Sensitivity Analysis of Biochemical Systems Using Bond Graphs: Additional Material for Pentose Phosphate Pathway

Peter Gawthrop. *peter.gawthrop@unimelb.edu.au*

April 3, 2023

Contents

1 Introduction

This notebook generates the figures for the paper: Sensitivity Analysis of Biochemical Systems Using Bond Graphs - Pentose Phosphate Pathway Example

2 Set up the code

2.1 Import packages

[1]:
```
## Some useful imports
import BondGraphTools as bgt
import numpy as np
```

```
import sympy as sp
import matplotlib.pyplot as plt
import control as con
import copy
import importlib
import time

## For reimporting: use imp.reload(module)
import importlib as imp

# ## For path etc: sys.path
import sys
sys.path.append("/home/peterg/WORK/Research/SystemsBiology/lib/python")

#
## Stoichiometric analysis
import stoich as st

## SVG
import svgBondGraph as sbg

## Stoichiometry to BG
import stoichBondGraph as stbg

## Sloppy parameters
import sloppy as slp

## Display (eg disp.SVG(), disp.
import IPython.display as disp

##
quiet = True
Plotting = False
Titles = False


## Normalisation
Normalise = True
```

```
Warning - scikit.odes not found. Simulations are disabled.
```

[2]:
```
def tfProps(tf,method='truncate'):

    ## Steady-state gain
    g = con.dcgain(tf)

    if not con.issiso(tf):
        g = g[0][0]

    ## Time constant
    ## Check if direct link
```

```
    direct_link = np.any(tf.D)
    ## Only set tau=0 if siso
    if direct_link and con.issiso(tf):
        ## Instant response
        tau = 0
    else:
        ## Reduce to first-order to estimate time constant
        ## Note that method='matchdc' can give a kernel crash - use 'truncate'
        tf1  = con.balred(sys,orders=1,method=method)
        poles = con.poles(tf1)
        realPoles = np.real(poles)
        tau = -1/min(realPoles)

    return g,tau
```

[3]:
```
## High frequency gain function (initial response to step)
def hfgain(sys):

    tf = con.ss2tf(sys)
    num = tf.num[0][0]
    den = tf.den[0][0]
    n = len(den)
    m = len(num)
#     print(den)
#     print(num)
#     print(n,m)

    if n==m:
        g = num[0]/den[0]
    elif n>m:
        g = 0
    elif n<m:
        g = float('inf')

    return g


# g = hfgain(sys)
# print(g)
```

[4]:
```
## Optional plotting
def Savefig(name,fontsize=20):
    if Plotting:
        plotname = 'Figs/'+name+'.pdf'
        print('Saving',plotname)
        plt.rcParams.update({'font.size': fontsize})
        plt.tight_layout()
        plt.savefig(plotname)
        plt.clf()
```

2.2 Steady-state by simulation

[5]:
```
## Steady-state by simulation
def SteadyState(s,sc,sf=None,parameter={},X0=None,t_ss=1000):
    t = np.linspace(0,t_ss)
    dat = st.
 ↪sim(s,sc=sc,sf=sf,t=t,parameter=parameter,X0=X0,tol=1e-4,quiet=quiet)
    X_ss = dat['X'][-1]
    V_ss = dat['V'][-1]
    #dX_ss = sc['N']@V_ss
    dX_ss = dat['dX'][-1]
    dXc_ss = dat['dXc'][-1]
    print(f'Steady-state finder error: {np.linalg.norm(dXc_ss):.2e}')
    return X_ss,V_ss
```

2.3 Stoichiometry

[6]:
```
def␣
 ↪Stoichiometry(model,chemostats=[],flowstats=[],CommonSpecies=None,sensitivity=False):
 ↪

    ## Stoichiometry
    s = st.stoich(model,quiet=quiet)

    ## Unify species
    if not (CommonSpecies is None):
        commonSpecies = st.merge(s,CommonSpecies=CommonSpecies)
#         print(commonSpecies)
        st.unify(s,commonSpecies=commonSpecies)

    ## Sensitivity
    if sensitivity:
        extra = st.stoichSensitivity(s)
    else:
        extra = []

#     print(chemostats+extra)

    ## Chemostats and flowstats
    sc = st.statify(s,chemostats=chemostats+extra)
    sf = st.statify(s,flowstats=flowstats)

    return s,sc,sf
```

2.4 Linearisation

[7]:
```
def Linear(s,sc,sf=None,parameter={},X0=None, invar='X', outvar = 'V',␣
 ↪quiet=False):

    ## Steady state
```

```
    if X0 is None:
        X0 = np.ones(s['n_X'])
#     print(len(X0))

    X_ss,V_ss = SteadyState(s,sc,sf=sf,parameter=parameter,X0=X0)
    dX_ss = s['N']@V_ss
#     print('X_ss =',X_ss)
    print('V_ss =', V_ss)
#     print('dX_ss =', dX_ss)

    ## Linearise
    sys = st.lin(s,sc,sf=sf,x_ss=X_ss,parameter=parameter,
                 invar=invar,outvar=outvar,quiet=quiet)

    return sys,X_ss,V_ss,dX_ss
```

2.5 Extract subsystem from linear system

[8]:
```
def Index(A,a):

    I = []
    for aa in a:
        i = A.index(aa)
        I.append(i)
    return np.array(I)

def zapSmall(x,tol=1e-10,quiet=True):

    xx = np.zeros(len(x))
    for i,val in enumerate(x):
        if abs(val)>tol:
            xx[i] = x[i]
        else:
            if not quiet:
                print(f'Setting {i}th coefficient {val:.2} to zero')
    return xx

def␣
 ↪extractSubsystem(SYS,sc,sf,inp,outp,tol=1e-3,order=None,minreal=False,quiet=False):
 ↪

    Sys = copy.copy(SYS)
    chemostats = sc['chemostats']
    if sf is None:
        flowstats = []
    else:
        flowstats = sf['flowstats']
    species = sc['species']
    reaction = sc['reaction']
```

```
    ## Index of input and output
    if inp[0] in chemostats:
        i_inp = Index(chemostats,inp)
#         print('Input:',i_inp,chemostats[i_inp[0]])
    else:
        i_inp = Index(flowstats,inp)+len(chemostats)

#     print(i_inp)

    if outp[0] in chemostats:
#         i_outp = Index(chemostats,outp)
        i_outp = Index(species,outp)
    elif outp[0] in species:
        i_outp = Index(species,outp)
    else:
        if outp[0] in reaction:
            i_outp = Index(reaction,outp)
        else:
            print(f'Output {outp} does not exist')

    ## Extract tf
    n_y = len(i_outp)
    n_u = len(i_inp)
    nn = Sys.A.shape
    n_x = nn[0]
    #print(n_x)

    sys = con.ss(Sys.A,
                 Sys.B[:,i_inp].reshape(n_x,n_u),
                 Sys.C[i_outp,:].reshape(n_y,n_x),
                 Sys.D[i_outp][:,i_inp].reshape(n_y,n_u))

    if minreal:
        sys = con.minreal(sys,tol=tol,verbose=True)

    ## Reduce order
    if not (order is None):
        sys = con.balred(sys,order,method='matchdc')

    return sys
```

2.6 Plotting

[9]:
```
def␣
↪plotSensitivity(dat,reactions=['r1','r2'],plotSim=True,name=None,labeling=True):
↪

    if plotSim:
        plt.plot(t,y_step,color='black')
```

```
    else:
        plt.plot(t,y_step,label=label)

    for reac in reactions:
        i = s['reaction'].index(reac)
        if plotSim:
#             label = reac + r' ($\tilde\lambda$ = ' + f'{lam-1:0.2f})'
            pc = int(round(100*(lam-1)))
            if labeling:
                label = f'{reac} ({pc}%)'
            else:
                label = None
            plt.plot(t,(dat['V'][:,i]-V_ss[i])/(lam-1),
                     lw=5,ls='dashed',label=label)

#     plt.grid()
    if labeling:
        plt.legend()
    plt.xlabel('$t$')
    plt.ylabel(r'$\tilde{v}/\tilde{\lambda}$')
#     plt.ylim(bottom=0)
#     plt.xlim(left=0)
#     plt.tight_layout()
```

[10]:
```
def␣
↪plotSensitivitydX(dat,species=['A','C'],plotSim=True,name=None,labeling=True,setZero=False
↪
    if plotSim:
        plt.plot(t,y_step,color='black')
    else:
        plt.plot(t,y_step,label=label)

    for spec in species:
        print(spec)
        i_s = s['species'].index(spec)
        if plotSim:
            pc = int(round(100*(lam-1)))
            if labeling:
                label = f'{spec} ({pc}%)'
            else:
                label=None
            plt.plot(t,(dat['dX'][:,i_s]-dX_ss[i_s])/(lam-1),
                     lw=5,ls='dashed',label=label)

#     plt.grid()
    if labeling:
        plt.legend()
    plt.xlabel('$t$')
    plt.ylabel(r'$\Delta \mathbf{f} /\Delta \lambda$')
```

```
    #plt.ylabel(r'$\tilde{\dot{x}}/\tilde{\lambda}$')
    if setZero:
        plt.ylim(bottom=0)
        plt.xlim(left=0)
#     plt.tight_layout()
```

[11]:
```
def plotLines(lw=4,ls='dotted'):
        plt.hlines(g,min(t),max(t),color='black',ls=ls,lw=lw)
        plt.vlines(tau,min(y_step.flatten()),1.1*max(y_step.
 ↪flatten()),color='black',ls=ls,lw=lw)
```

3 Normalisation constants

[12]:
```
T_human = 37 # Human body temperature
K_0 = 273.15
print(f'T_human = {T_human} degC = {T_human+K_0} K')

mu_0 = RT = st.RT(T_cent=T_human)
print(f'mu_0 = {mu_0*1e-3:0.3f} kJ/mol')

F = st.F() # Faraday's constant
print(f'F = {F*1e-3:0.2f} kC/mol')

V_0 = RT/F
print(f'V_0 = {V_0*1e3:0.2f} mV')

P_0 = 1e-3

v_0 = P_0/mu_0
print(f'v_0 = {v_0*1e6:0.4f} micro mol /s')

i_0 = F*v_0
print(f'i_0 = {i_0*1e3:0.2f} mA')
```

```
T_human = 37 degC = 310.15 K
mu_0 = 2.579 kJ/mol
F = 96.49 kC/mol
V_0 = 26.73 mV
v_0 = 0.3878 micro mol /s
i_0 = 37.42 mA
```

4 Stoichiometric approach

[13]:
```
def simSensitivity(s,sc,sf,Sys,X_ss,V_ss,dX_ss,chemostats=['A','C'],
                   inp=['sr1'],outp=['r1','r2'],t_last=2,parameter={},lam=1.
 ↪2,order=None,tol=None):

    ## Extract sensitivity system
```

```
    sys = extractSubsystem(Sys,sc,sf,inp=inp,outp=outp,order=order,tol=tol)

    ## Time
    t = np.linspace(0,t_last,500)

    ## Sensitivity step response
    step = con.step_response(sys,T=t)
    y_step = step.y[:,0,:].T

    ## Exact simulation with changed parameter or state
    X_ss_1 = copy.copy(X_ss)
    parameter1 = copy.copy(parameter)
    inComp = inp[0][1:]

    if inComp in chemostats:
        ## Perturb state
        iComp = s['species'].index(inComp)
        X_ss_1[iComp] = lam*X_ss_1[iComp]
    else:
        ## Perturb parameter
        if inComp in s['species']:
            parname = 'K_'+inComp
        else:
            parname = 'kappa_'+inComp

        print(parname)

        if parname in list(parameter.keys()):
            parameter1[parname] = lam*parameter[parname]
        else:
            parameter1[parname] = lam

    dat = st.
↪sim(s,sc=sc,sf=sf,t=t,parameter=parameter1,X0=X_ss_1,quiet=quiet,Normalise=Normalise)

    return dat,y_step,t,sys
```

5 Import Pentose Phosphate Pathway information from:

Peter J. Gawthrop, Michael Pan, and Edmund J. Crampin. *Modular dynamic biomolecular modelling with bond graphs: the unification of stoichiometry, thermodynamics*, kinetics and data. Journal of The Royal Society Interface, 18(181):20210478, 2021. doi:10.1098/rsif.2021.0478.

[14]:
```
## Load data
import pickle
filename = 'GlyPPP.pic'
datafile = open(filename, 'rb')
GlyPPP = pickle.load(datafile)

s0 = GlyPPP['s']
```

```
parameter = GlyPPP['parameter']
print(parameter)

sc0 = GlyPPP['sc']
chemostats = sc0['chemostats']
print(chemostats)

conc = GlyPPP['conc']

import json
print(json.dumps(parameter,indent=2))
```

```
{'kappa_PGI': 154.38997002578546, 'kappa_PFK': 54.85043355391926, 'kappa_FBA':
160.07691981318658, 'kappa_TPI': 353.9323765134668, 'kappa_G6PDH2R':
4.667672687418436, 'kappa_PGL': 291.6414393949941, 'kappa_GND':
1.2695594575280742, 'kappa_RPI': 4206.975085828208, 'kappa_TKT2':
9.170830408091838, 'kappa_TALA': 1.6563170906188465, 'kappa_TKT1':
8.819978413932159, 'kappa_RPE': 96.07053469867635, 'K_6PGC': 6.233520699319548,
'K_6PGL': 1.0397283862123778, 'K_ADP': 5.154605959675528, 'K_ATP':
2.2539233202427162, 'K_CO2': 33.794182951886775, 'K_DHAP': 1.778985981513151,
'K_E4P': 57.93534155565906, 'K_F6P': 1.41401983409913, 'K_FDP':
0.38795235349448437, 'K_G3P': 14.902044199279699, 'K_G6P': 0.8377378837717493,
'K_H': 0.3491739027654742, 'K_H2O': 1.0397283862123778, 'K_NADP':
11747.063287701598, 'K_NADPH': 21.002665768445333, 'K_R5P': 12.241945837528652,
'K_RU5PD': 86.15506601742248, 'K_S7P': 21.751324335743256, 'K_XU5PD':
51.68293204643235}
['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP', 'NADPH', 'R5P', 'G3P']
{
  "kappa_PGI": 154.38997002578546,
  "kappa_PFK": 54.85043355391926,
  "kappa_FBA": 160.07691981318658,
  "kappa_TPI": 353.9323765134668,
  "kappa_G6PDH2R": 4.667672687418436,
  "kappa_PGL": 291.6414393949941,
  "kappa_GND": 1.2695594575280742,
  "kappa_RPI": 4206.975085828208,
  "kappa_TKT2": 9.170830408091838,
  "kappa_TALA": 1.6563170906188465,
  "kappa_TKT1": 8.819978413932159,
  "kappa_RPE": 96.07053469867635,
  "K_6PGC": 6.233520699319548,
  "K_6PGL": 1.0397283862123778,
  "K_ADP": 5.154605959675528,
  "K_ATP": 2.2539233202427162,
  "K_CO2": 33.794182951886775,
  "K_DHAP": 1.778985981513151,
  "K_E4P": 57.93534155565906,
  "K_F6P": 1.41401983409913,
  "K_FDP": 0.38795235349448437,
  "K_G3P": 14.902044199279699,
```

```
  "K_G6P": 0.8377378837717493,
  "K_H": 0.3491739027654742,
  "K_H2O": 1.0397283862123778,
  "K_NADP": 11747.063287701598,
  "K_NADPH": 21.002665768445333,
  "K_R5P": 12.241945837528652,
  "K_RU5PD": 86.15506601742248,
  "K_S7P": 21.751324335743256,
  "K_XU5PD": 51.68293204643235
}
```

[15]:
```
## Reactions
disp.Latex(st.sprintrl(s0,chemformula=True))
```

[15]:

$$\mathrm{G_6P} \rightleftharpoons^{\mathrm{PGI}} \mathrm{F_6P} \tag{1}$$

$$\mathrm{ATP} + \mathrm{F_6P} \rightleftharpoons^{\mathrm{PFK}} \mathrm{ADP} + \mathrm{FDP} + \mathrm{H} \tag{2}$$

$$\mathrm{FDP} \rightleftharpoons^{\mathrm{FBA}} \mathrm{DHAP} + \mathrm{G_3P} \tag{3}$$

$$\mathrm{DHAP} \rightleftharpoons^{\mathrm{TPI}} \mathrm{G_3P} \tag{4}$$

$$\mathrm{G_6P} + \mathrm{NADP} \rightleftharpoons^{\mathrm{G_6PDH_2R}} {}_6\mathrm{PGL} + \mathrm{H} + \mathrm{NADPH} \tag{5}$$

$${}_6\mathrm{PGL} + \mathrm{H_2O} \rightleftharpoons^{\mathrm{PGL}} {}_6\mathrm{PGC} + \mathrm{H} \tag{6}$$

$${}_6\mathrm{PGC} + \mathrm{NADP} \rightleftharpoons^{\mathrm{GND}} \mathrm{CO_2} + \mathrm{NADPH} + \mathrm{RU_5PD} \tag{7}$$

$$\mathrm{RU_5PD} \rightleftharpoons^{\mathrm{RPI}} \mathrm{R_5P} \tag{8}$$

$$\mathrm{E_4P} + \mathrm{XU_5PD} \rightleftharpoons^{\mathrm{TKT_2}} \mathrm{F_6P} + \mathrm{G_3P} \tag{9}$$

$$\mathrm{G_3P} + \mathrm{S_7P} \rightleftharpoons^{\mathrm{TALA}} \mathrm{E_4P} + \mathrm{F_6P} \tag{10}$$

$$\mathrm{R_5P} + \mathrm{XU_5PD} \rightleftharpoons^{\mathrm{TKT_1}} \mathrm{G_3P} + \mathrm{S_7P} \tag{11}$$

$$\mathrm{RU_5PD} \rightleftharpoons^{\mathrm{RPE}} \mathrm{XU_5PD} \tag{12}$$

[16]:
```
## Pathway reactions
sp0 = st.path(s0,sc0)
disp.Latex(st.sprintrl(sp0,chemformula=True))
```

[16]:

$$\mathrm{ATP} + \mathrm{G_6P} \rightleftharpoons^{\mathrm{pr_1}} \mathrm{ADP} + 2\,\mathrm{G_3P} + \mathrm{H} \tag{13}$$

$$\mathrm{G_6P} + \mathrm{H_2O} + 2\,\mathrm{NADP} \rightleftharpoons^{\mathrm{pr_2}} \mathrm{CO_2} + 2\,\mathrm{H} + 2\,\mathrm{NADPH} + \mathrm{R_5P} \tag{14}$$

$$2\,\mathrm{H_2O} + 4\,\mathrm{NADP} + \mathrm{R_5P} \rightleftharpoons^{\mathrm{pr_3}} 2\,\mathrm{CO_2} + \mathrm{G_3P} + 4\,\mathrm{H} + 4\,\mathrm{NADPH} \tag{15}$$

5.1 Repeat simulations for sanity check

[17]:
```
def setPath(s,path='R5P'):

    print('\n Path =', path)

    if path == 'R5P':
        chemostats = ['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP',␣
↪'NADPH', 'R5P']
        flowstats =['G6PDH2R']
        dX_G6P = 5
    elif path == 'NADPH':
        chemostats = ['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP', 'NADPH']
        flowstats = []
        dX_G6P = 1
    elif path == 'both':
        chemostats = ['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP',␣
↪'NADPH', 'R5P']
        flowstats =  ['PGI', 'TKT2']
        dX_G6P = 1
    elif path == 'all':
        chemostats = ['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP',␣
↪'NADPH', 'R5P','G3P']
        flowstats = []
        dX_G6P = 10
    elif path == 'all_e4p':
        chemostats = ['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP',␣
↪'NADPH', 'R5P','G3P','E4P']
        flowstats = []
        dX_G6P = 10

    sc = st.statify(s,chemostats=chemostats)
    sf = st.statify(s,flowstats=flowstats)

    return sc,sf,dX_G6P
```

[18]:
```
## Data normalisation
c_0 = 7.88 # mM
f_0 = 0.9916666666666667 # mM/sec
t_0 = 7.946218487394957 # sec
```

[19]:
```
## Simulate for each path
s = copy.copy(s0) ## for compatibility
spec = s['species']
approximateFlowstats = True

Spec = ['G6P','R5P','NADPH','ADP','CO2','H','H2O']
#paths = ['all','both','R5P','NADPH']
#paths = ['R5P']
#paths = ['all','both','NADPH','R5P']
paths = ['both','NADPH','R5P','all_e4p','all']
```

```
RATIO = {}
for path in paths:
    Ratio = {}
    normalisedRatio = {}

    ## Set up pathway
    sc,sf,dX_G6P_0 = setPath(s,path=path)

    ## Print pathway
    sp = st.path(s,sc)
    print(st.sprintrl(sp))

    ## Set up parameters
    par = copy.copy(parameter)
    if approximateFlowstats:
        small = 1e-3
        par = copy.copy(parameter)
        for fs in sf['flowstats']:
            par['kappa_'+fs] = small
        sf = None

    ## Simulate
    t = np.linspace(0,3*t_0,1000)

#     ## Find steady-state with no flowstats
#     dat_ss = st.sim(s,sc=sc,sf=sf,t=t,parameter=parameter,X0=conc)
#     X_ss = dat_ss['X'][-1,:]

    dat = st.sim(s,sc=sc,sf=sf,t=t,parameter=par,X0=conc)
    #st.plot(s,dat,species=[])
#     st.plot(s,dat,reaction=[],species=Spec,dX=True)

    ## Extract some external flows
    DX = dat['dX']
    dX = {}
    for Sp in Spec:
        dX[Sp] = DX[:,spec.index(Sp)]
        Ratio[Sp] = -dX[Sp]/dX['G6P']
        normalisedRatio[Sp] =  -dX_G6P_0*dX[Sp]/dX['G6P']

    RATIO[path] = normalisedRatio


    ## Print steady-state values
    for Sp in Spec:
        ratio = Ratio[Sp][-1]
        print(f'{Sp}:\t{dX[Sp][0]:3.1f} \t{dX[Sp][-1]:3.
↪1f}\t{(dX_G6P_0*ratio):3.1f}\t{100*ratio:3.1f}%')
```

```
Path = both
```

```
\begin{align}
G6P + H2O + 2 NADP  &\Leftrightarrow CO2 + 2 H + 2 NADPH + R5P \\
ADP + 4 H2O + 8 NADP + 2 R5P  &\Leftrightarrow ATP + 4 CO2 + G6P + 7 H + 8 NADPH
\end{align}

G6P:    -11.6   -11.6   -1.0    -100.0%
R5P:    6.2     11.5    1.0     99.4%
NADPH:  23.2    23.2    2.0     200.0%
ADP:    63.1    0.0     0.0     0.4%
CO2:    11.6    11.6    1.0     100.0%
H:      86.3    23.2    2.0     200.4%
H2O:    -11.6   -11.6   -1.0    -100.0%

 Path = NADPH
\begin{align}
ADP + G6P + 6 H2O + 12 NADP  &\Leftrightarrow ATP + 6 CO2 + 11 H + 12 NADPH
\end{align}

G6P:    -71.1   -1.9    -1.0    -100.0%
R5P:    6.2     -0.0    -0.0    -0.0%
NADPH:  23.2    23.1    12.0    1200.0%
ADP:    63.1    -1.9    -1.0    -100.0%
CO2:    11.6    11.6    6.0     600.0%
H:      86.3    21.2    11.0    1100.0%
H2O:    -11.6   -11.6   -6.0    -600.0%

 Path = R5P
\begin{align}
G6P + H2O + 2 NADP  &\Leftrightarrow CO2 + 2 H + 2 NADPH + R5P \\
ADP + 4 H2O + 8 NADP + 2 R5P  &\Leftrightarrow ATP + 4 CO2 + G6P + 7 H + 8 NADPH
\end{align}

G6P:    -59.5   -2.4    -5.0    -100.0%
R5P:    6.2     2.9     6.0     120.0%
NADPH:  11.6    0.0     0.0     0.2%
ADP:    63.1    0.5     1.0     20.0%
CO2:    11.6    0.0     0.0     0.1%
H:      74.7    0.5     1.0     20.2%
H2O:    -11.6   -0.0    -0.0    -0.1%

 Path = all_e4p
\begin{align}
ATP + G6P  &\Leftrightarrow ADP + 2 G3P + H \\
G6P + H2O + 2 NADP  &\Leftrightarrow CO2 + 2 H + 2 NADPH + R5P \\
G3P + R5P  &\Leftrightarrow 2 E4P \\
E4P + H2O + 2 NADP  &\Leftrightarrow CO2 + G3P + 2 H + 2 NADPH
\end{align}

G6P:    -71.1   -71.1   -10.0   -100.0%
R5P:    6.2     6.2     0.9     8.7%
NADPH:  23.2    23.2    3.3     32.6%
```

```
ADP:    63.1    63.1    8.9     88.8%
CO2:    11.6    11.6    1.6     16.3%
H:      86.3    86.3    12.1    121.3%
H2O:    -11.6   -11.6   -1.6    -16.3%

 Path = all
\begin{align}
ATP + G6P  &\Leftrightarrow ADP + 2 G3P + H \\
G6P + H2O + 2 NADP  &\Leftrightarrow CO2 + 2 H + 2 NADPH + R5P \\
2 H2O + 4 NADP + R5P  &\Leftrightarrow 2 CO2 + G3P + 4 H + 4 NADPH
\end{align}

G6P:    -71.1   -71.1   -10.0   -100.0%
R5P:    6.2     6.2     0.9     8.7%
NADPH:  23.2    23.2    3.3     32.6%
ADP:    63.1    63.1    8.9     88.8%
CO2:    11.6    11.6    1.6     16.3%
H:      86.3    86.3    12.1    121.3%
H2O:    -11.6   -11.6   -1.6    -16.3%
```

[20]:
```
## Set up chemostats to correspond to above path
chemostats = sc['chemostats']
Chemostats = copy.copy(sc['chemostats'])
print(path)
print(chemostats)
```

```
all
['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP', 'NADPH', 'R5P', 'G3P']
```

[21]:
```
## Plot ratios
plottingRatios = False
if plottingRatios:
    name = ['i','ii','iii']
    for sp in ['R5P','NADPH']:
        BigFont = 24
        plt.rcParams.update({'font.size': BigFont})
        for i,path in enumerate(['both','R5P','NADPH']):
            Ratio = RATIO[path]
            label = f'Path {name[i]} ({path})'
            plt.plot(t/t_0,Ratio[sp],label=label,linewidth=5)
        if sp=='R5P':
            ylim = 8
        else:
            ylim=15
        plt.ylim((0,ylim))
        ylabel = r'$\rho_{'+sp+'}$'
        plt.ylabel(ylabel)
        plt.xlabel('$t/t_0$')
        plt.legend()
#         plt.grid()
    #     if SaveFig:
```

```
    #         plt.savefig(f'Figs/{sp}.pdf',bbox_inches='tight')

        # plt.show()
```

6 Sensitivity System

[22]:
```
## Create sensitivity system
s = copy.copy(s0)
extra = st.stoichSensitivity(s)
chemostats += extra
sc = st.statify(s,chemostats=chemostats)
```

[23]:
```
## Reactions
disp.Latex(st.sprintrl(s,all=True,chemformula=True))
```

[23]:

$$G_6P + sG_6P + sPGI \xrightleftharpoons{PGI} F_6P + sF_6P + sPGI \tag{16}$$

$$ATP + F_6P + sATP + sF_6P + sPFK \xrightleftharpoons{PFK} ADP + FDP + H + sADP + sFDP + sH + sPFK \tag{17}$$

$$FDP + sFDP + sFBA \xrightleftharpoons{FBA} DHAP + G_3P + sDHAP + sG_3P + sFBA \tag{18}$$

$$DHAP + sDHAP + sTPI \xrightleftharpoons{TPI} G_3P + sG_3P + sTPI \tag{19}$$

$$G_6P + NADP + sG_6P + sNADP + sG_6PDH_2R \xrightleftharpoons{G_6PDH_2R} {}_6PGL + H + NADPH + s_6PGL + sH + sNAI \tag{20}$$

$${}_6PGL + H_2O + s_6PGL + sH_2O + sPGL \xrightleftharpoons{PGL} {}_6PGC + H + s_6PGC + sH + sPGL \tag{21}$$

$${}_6PGC + NADP + s_6PGC + sNADP + sGND \xrightleftharpoons{GND} CO_2 + NADPH + RU_5PD + sCO_2 + sNADPH + \tag{22}$$

$$RU_5PD + sRU_5PD + sRPI \xrightleftharpoons{RPI} R_5P + sR_5P + sRPI \tag{23}$$

$$E_4P + XU_5PD + sE_4P + sXU_5PD + sTKT_2 \xrightleftharpoons{TKT_2} F_6P + G_3P + sF_6P + sG_3P + sTKT_2 \tag{24}$$

$$G_3P + S_7P + sG_3P + sS_7P + sTALA \xrightleftharpoons{TALA} E_4P + F_6P + sE_4P + sF_6P + sTALA \tag{25}$$

$$R_5P + XU_5PD + sR_5P + sXU_5PD + sTKT_1 \xrightleftharpoons{TKT_1} G_3P + S_7P + sG_3P + sS_7P + sTKT_1 \tag{26}$$

$$RU_5PD + sRU_5PD + sRPE \xrightleftharpoons{RPE} XU_5PD + sXU_5PD + sRPE \tag{27}$$

[24]:
```
## Initial conditions
X0 = np.hstack((conc,np.ones(s0['n_X']),np.ones(s0['n_V'])))
# print(X0)
print(X0.shape)
```

```
(50,)
```

[25]:
```
## Linearise
# Sys,X_ss,V_ss,dX_ss = Linear(s,sc,X0=X0)
# Sys,X_ss,V_ss,dX_ss = Linear(s,sc,parameter=parameter,X0=X0, invar='X',␣
 ↪outvar = 'V', quiet=False)
```

[26]:
```
# species = s['species']
# reaction = s['reaction']
```

[27]:
```
# print(chemostats)
# print(reaction)
# print(species)
# # print(s0['Z'])
```

[28]:
```
# sys = extractSubsystem(Sys,sc,sf,inp=['sG6P'],outp=['RPI'],order=3)
```

6.1 Linearise sensitivity system

[29]:
```
## Linearise dX as outvar
Sys,X_ss,V_ss,dX_ss = Linear(s,sc,parameter=parameter,X0=X0, invar='X',␣
 ↪outvar = 'dX', quiet=False)
```

```
Steady-state finder error: 9.65e-13
V_ss = [59.52330023 63.11688312 63.11688312 63.11688312 11.57830405 11.57830405
 11.57830405  7.98472116  1.79679144  1.79679144  1.79679144  3.59358289]
Setting K_6PGC to 6.233520699319548
Setting K_6PGL to 1.0397283862123778
Setting K_ADP to 5.154605959675528
Setting K_ATP to 2.2539233202427162
Setting K_CO2 to 33.794182951886775
Setting K_DHAP to 1.778985981513151
Setting K_E4P to 57.93534155565906
Setting K_F6P to 1.41401983409913
Setting K_FDP to 0.38795235349448437
Setting K_G3P to 14.902044199279699
Setting K_G6P to 0.8377378837717493
Setting K_H to 0.3491739027654742
Setting K_H2O to 1.0397283862123778
Setting K_NADP to 11747.063287701598
Setting K_NADPH to 21.002665768445333
Setting K_R5P to 12.241945837528652
Setting K_RU5PD to 86.15506601742248
Setting K_S7P to 21.751324335743256
Setting K_XU5PD to 51.68293204643235
Setting kappa_PGI to 154.38997002578546
```

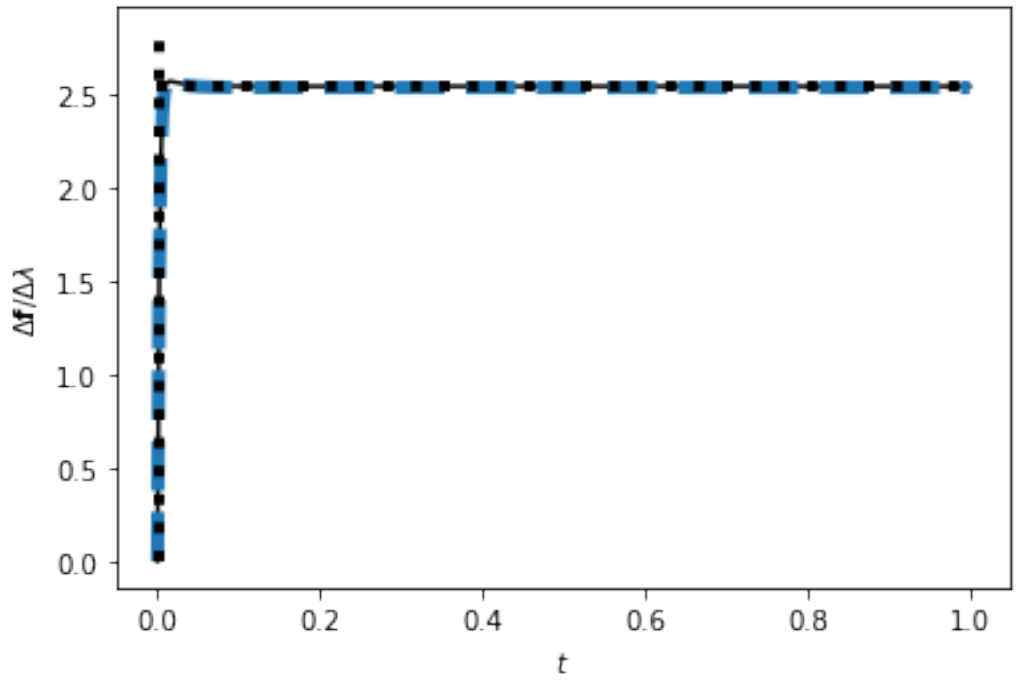
```
Setting kappa_PFK to 54.85043355391926
Setting kappa_FBA to 160.07691981318658
Setting kappa_TPI to 353.9323765134668
Setting kappa_G6PDH2R to 4.667672687418436
Setting kappa_PGL to 291.6414393949941
Setting kappa_GND to 1.2695594575280742
Setting kappa_RPI to 4206.975085828208
Setting kappa_TKT2 to 9.170830408091838
Setting kappa_TALA to 1.6563170906188465
Setting kappa_TKT1 to 8.819978413932159
Setting kappa_RPE to 96.07053469867635
```

6.2 Compute chemostat sensitivities

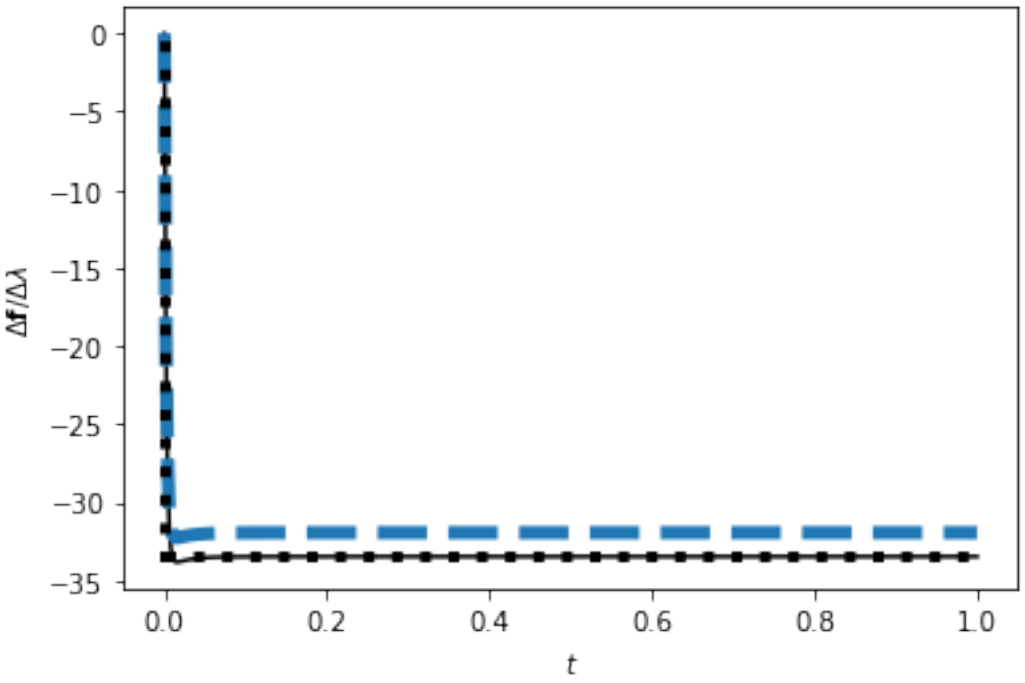
[30]:
```
## Chemostat sensitivities
# Inp = ['sG6P','sR5P','sNADPH','sG3P']
Outp = ['G6P','R5P','NADPH','G3P']
Inp = []
for chemo in sc['chemostats']:
    if not chemo[0] in ['s','H']:
        Inp.append('s'+chemo)
print(Inp)
Inp_chemo = Inp
# Inp = []
# for reac in s['reaction']:
#     Inp.append('s'+reac)
# Inp = ['sG6PDH2R']

t_last=1
lam = 1.1 # Perturbation parameter

dcgain = {}
Tau = {}
for outp in Outp:
    for inp in Inp:
        print('Doing:',inp)
        sf = None
        order = None
        dat,y_step,t,sys =␣
↪simSensitivity(s,sc,sf,Sys,X_ss,V_ss,dX_ss,chemostats=Chemostats,parameter=parameter,
                                                    ␣
↪inp=[inp],outp=[outp],lam=lam,t_last=t_last,order=order)


        g = con.dcgain(sys)
        g,tau = tfProps(sys)
        print('g =',g)
        name = inp[1:]

        if outp=='G6P':
            dcgain[name] = -g
```

```
            ylabel = '$-g_{\infty}$'
        else:
            dcgain[name] = g
            ylabel = '$g_{\infty}$'

        Tau[name] = tau

        dlam = int((lam-1)*100)
        print('dlam =',dlam)
        if Titles:
            plt.title(f'{name}'+r' ($\tilde\lambda$ = ' + f'{dlam}%; g={g:.
↪2f})')
        plotSensitivitydX(dat,species=[outp],labeling=False)
        plotLines()
        Savefig('PPPdX_'+name+'_'+outp)
        plt.show()

    ## Barcharts
    ## DC gain
    if Titles:
        plt.title(outp)
    plt.tick_params(axis='x', rotation=90)
#     plt.grid(axis='y')
    plt.bar(range(len(dcgain)), dcgain.values(), align='center',color='black')
    plt.xticks(range(len(dcgain)), list(dcgain.keys()))
    plt.ylabel(ylabel)
    plt.axhline(0,color='black',ls='dashed')
    Savefig(f'PPPdX_{outp}_chem_bar')
    plt.show()

    ## Time constant
    if Titles:
        plt.title(outp)
    plt.tick_params(axis='x', rotation=90)
#     plt.grid(axis='y')
    plt.bar(range(len(Tau)), Tau.values(), align='center',color='black')
    plt.xticks(range(len(Tau)), list(Tau.keys()))
    plt.ylabel(r'$\tau$')
#     plt.axhline(0,color='black',ls='dashed')
    Savefig(f'PPPdX_{outp}_chem_tau_bar')
    plt.show()
```

```
['sADP', 'sATP', 'sCO2', 'sG6P', 'sNADP', 'sNADPH', 'sR5P', 'sG3P']
Doing: sADP
g = 2.5484963469688195
dlam = 10
G6P
```

