)
829
830         for j in range(max(0, i - dg), min(df, i) + 1):
--> 831             coeff = dmp_add(coeff, dmp_mul(f[j], g[i - j], v, K), v, K)
832
833         h.append(coeff)


/usr/local/lib/python2.7/dist-packages/sympy/polys/densearith.pyc in dmp_mul(f, g, u, K)
829
830         for j in range(max(0, i - dg), min(df, i) + 1):
--> 831             coeff = dmp_add(coeff, dmp_mul(f[j], g[i - j], v, K), v, K)
832
833         h.append(coeff)


KeyboardInterrupt:
```

```
In [ ]: a
```

```
In [141]: exp(a).diff(t)
```

Out[141]:

$$-\frac{1}{\tau_{er}e^{\frac{t}{\tau_{er}}}}$$

```
In [140]: mpmath.lambertw(1)
```

```
Out[140]: mpf('0.56714329040978384')
```

```
In [138]: C.diff(t)
```

Out[138]:

$$\frac{\bar{g}_e\left(-\frac{1}{\tau_{er}e^{\frac{t}{\tau_{er}}}}+\frac{1}{\tau_{ed}e^{\frac{t}{\tau_{ed}}}}\right)}{-\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{ed}}{\tau_{ed}-\tau_{er}}}+\left(\frac{\tau_{er}}{\tau_{ed}}\right)^{\frac{\tau_{er}}{\tau_{ed}-\tau_{er}}}}-\frac{\bar{g}_i\left(-\frac{1}{\tau_{ir}}e^{\frac{1}{\tau_{ir}}(\delta_i-t)}+\frac{1}{\tau_{id}}e^{\frac{1}{\tau_{id}}(\delta_i-t)}\right)}{-\left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{id}}{\tau_{id}-\tau_{ir}}}+\left(\frac{\tau_{ir}}{\tau_{id}}\right)^{\frac{\tau_{ir}}{\tau_{id}-\tau_{ir}}}}$$

```
In [131]: powsimp(C.subs({-t/e_r:a, -t/e_d:b, -(t - delta_i)/i_r:c, -(t - delta_i)/i_d:d}))
```

Out[131]:

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

In [132]:

Out[132]:

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

```
In [129]: piecewise_C_star = simplify(ratsimp(factor(C_prime))).args
```

```
In [ ]: C_star_1 = simplify(piecewise_C_star[1][0])
```

```
In [ ]: C_star_1.args
```

```
In [ ]: simplify(solveset(C_star_1.args[5],t).doit())
```

```
In [ ]: factor(C_star_1).collect(exp(t)).args
```

```
In [ ]: expand_log(factor(C_prime))
```

```
In [ ]: denominator
```

```
In [ ]: -x[5]/((i_d*exp(x[13]*x[9])) - (i_r*(exp(x[13]/i_d))))
```

```
In [ ]: j,k = symbols({'J','K'})
```

```
In [ ]: new_eq = simplify(C_prime.subs({e_d:e_r*((j+1)/(j-1)), i_d:i_r*((k+1)/(k-1))}))
```

```
In [ ]: refine(powsimp(new_eq.as_numer_denom()))
```

```
In [ ]: eq_1 = latex("\bar{g}_e*\tau_{er}*(-P*\rho_e*\rho_i**(\rho_i/(\rho_i - 1) - 1 + 1/(\rho_
```

```
In [ ]: eq_1
```

```
In [ ]: C.diff(t).diff(t)
```

```
In [ ]: t_star = (delta_i - (b*e_r*log((P*(rho_i+1)*rho_e)/((rho_e+1)*rho_i))))/(b+1)
```

```
In [ ]: t_star.subs(averageEstimateDict)
```

```
In [ ]: N(delta_i.subs(averageEstimateDict))
```

1.0.11 Unfortunately this is not possible: Since the τ_{decay} will not contribute to the first peak, we can eliminate them.

```
In [ ]: erise = Piecewise((0, t < delta_e), (g_e * (exp(-(t-delta_e)/e_r)/alpha_star), True))
```

```
In [ ]: efall = Piecewise((0, t < delta_e), (g_e * (exp(-(t-delta_e)/e_d)/alpha_star), True))
```

```
In [ ]: irise = erise.subs({g_e: g_i, rho_e: rho_i, e_r:i_r, e_d: i_d, delta_e: delta_i})
```

```
In [ ]: ifall = efall.subs({g_e: g_i, rho_e: rho_i, e_r:i_r, e_d: i_d, delta_e: delta_i})
```

```
In [ ]: C = C.subs({g_i: g_e*P, i_r : e_r*b}) # Replacing g_i with P*ge
        C = C.subs({delta_e:0})
```

```
In [ ]: C
```

```
In [ ]: C_check = C.subs({P:0, delta_i:2})
```

```
In [ ]: C_check
```

```
In [ ]: C
```

```
In [ ]: C.diff(t)
```

```
In [ ]: averageEstimateDict
```

```
In [ ]: C_check = N(C.subs(averageEstimateDict))
```

```
In [ ]: C_check
```

1.0.12 Verifying that C behaves

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

```
In [ ]: plot(erise.subs(averageEstimateDict),(t,0,100))
```

```
In [ ]: plot(((efall-erise)-(-irise)).subs(averageEstimateDict),(t,0,100))
```

```
In [ ]: plot(((efall-erise)-(ifall-irise)).subs(averageEstimateDict),(t,0,100))
```

```
In [ ]: plot((E-I).subs(averageEstimateDict),(t,0,100))
```

```
In [ ]: C_check = N(C.subs({rho_e:7, rho_i: 15}))
```

```
In [ ]: C_check
```

```
In [ ]: C_check = C_check.subs(averageEstimateDict)
```

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

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

```
In [ ]: C_prime_check = N(C_prime.subs(averageEstimateDict))
```

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

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

```
In [ ]: C_prime_prime_check = N(C_prime_prime.subs(averageEstimateDict))
```

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

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

In [ ]:

In [ ]:

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])

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

1.0.13 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)

In [ ]: log(-2)

In [ ]: