kpp_vs_kppa

Unknown Author

April 10, 2015

In [26]:
```
kpp_file_1 = '0-dim_box_model_Exa2Green/kpp-2.2.3.dat'
kpp_file_2 = '0-dim_box_model_Exa2Green/box_model.dat_CUDA'
```

In [27]:
```
%matplotlib inline
import re
from itertools import cycle
from pylab import *
from matplotlib.markers import MarkerStyle
import matplotlib.pyplot as plt

ATOL = 1.0e-2
RTOL = 1.0e-2
EPS = 2.2204460492503131E-016
REGEX = re.compile('^([+\-]?)([0-9.]+)e?([+\-])([0-9.]+)$')
def convert(s):
    """
    Converts a number in Fortran E24.16 format to a Python float
    """
    m = re.search(REGEX, s)
    if m:
        s = ''.join([m.group(1), m.group(2), 'e', m.group(3), m.group(4)])
    try:
        fval = float(s)
    except ValueError:
        print '================> %s' % s
        fval = 0.0
    if fval < EPS:
        return 0.0
    else:
        return fval

def read_datfile(fname, tstart, cstart):
    """
    Read data from fname beginning on line tstart with concentration data beginning in
    Returns a tuple: (time, concentrations)
    Time data:
    [t0 t1 ... tN]
    Concentration data:
    [ [SPC_0(t0) SPC_1(t0) ... SPC_N(t0)]
    [SPC_0(t1) SPC_1(t1) ... SPC_N(t1)]
    : : :
    [SPC_0(tN) SPC_1(tN) ... SPC_N(tN)] ]
    """
    t = []
    c = []
    with open(fname, 'r') as f:
        while tstart:
            f.readline()
            tstart -= 1
        for line in f:
            parts = line.split()
```

```
            t.append(convert(parts[0]))
            c.append([convert(x) for x in parts[cstart:]])
    return t, c

def plot_dat(data, xlabel='Time', ylabel='Conc', names=None, titles=None):
    """
    Draw a plot of data read from read_datfile
    """
    lines = ['-', '--', '-.', ':']
    markers = MarkerStyle.filled_markers
    linecycler = cycle(lines)
    markercycler = cycle(markers)
    datastyles = ['%s%s' % (linecycler.next(), markercycler.next()) for _ in data]
    ndat = len(data)
    nspec = len(data[0][1][0])
    x = data[0][0]
    for i in xrange(0, nspec):
        fig, ax = plt.subplots()
        for j, dat in enumerate(data):
            t, c = dat
            y = [ct[i] for ct in c]
            style = datastyles[j]
            if names:
                label = '%s' % names[j]
            else:
                label = '%d' % j
            ax.plot(x, y, style, label=label)
        if ndat > 1:
            ax.legend(loc=2)
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)
        if titles:
            ax.set_title(titles[i])
        else:
            ax.set_title('Species %d' % i)
        show()

def scaled_err(x, y):
    if x or y:
        return abs(x-y)/max(x, y)
    elif x == y:
        return 0.0
    else:
        return float('inf')

def calc_err(d0, d1):
    c0 = d0[1]
    c1 = d1[1]
    err = []
    nsteps = len(c0)
    nspec = len(c0[0])
    sigPow = 0.0
    errPow = 0.0
    errCount = 0.0
    for i in xrange(0, nsteps):
        e = []
        for j in xrange(0, nspec):
            x = c0[i][j]
            y = c1[i][j]
            sigPow += x*x
            errPow += (x-y)*(x-y)
            serr = scaled_err(x,y)
            if serr > RTOL:
                print '%g > %g: %g, %g' % (serr, RTOL, x, y)
                errCount += 1
            e.append(serr)
```

```
            err.append(e)
        if errPow > 0:
            snr = 20 * log10(sigPow / errPow)
        else:
            snr = float('inf')
        print 'SNR: %fdb' % snr
        if errCount:
            print '%d samples with relative error > %g' % (errCount, RTOL)
        return d1[0], err
```