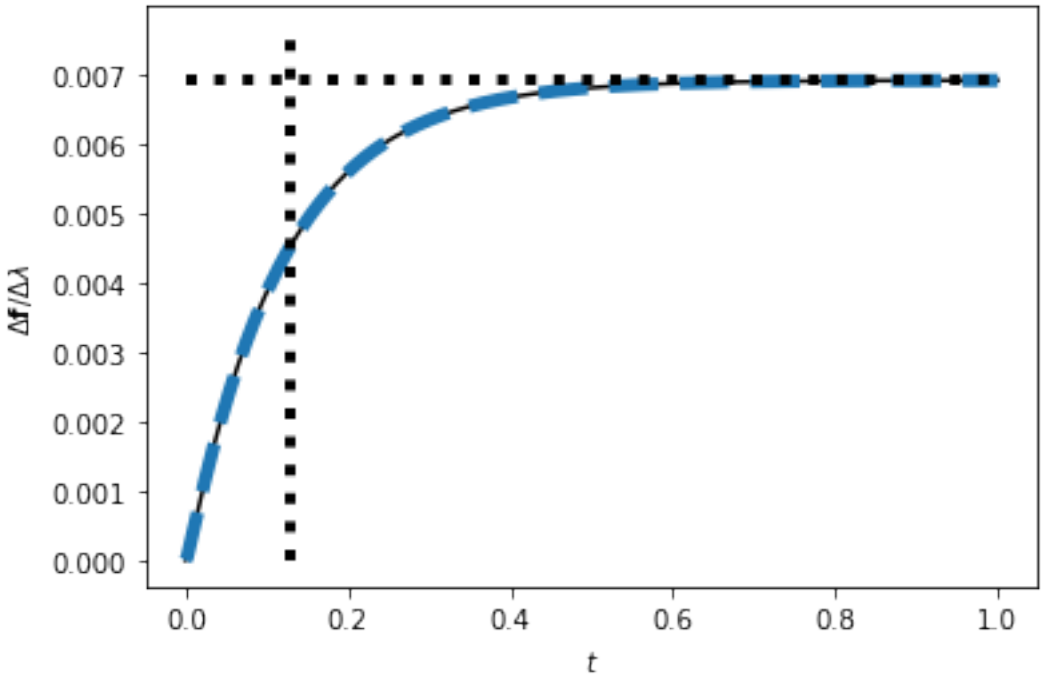



```
Doing: sATP
g = -33.46210603034231
dlam = 10
G6P
```

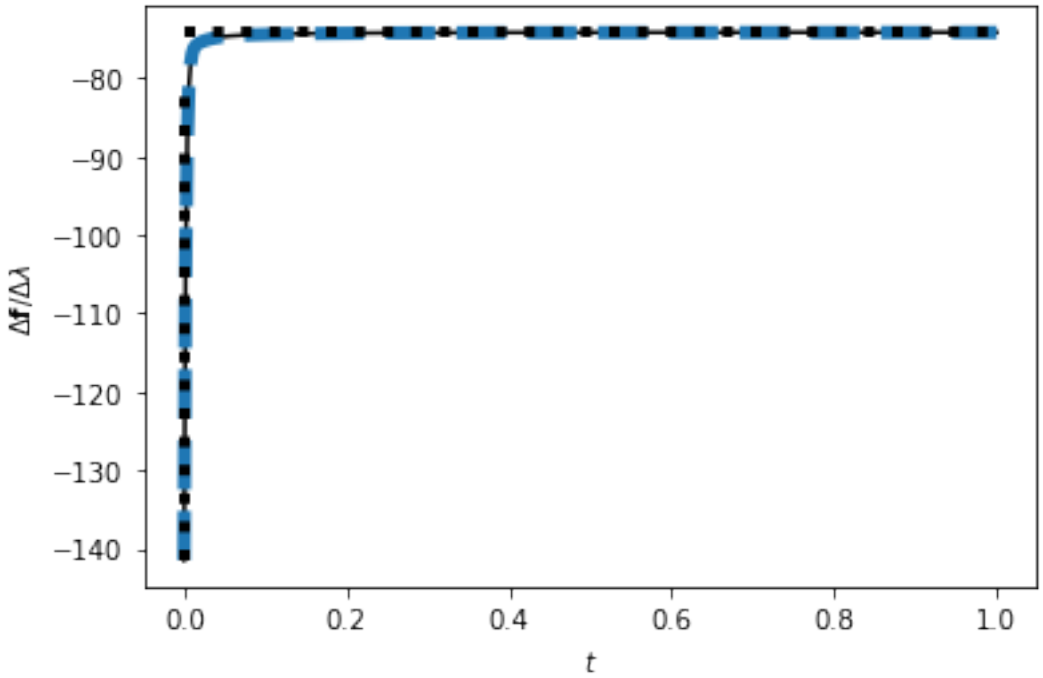



```
Doing: sCO2
```

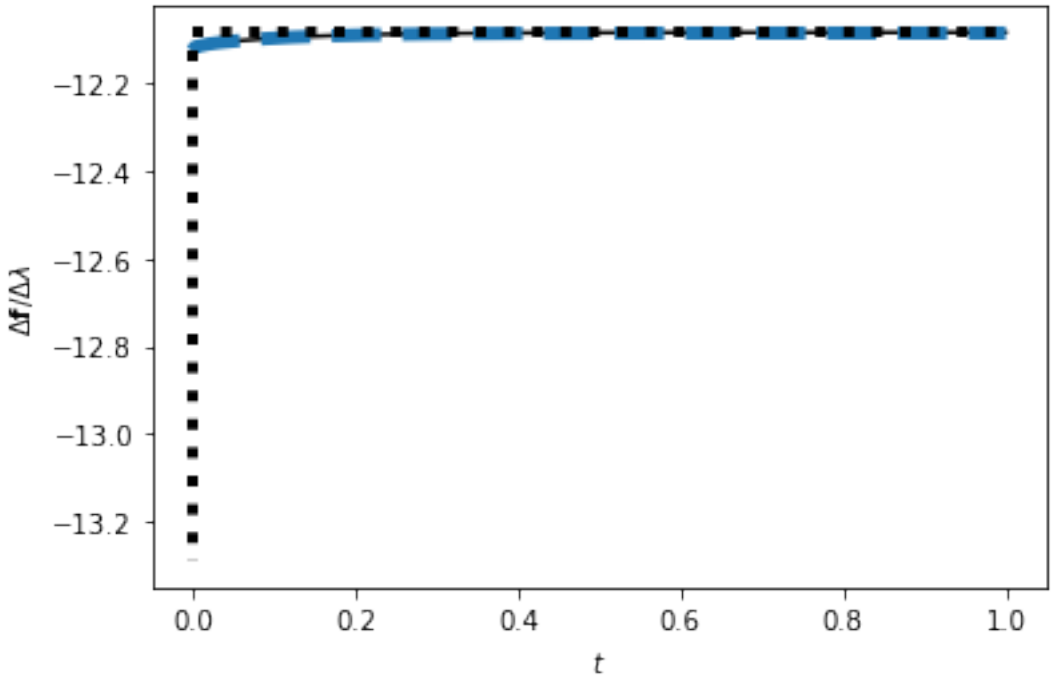
```
g = 0.00692494242981384
dlam = 10
G6P
```



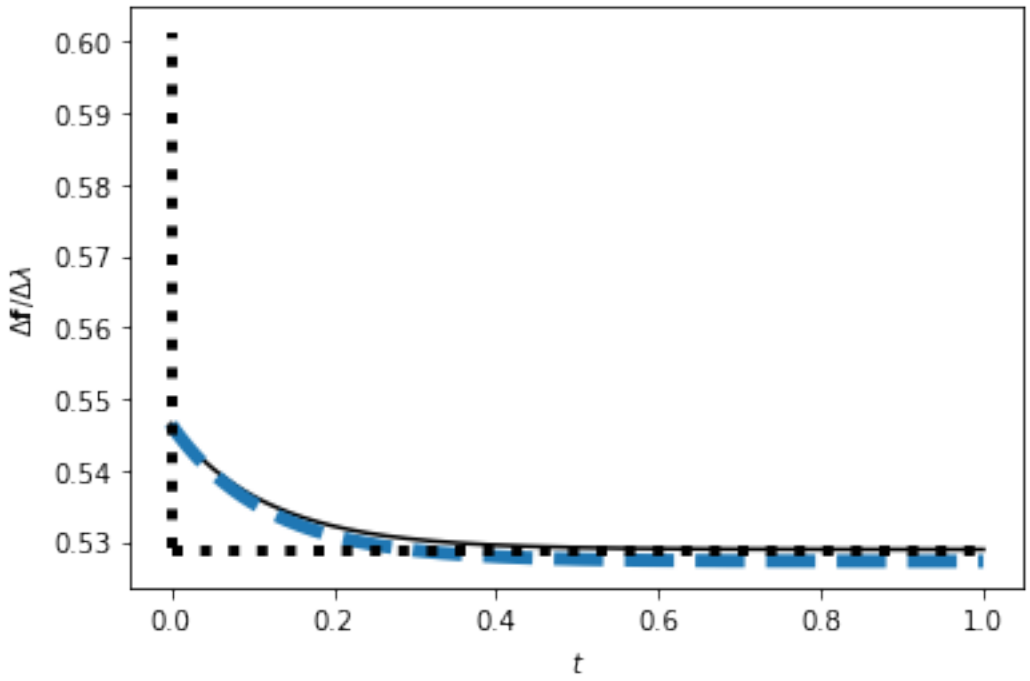

```
Doing: sG6P
g = -74.18144575346916
dlam = 10
G6P
```



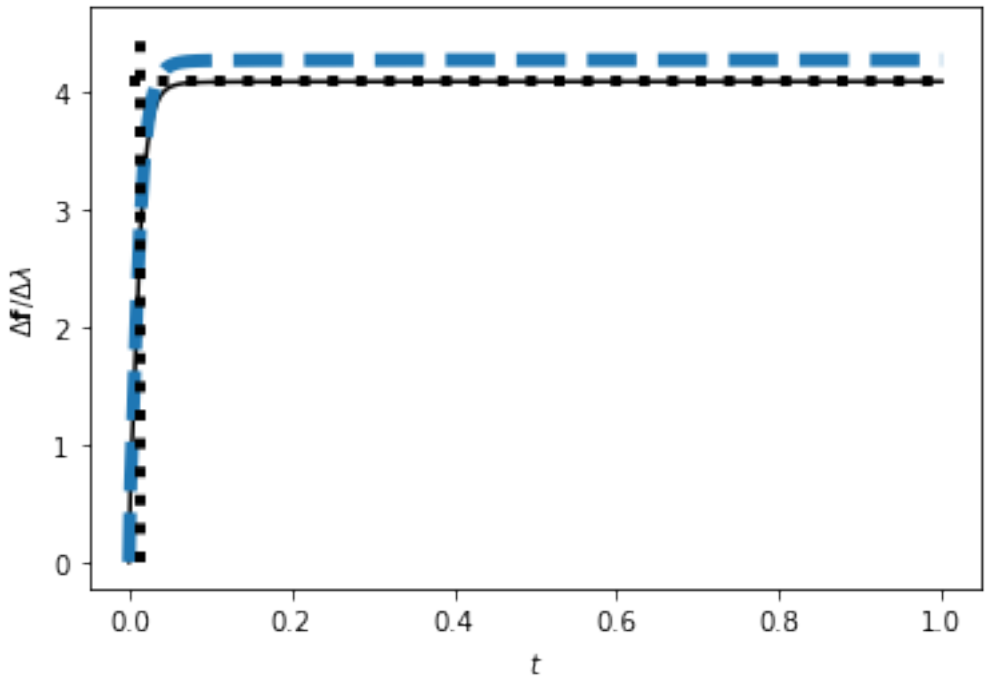
```
Doing: sNADP
g = -12.083105188911862
dlam = 10
G6P
```




```
Doing: sNADPH
g = 0.5288892968979388
dlam = 10
G6P
```

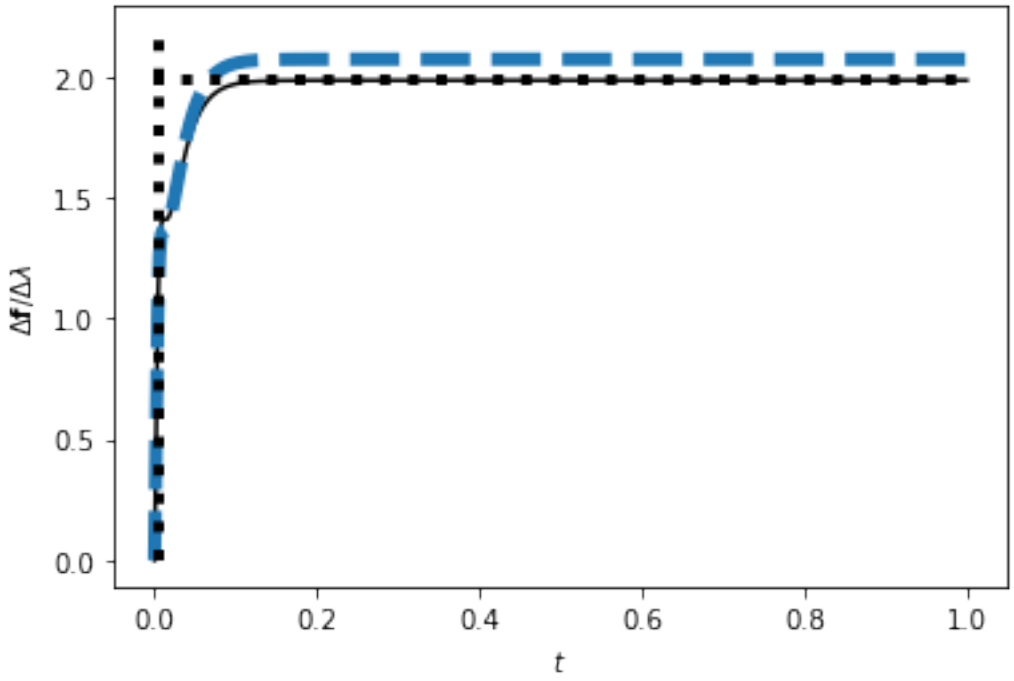


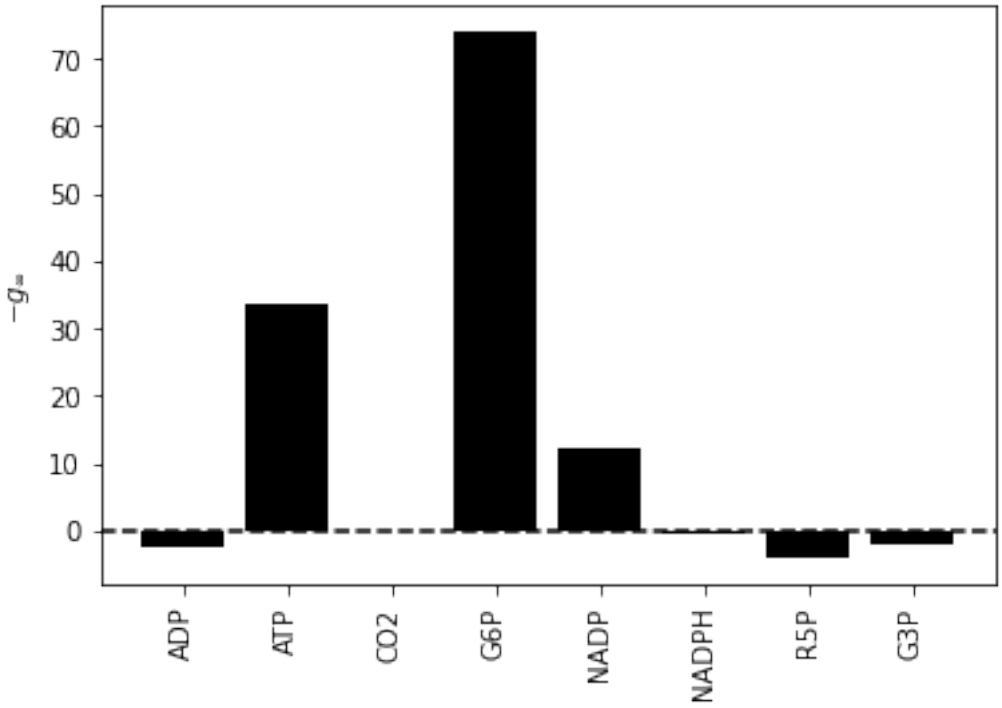

```
Doing: sR5P
g = 4.087111202682657
dlam = 10
G6P
```

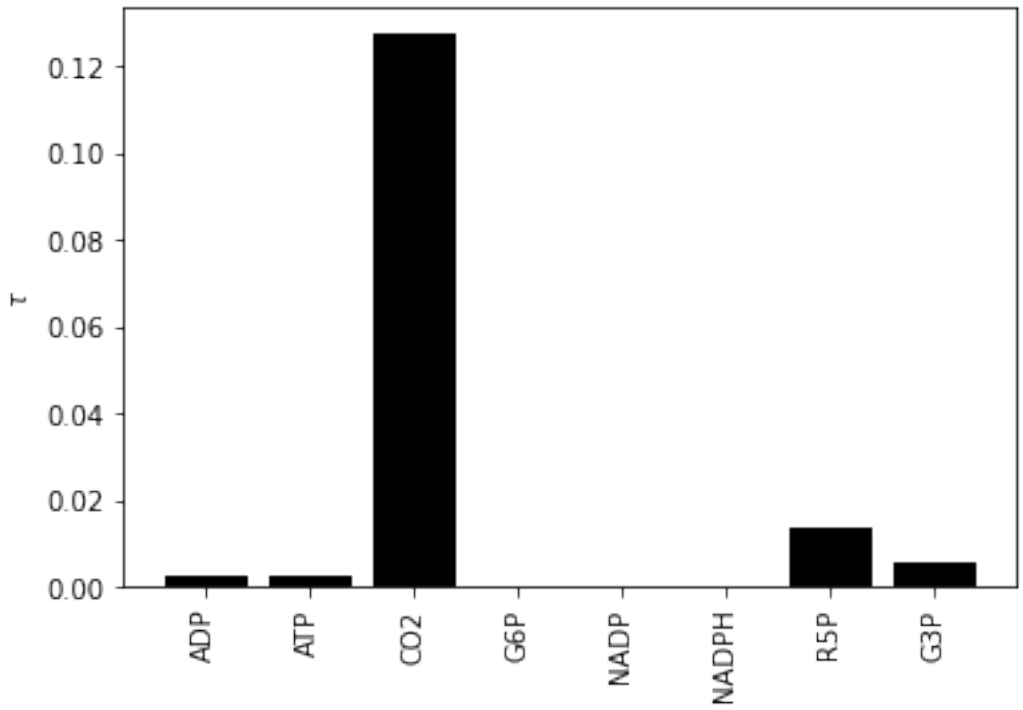



```
Doing: sG3P
```

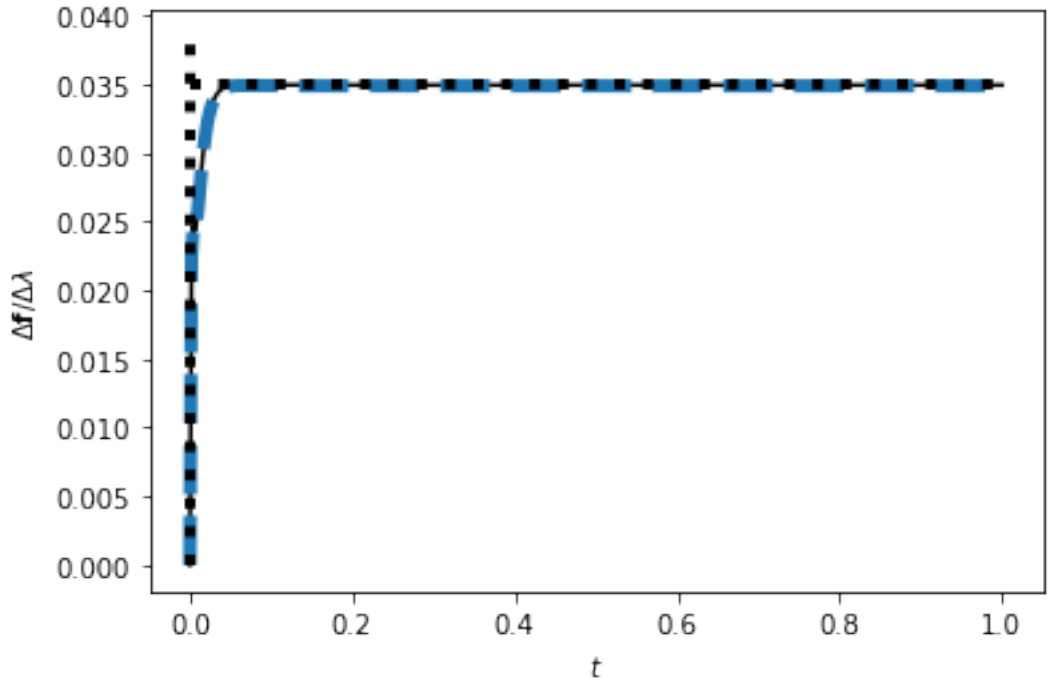
```
g = 1.9845213690294017
dlam = 10
G6P
```



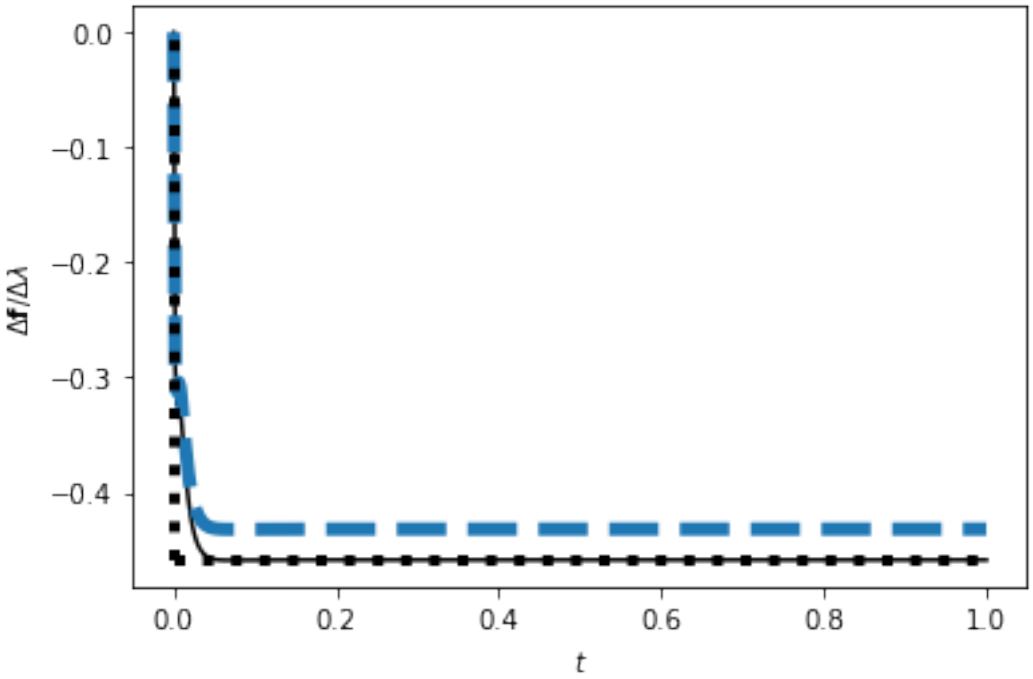




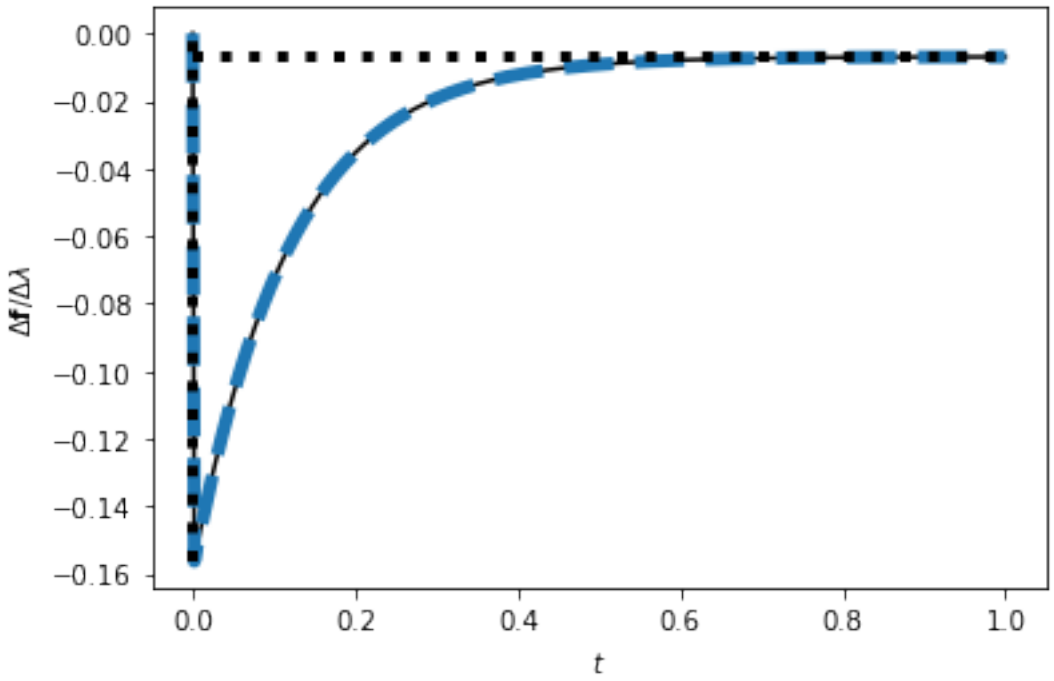

```
Doing: sADP
g = 0.034924802389062704
dlam = 10
R5P
```



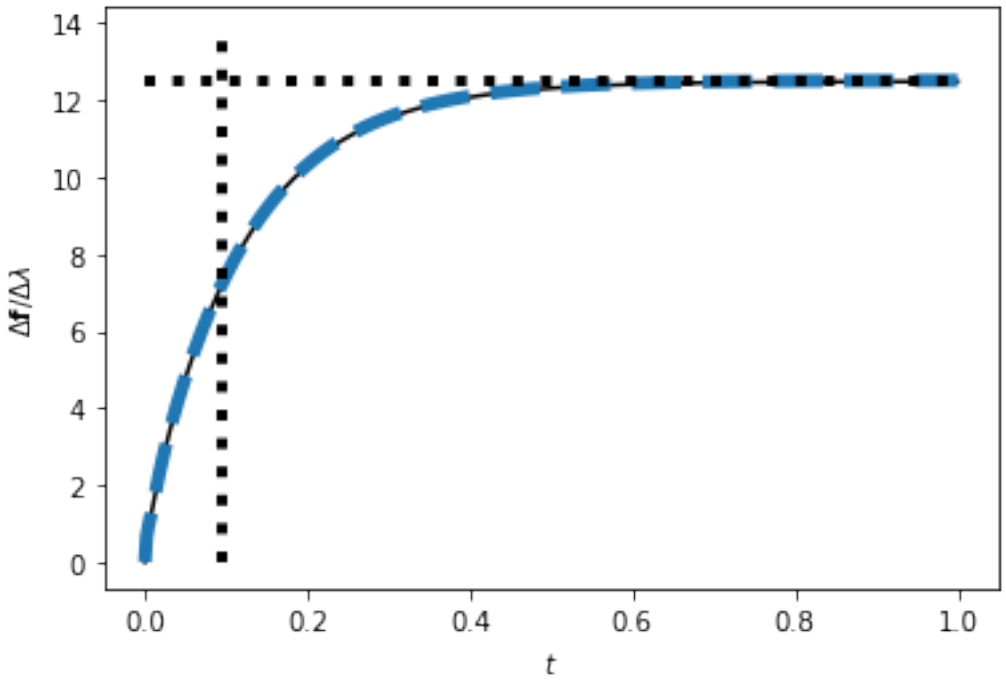
```
Doing: sATP
g = -0.4585674380194895
dlam = 10
R5P
```




```
Doing: sCO2
g = -0.006919387119862225
dlam = 10
R5P
```

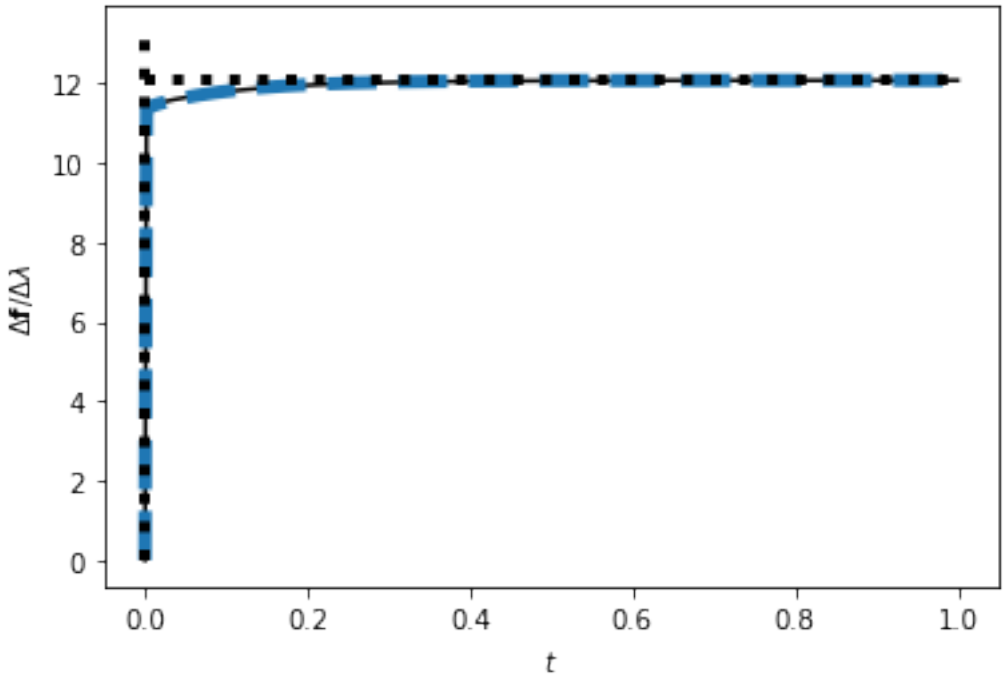



```
Doing: sG6P
g = 12.485591708852539
dlam = 10
R5P
```

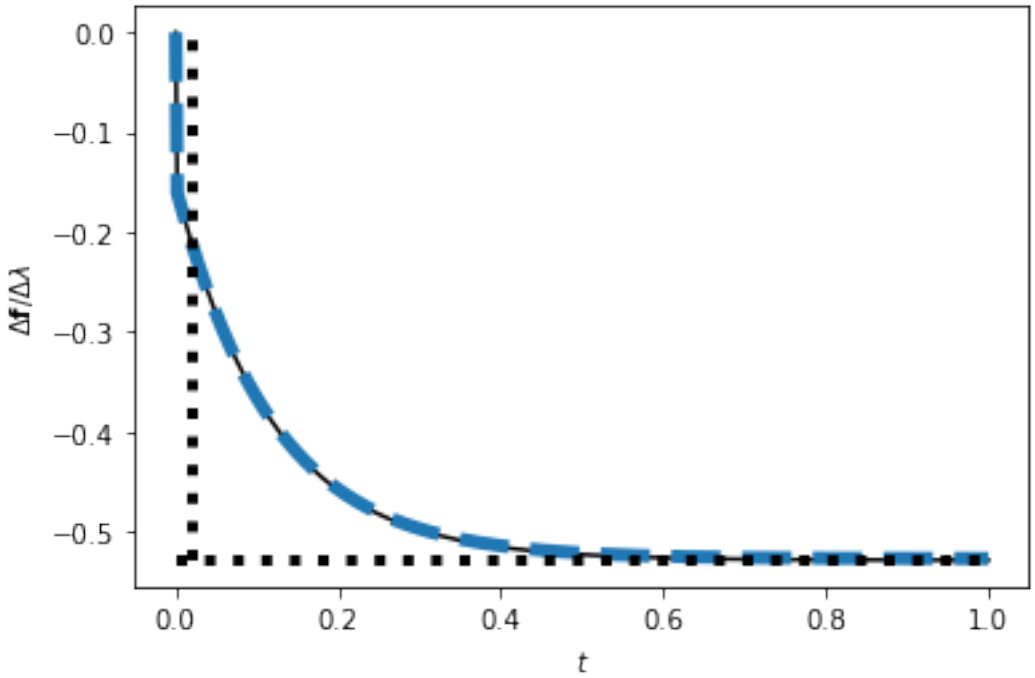



```
Doing: sNADP
```

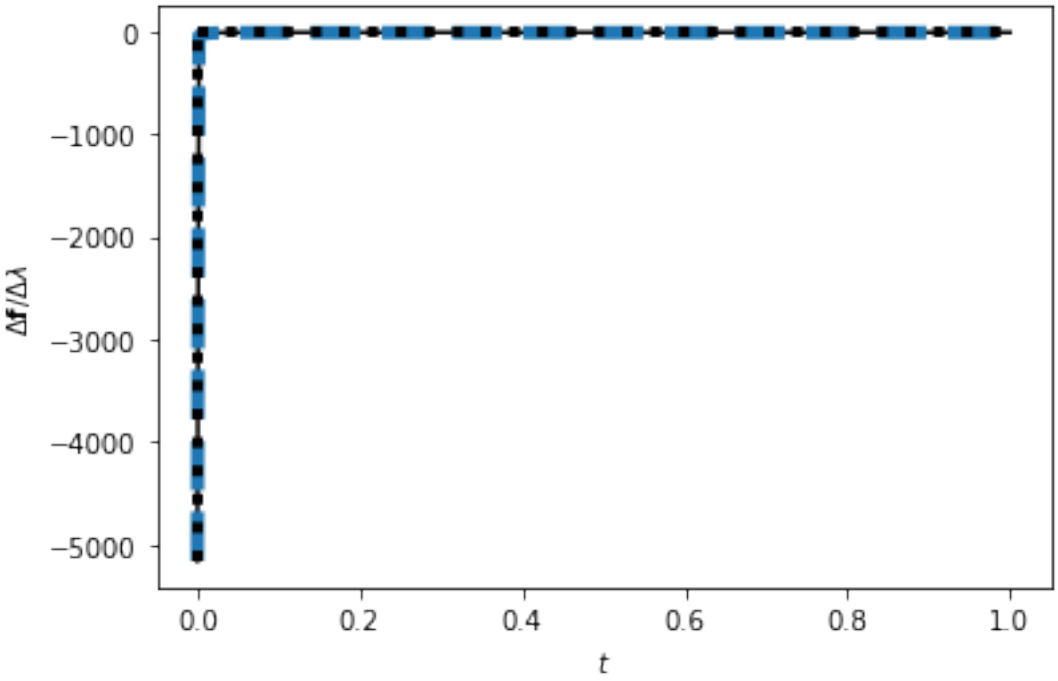
```
g = 12.073411910653313
dlam = 10
R5P
```



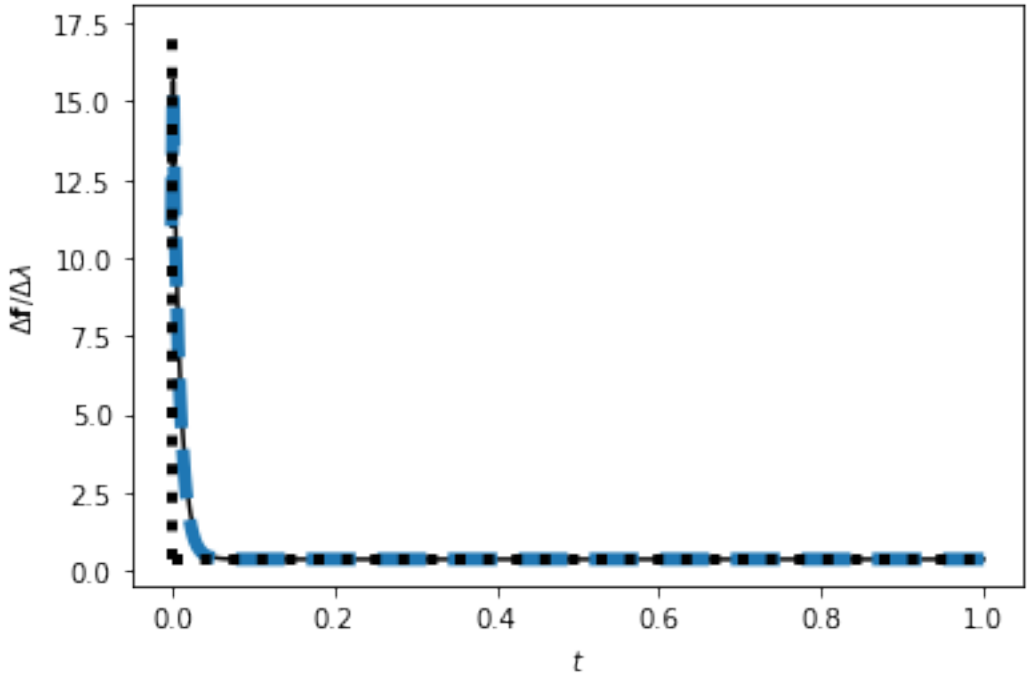

```
Doing: sNADPH
g = -0.5284650126562568
dlam = 10
R5P
```

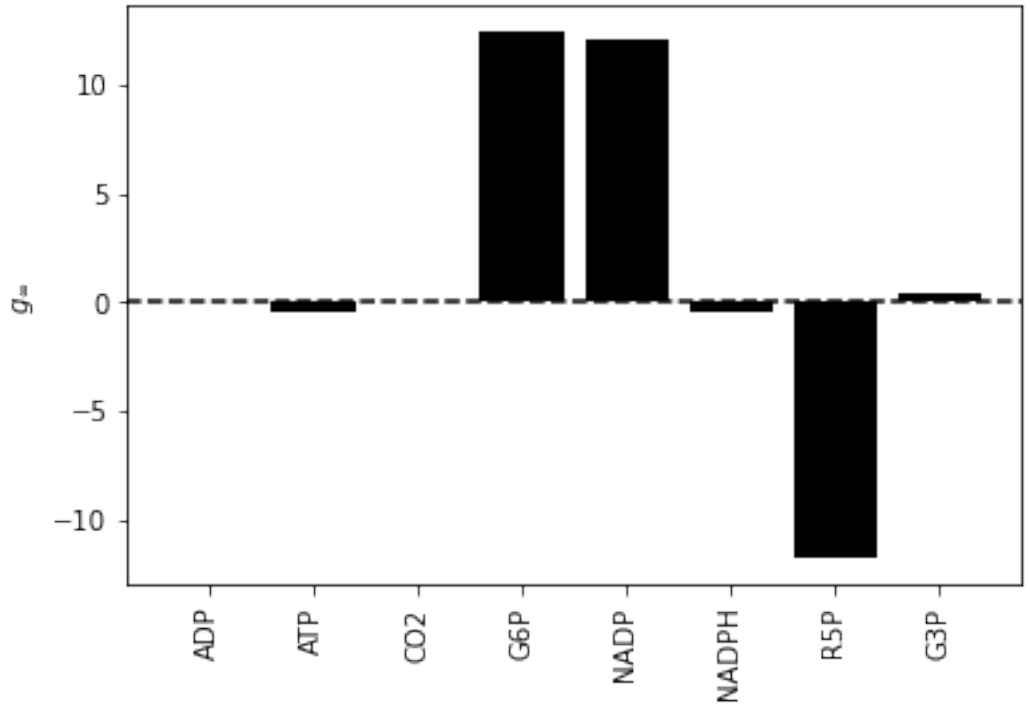


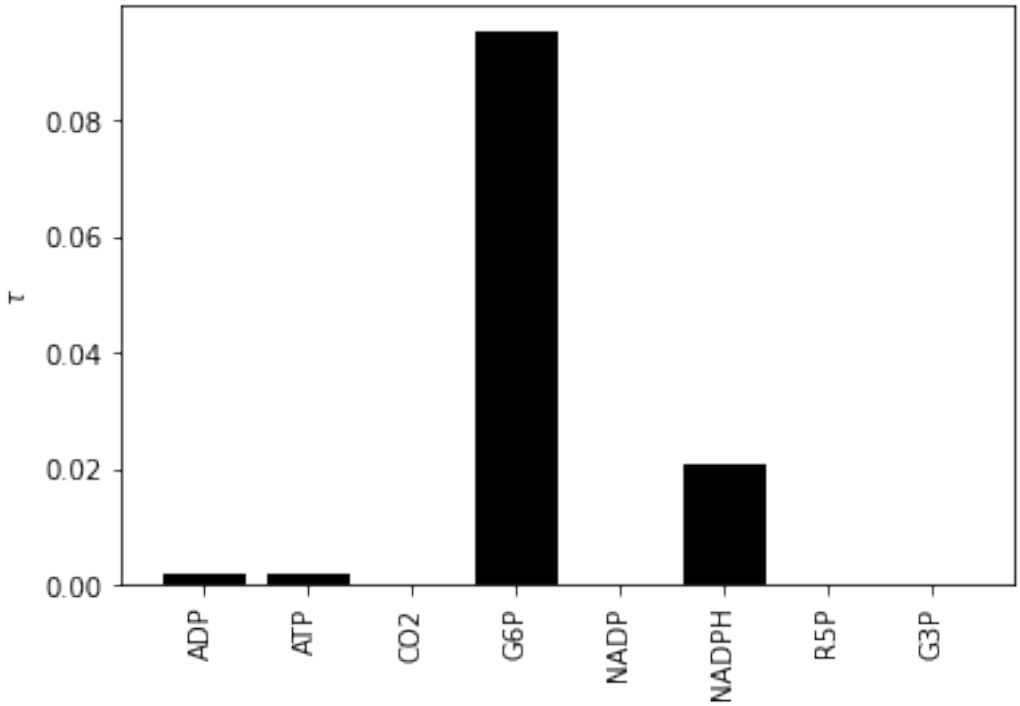
```
Doing: sR5P
g = -11.81229752461877
dlam = 10
R5P
```



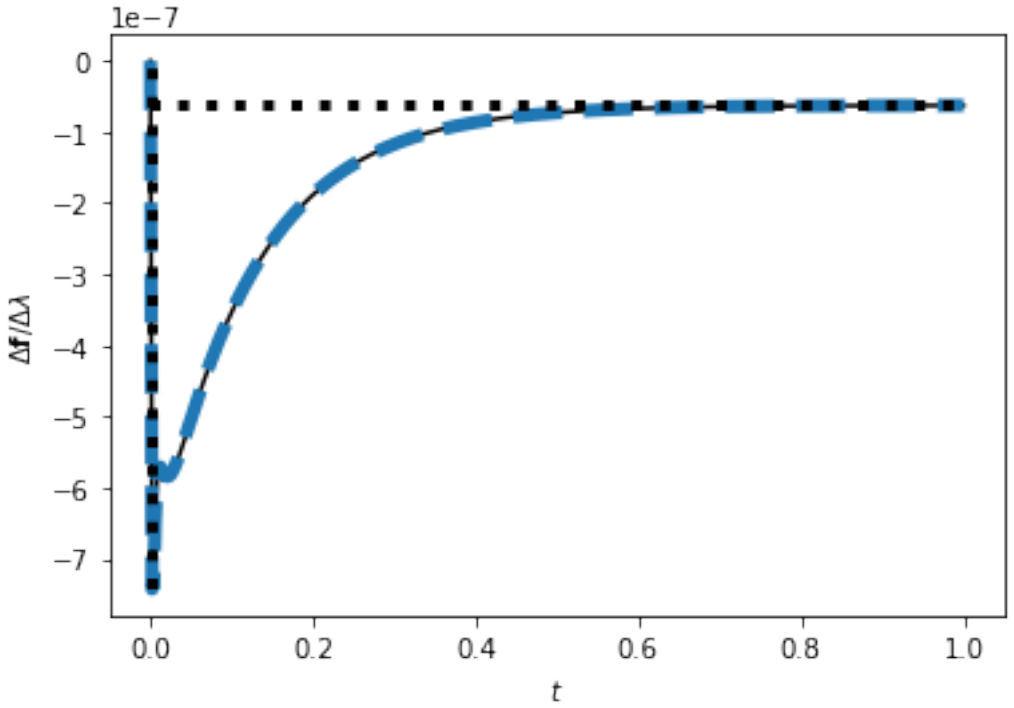

```
Doing: sG3P
g = 0.3634389561698672
dlam = 10
R5P
```


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
$\Delta f/\Delta\lambda$
0.0
0.2
0.4
0.6
0.8
1.0
t

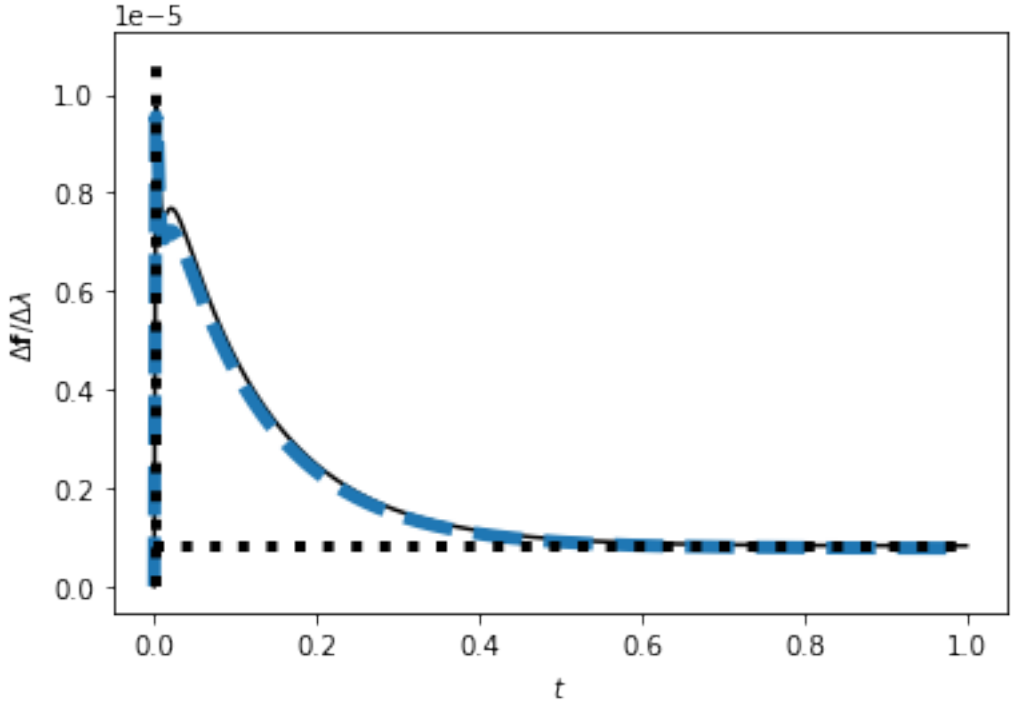

10
5
0
−5
−10
g_∞
ADP
ATP
CO2
G6P
NADP
NADPH
R5P
G3P



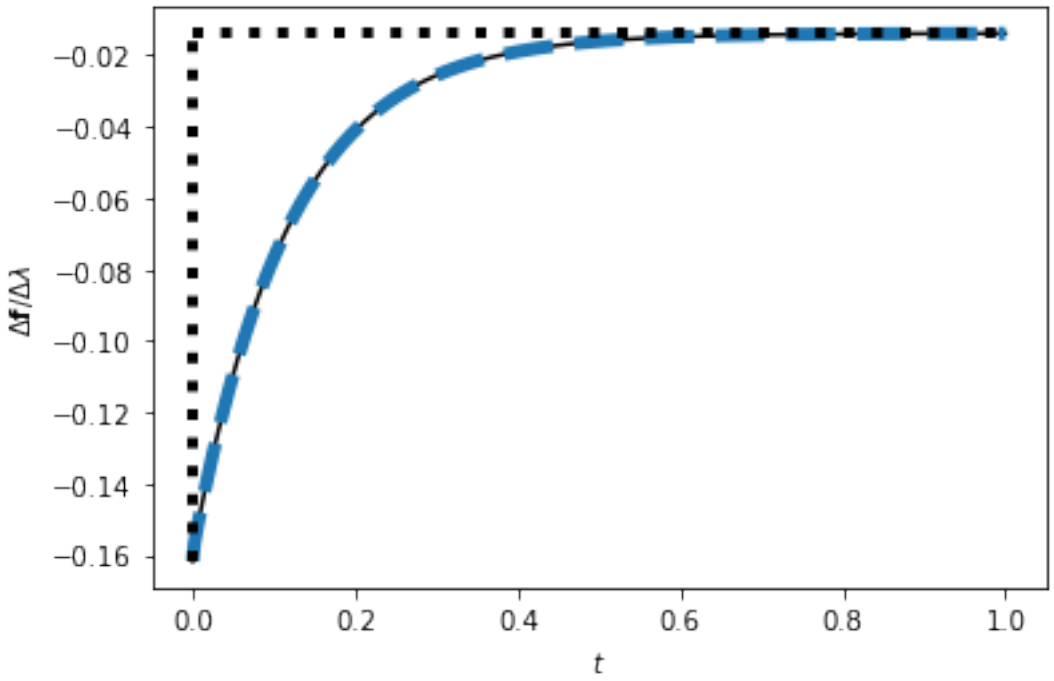

```
Doing: sADP
g = -6.262254210625168e-08
dlam = 10
NADPH
```



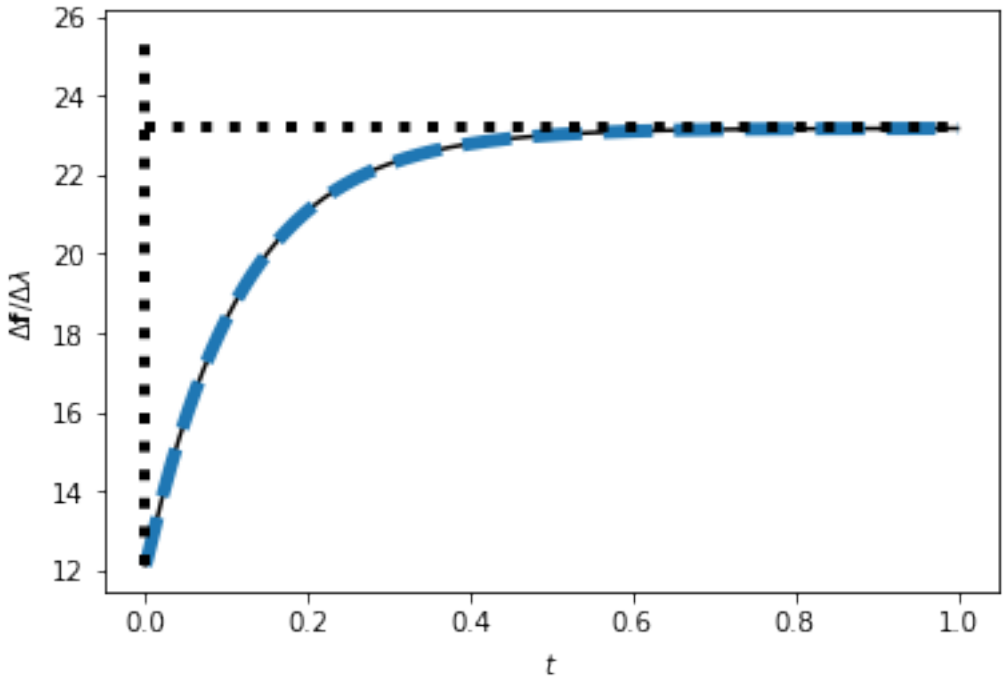
```
Doing: sATP
g = 8.222425534867763e-07
dlam = 10
NADPH
```




```
Doing: sCO2
g = -0.013855753145604471
dlam = 10
NADPH
```

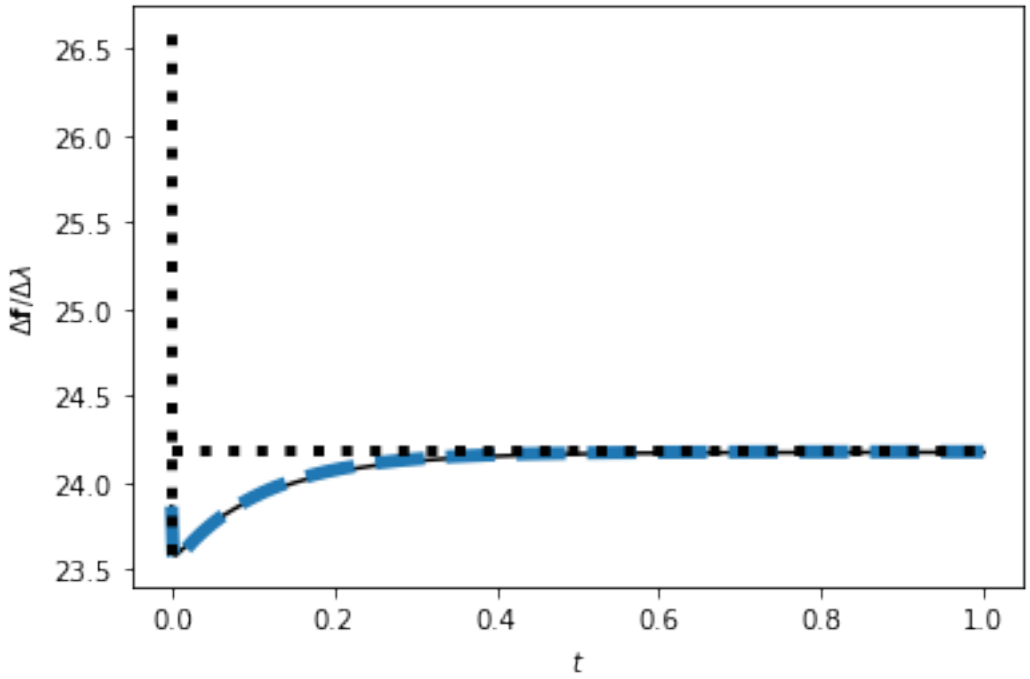



```
Doing: sG6P
g = 23.17043109548335
dlam = 10
NADPH
```

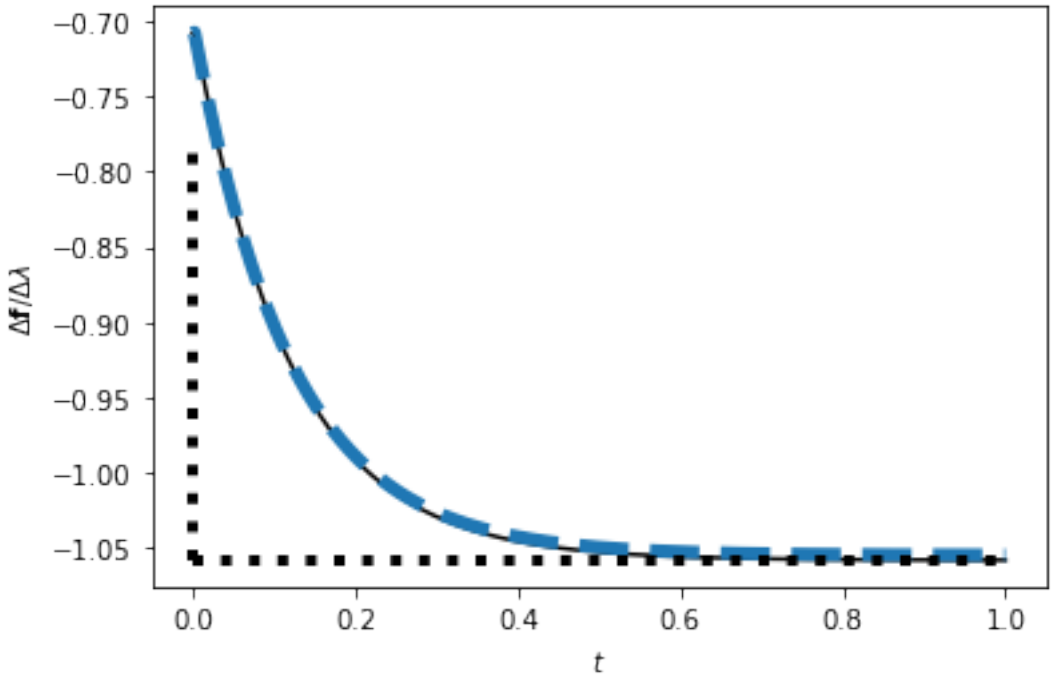



```
Doing: sNADP
```

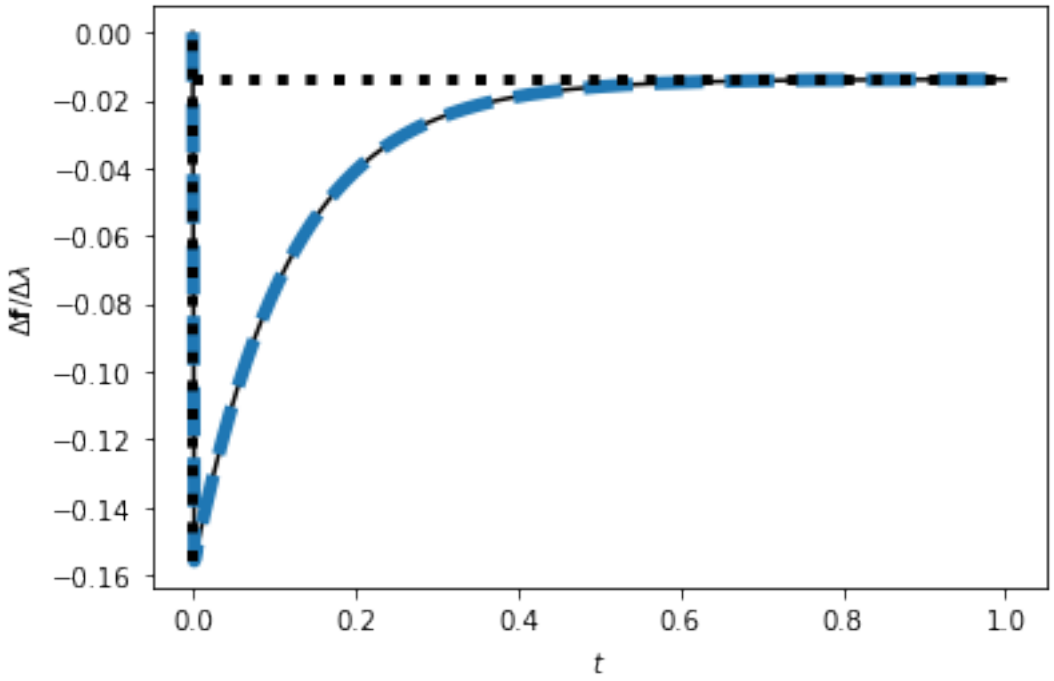
```
g = 24.176449757783384
dlam = 10
NADPH
```



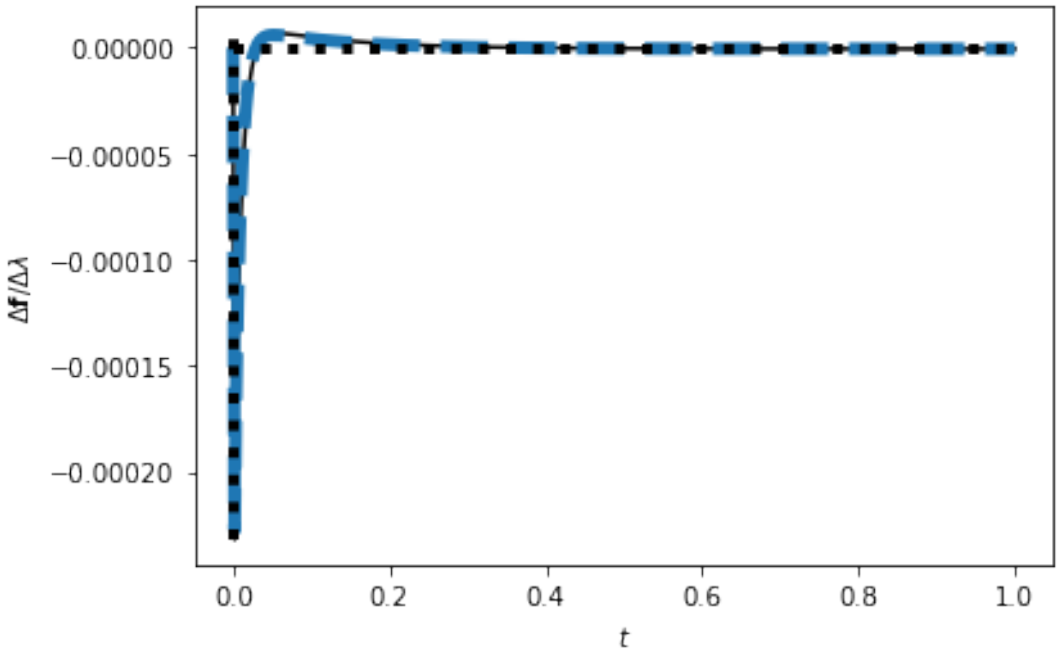

```
Doing: sNADPH
g = -1.058226781441592
dlam = 10
NADPH
```

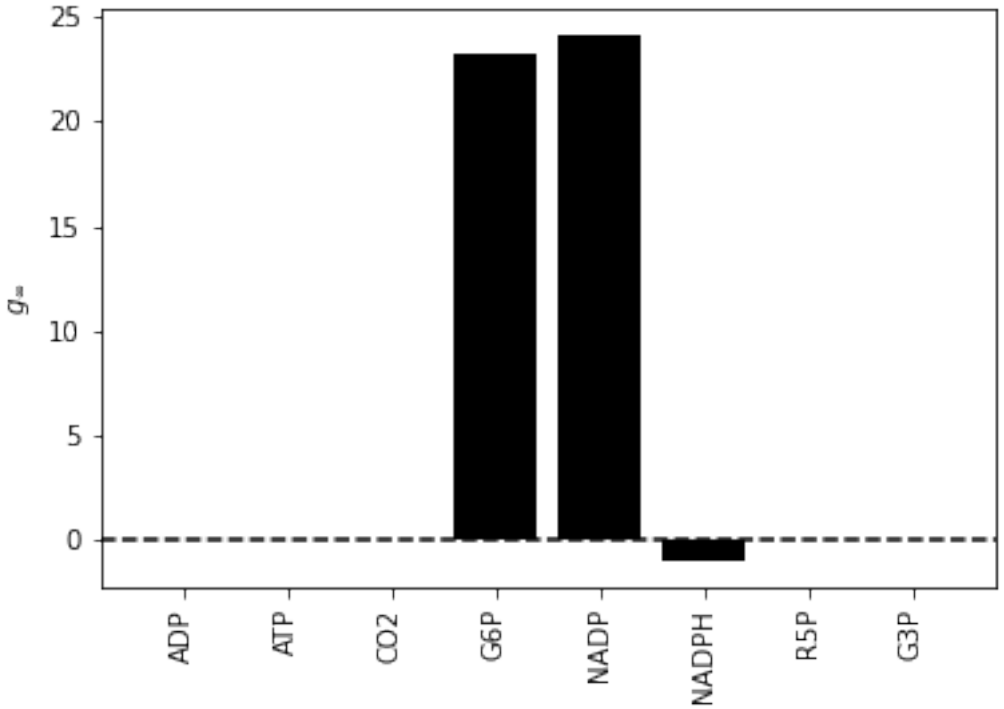


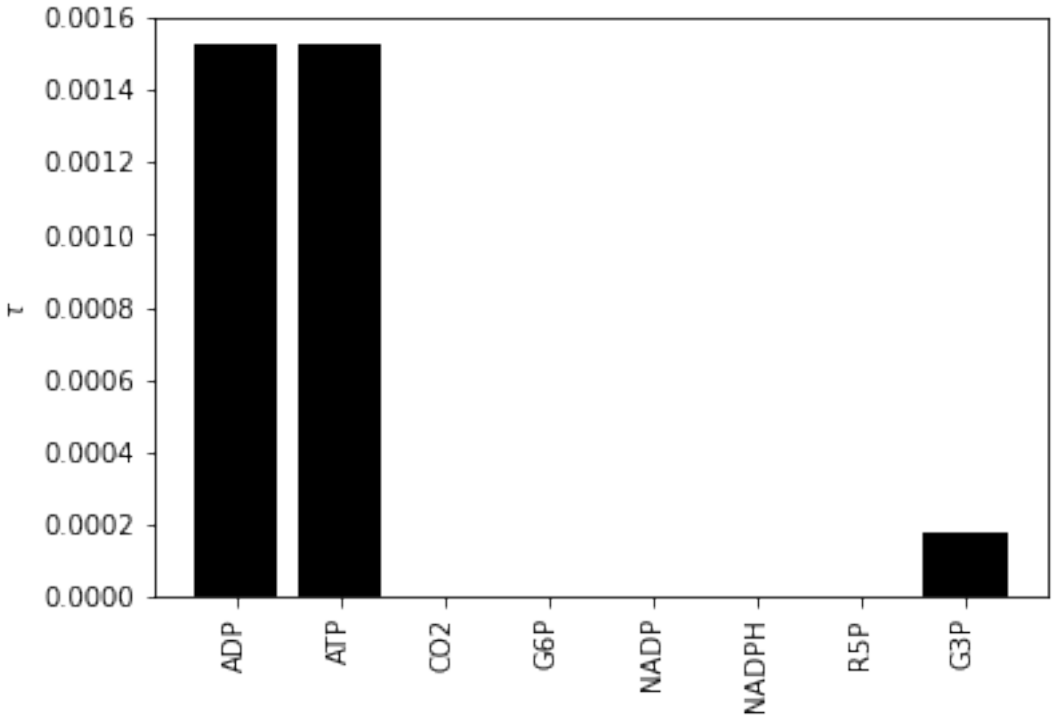
```
Doing: sR5P
g = -0.013813109583457531
dlam = 10
NADPH
```



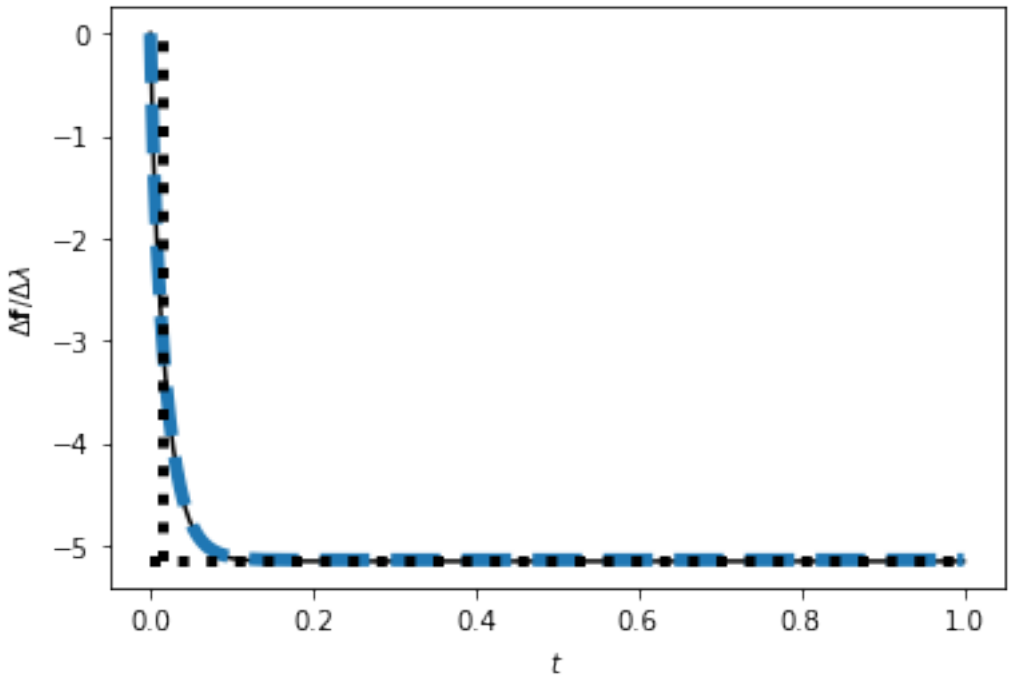

```
Doing: sG3P
g = -6.516703408418131e-07
dlam = 10
NADPH
```


0.00000
−0.00005
−0.00010
−0.00015
−0.00020
Δf/Δλ
0.0
0.2
0.4
0.6
0.8
1.0
t

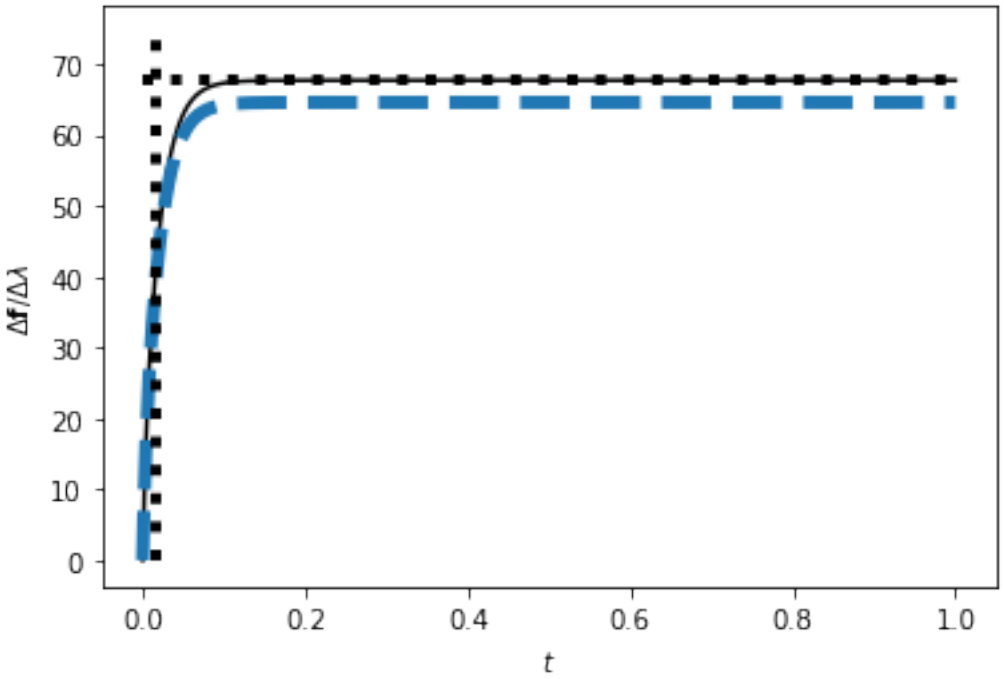

25
20
15
10
5
0
g_∞
ADP
ATP
CO2
G6P
NADP
NADPH
R5P
G3P



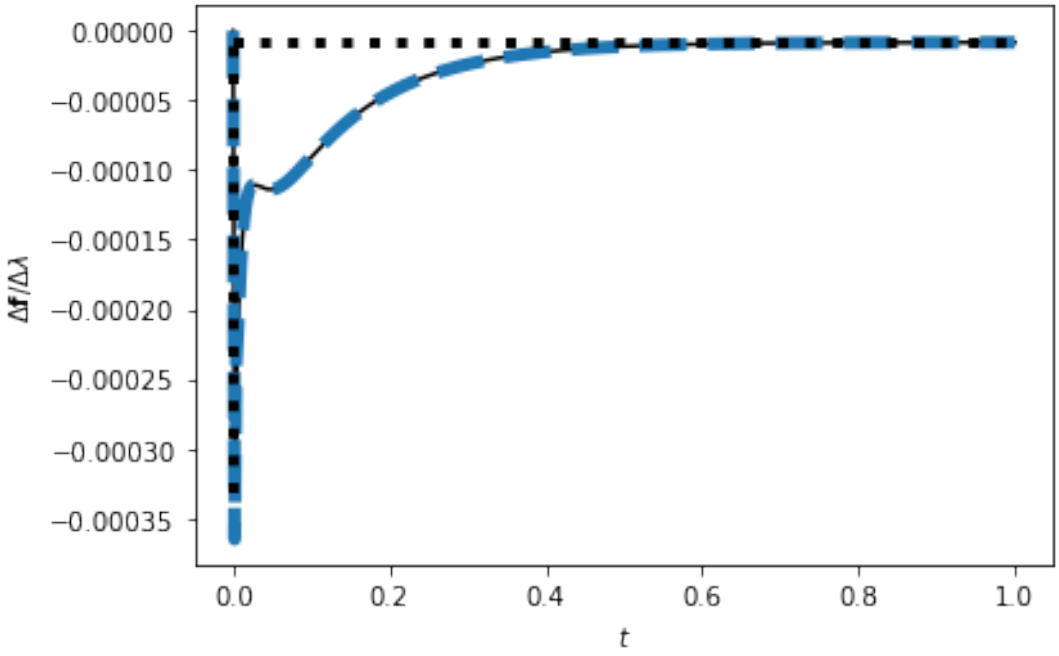

```
Doing: sADP
g = -5.155200687482166
dlam = 10
G3P
```



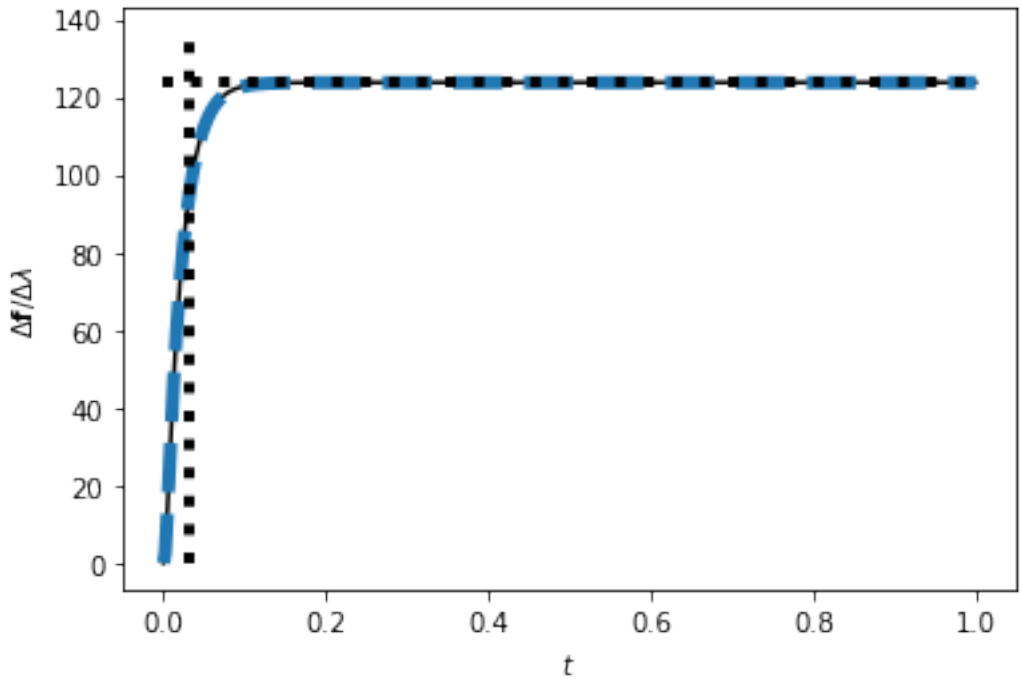
```
Doing: sATP
g = 67.68849098700797
dlam = 10
G3P
```



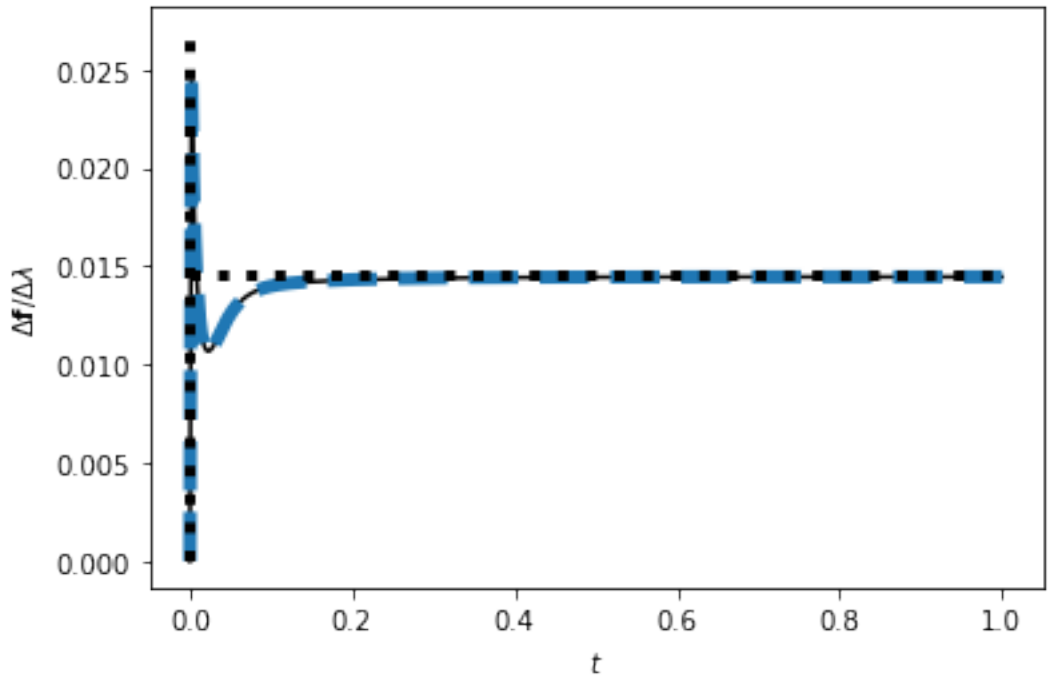

```
Doing: sCO2
g = -8.280802150611012e-06
dlam = 10
G3P
```



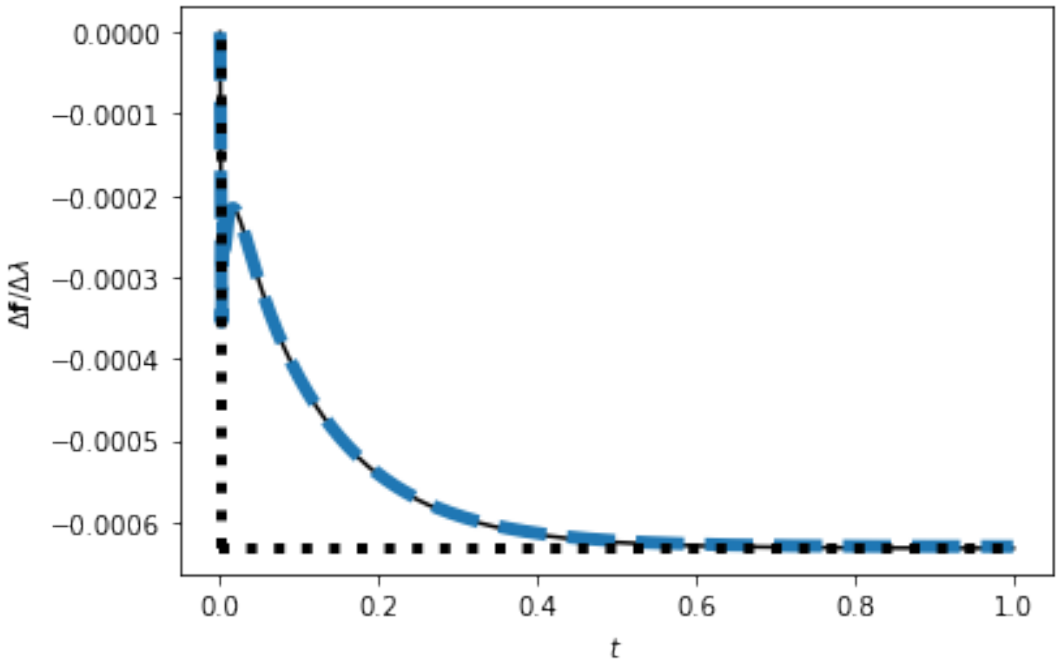
```
Doing: sG6P
g = 123.69183347627416
dlam = 10
G3P
```




```
Doing: sNADP
g = 0.014448900438123411
dlam = 10
G3P
```

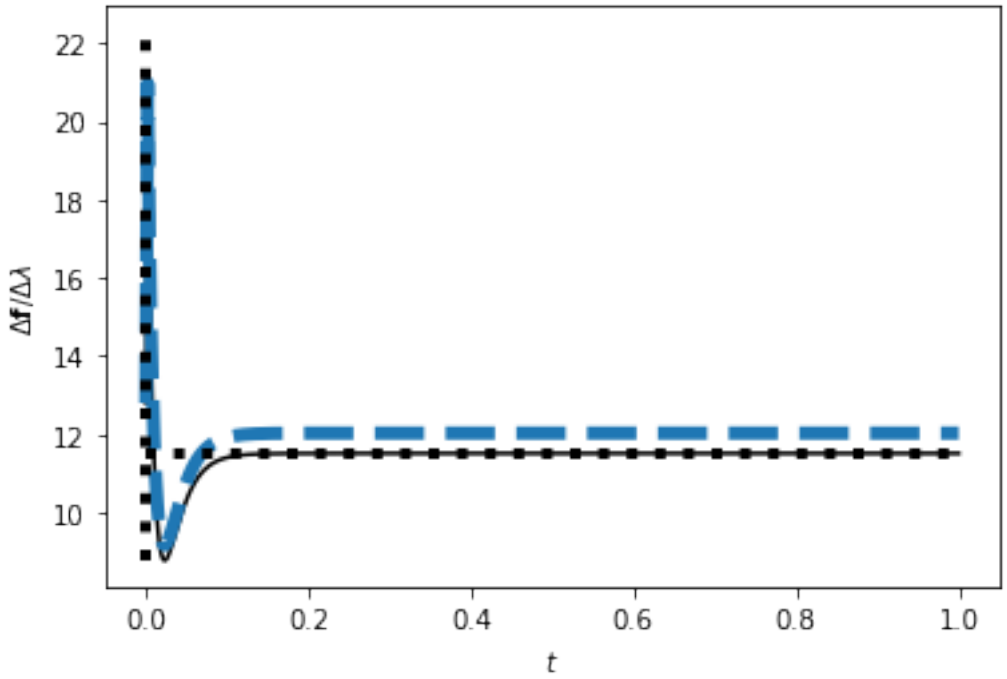



```
Doing: sNADPH
g = -0.0006324424620007293
dlam = 10
G3P
```

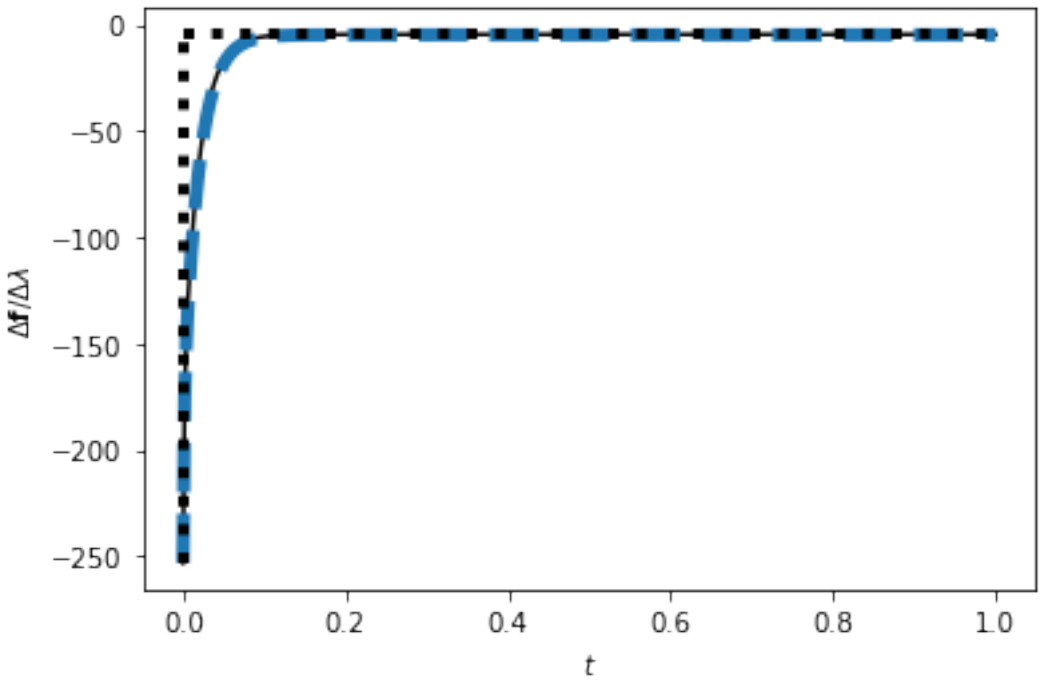


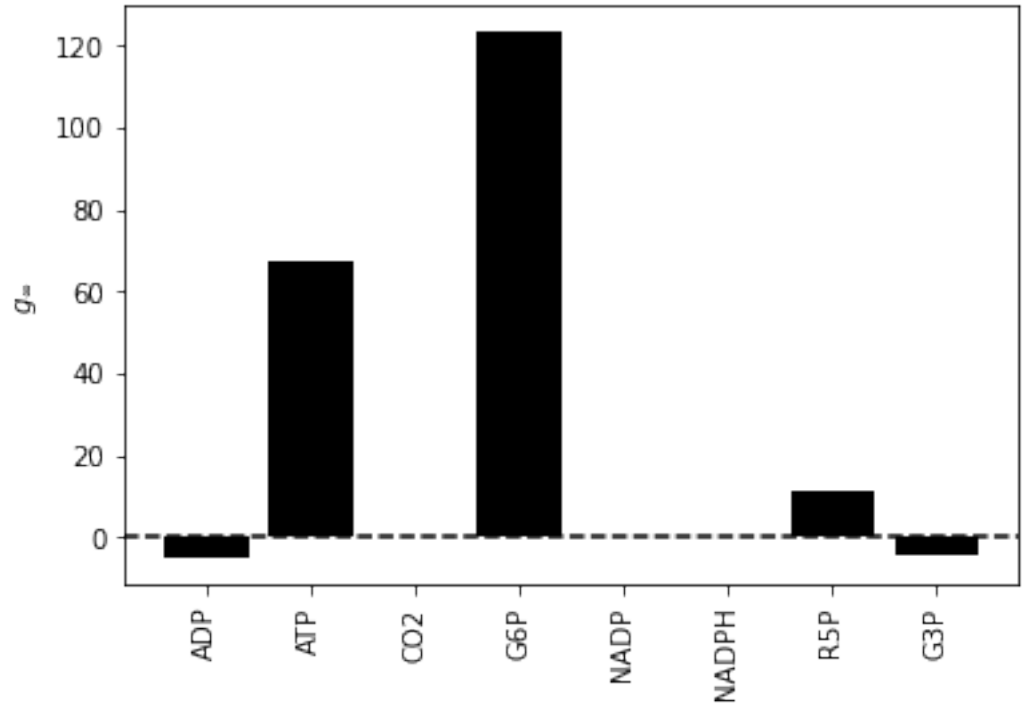

```
Doing: sR5P
g = 11.51524232059544
```

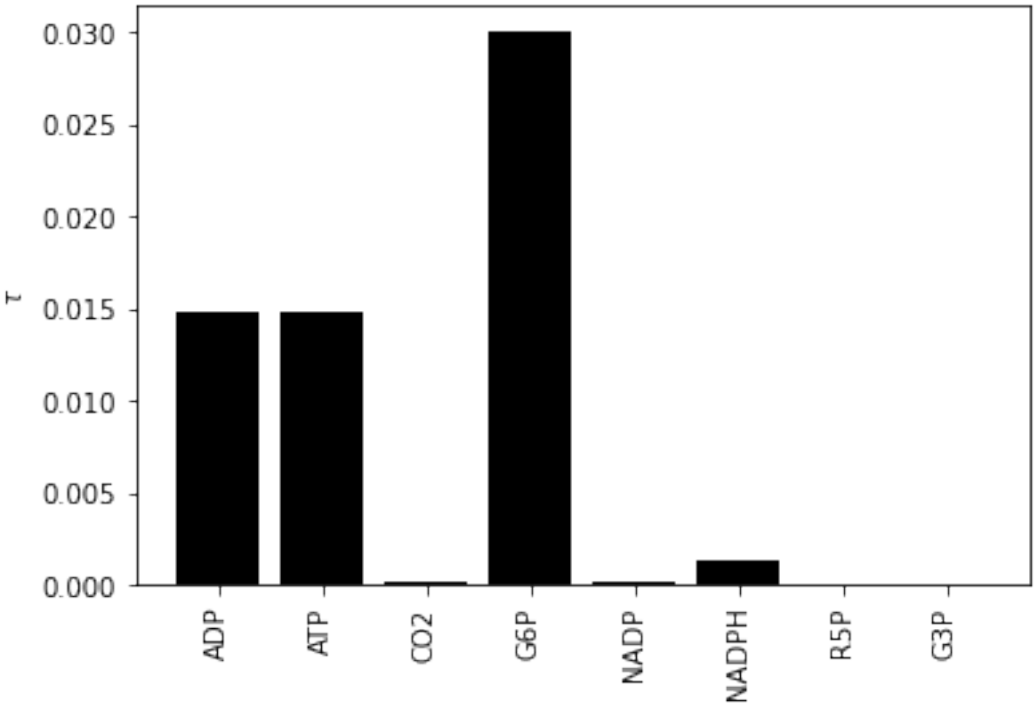
```
dlam = 10
G3P
```




```
Doing: sG3P
g = -4.574774223072978
dlam = 10
G3P
```



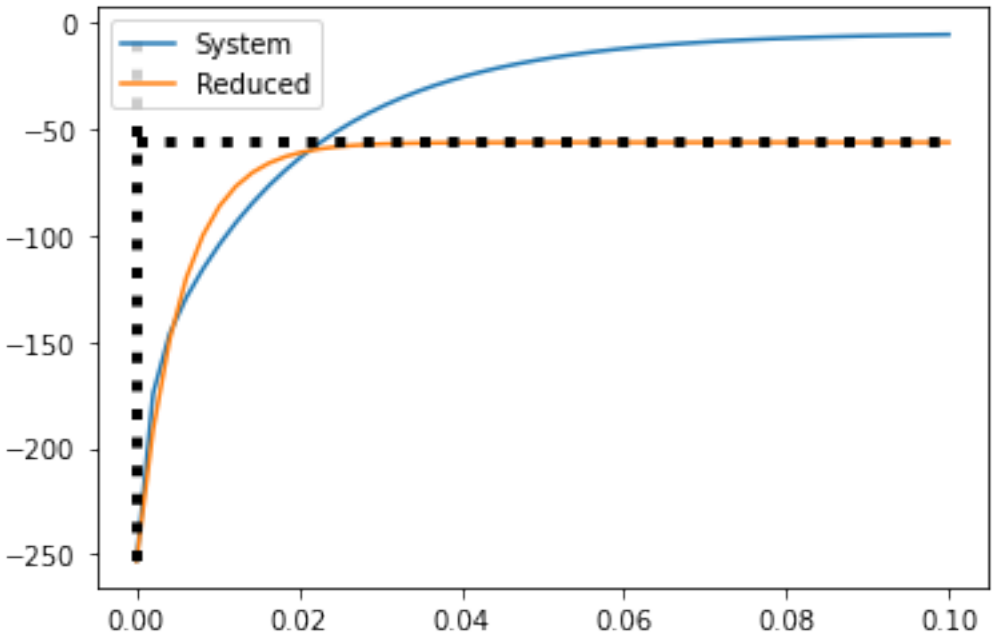

120
100
80
60
40
20
0
g_∞
ADP
ATP
CO2
G6P
NADP
NADPH
R5P
G3P


0.030
0.025
0.020
0.015
0.010
0.005
0.000
τ
ADP
ATP
CO2
G6P
NADP
NADPH
R5P
G3P

6.3 Test model reduction

[31]:
```
## Test model reduction
g,tau = tfProps(sys)
print(f'g = {g:.2f}, tau = {tau:.2f}')
sys1 = con.balred(sys,orders=1,method='truncate')
# print(con.dcgain(sys))
# print(con.dcgain(sys1))
con.tf(sys1)
g,tau = tfProps(sys1)
print(f'g = {g:.2f}, tau = {tau:.2f}')

t = np.linspace(0,0.1)
step = con.step_response(sys,T=t)
step1 = con.step_response(sys1,T=t)
plt.plot(t,step.outputs.T,label='System')
plt.plot(t,step1.outputs.T,label='Reduced')
plt.legend()
# plt.grid()
plotLines()
```

```
g = -4.57, tau = 0.00
g = -56.31, tau = 0.00
```



[32]:
```
print(Tau)
```

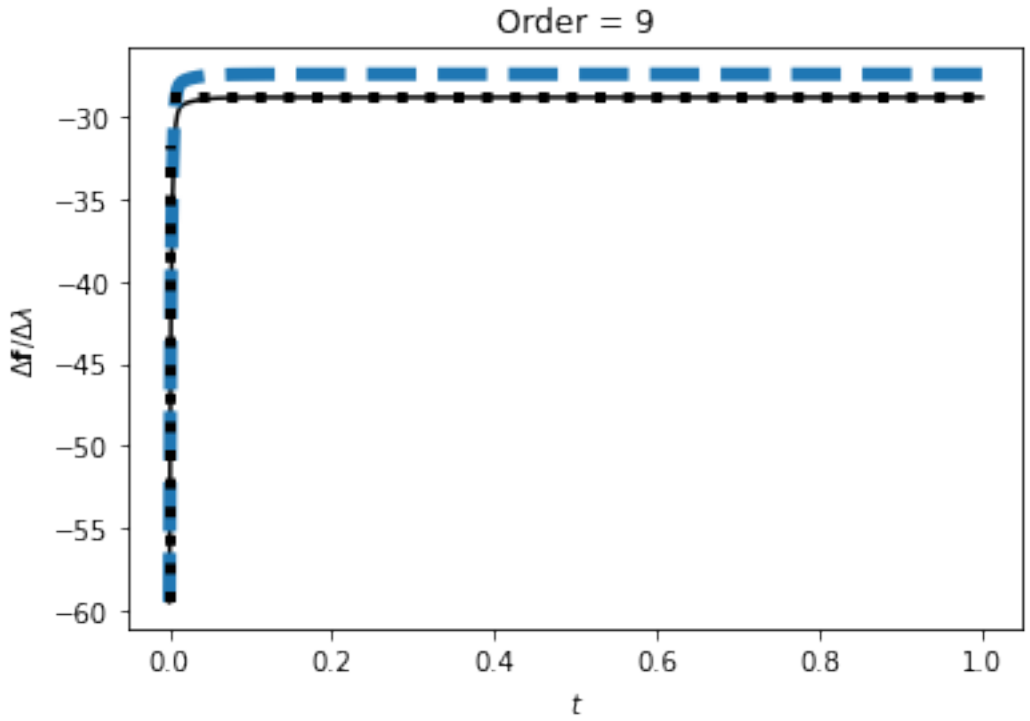
```
{'ADP': 0.01486615676575225, 'ATP': 0.014866156765752266, 'CO2':
0.00017684665624203704, 'G6P': 0.0300413900809015, 'NADP':
0.0001764287195764643, 'NADPH': 0.0013662874823763052, 'R5P': 0, 'G3P': 0}
```

6.4 Compute reaction sensitivities

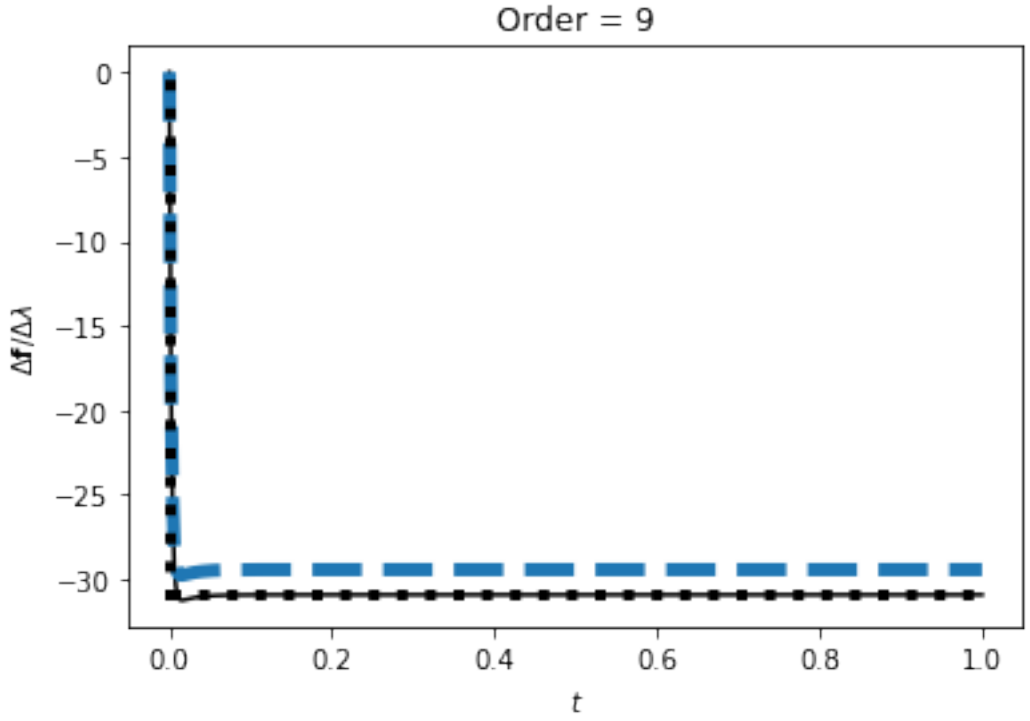
[33]:
```
## Reaction sensitivities
Outp = ['G6P','R5P','NADPH','G3P']
Inp = []
for reac in s['reaction']:
    Inp.append('s'+reac)

Inp_reac = Inp
# Inp = []
# for reac in s['reaction']:
#     Inp.append('s'+reac)
# Inp = ['sG6PDH2R']

t_last=1
lam = 1.1 # Perturbation parameter

dcgain = {}
Tau = {}
for outp in Outp:
    for inp in Inp:
        print('Doing:',inp)
        sf = None
        order = None
        dat,y_step,t,sys =␣
↪simSensitivity(s,sc,sf,Sys,X_ss,V_ss,dX_ss,chemostats=Chemostats,parameter=parameter,
                                                ␣
↪inp=[inp],outp=[outp],lam=lam,t_last=t_last,order=order,tol=None)

#         g = con.dcgain(sys)
        g,tau = tfProps(sys)
        print('g =',g, 'tau =',tau)
        poles = con.poles(sys)
#         print('Poles:', poles)
#         print('Zeros:', con.zeros(sys))
#         print(con.ss2tf(sys))
        name = inp[1:]
        if name == 'G6PDH2R':
            name = 'G6PD.'
        if outp=='G6P':
            dcgain[name] = -g
            ylabel = '$-g_{\infty}$'
        else:
            dcgain[name] = g
            ylabel = '$g_{\infty}$'

        Tau[name] = tau

        dlam = int((lam-1)*100)
        print('dlam =',dlam)
        plt.title(f'Order = {poles.size}')
```

```
        if Titles:
            plt.title(f'{name}'+r' ($\tilde\lambda$ = ' + f'{dlam}%; g={g:.
↪2f})')
        plotSensitivitydX(dat,species=[outp],labeling=False)
#         plt.hlines(g,min(t),max(t),color='black',ls='dashed')
        plotLines()
        Savefig('PPPdX_'+name+'_'+outp)

        plt.show()
        #print(con.tf(sys))


    ## Bar chart of gains
    if Titles:
        plt.title(outp)
    plt.tick_params(axis='x', rotation=90)
#     plt.grid(axis='y')
    plt.bar(range(len(dcgain)), dcgain.values(), align='center',color='black')
    plt.xticks(range(len(dcgain)), list(dcgain.keys()))
    plt.ylabel(ylabel)
    plt.axhline(0,color='black',ls='dashed')
    Savefig(f'PPPdX_{outp}_reac_bar')
    plt.show()

    ## Time constant
    if Titles:
        plt.title(outp)
    plt.tick_params(axis='x', rotation=90)
#     plt.grid(axis='y')
    plt.bar(range(len(Tau)), Tau.values(), align='center',color='black')
    plt.xticks(range(len(Tau)), list(Tau.keys()))
    plt.ylabel(r'$\tau$')
    Savefig(f'PPPdX_{outp}_reac_tau_bar')
    plt.show()
```

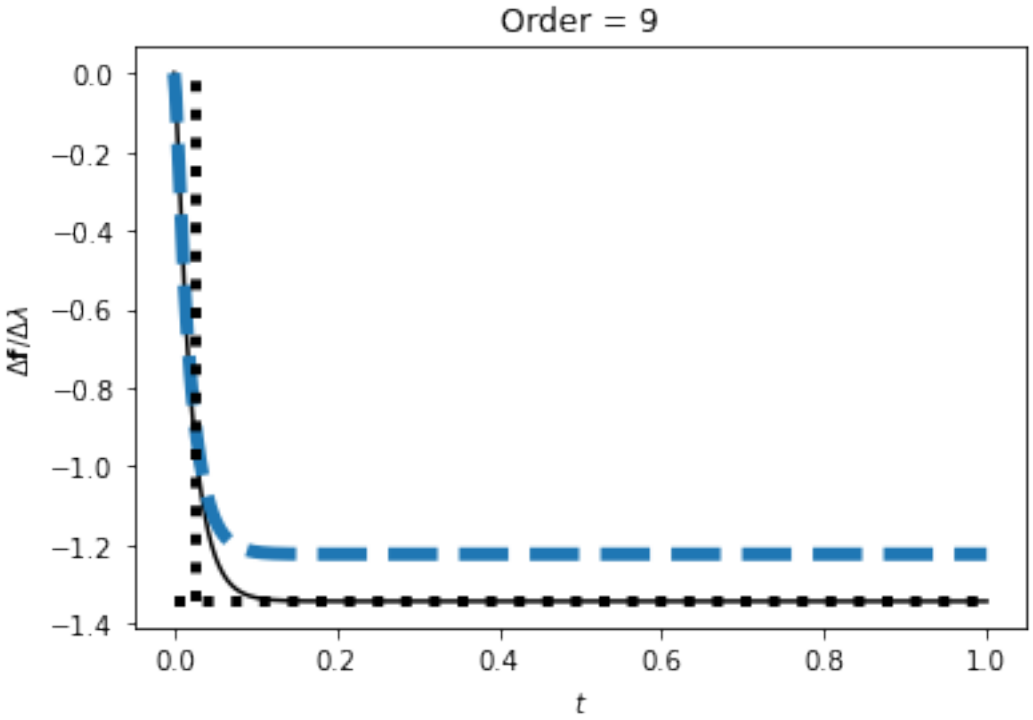
```
Doing: sPGI
kappa_PGI
g = -28.80991511396472 tau = 0
dlam = 10
G6P
```



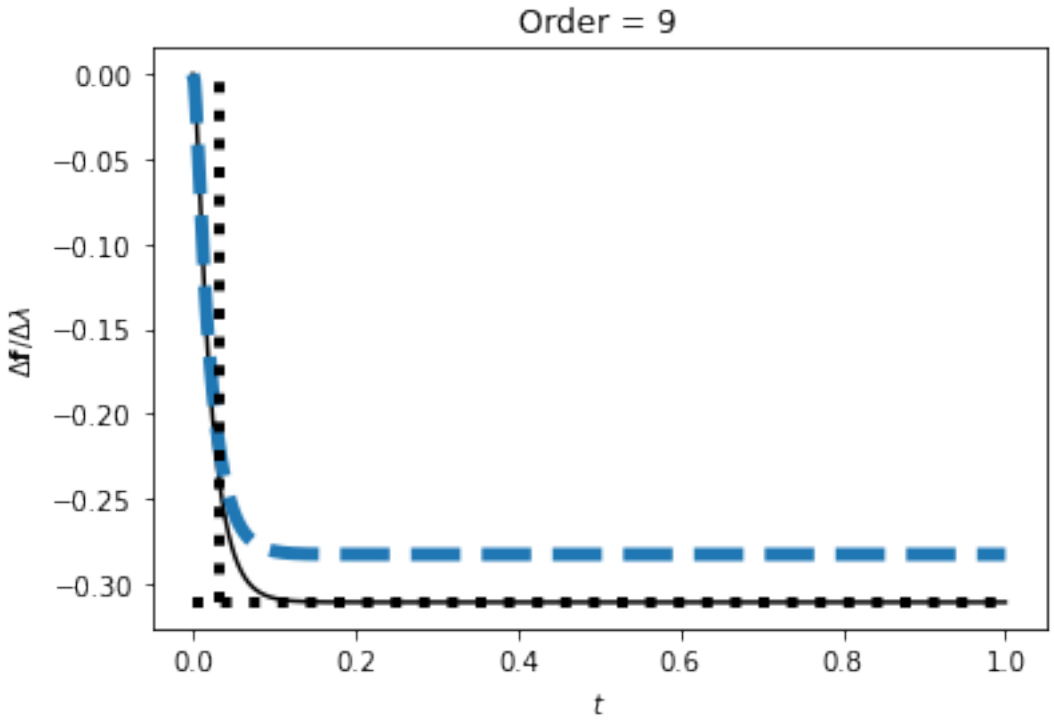

```
Doing: sPFK
kappa_PFK
g = -30.913609683373508 tau = 0.0022780158257760842
dlam = 10
G6P
```



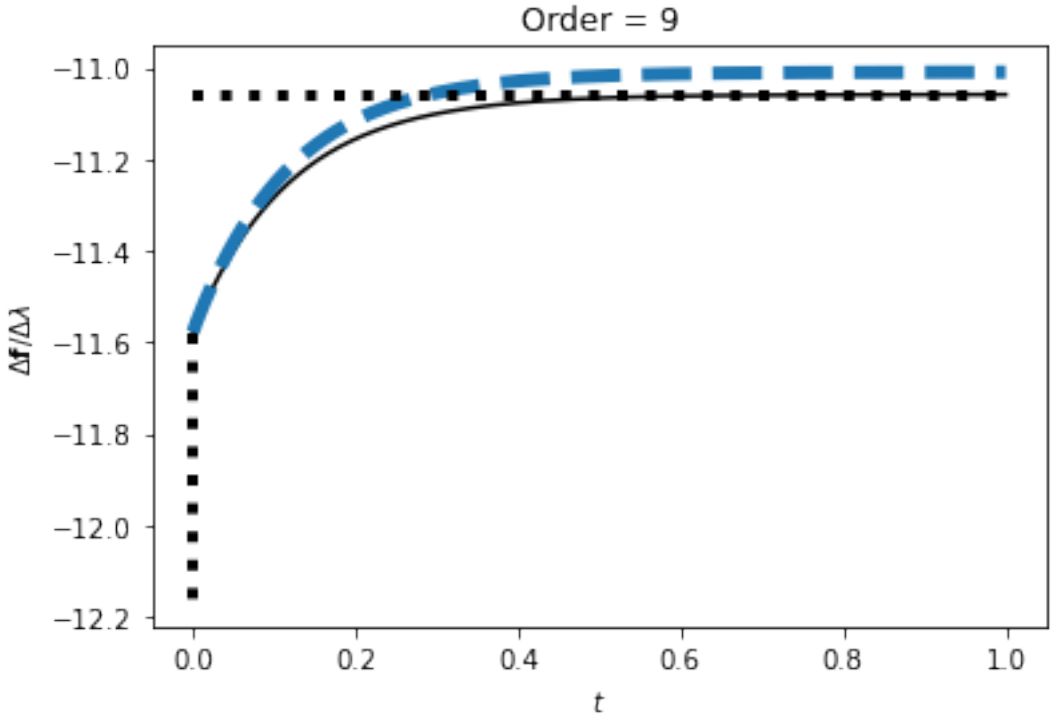
```
Doing: sFBA
kappa_FBA
g = -1.3427806651015182 tau = 0.026921154867136747
dlam = 10
G6P
```



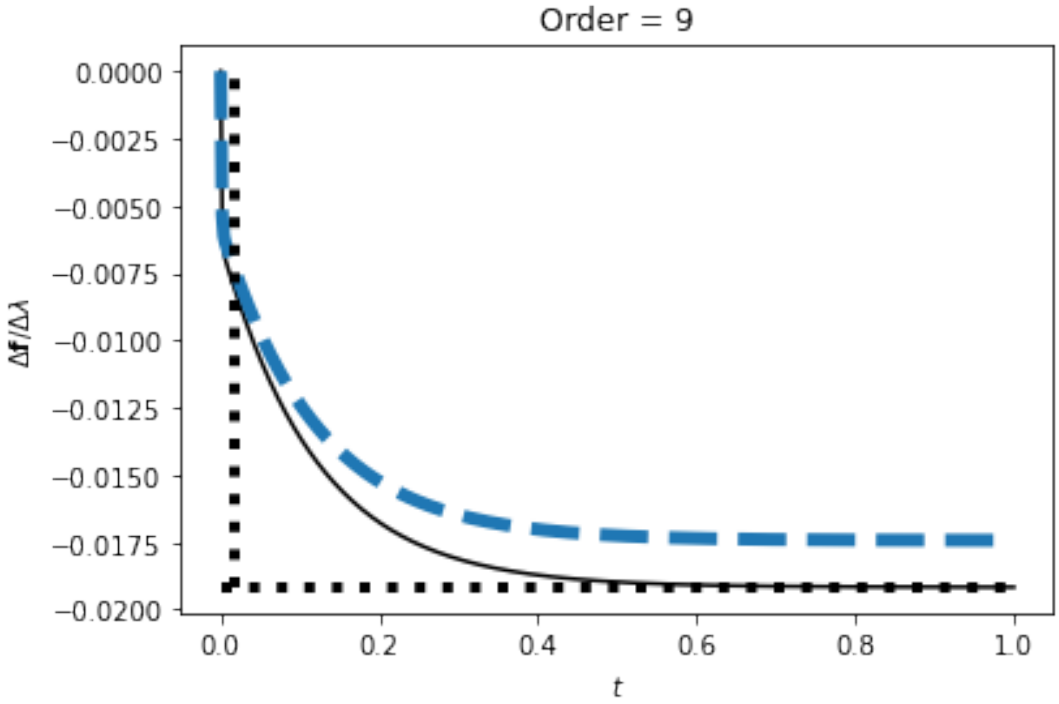

```
Doing: sTPI
kappa_TPI
g = -0.311245030898915 tau = 0.032926169765514844
dlam = 10
G6P
```



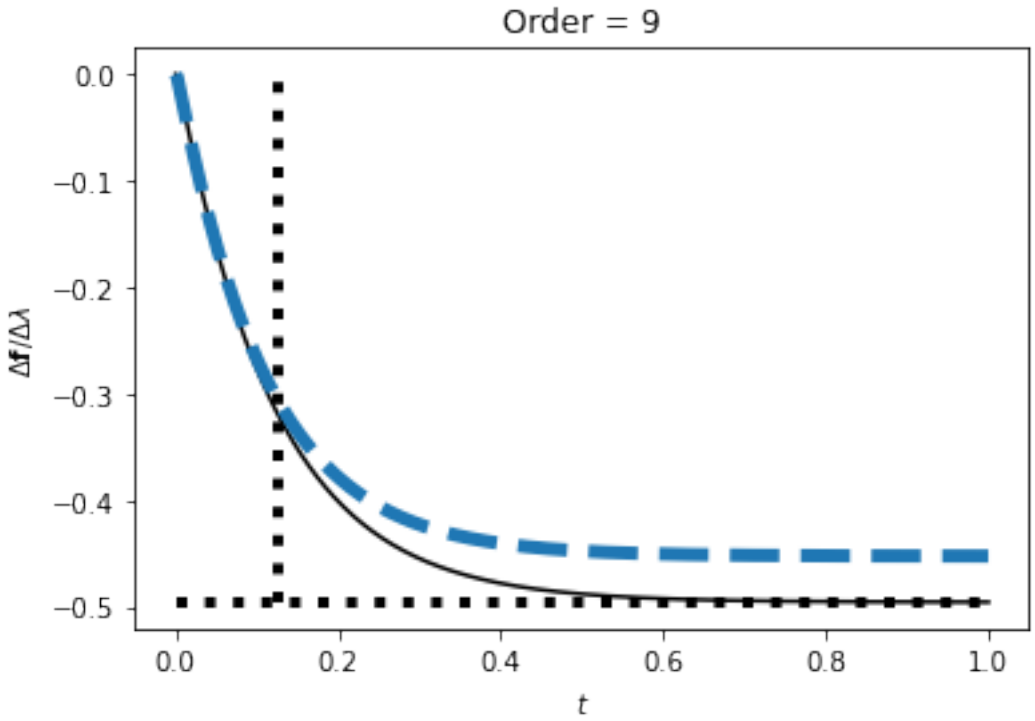

```
Doing: sG6PDH2R
kappa_G6PDH2R
g = -11.058345360939143 tau = 0
dlam = 10
G6P
```



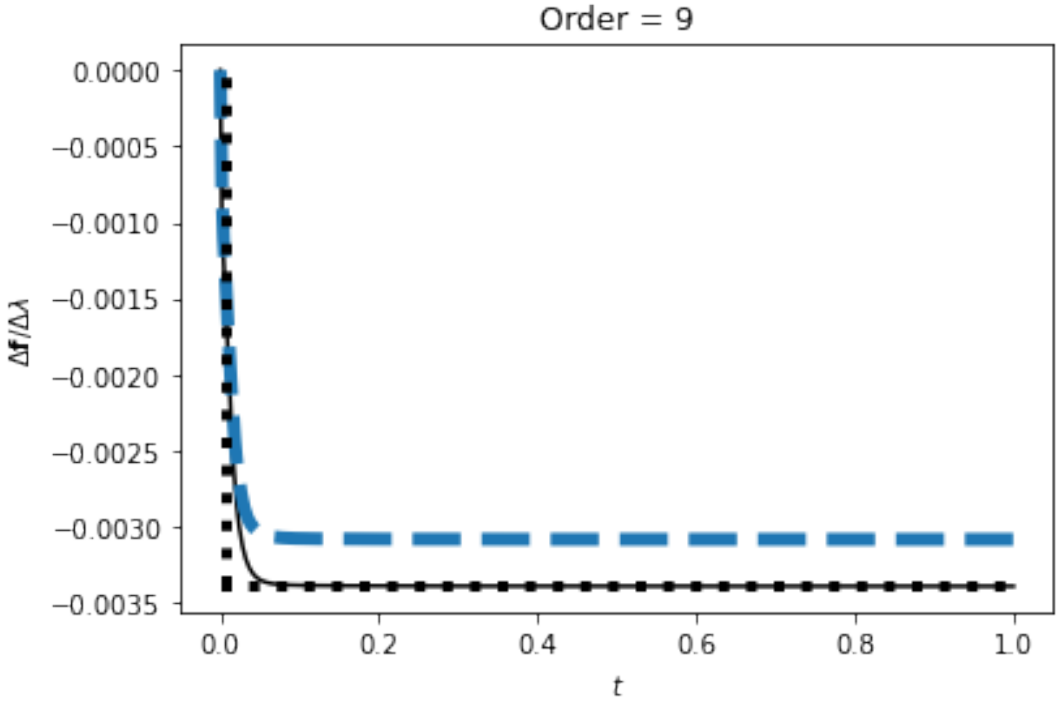
```
Doing: sPGL
kappa_PGL
g = -0.01916888096353947 tau = 0.016293783108218045
dlam = 10
G6P
```



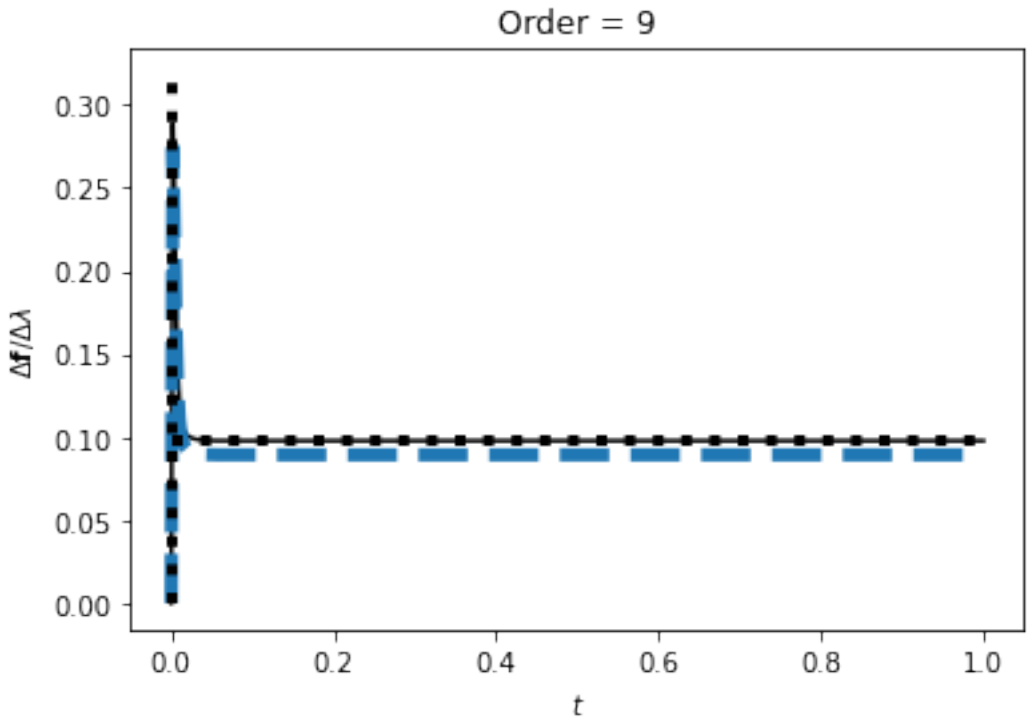

```
Doing: sGND
kappa_GND
g = -0.49587053107478085 tau = 0.12751609732060556
dlam = 10
G6P
```



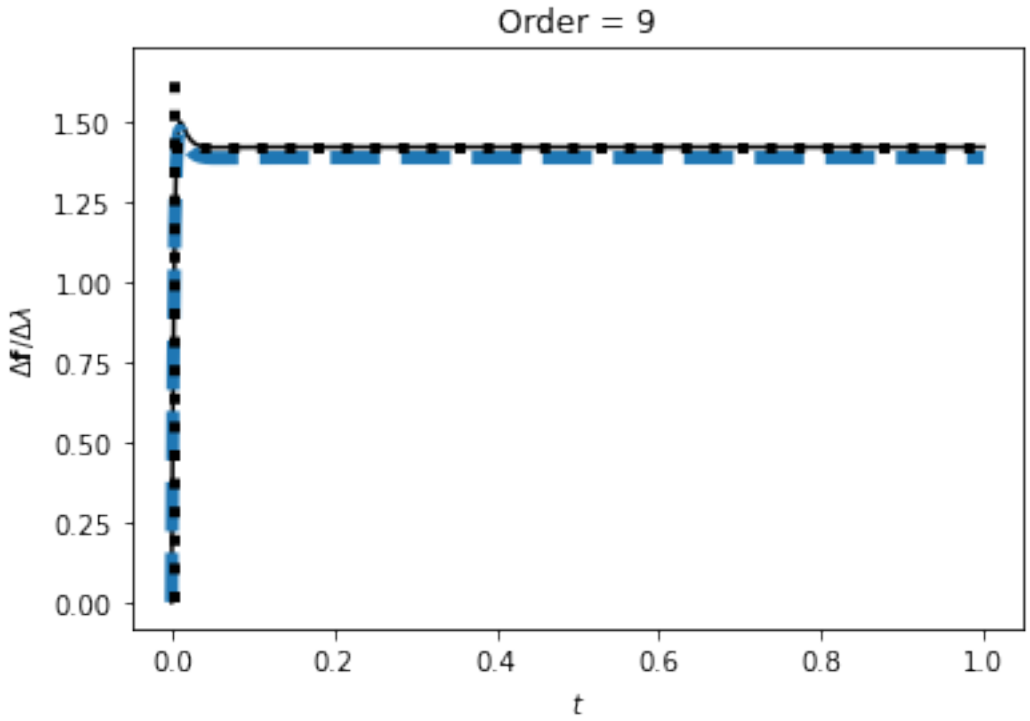

```
Doing: sRPI
kappa_RPI
g = -0.003392469771817826 tau = 0.008244435412417003
dlam = 10
G6P
```



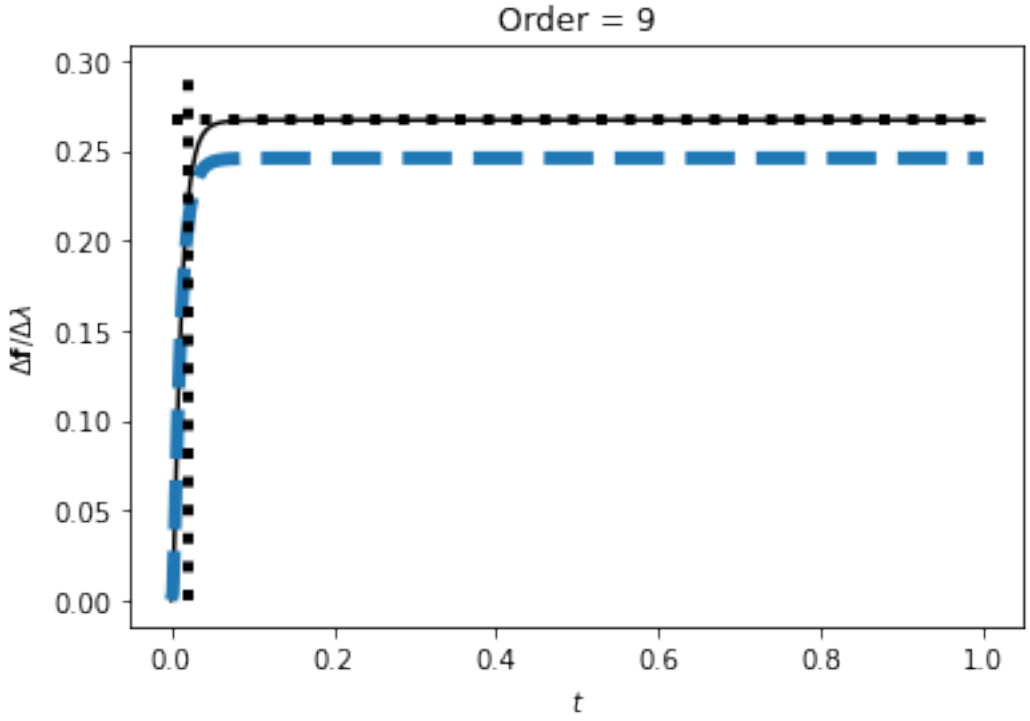
```
Doing: sTKT2
kappa_TKT2
g = 0.0977849661428157 tau = 0.0009111118563853972
dlam = 10
G6P
```



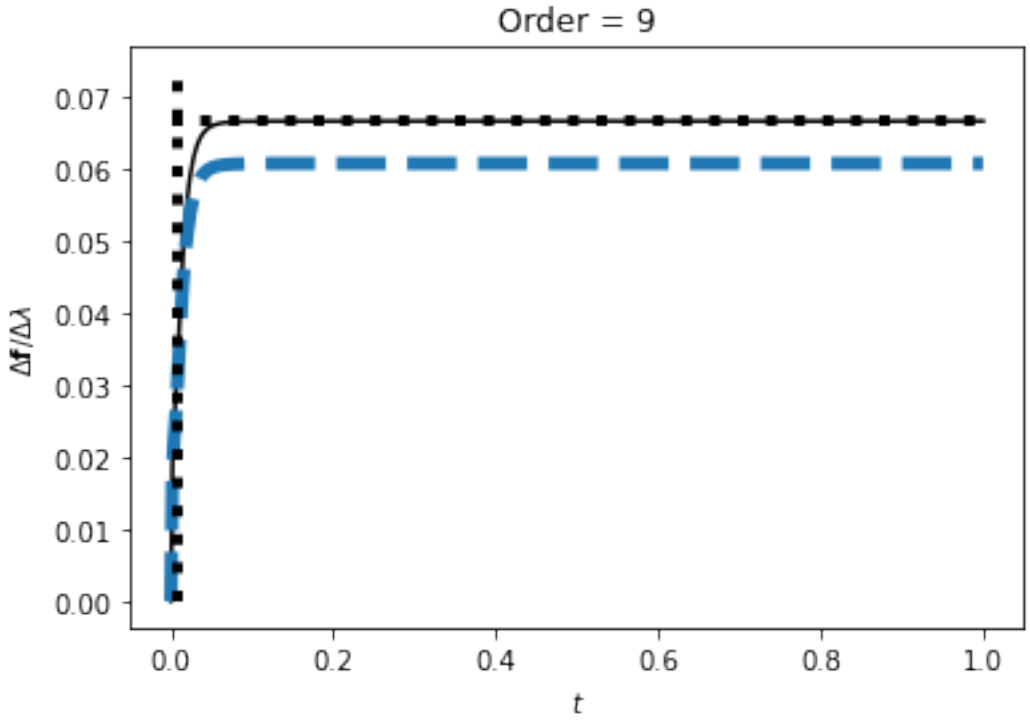

```
Doing: sTALA
kappa_TALA
g = 1.4213693479775227 tau = 0.003398714973903035
dlam = 10
G6P
```

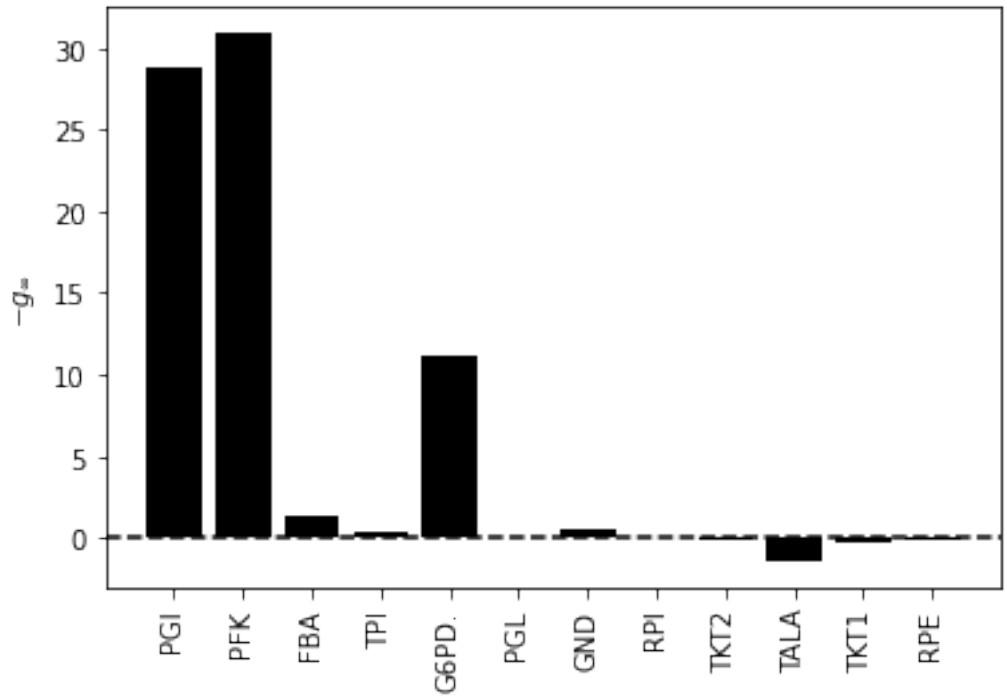


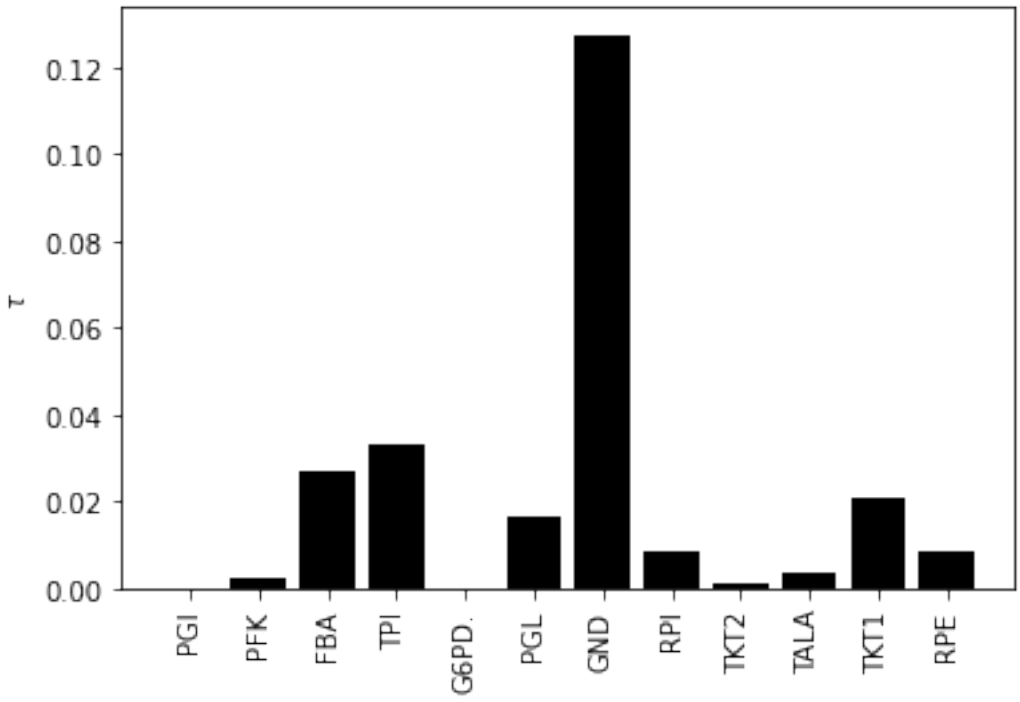

```
Doing: sTKT1
kappa_TKT1
g = 0.26692110033037636 tau = 0.02100117400639687
dlam = 10
G6P
```



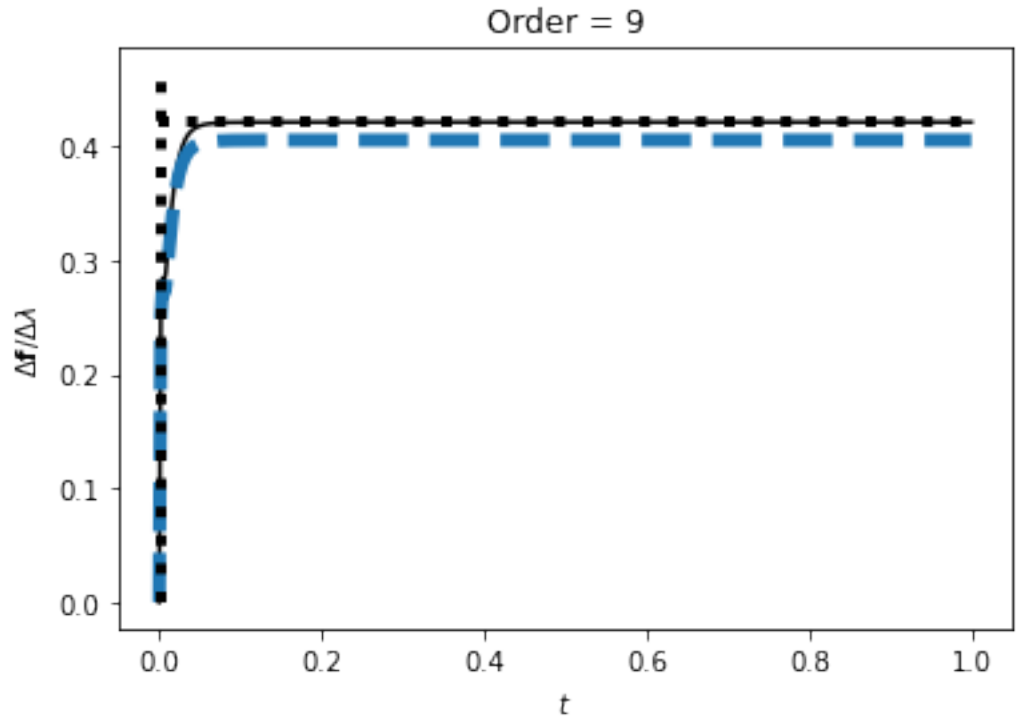
```
Doing: sRPE
kappa_RPE
g = 0.06664804356236083 tau = 0.008221010075271623
dlam = 10
G6P
```



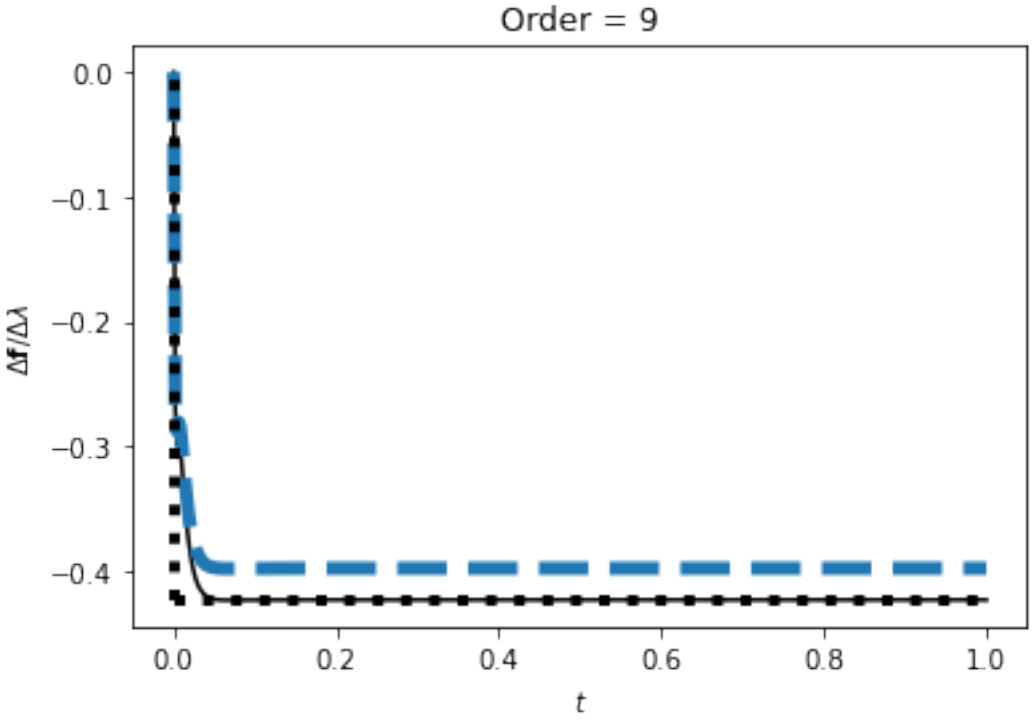




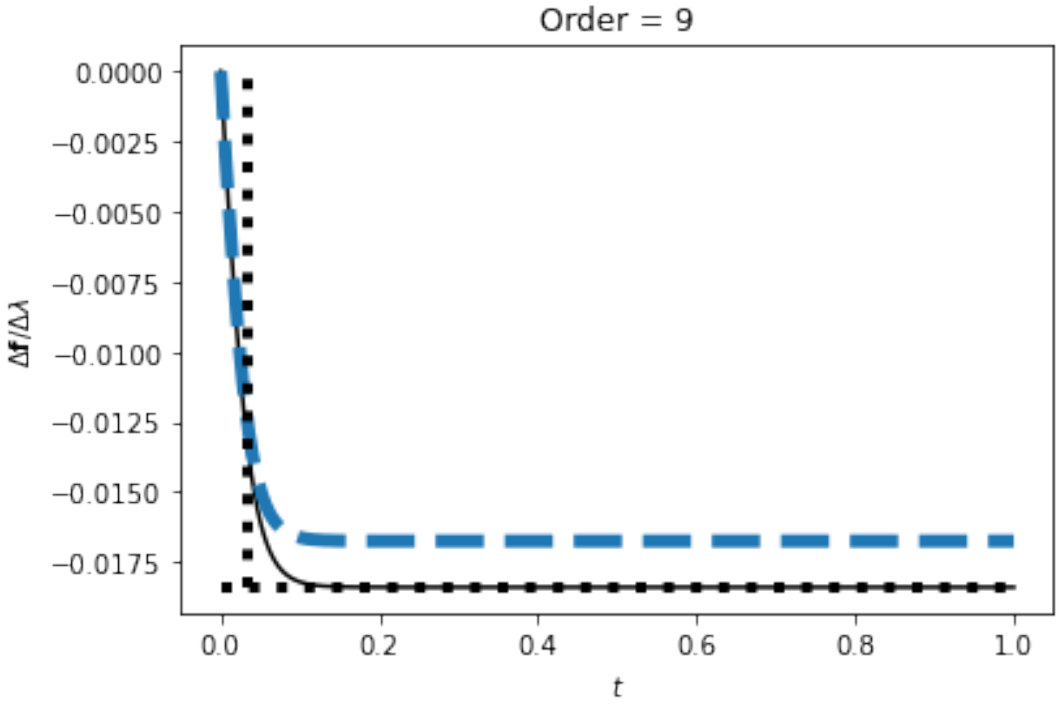

```
Doing: sPGI
kappa_PGI
g = 0.42089874176955533 tau = 0.0017840191168220973
dlam = 10
R5P
```



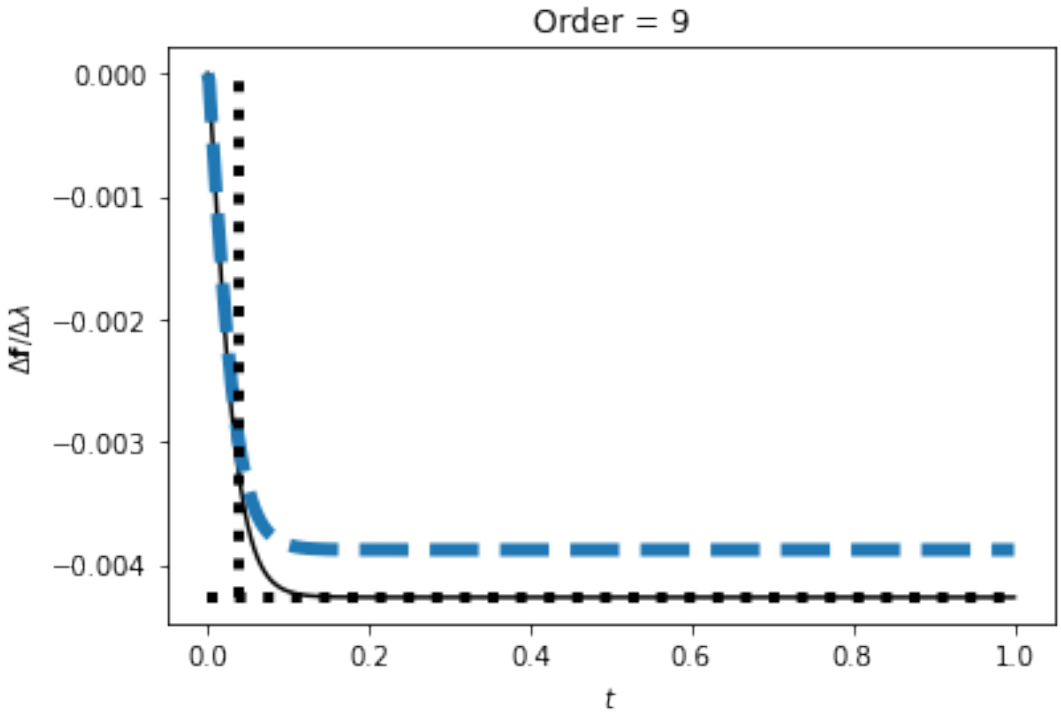

```
Doing: sPFK
kappa_PFK
g = -0.42364263563040794 tau = 0.0017179649901805588
dlam = 10
R5P
```



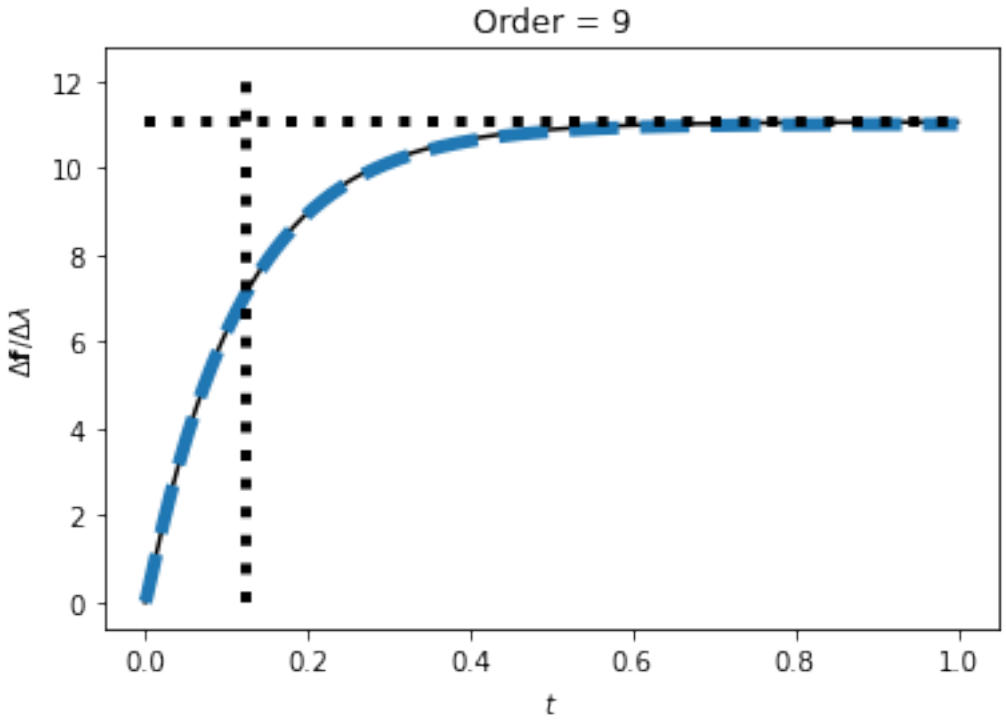
```
Doing: sFBA
kappa_FBA
g = -0.018401576065087966 tau = 0.033394801224602685
dlam = 10
R5P
```



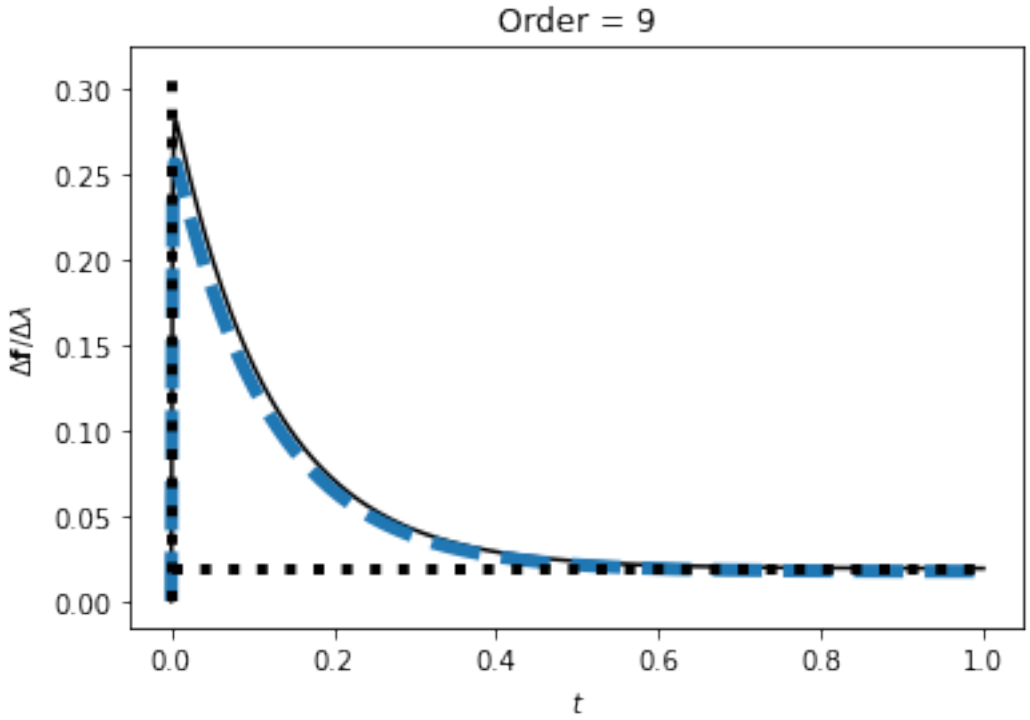

```
Doing: sTPI
kappa_TPI
g = -0.004265327359387783 tau = 0.03946956823549017
dlam = 10
R5P
```



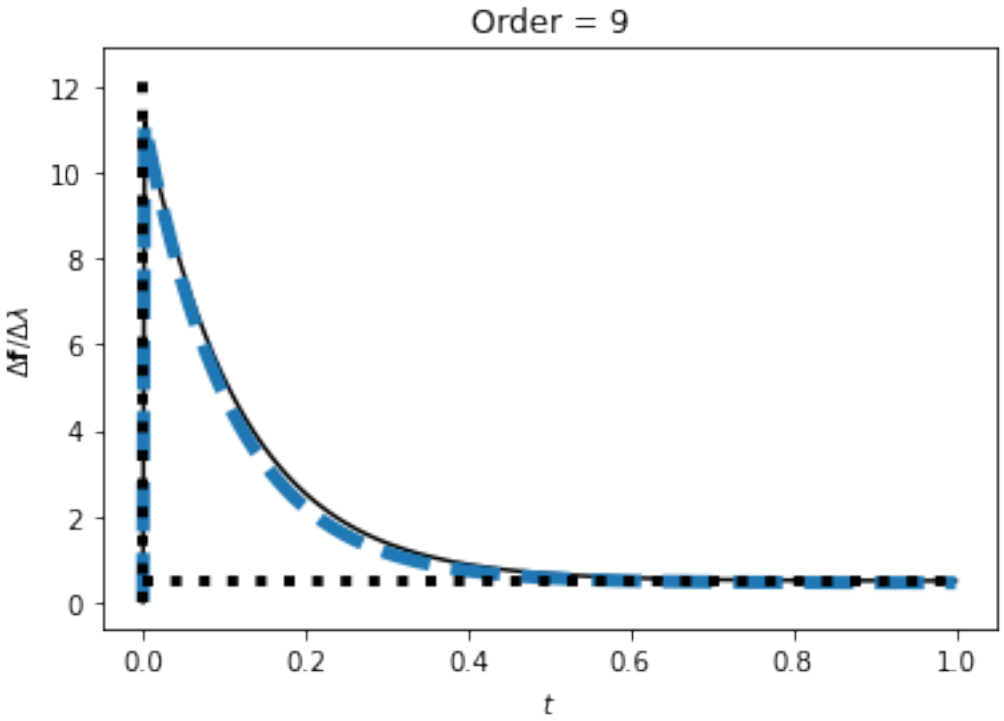

```
Doing: sG6PDH2R
kappa_G6PDH2R
g = 11.049474162936548 tau = 0.12364579561291006
dlam = 10
R5P
```



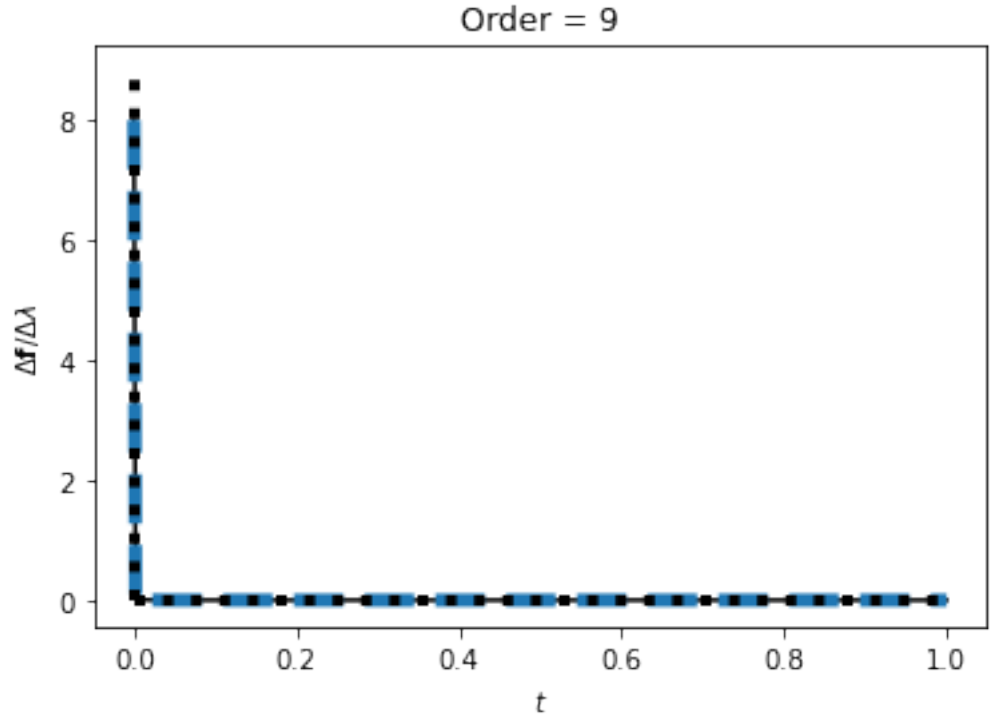
```
Doing: sPGL
kappa_PGL
g = 0.019153503351911308 tau = 0.001047059603233129
dlam = 10
R5P
```



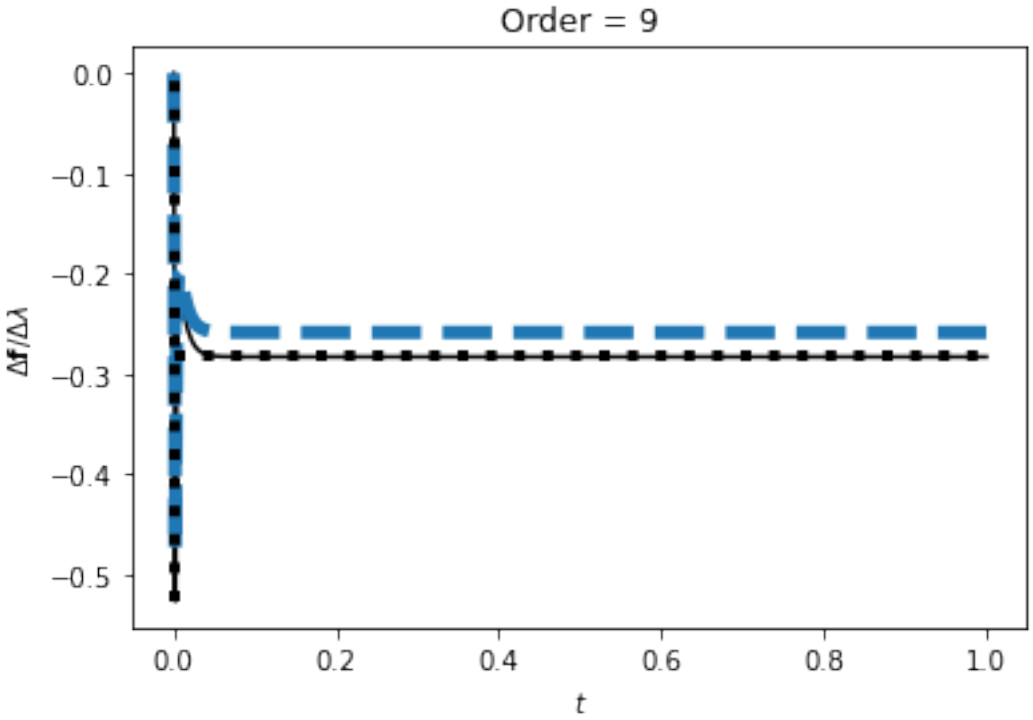

```
Doing: sGND
kappa_GND
g = 0.49547273506016615 tau = 2.6988194236854522e-06
dlam = 10
R5P
```



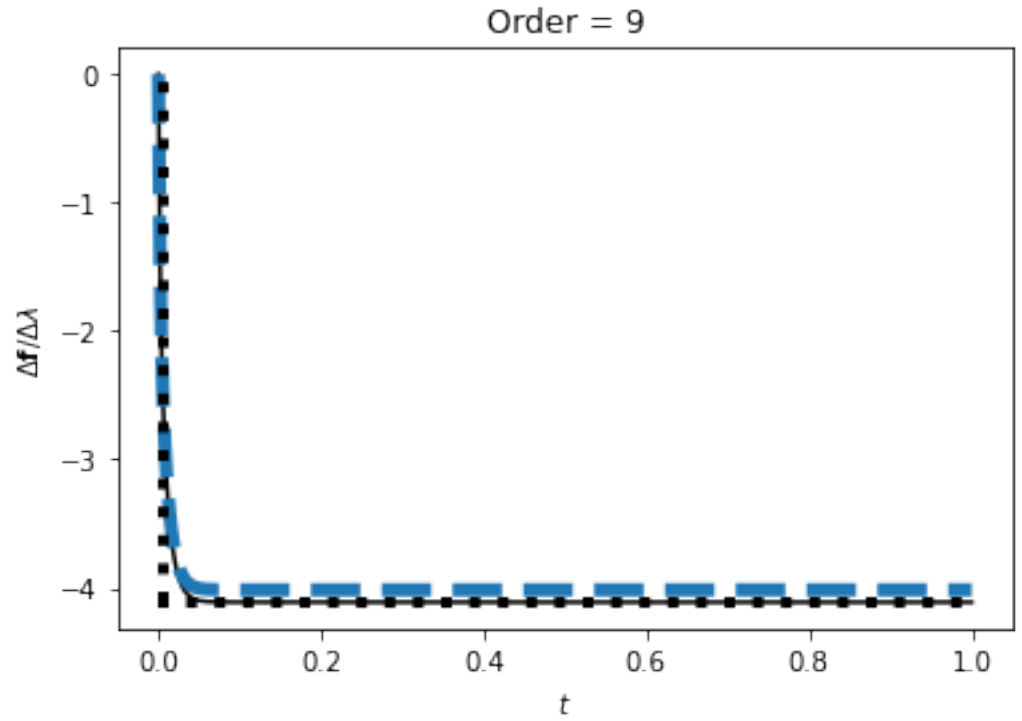

```
Doing: sRPI
kappa_RPI
g = 0.009795231203676202 tau = 0
dlam = 10
R5P
```



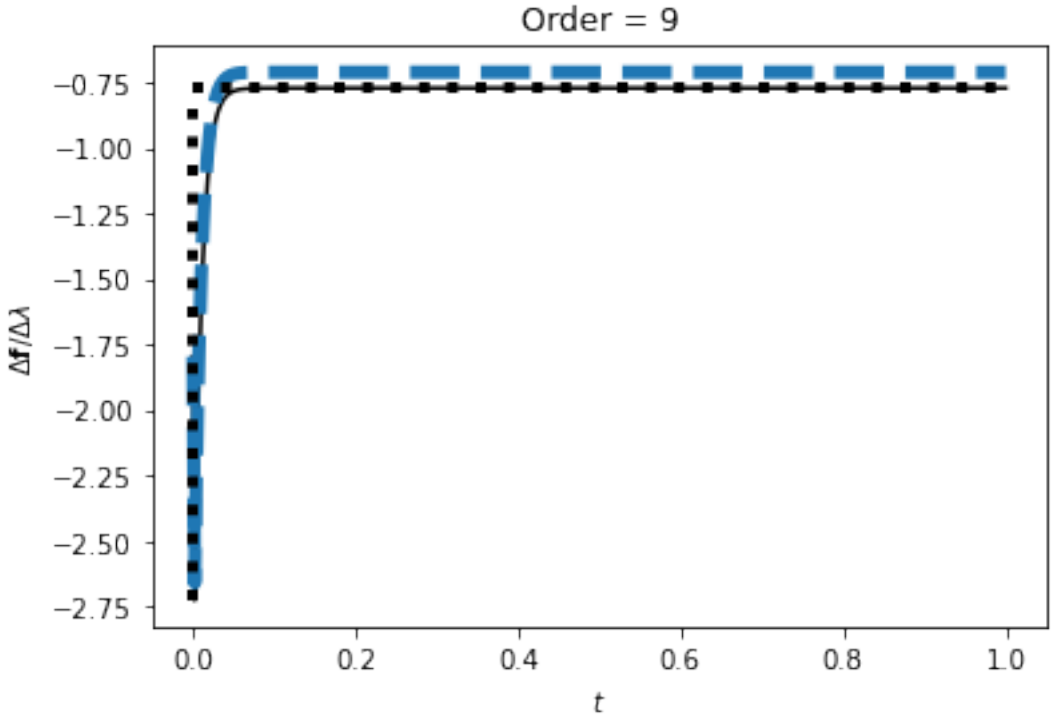
```
Doing: sTKT2
kappa_TKT2
g = -0.28292495485280156 tau = 0.00017126550296392347
dlam = 10
R5P
```



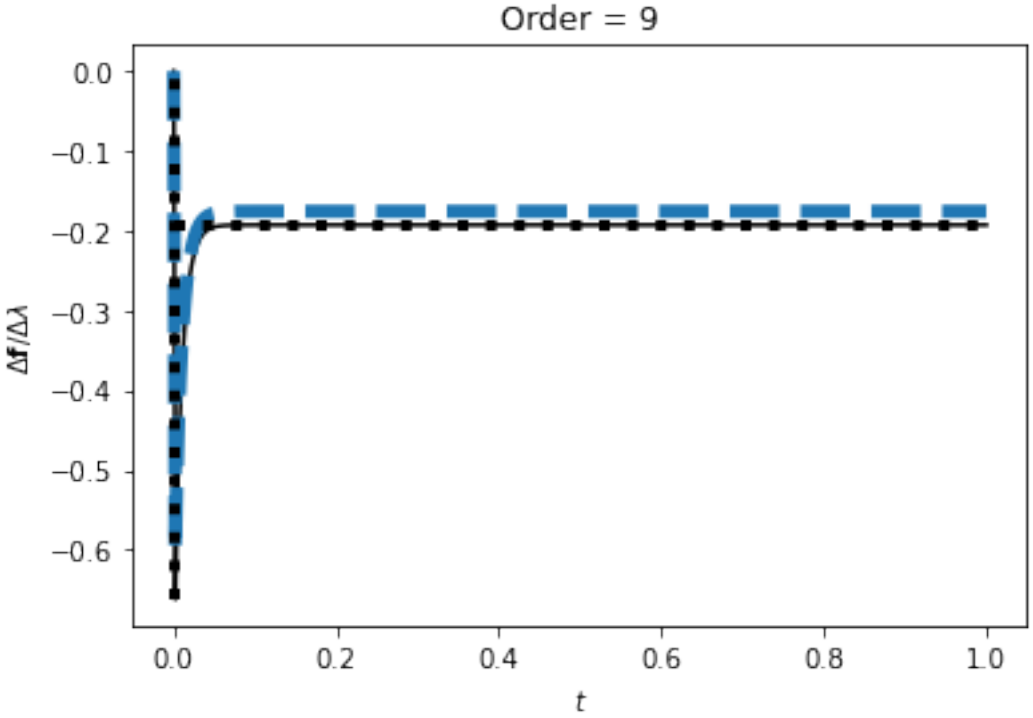

```
Doing: sTALA
kappa_TALA
g = -4.112501895417795 tau = 0.004516099692879739
dlam = 10
R5P
```

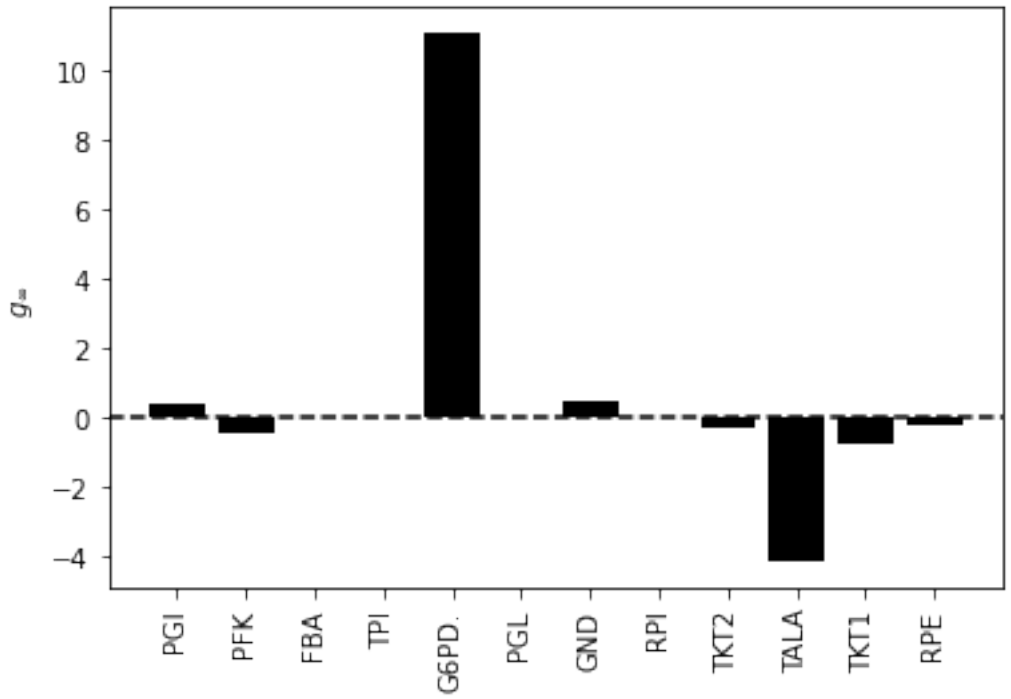


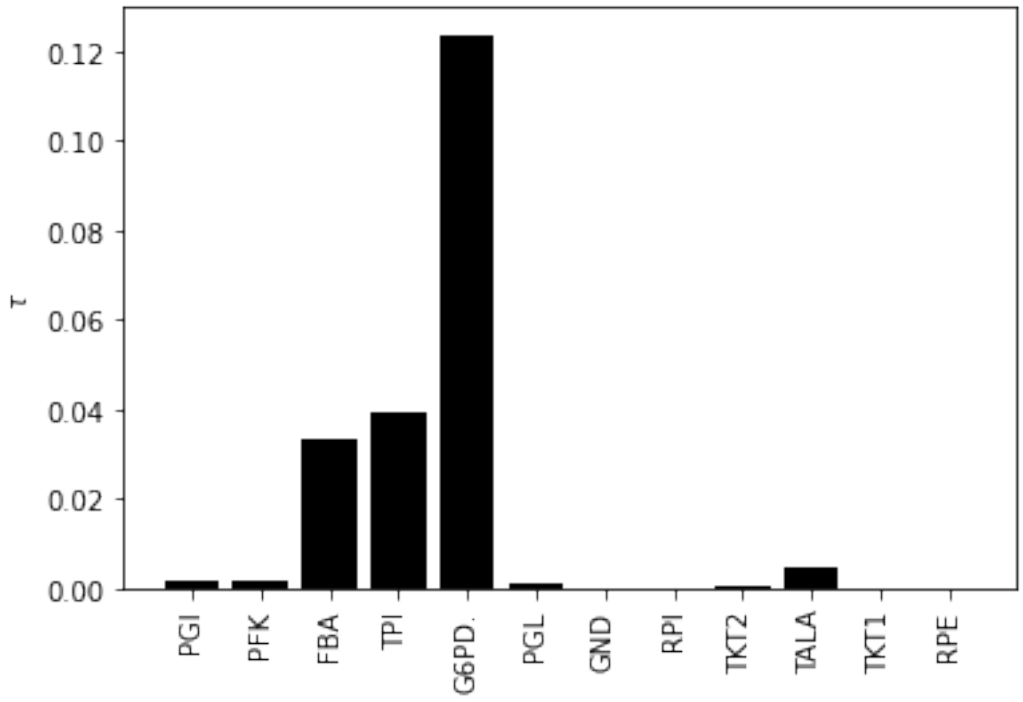

```
Doing: sTKT1
kappa_TKT1
g = -0.7722929529876423 tau = 0
dlam = 10
R5P
```



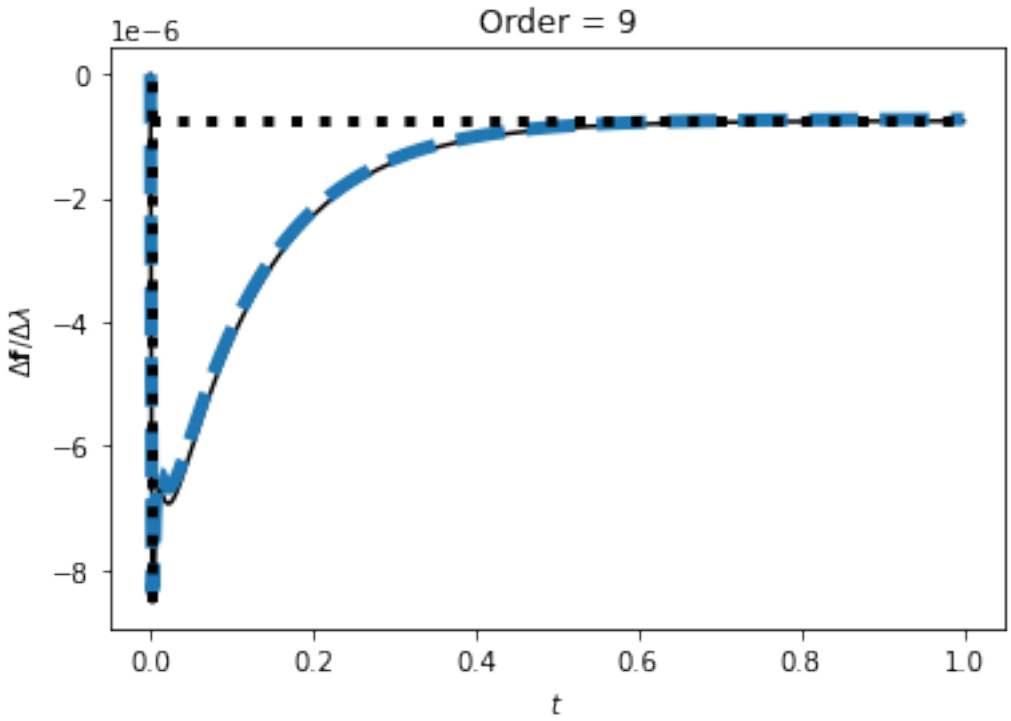
```
Doing: sRPE
kappa_RPE
g = -0.19283531466757514 tau = 2.657174406414892e-06
dlam = 10
R5P
```



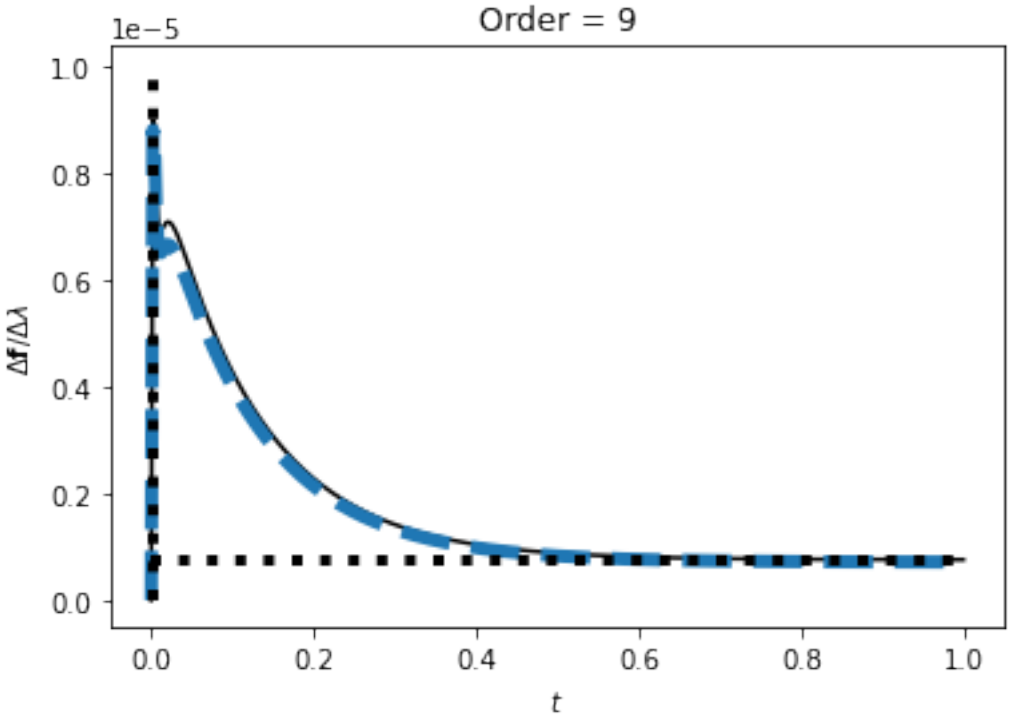




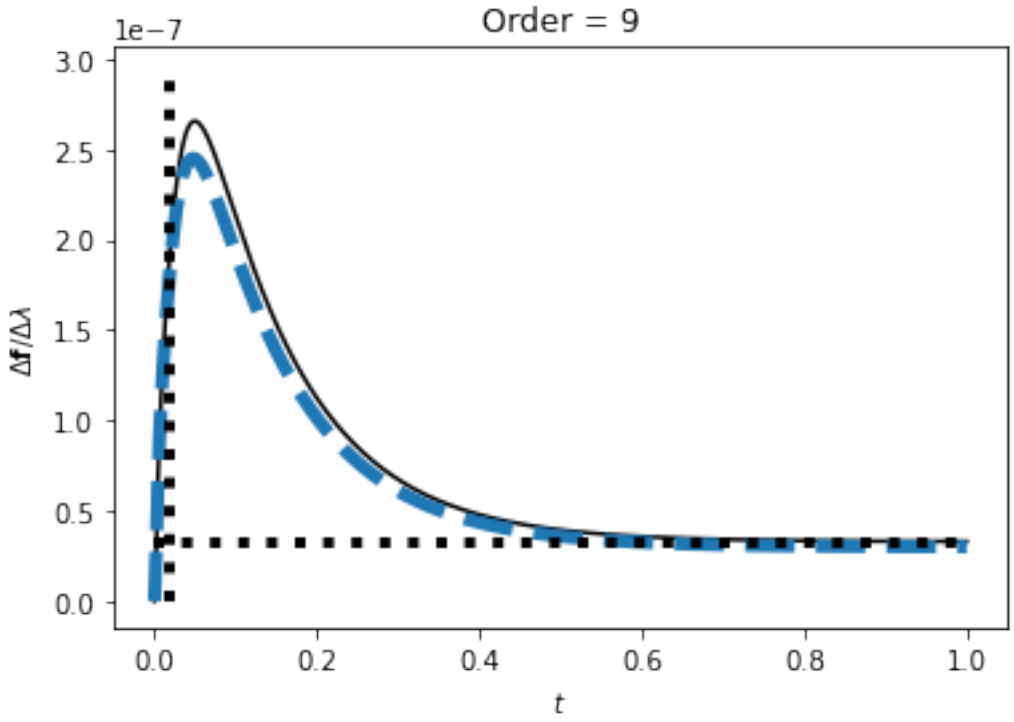

```
Doing: sPGI
kappa_PGI
g = -7.546999116173243e-07 tau = 0.0015271411542590262
dlam = 10
NADPH
```



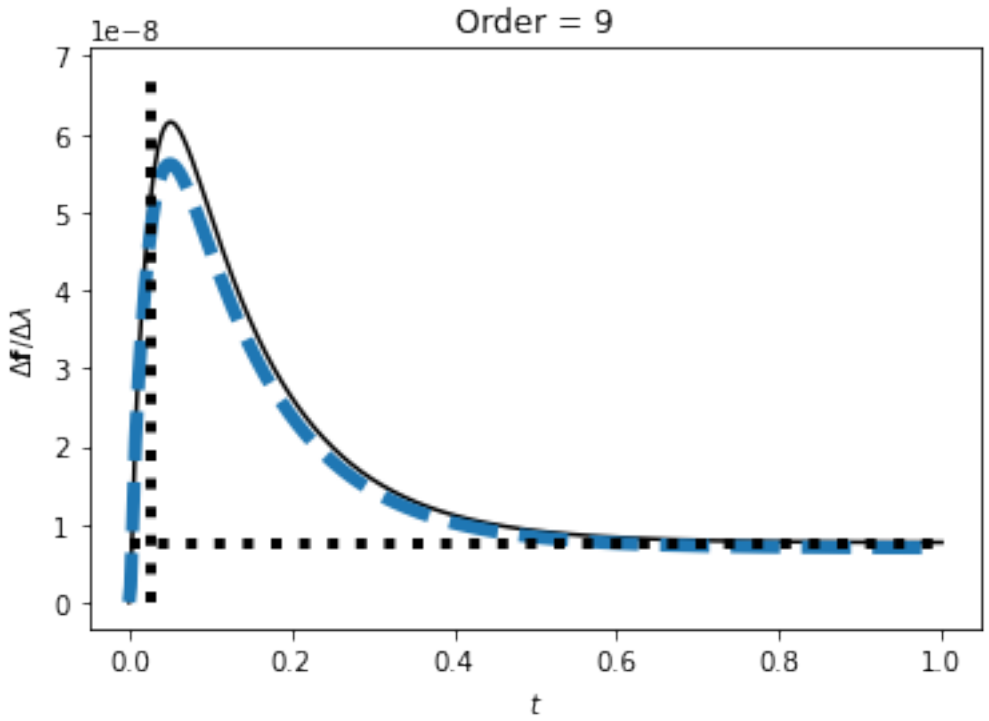

```
Doing: sPFK
kappa_PFK
g = 7.596200113605279e-07 tau = 0.0015261943288074609
dlam = 10
NADPH
```



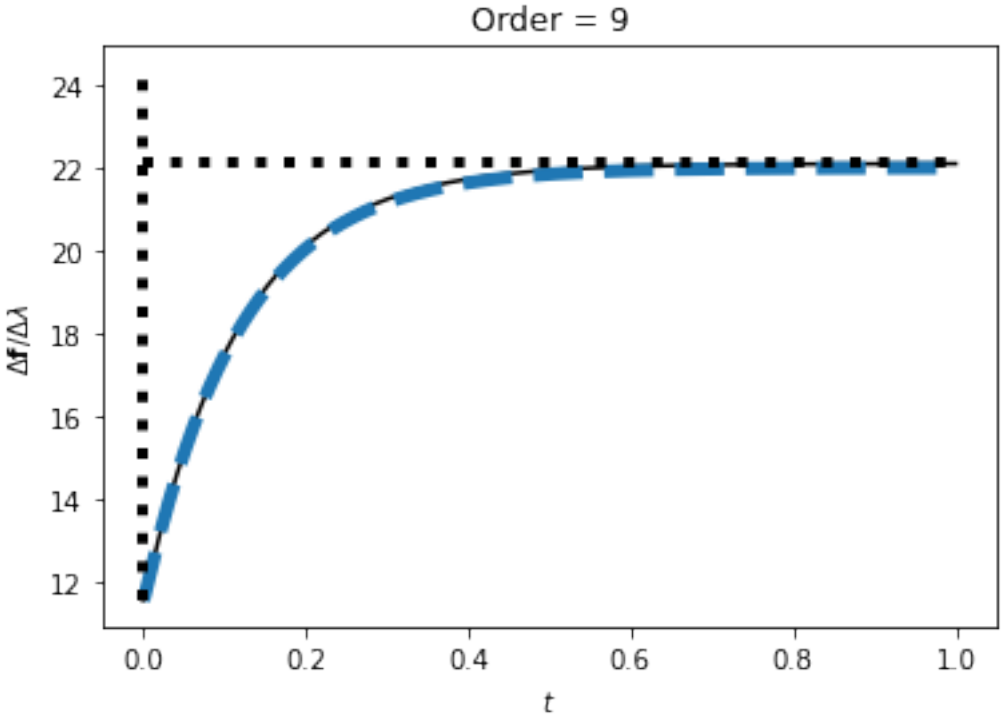
```
Doing: sFBA
kappa_FBA
g = 3.299527473205749e-08 tau = 0.01952750730586499
dlam = 10
NADPH
```



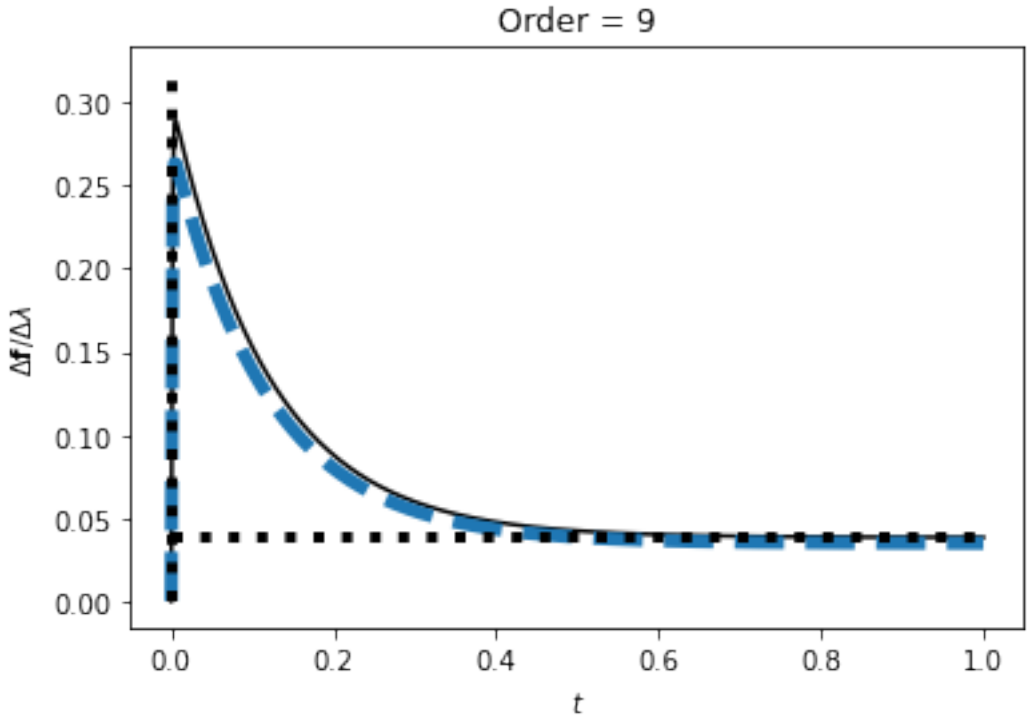

```
Doing: sTPI
kappa_TPI
g = 7.647903722590634e-09 tau = 0.02541573002381988
dlam = 10
NADPH
```



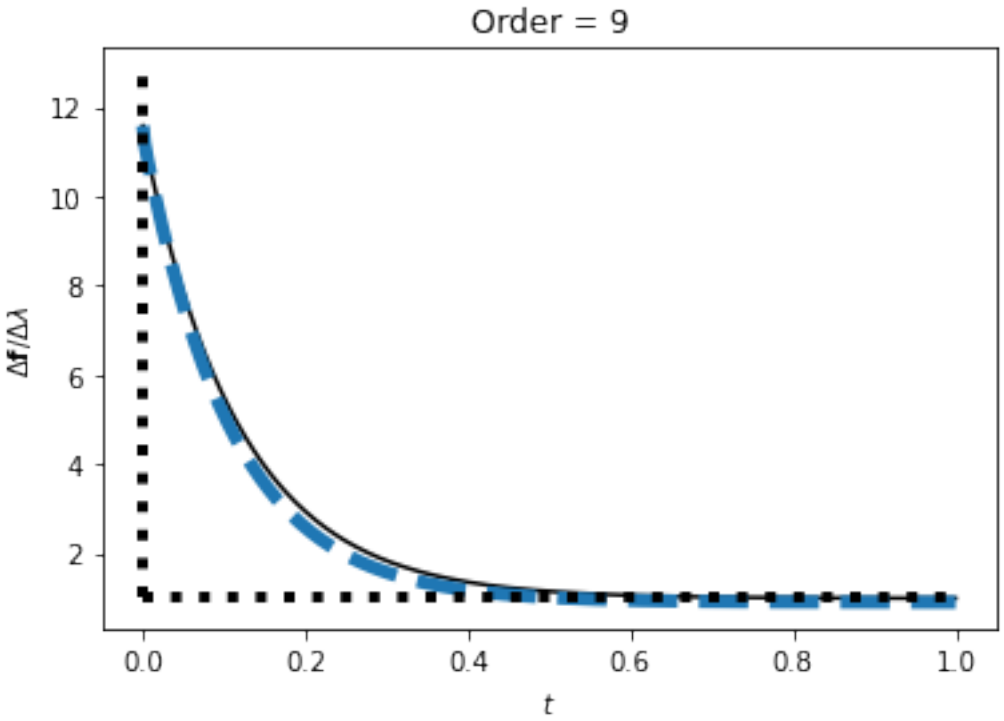

```
Doing: sG6PDH2R
kappa_G6PDH2R
g = 22.12606170707668 tau = 0
dlam = 10
NADPH
```



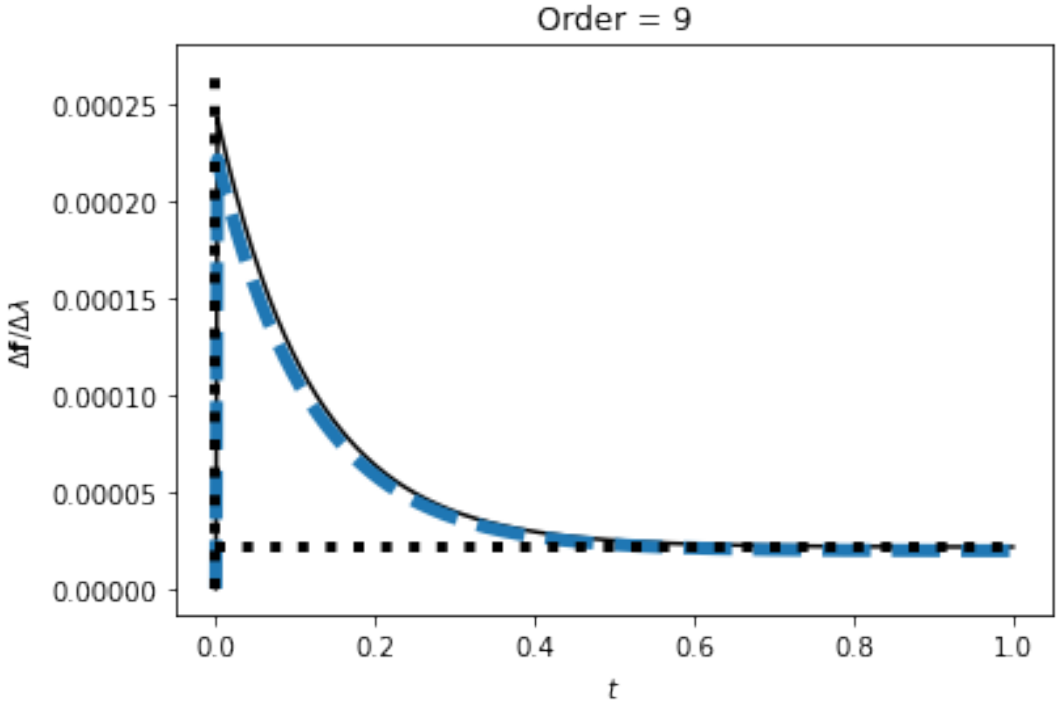
```
Doing: sPGL
kappa_PGL
g = 0.03835400588527505 tau = 0.001025533527882495
dlam = 10
NADPH
```



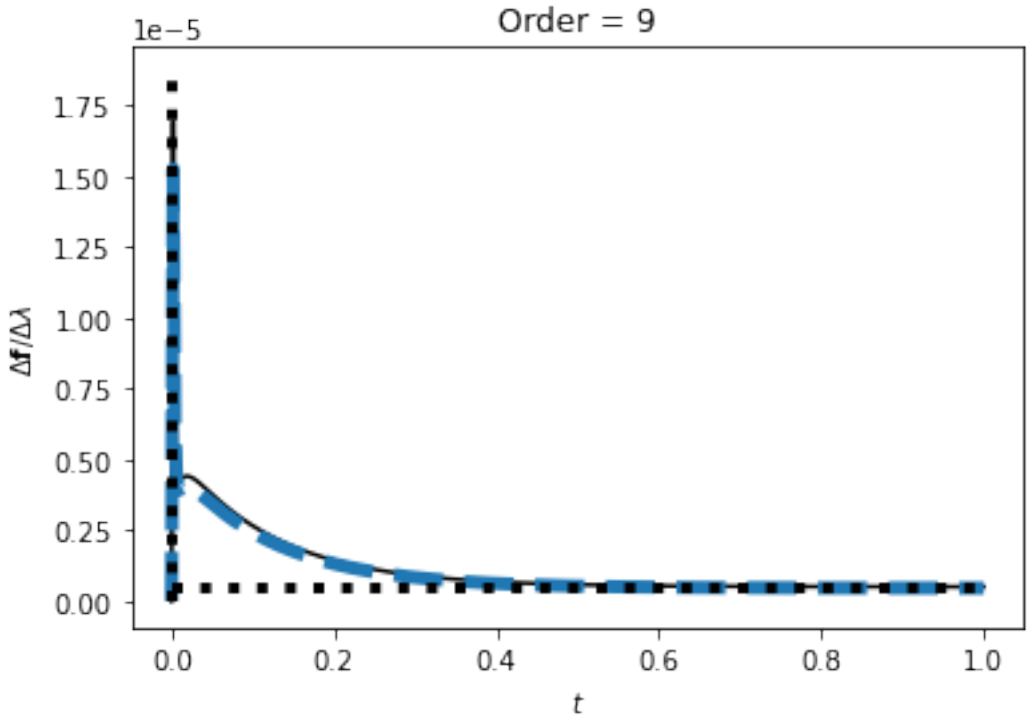

```
Doing: sGND
kappa_GND
g = 0.9921612692651145 tau = 0
dlam = 10
NADPH
```



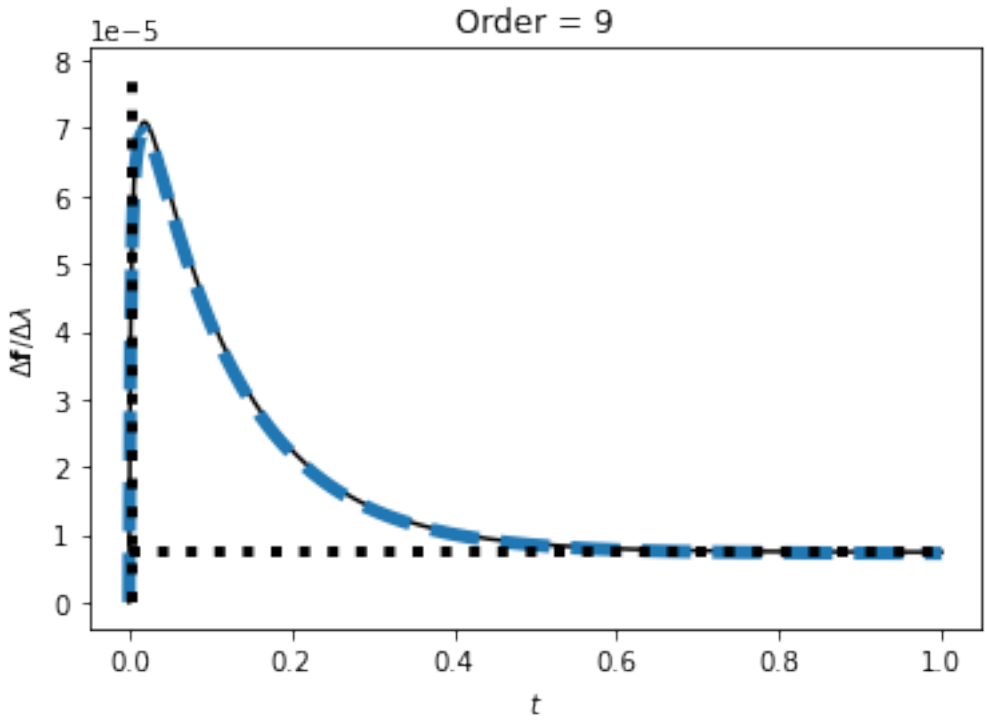

```
Doing: sRPI
kappa_RPI
g = 2.145815564140889e-05 tau = 2.699186974424083e-06
dlam = 10
NADPH
```



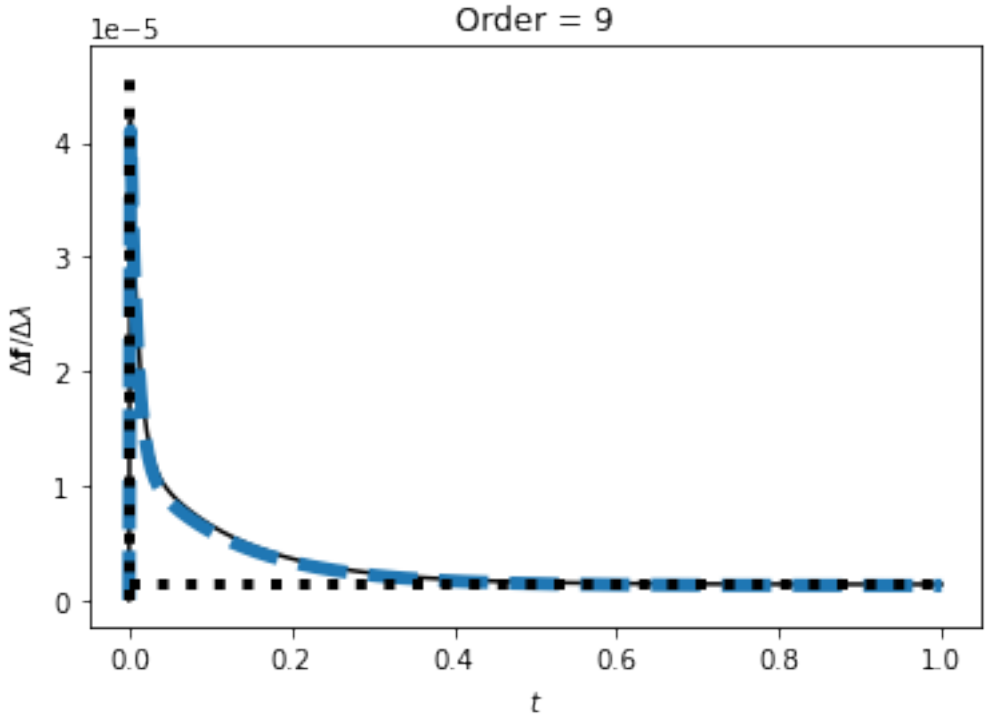
```
Doing: sTKT2
kappa_TKT2
g = 5.073036158009343e-07 tau = 0.00016943022110074563
dlam = 10
NADPH
```



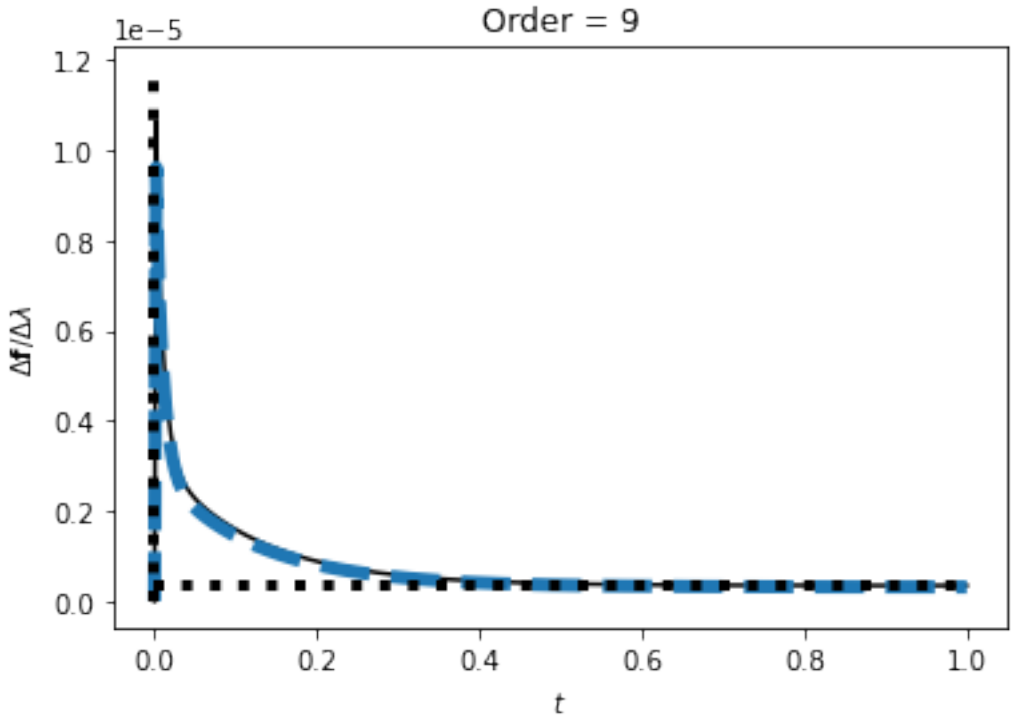

```
Doing: sTALA
kappa_TALA
g = 7.373994572595707e-06 tau = 0.0028552247467923974
dlam = 10
NADPH
```

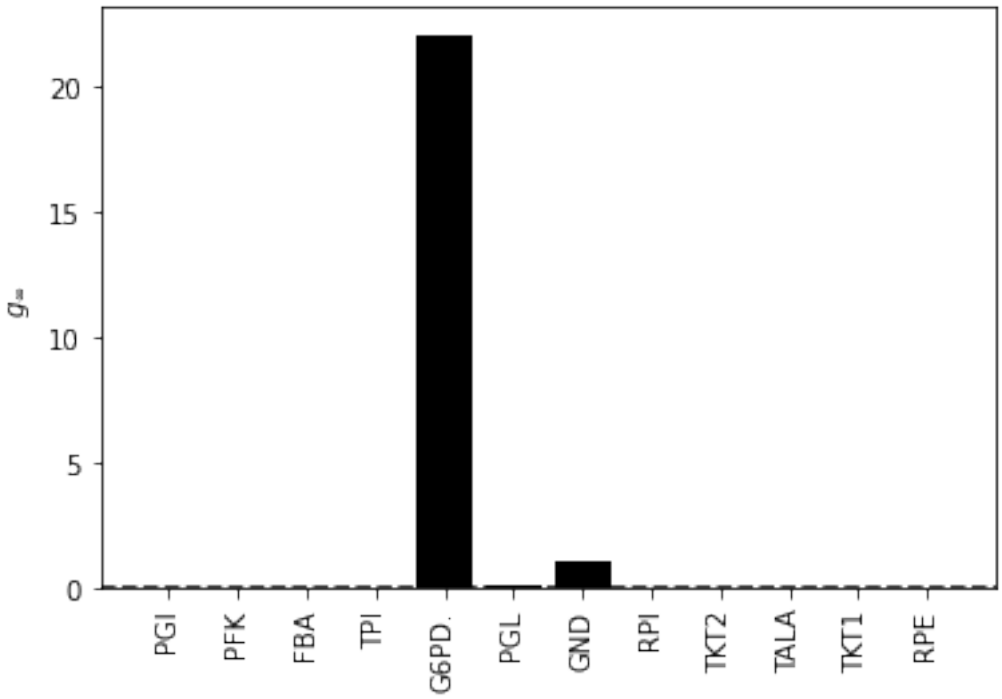


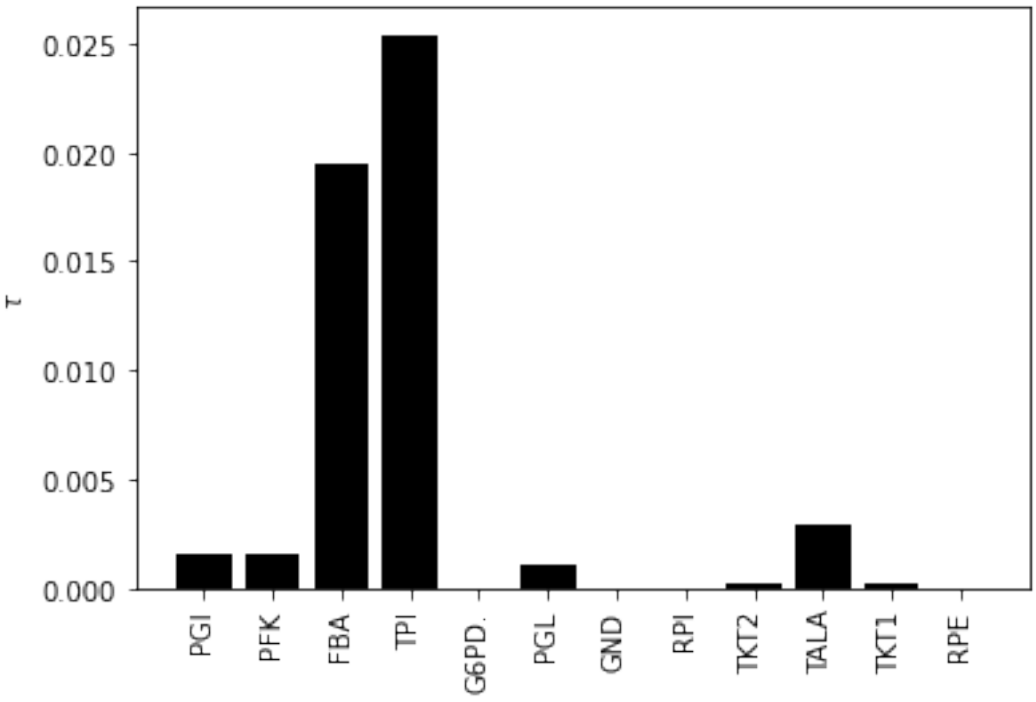

```
Doing: sTKT1
kappa_TKT1
g = 1.384773597330272e-06 tau = 0.00018475667332610188
dlam = 10
NADPH
```



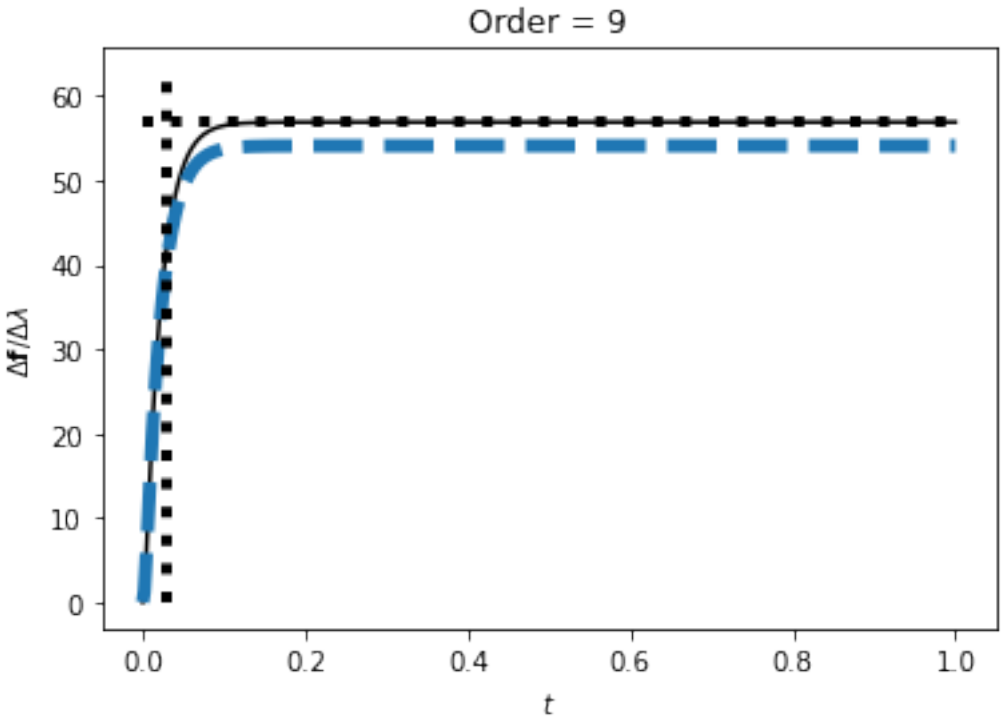
```
Doing: sRPE
kappa_RPE
g = 3.4576678613673054e-07 tau = 2.656594534459993e-06
dlam = 10
NADPH
```



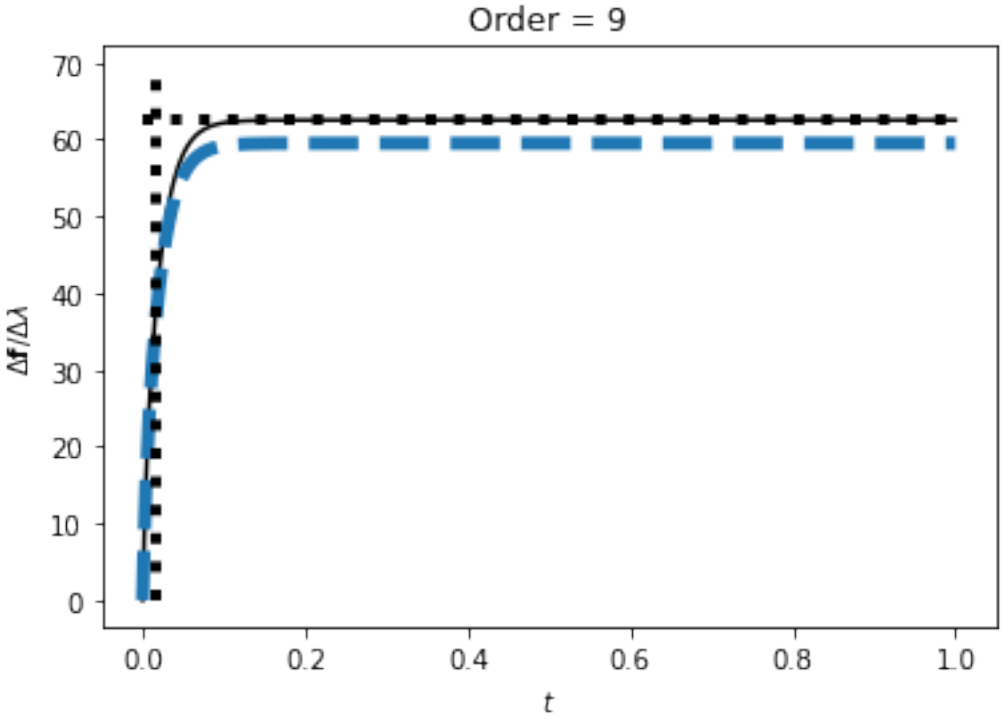




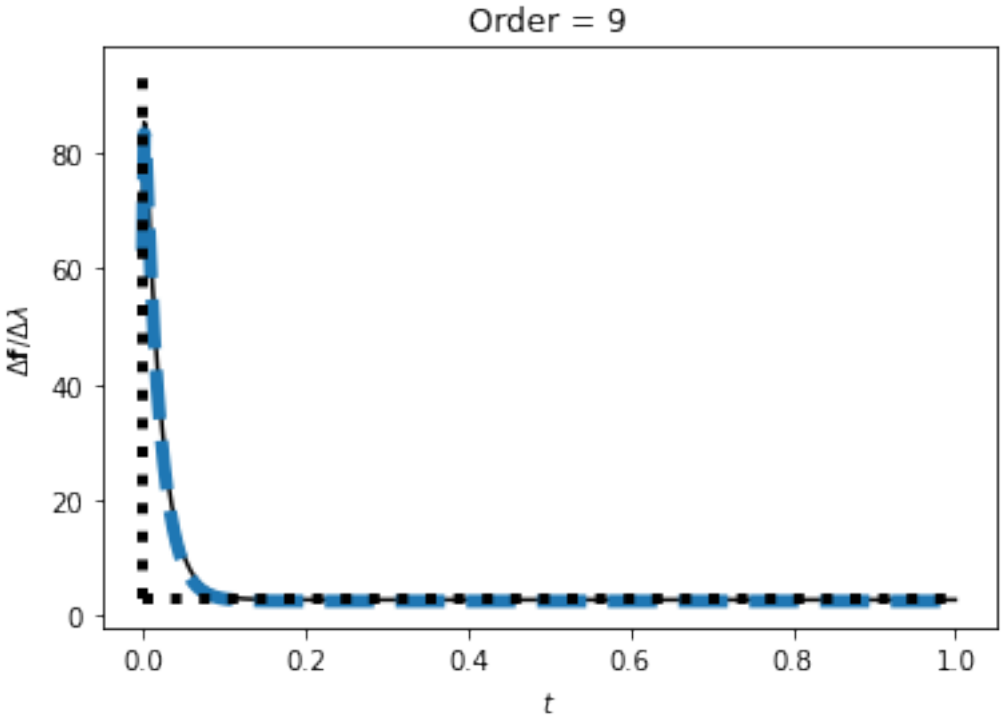

```
Doing: sPGI
kappa_PGI
g = 56.91833245076178 tau = 0.0300410381070839
dlam = 10
G3P
```



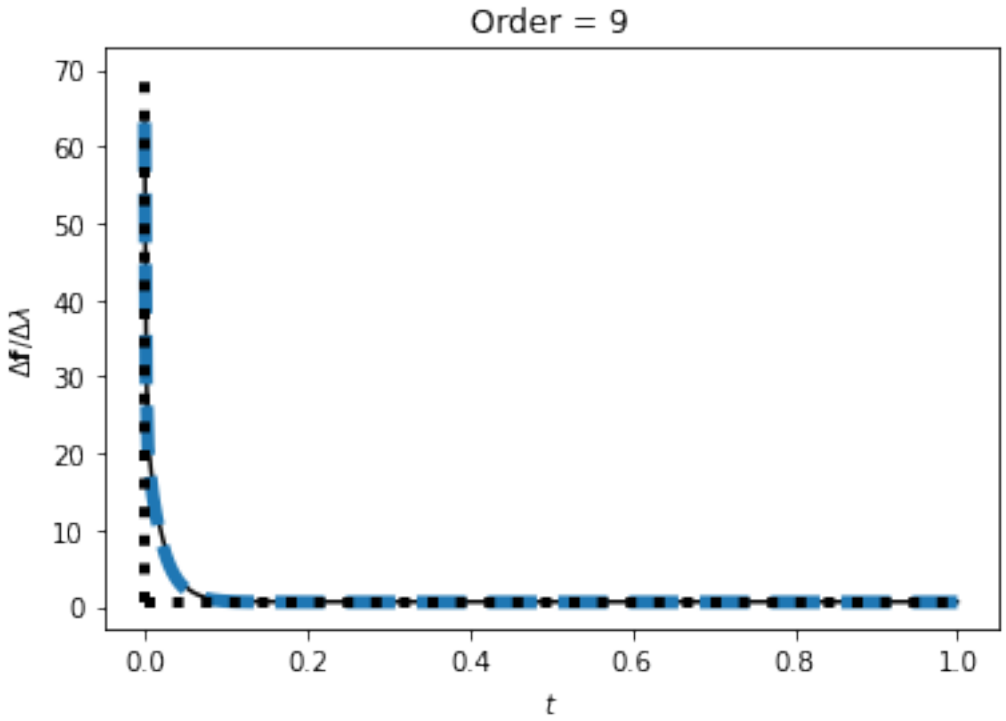

```
Doing: sPFK
kappa_PFK
g = 62.53329029952575 tau = 0.014866156765752288
dlam = 10
G3P
```



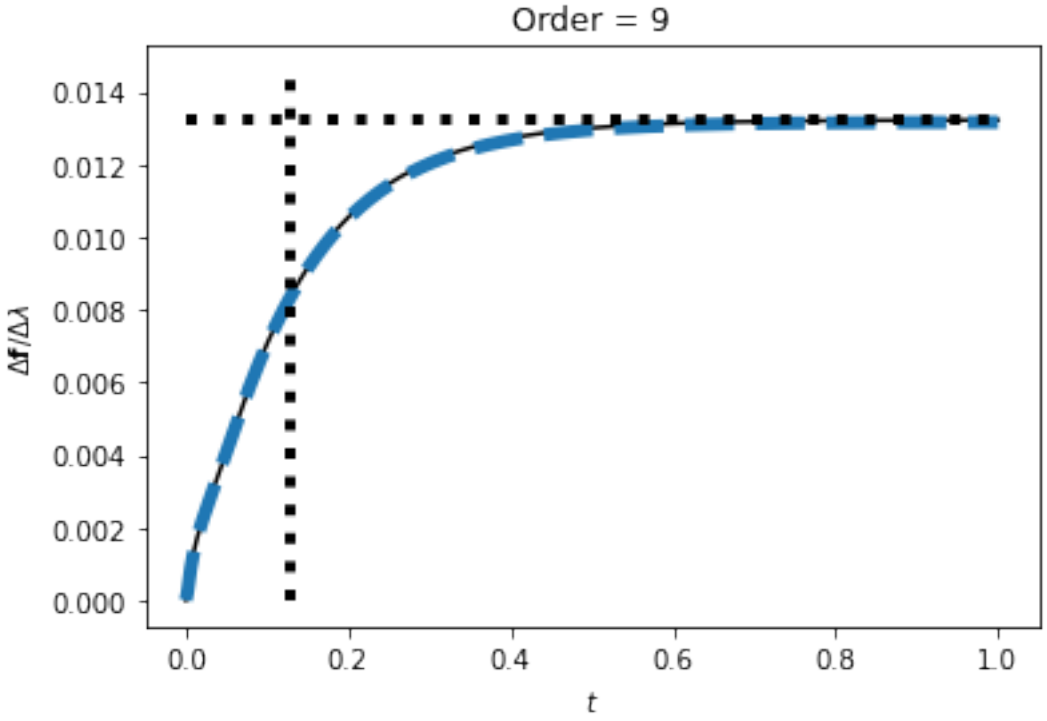
```
Doing: sFBA
kappa_FBA
g = 2.716230618146149 tau = 0
dlam = 10
G3P
```



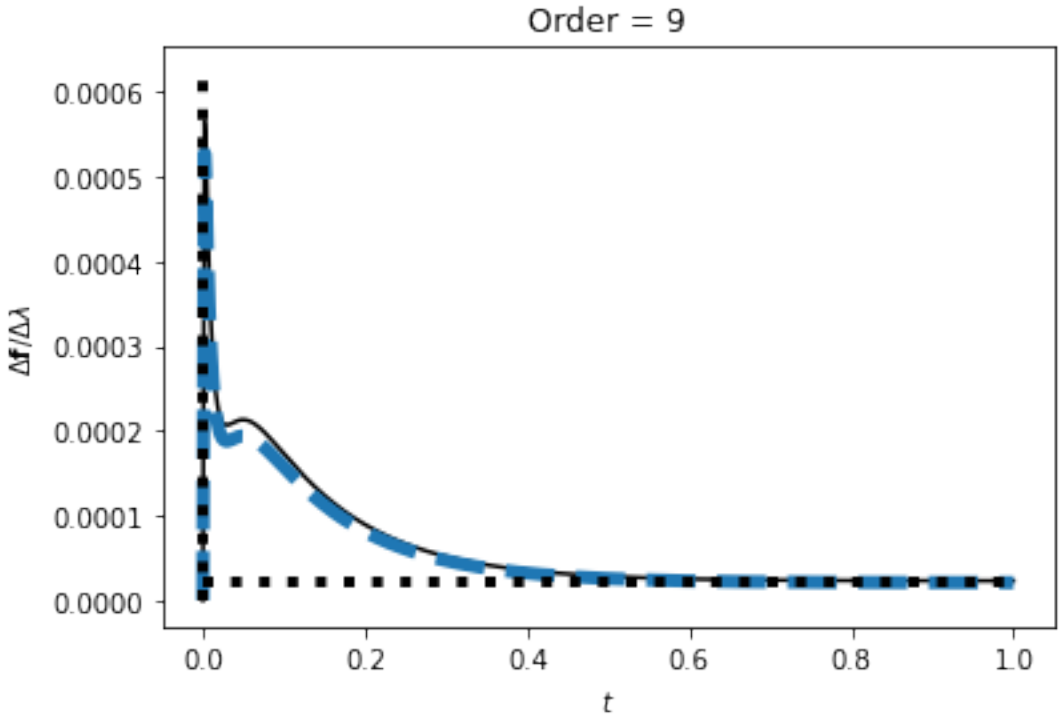

```
Doing: sTPI
kappa_TPI
g = 0.6295989394586812 tau = 0
dlam = 10
G3P
```



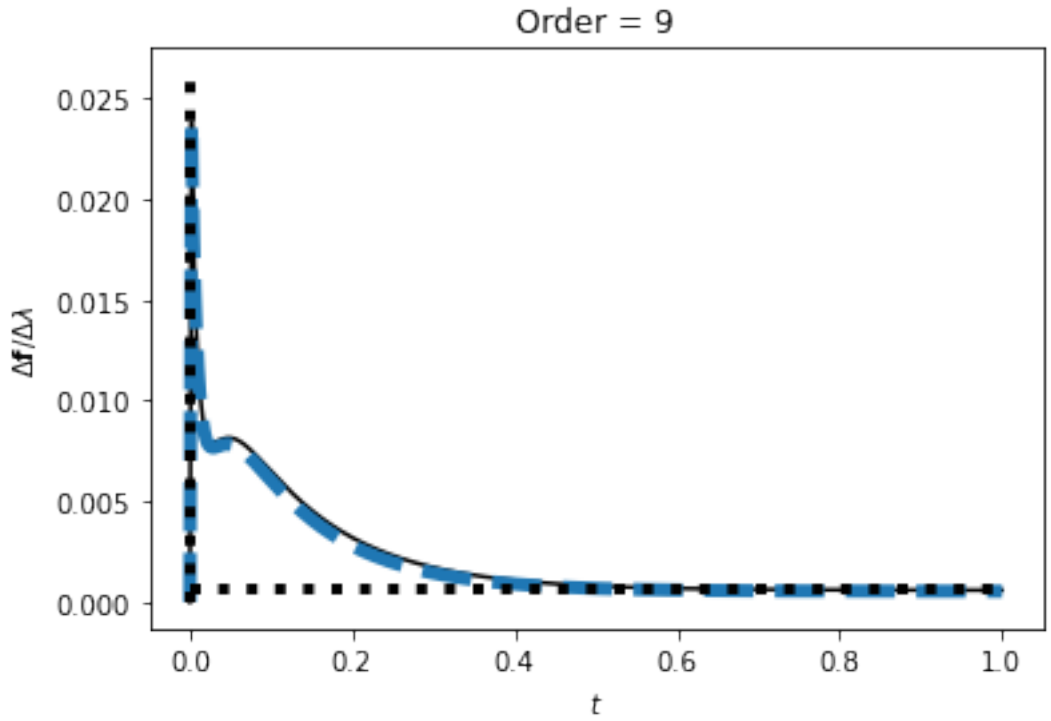

```
Doing: sG6PDH2R
kappa_G6PDH2R
g = 0.013223499146244906 tau = 0.1280086430294219
dlam = 10
G3P
```



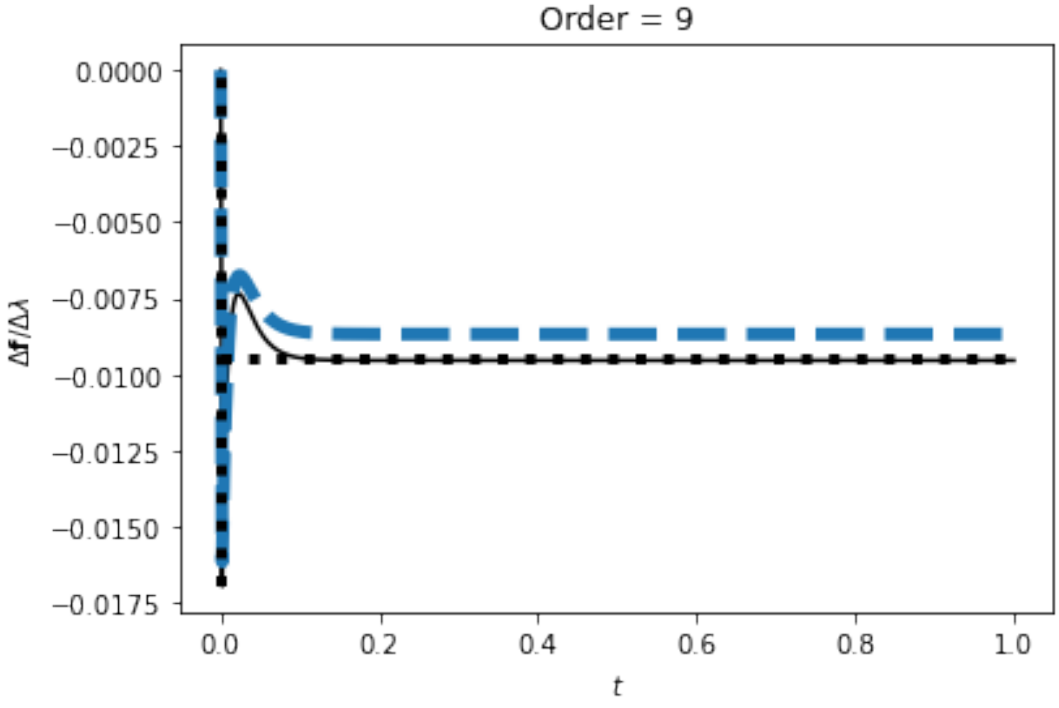
```
Doing: sPGL
kappa_PGL
g = 2.292202604095388e-05 tau = 0.0014045237898684167
dlam = 10
G3P
```



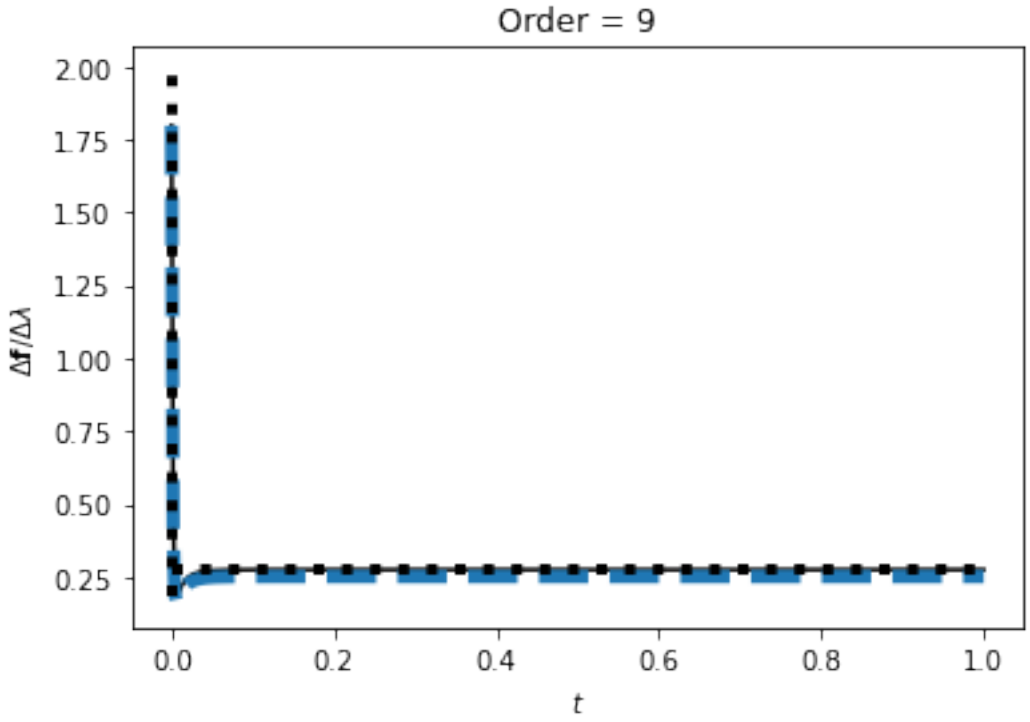

```
Doing: sGND
kappa_GND
g = 0.0005929588298785541 tau = 0.00017684665624203715
dlam = 10
G3P
```



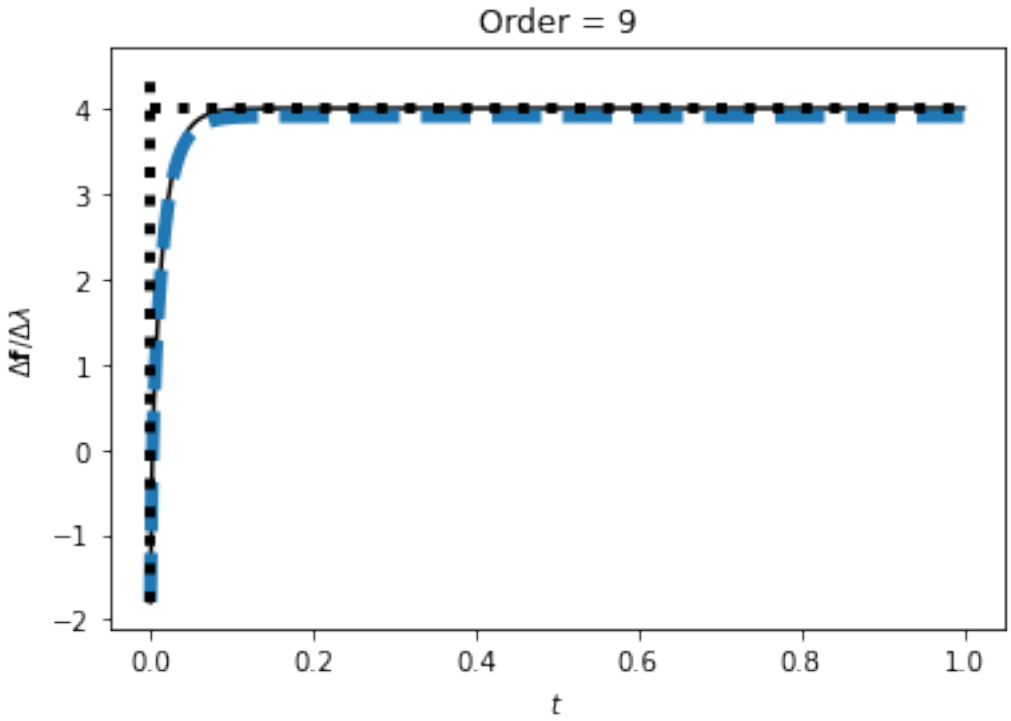

```
Doing: sRPI
kappa_RPI
g = -0.009544022155095408 tau = 0.00017769464683750914
dlam = 10
G3P
```



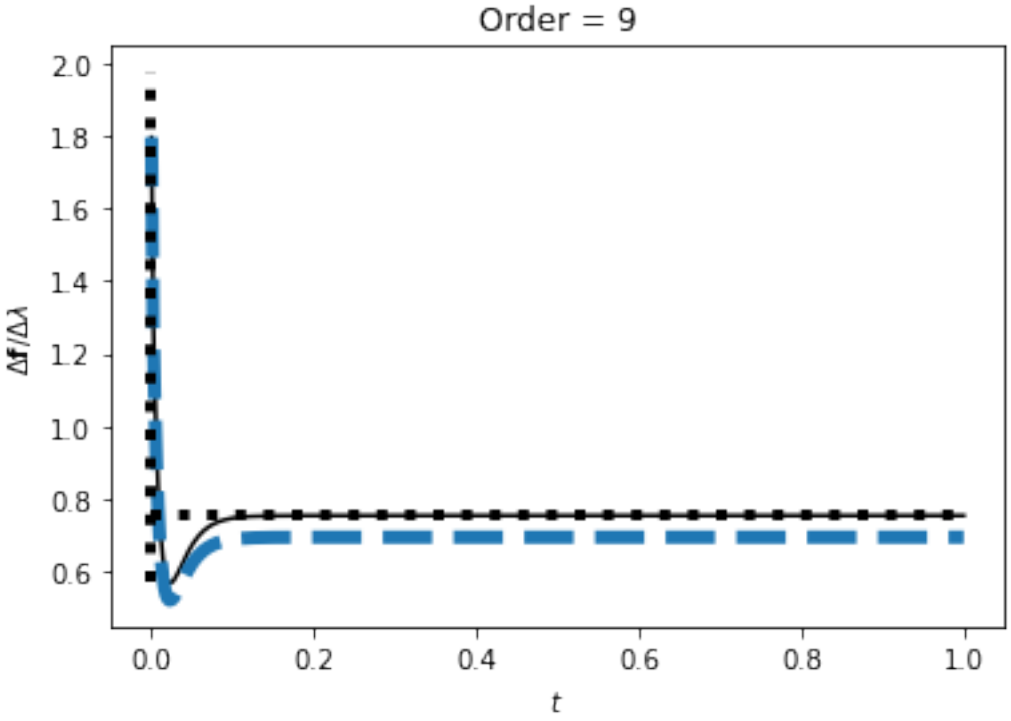
```
Doing: sTKT2
kappa_TKT2
g = 0.2759715745850928 tau = 0
dlam = 10
G3P
```



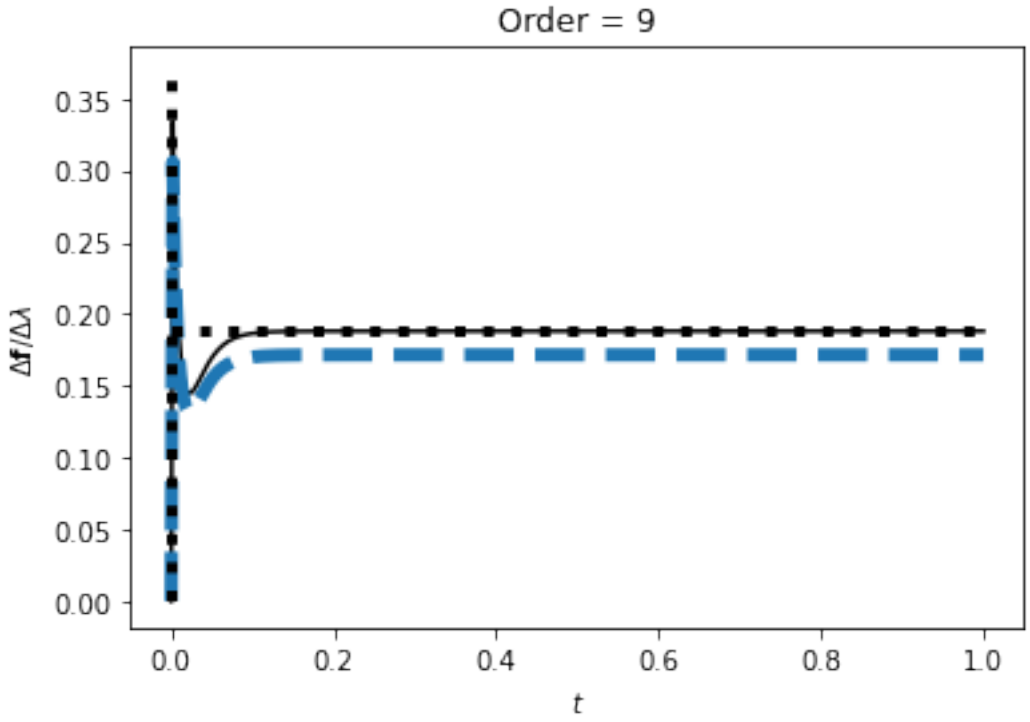

```
Doing: sTALA
kappa_TALA
g = 4.011429900742106 tau = 0
dlam = 10
G3P
```

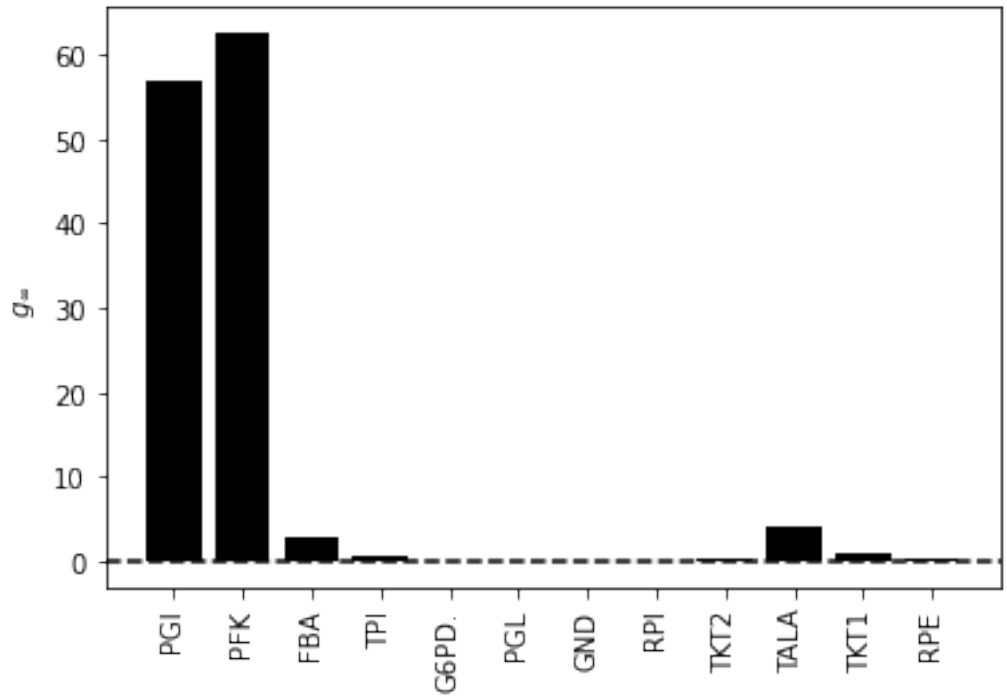


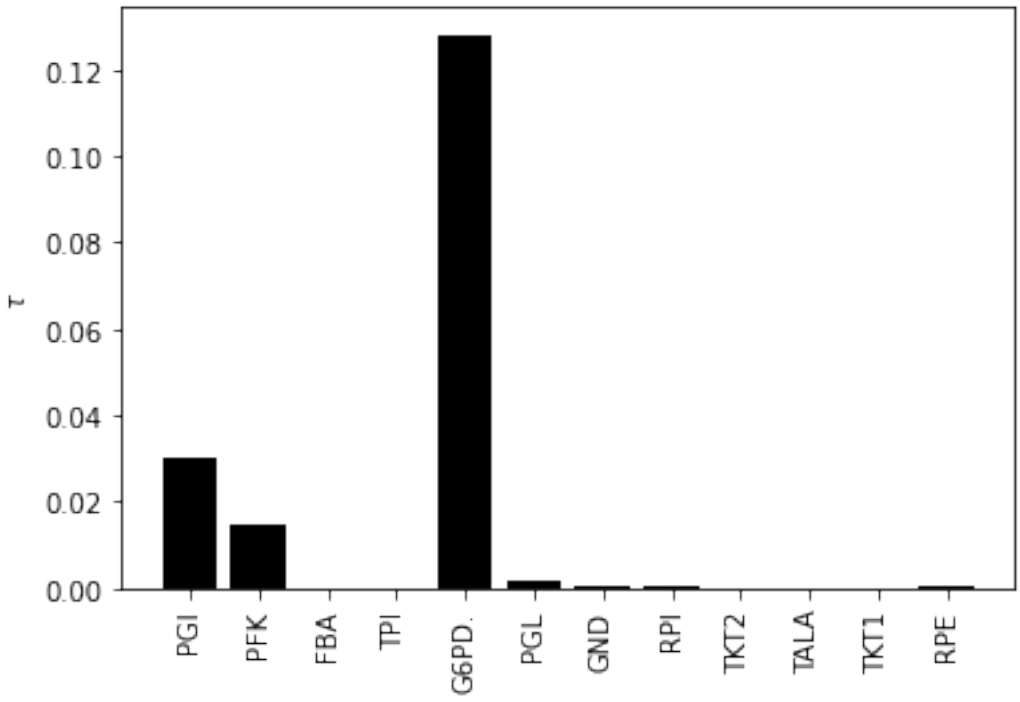

```
Doing: sTKT1
kappa_TKT1
g = 0.753312490189695 tau = 0
dlam = 10
G3P
```



```
Doing: sRPE
kappa_RPE
g = 0.18809604636010788 tau = 0.0001668257471471642
dlam = 10
G3P
```




60
50
40
30
20
10
0
g_∞
PGI
PFK
FBA
TPI
G6PD.
PGL
GND
RPI
TKT2
TALA
TKT1
RPE

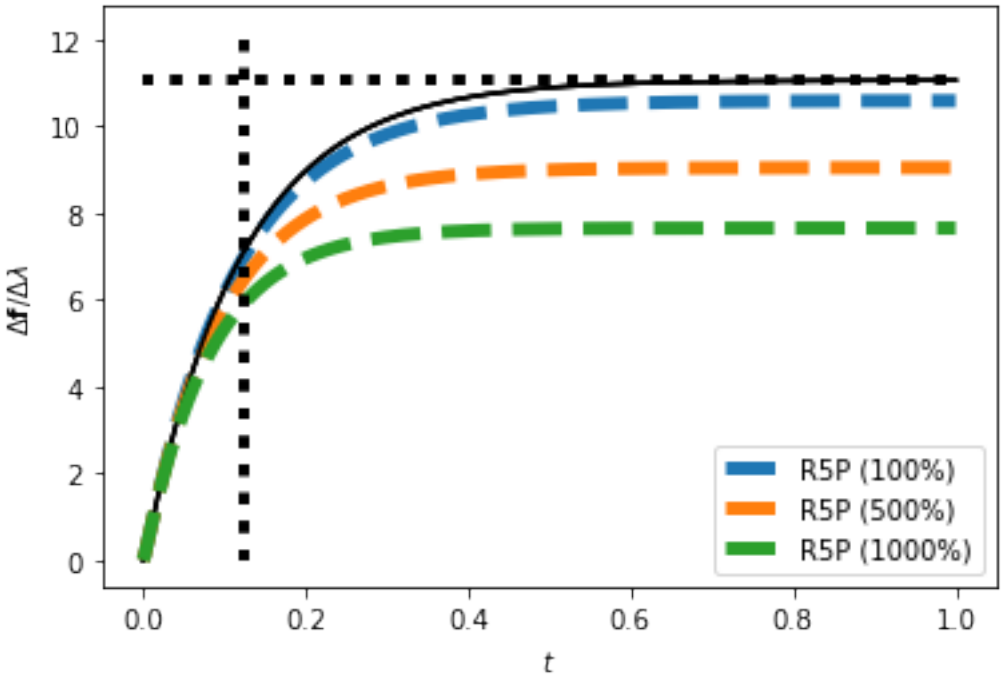

0.12
0.10
0.08
0.06
0.04
0.02
0.00
τ
PGI
PFK
FBA
TPI
G6PD.
PGL
GND
RPI
TKT2
TALA
TKT1
RPE

6.5 Vary Lambda

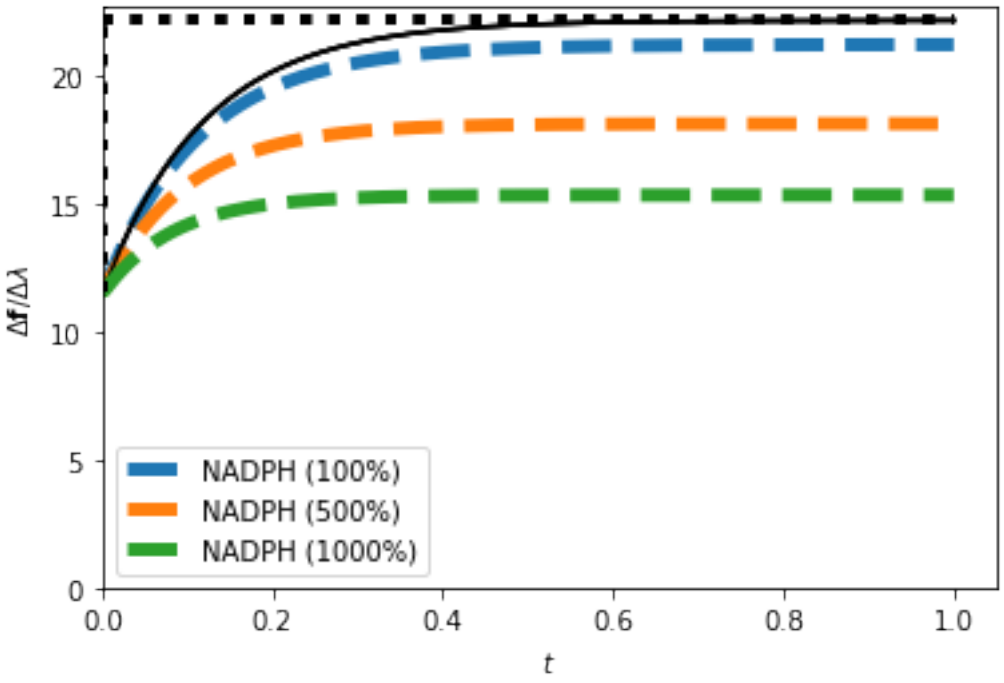
[34]:
```
## Vary Lambda

# inp = 'sG6PDH2R'
# outp = ['R5P']
Lin = {}
Nlin = {}
Lam = [1.1,2,6,11]
species = s['species']
for inp in ['sG6PDH2R','sG6P']:
    for outp in ['R5P','NADPH']:
        lin = []
        nlin = []
        for lam in Lam:
            dat,y_step,t,sys = simSensitivity(s,sc,sf,Sys,X_ss,V_ss,dX_ss,
 ↪parameter=parameter,inp=[inp],outp=[outp],lam=lam,t_last=t_last)
#             g = con.dcgain(sys)
            g,tau = tfProps(sys)
            g_hf = hfgain(sys)

            lin.append(g)
            i = species.index(outp)
            dX = dat['dX'][-1,i]
            ng = (dX-dX_ss[i])/(lam-1)
            nlin.append(ng)

            name = inp[1:]
            Name = f'{name}-{outp}'
            print(f'{Name} (DC gain: {g:.3f}, HF gain = {g_hf:.3f}')
            if Titles:
                plt.title(f'{Name} (DC gain: {g:.3f}, HF gain = {g_hf:.3f}')
            if lam>1.5:
                setZero = outp in ['NADPH']
                plotSensitivitydX(dat,species=[outp],setZero=setZero)
                plotLines()
        Lin[Name] = lin
        Nlin[Name] = nlin
        Savefig(f'PPPdX_{name}_{outp}_lambda')
        plt.show()
# print(Lin)
# print(Nlin)
```

```
kappa_G6PDH2R
G6PDH2R-R5P (DC gain: 11.049, HF gain = 0.000
kappa_G6PDH2R
G6PDH2R-R5P (DC gain: 11.049, HF gain = 0.000
R5P
kappa_G6PDH2R
G6PDH2R-R5P (DC gain: 11.049, HF gain = 0.000
```

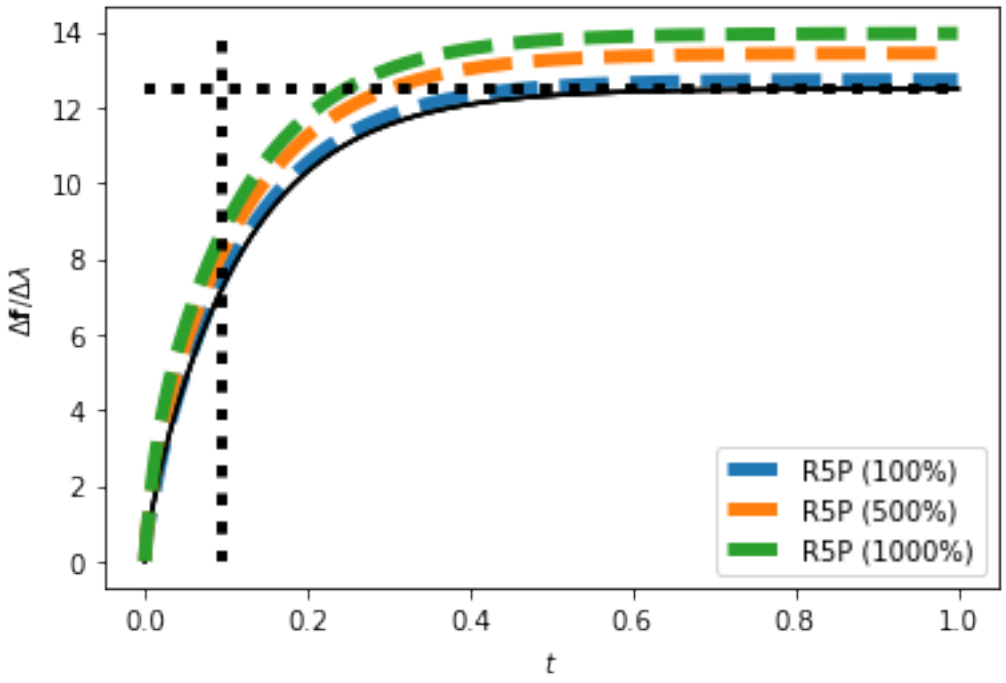
```
R5P
kappa_G6PDH2R
G6PDH2R-R5P (DC gain: 11.049, HF gain = 0.000
R5P
```



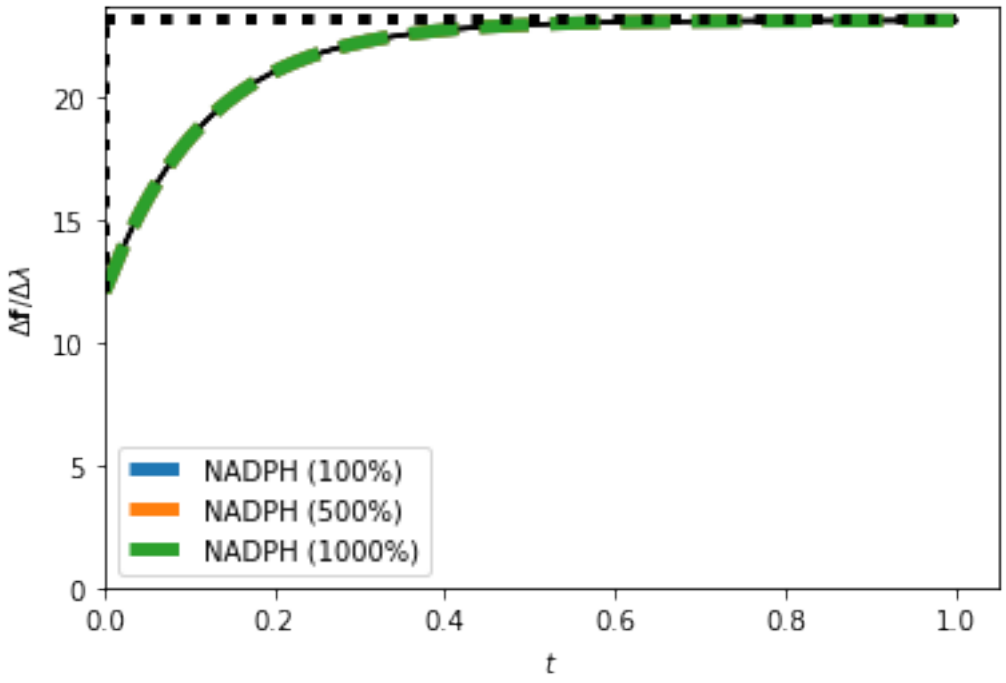

```
kappa_G6PDH2R
G6PDH2R-NADPH (DC gain: 22.126, HF gain = 11.578
kappa_G6PDH2R
G6PDH2R-NADPH (DC gain: 22.126, HF gain = 11.578
NADPH
kappa_G6PDH2R
G6PDH2R-NADPH (DC gain: 22.126, HF gain = 11.578
NADPH
kappa_G6PDH2R
G6PDH2R-NADPH (DC gain: 22.126, HF gain = 11.578
NADPH
```




```
K_G6P
G6P-R5P (DC gain: 12.486, HF gain = 0.000
K_G6P
G6P-R5P (DC gain: 12.486, HF gain = 0.000
R5P
K_G6P
G6P-R5P (DC gain: 12.486, HF gain = 0.000
R5P
K_G6P
G6P-R5P (DC gain: 12.486, HF gain = 0.000
R5P
```



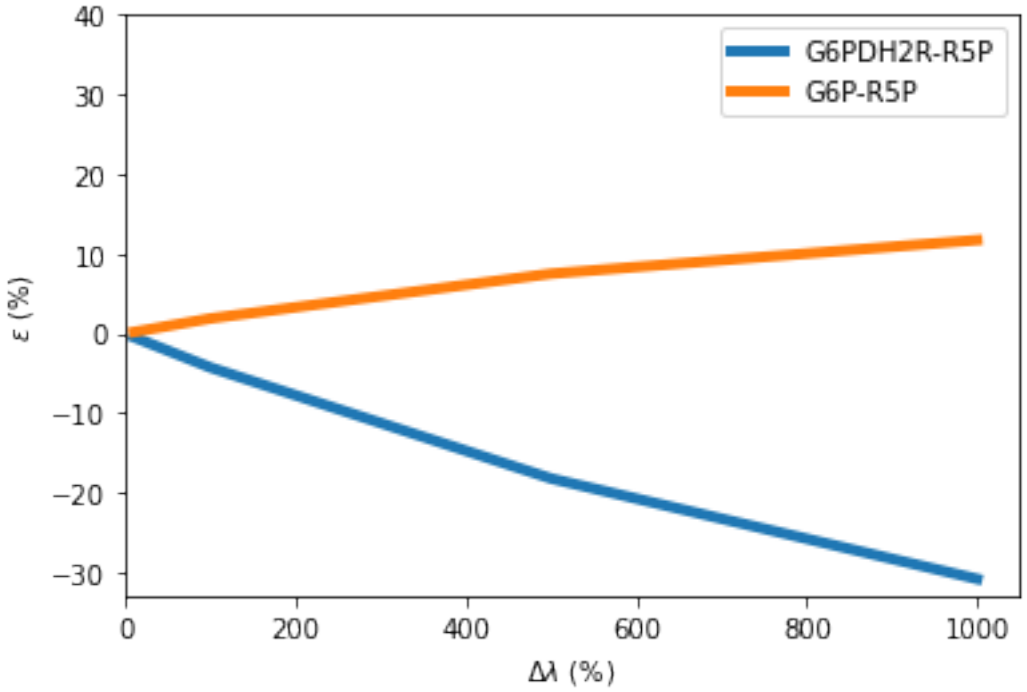

```
K_G6P
G6P-NADPH (DC gain: 23.170, HF gain = 12.125
K_G6P
G6P-NADPH (DC gain: 23.170, HF gain = 12.125
NADPH
K_G6P
G6P-NADPH (DC gain: 23.170, HF gain = 12.125
NADPH
K_G6P
G6P-NADPH (DC gain: 23.170, HF gain = 12.125
NADPH
```



[35]:
```
## Plot error curves
name = ['a','b','c','d']
lw = [10,4,4,4]
for i,Name in enumerate(Nlin):
    print(Name)
    G = np.array(Lin[Name])
    NG = np.array(Nlin[Name])
    Err = (NG-G)/G
    ErrPC = 100*Err
    LamPC = (Lam-np.ones(len(Lam)))*100

    ## Just plot 1 and 3
    if i in [0,2]:
#         plt.plot(LamPC,ErrPC,label=name[i],lw=lw[i])
        plt.plot(LamPC,ErrPC,label=Name,lw=4)

plt.legend()
# plt.grid()
plt.ylim(top=40)
plt.xlim(left=0)
plt.xlabel(r'$\Delta \lambda$ (%)')
# plt.ylabel(r'$\tilde{\dot{x}}/\tilde{\lambda}-g$ (%)')
plt.ylabel(r'$\epsilon$ (%)')
Savefig(f'PPPdX_lambda',fontsize=20)
plt.show()
```

```
G6PDH2R-R5P
G6PDH2R-NADPH
G6P-R5P
```

G6P-NADPH



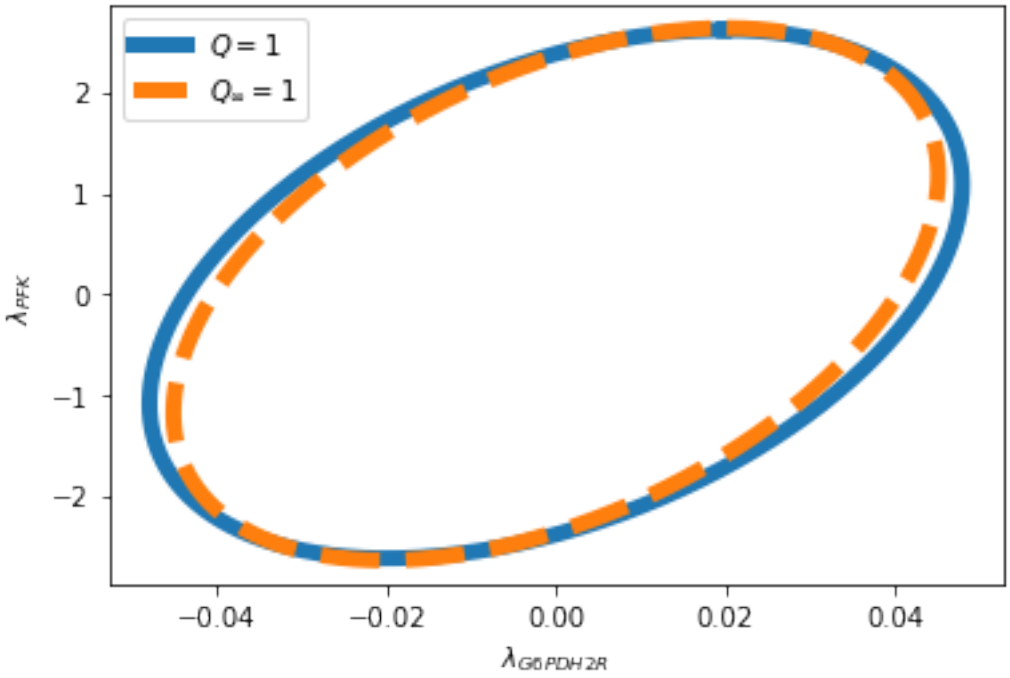
7 Sloppy parameter analysis

[36]:
```
imp.reload(slp)
def sloppy(Sys,inp,outp,t=None):

    sys = extractSubsystem(Sys,sc,sf,inp,outp)
    #print(sys)
    H,eig,eigv,t = slp.Sloppy(sys,t=t)
    cond = np.linalg.cond(H)
    print(f'Log10 condition number: {np.log10(cond):.1f}')
    slp.SloppyPrint(eig,eigv,inp)
    slp.SloppyPlot(eig,eigv,inp,square=False)
    #slp.SloppyPlotData(t,y,inp,outp)


def sloppyBoth(Sys,inp,outp,t=None):

    sys = extractSubsystem(Sys,sc,sf,inp,outp)
    #SysName = sc['name']
    blurb = '\n*******************************\n'
    print(blurb,SysName,blurb)
    if t is None:
        name = f'{SysName}_sloppy_{inp[0]}'
    else:
        name = f'{SysName}_sloppy_{inp[0]}_long'
```

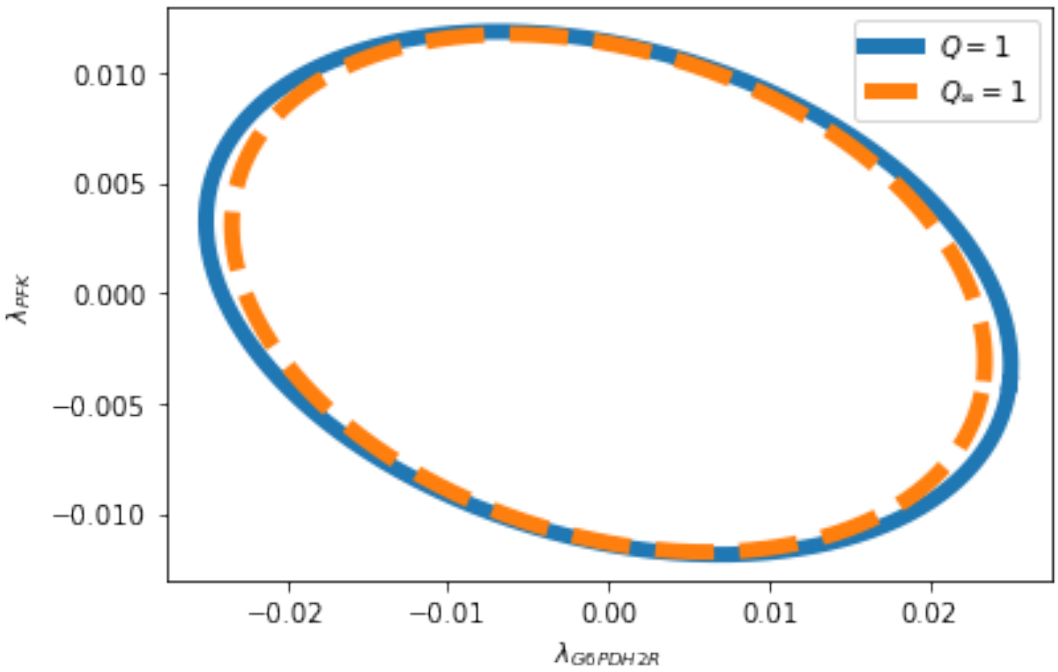
```
    #print(sys)
    H,eig,eigv,t = slp.Sloppy(sys,t=t,GainOnly=False)
    print('H:\n',H)
    slp.SloppyPrint(eig,eigv,inp,min_eig=0)
    H,Eig,Eigv,t = slp.Sloppy(sys,t=t,GainOnly=True)
    print('H_ss:\n',H)
    slp.SloppyPrint(Eig,Eigv,inp,min_eig=0)
    slp.SloppyPlot(eig,eigv,inp,Eig=Eig,Eigv=Eigv)

    Savefig(name)
    plt.show()


for t_last in [0,1e1]:
    if t_last==0:
        t = None
    else:
        t = np.linspace(0,t_last,100)

    SysName = 'GlyPPP_PFK_2'
    inp = ['sG6PDH2R','sPFK']
    outp = ['NADPH','R5P']
    sloppyBoth(Sys,inp,outp,t=t)

    SysName = 'GlyPPP_PFK_all'
    Outp = []
    for chemo in sc['chemostats']:
        if not chemo[0] in ['s']:
            Outp.append(chemo)
    print(Outp)
    sloppyBoth(Sys,inp,Outp,t=t)

    SysName = 'GlyPPP_PGI_all'
    inp = ['sG6PDH2R','sPGI']
    sloppyBoth(Sys,inp,Outp,t=t)
```

```
*******************************
 GlyPPP_PFK_2
*******************************

H:
 [[ 5.23558279e+02 -3.97898795e+00]
 [-3.97898795e+00  1.76090966e-01]]
\sqrt\sigma_1 &= 23 & V_1\Lambda &= + 1.000 \lambda_{G6PDH2R} - 0.008
\lambda_{PFK}
\sqrt\sigma_2 &= 0.38 & V_2\Lambda &= + 1.000 \lambda_{PFK} + 0.008
\lambda_{G6PDH2R}
H_ss:
 [[ 6.117e+02 -4.681e+00]
```

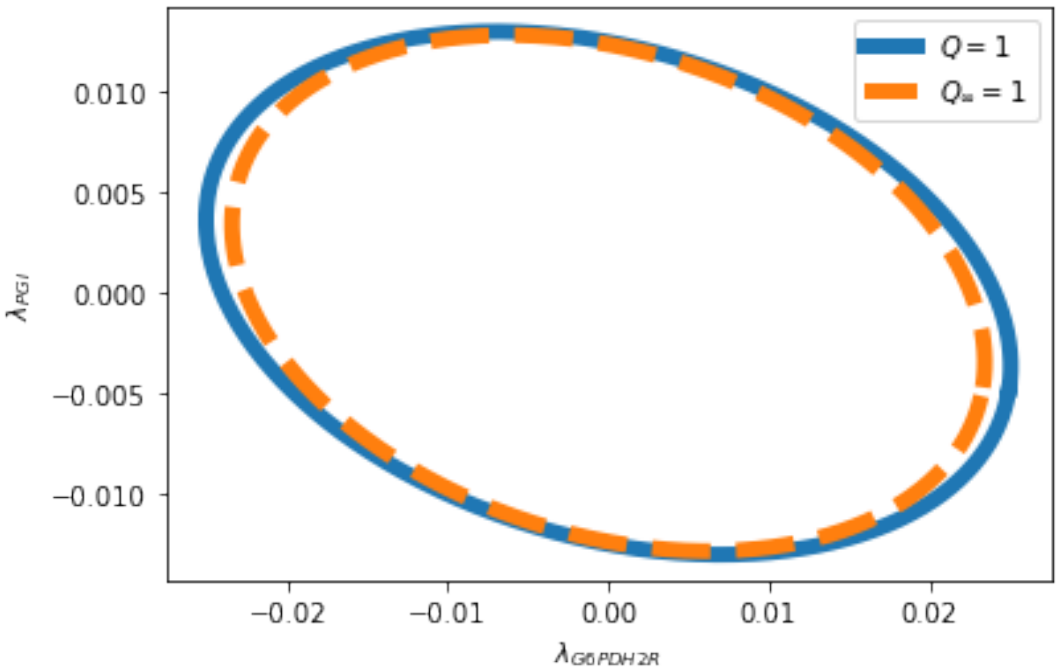
```
 [-4.681e+00  1.795e-01]]
\sqrt\sigma_1 &= 25 & V_1\Lambda &= + 1.000 \lambda_{G6PDH2R} - 0.008
\lambda_{PFK}
\sqrt\sigma_2 &= 0.38 & V_2\Lambda &= + 1.000 \lambda_{PFK} + 0.008
\lambda_{G6PDH2R}
```



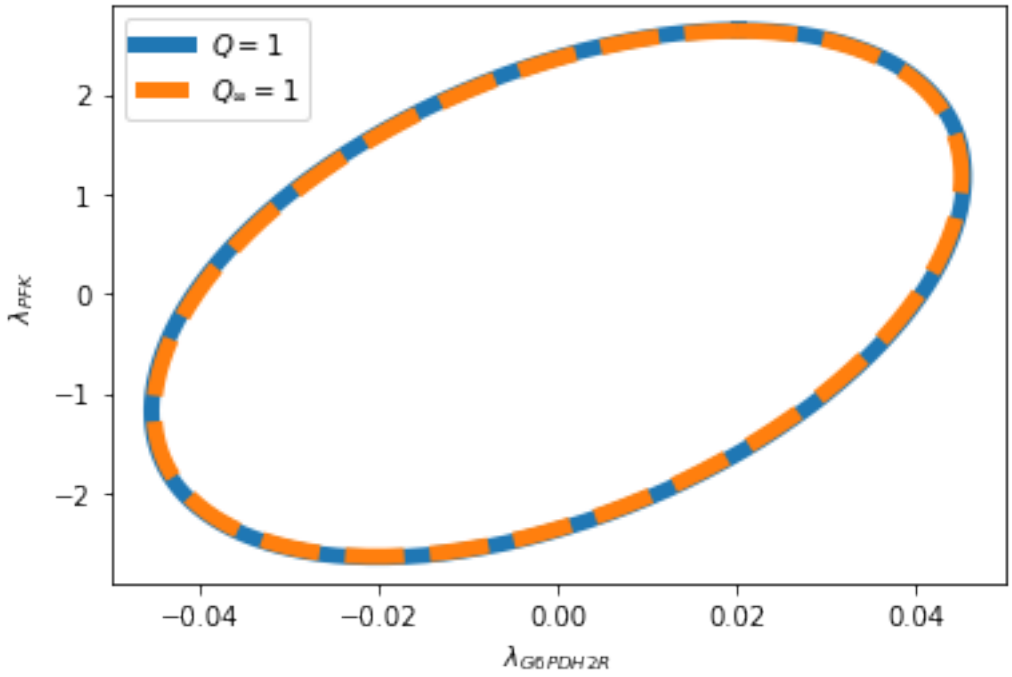

```
['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP', 'NADPH', 'R5P', 'G3P']

*******************************
 GlyPPP_PFK_all
*******************************

H:
 [[1726.16  1001.115]
 [1001.115 7677.688]]
\sqrt\sigma_1 &= 89 & V_1\Lambda &= + 0.987 \lambda_{PFK} + 0.162
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 40 & V_2\Lambda &= + 0.987 \lambda_{G6PDH2R} - 0.162
\lambda_{PFK}
H_ss:
 [[1958.04  1028.652]
 [1028.652 7785.822]]
\sqrt\sigma_1 &= 89 & V_1\Lambda &= + 0.986 \lambda_{PFK} + 0.169
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 42 & V_2\Lambda &= + 0.986 \lambda_{G6PDH2R} - 0.169
\lambda_{PFK}
```



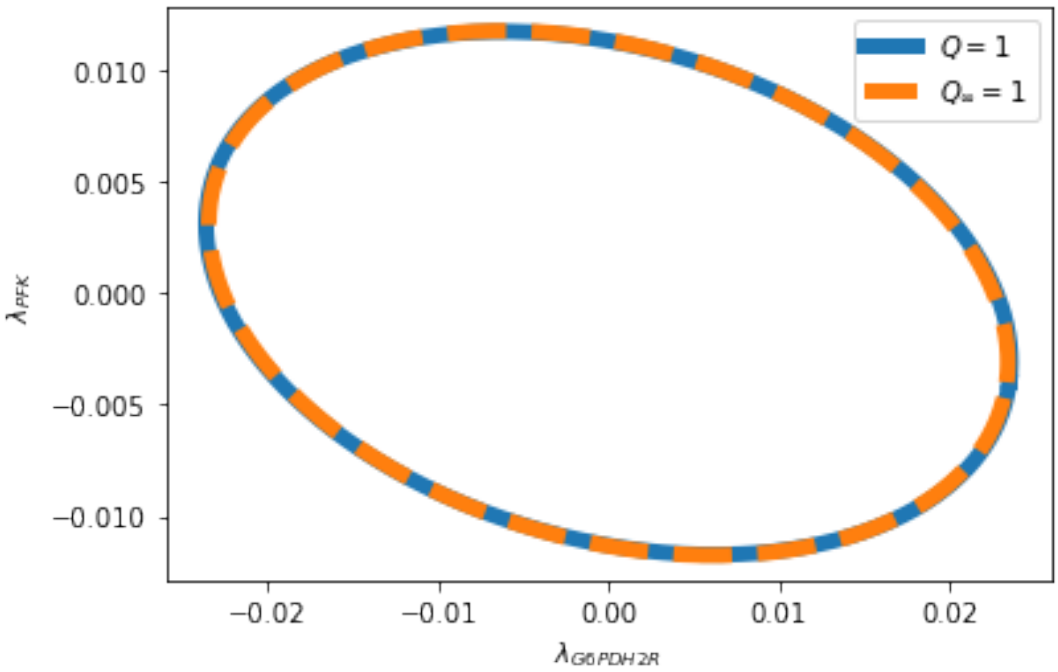

```
*******************************
 GlyPPP_PGI_all
*******************************

H:
 [[1726.16   928.876]
 [ 928.876 6383.276]]
\sqrt\sigma_1 &= 81 & V_1\Lambda &= + 0.982 \lambda_{PGI} + 0.189
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 39 & V_2\Lambda &= + 0.982 \lambda_{G6PDH2R} - 0.189
\lambda_{PGI}
H_ss:
 [[1958.04   955.607]
 [ 955.607 6511.651]]
\sqrt\sigma_1 &= 82 & V_1\Lambda &= + 0.980 \lambda_{PGI} + 0.197
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 42 & V_2\Lambda &= + 0.980 \lambda_{G6PDH2R} - 0.197
\lambda_{PGI}
```



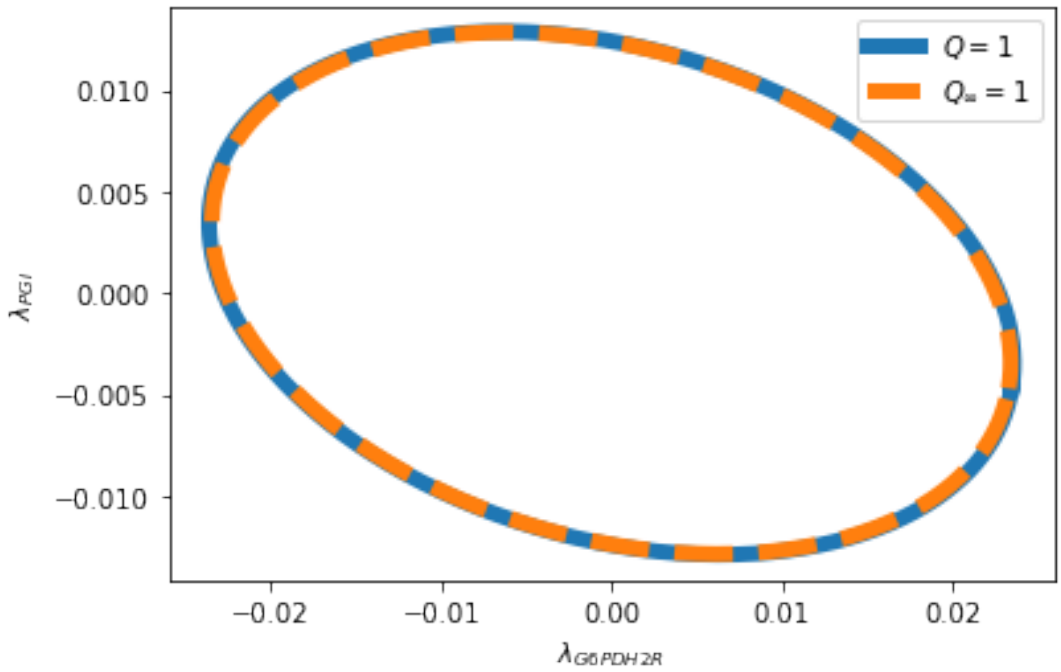

```
*******************************
 GlyPPP_PFK_2
*******************************

H:
 [[ 6.020e+02 -4.599e+00]
 [-4.599e+00  1.777e-01]]
\sqrt\sigma_1 &= 25 & V_1\Lambda &= + 1.000 \lambda_{G6PDH2R} - 0.008
\lambda_{PFK}
\sqrt\sigma_2 &= 0.38 & V_2\Lambda &= + 1.000 \lambda_{PFK} + 0.008
\lambda_{G6PDH2R}
H_ss:
 [[ 6.117e+02 -4.681e+00]
 [-4.681e+00  1.795e-01]]
\sqrt\sigma_1 &= 25 & V_1\Lambda &= + 1.000 \lambda_{G6PDH2R} - 0.008
\lambda_{PFK}
\sqrt\sigma_2 &= 0.38 & V_2\Lambda &= + 1.000 \lambda_{PFK} + 0.008
\lambda_{G6PDH2R}
```




```
['ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP', 'NADPH', 'R5P', 'G3P']

*******************************
 GlyPPP_PFK_all
*******************************

H:
 [[1930.918 1024.217]
 [1024.217 7827.073]]
\sqrt\sigma_1 &= 89 & V_1\Lambda &= + 0.986 \lambda_{PFK} + 0.166
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 42 & V_2\Lambda &= + 0.986 \lambda_{G6PDH2R} - 0.166
\lambda_{PFK}
H_ss:
 [[1958.04  1028.652]
 [1028.652 7785.822]]
\sqrt\sigma_1 &= 89 & V_1\Lambda &= + 0.986 \lambda_{PFK} + 0.169
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 42 & V_2\Lambda &= + 0.986 \lambda_{G6PDH2R} - 0.169
\lambda_{PFK}
```




```
*******************************
 GlyPPP_PGI_all
*******************************

H:
 [[1930.918  951.546]
 [ 951.546 6481.527]]
\sqrt\sigma_1 &= 82 & V_1\Lambda &= + 0.980 \lambda_{PGI} + 0.197
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 42 & V_2\Lambda &= + 0.980 \lambda_{G6PDH2R} - 0.197
\lambda_{PGI}
H_ss:
 [[1958.04   955.607]
 [ 955.607 6511.651]]
\sqrt\sigma_1 &= 82 & V_1\Lambda &= + 0.980 \lambda_{PGI} + 0.197
\lambda_{G6PDH2R}
\sqrt\sigma_2 &= 42 & V_2\Lambda &= + 0.980 \lambda_{G6PDH2R} - 0.197
\lambda_{PGI}
```



[37]:
```
imp.reload(slp)
Outp = ['R5P','NADPH','G3P']
# Outp = ['R5P']
def PrintSloppy(Inp,Outp,GainOnly=True):
    blurb = '\n*********\n'
    for outp in Outp:
        print(blurb,outp,blurb)
        sys = extractSubsystem(Sys,sc,sf,Inp,[outp])
#         print(con.dcgain(sys))
        gain = con.dcgain(sys)[0]
        norm = np.sum(gain*gain)
        ngain = gain/np.sqrt(norm)
#         print(norm,gain/np.sqrt(norm))
        H,eig,eigv,t = slp.Sloppy(sys,GainOnly=GainOnly,small=1e-10)
        slp.SloppyPrint(eig,eigv,Inp,min_eig=0.01,min_eigv=0.1,max_eigs=5)
        if GainOnly:
            print('Direct')
            slp.SloppyPrint([norm],np.array([ngain]).T,Inp,min_eig=0.
 ↪01,min_eigv=0.1,max_eigs=5)
#         print(eigv[:,0])
#         print(ngain)

## Reactions
for GainOnly in [True,False]:
    print('\nGainOnly =',GainOnly)
    PrintSloppy(Inp_reac,Outp,GainOnly=GainOnly)
```

```
GainOnly = True

*********
 R5P
*********

\sqrt\sigma_1 &= 12 & V_1\Lambda &= + 0.933 \lambda_{G6PDH2R} - 0.347
\lambda_{TALA}
Direct
\sqrt\sigma_1 &= 12 & V_1\Lambda &= + 0.933 \lambda_{G6PDH2R} - 0.347
\lambda_{TALA}

*********
 NADPH
*********

\sqrt\sigma_1 &= 22 & V_1\Lambda &= + 0.999 \lambda_{G6PDH2R}
Direct
\sqrt\sigma_1 &= 22 & V_1\Lambda &= + 0.999 \lambda_{G6PDH2R}

*********
 G3P
*********

\sqrt\sigma_1 &= 85 & V_1\Lambda &= + 0.738 \lambda_{PFK} + 0.672 \lambda_{PGI}
Direct
\sqrt\sigma_1 &= 85 & V_1\Lambda &= + 0.738 \lambda_{PFK} + 0.672 \lambda_{PGI}

GainOnly = False

*********
 R5P
*********

\sqrt\sigma_1 &= 11 & V_1\Lambda &= + 0.910 \lambda_{G6PDH2R} - 0.376
\lambda_{TALA} + 0.147 \lambda_{GND}
\sqrt\sigma_2 &= 3 & V_2\Lambda &= + 0.940 \lambda_{GND} - 0.250
\lambda_{G6PDH2R} - 0.214 \lambda_{TALA}

*********
 NADPH
*********

\sqrt\sigma_1 &= 21 & V_1\Lambda &= + 0.994 \lambda_{G6PDH2R} + 0.109
\lambda_{GND}
\sqrt\sigma_2 &= 2.7 & V_2\Lambda &= + 0.994 \lambda_{GND} - 0.109
\lambda_{G6PDH2R}

*********
 G3P
*********
```

```
\sqrt\sigma_1 &= 76 & V_1\Lambda &= + 0.738 \lambda_{PFK} + 0.653 \lambda_{PGI}
+ 0.161 \lambda_{FBA}
\sqrt\sigma_2 &= 26 & V_2\Lambda &= + 0.938 \lambda_{FBA} + 0.293 \lambda_{TPI}
- 0.173 \lambda_{PGI}
```

[38]:
```
## Chemostats
for GainOnly in [True,False]:
    print('\nGainOnly =',GainOnly)
    PrintSloppy(Inp_chemo,Outp,GainOnly=GainOnly)
```

```
GainOnly = True

*********
 R5P
*********

\sqrt\sigma_1 &= 21 & V_1\Lambda &= + 0.594 \lambda_{G6P} + 0.574 \lambda_{NADP}
- 0.562 \lambda_{R5P}
Direct
\sqrt\sigma_1 &= 21 & V_1\Lambda &= + 0.594 \lambda_{G6P} + 0.574 \lambda_{NADP}
- 0.562 \lambda_{R5P}

*********
 NADPH
*********

\sqrt\sigma_1 &= 34 & V_1\Lambda &= + 0.722 \lambda_{NADP} + 0.692 \lambda_{G6P}
Direct
\sqrt\sigma_1 &= 34 & V_1\Lambda &= + 0.722 \lambda_{NADP} + 0.692 \lambda_{G6P}

*********
 G3P
*********

\sqrt\sigma_1 &= 1.4e+02 & V_1\Lambda &= + 0.873 \lambda_{G6P} + 0.478
\lambda_{ATP}
Direct
\sqrt\sigma_1 &= 1.4e+02 & V_1\Lambda &= + 0.873 \lambda_{G6P} + 0.478
\lambda_{ATP}

GainOnly = False

*********
 R5P
*********

\sqrt\sigma_1 &= 1.5e+03 & V_1\Lambda &= + 1.000 \lambda_{R5P}

*********
```

```
 NADPH
*********

\sqrt\sigma_1 &= 32 & V_1\Lambda &= + 0.743 \lambda_{NADP} + 0.668 \lambda_{G6P}

*********
 G3P
*********

\sqrt\sigma_1 &= 1.3e+02 & V_1\Lambda &= + 0.857 \lambda_{G6P} + 0.482
\lambda_{ATP} - 0.158 \lambda_{G3P}
\sqrt\sigma_2 &= 48 & V_2\Lambda &= + 0.982 \lambda_{G3P} + 0.173 \lambda_{G6P}
```

[39]:
```
## Three outputs
Outp = ['R5P','NADPH','G3P']
sys = extractSubsystem(Sys,sc,sf,Inp,Outp)
for GainOnly in [True,False]:
    print('\nGainOnly =',GainOnly)
    H,eig,eigv,t = slp.Sloppy(sys,GainOnly=GainOnly,small=1e-10)
    slp.SloppyPrint(eig,eigv,Inp,min_eig=1e-3,min_eigv=0.1,max_eigs=4)
```

```
GainOnly = True
\sqrt\sigma_1 &= 85 & V_1\Lambda &= + 0.738 \lambda_{PFK} + 0.672 \lambda_{PGI}
\sqrt\sigma_2 &= 25 & V_2\Lambda &= + 0.996 \lambda_{G6PDH2R}
\sqrt\sigma_3 &= 3.8 & V_3\Lambda &= + 0.966 \lambda_{TALA} + 0.181
\lambda_{TKT1} - 0.136 \lambda_{PGI}

GainOnly = False
\sqrt\sigma_1 &= 83 & V_1\Lambda &= + 0.739 \lambda_{PFK} + 0.670 \lambda_{PGI}
\sqrt\sigma_2 &= 23 & V_2\Lambda &= + 0.990 \lambda_{G6PDH2R} + 0.115
\lambda_{GND}
\sqrt\sigma_3 &= 11 & V_3\Lambda &= + 0.951 \lambda_{FBA} + 0.292 \lambda_{TPI}
\sqrt\sigma_4 &= 4.3 & V_4\Lambda &= + 0.731 \lambda_{GND} - 0.643
\lambda_{TALA} - 0.138 \lambda_{G6PDH2R} - 0.138 \lambda_{TKT1}
```

[40]:
```
## All outputs
for chemo in sc['chemostats']:
    if not chemo[0] in ['s']:
        Outp.append(chemo)
print(Outp)
sys = extractSubsystem(Sys,sc,sf,Inp,Outp)

for GainOnly in [True,False]:
    print('\nGainOnly =',GainOnly)
    H,eig,eigv,t = slp.Sloppy(sys,GainOnly=GainOnly,small=1e-10)
    slp.SloppyPrint(eig,eigv,Inp,min_eig=0.001,min_eigv=0.1,max_eigs=4)
```

```
['R5P', 'NADPH', 'G3P', 'ADP', 'ATP', 'CO2', 'G6P', 'H', 'H2O', 'NADP', 'NADPH',
'R5P', 'G3P']

GainOnly = True
```

```
\sqrt\sigma_1 &= 1.5e+02 & V_1\Lambda &= + 0.736 \lambda_{PFK} + 0.672
\lambda_{PGI}
\sqrt\sigma_2 &= 50 & V_2\Lambda &= + 0.995 \lambda_{G6PDH2R}
\sqrt\sigma_3 &= 6.3 & V_3\Lambda &= + 0.968 \lambda_{TALA} + 0.182
\lambda_{TKT1} - 0.131 \lambda_{PGI}

GainOnly = False
\sqrt\sigma_1 &= 1.5e+02 & V_1\Lambda &= + 0.736 \lambda_{PFK} + 0.670
\lambda_{PGI}
\sqrt\sigma_2 &= 47 & V_2\Lambda &= + 0.990 \lambda_{G6PDH2R} + 0.107
\lambda_{GND}
\sqrt\sigma_3 &= 16 & V_3\Lambda &= + 0.951 \lambda_{FBA} + 0.291 \lambda_{TPI}
\sqrt\sigma_4 &= 7.4 & V_4\Lambda &= + 0.924 \lambda_{GND} - 0.349
\lambda_{TALA} - 0.120 \lambda_{G6PDH2R}
```

[ ]: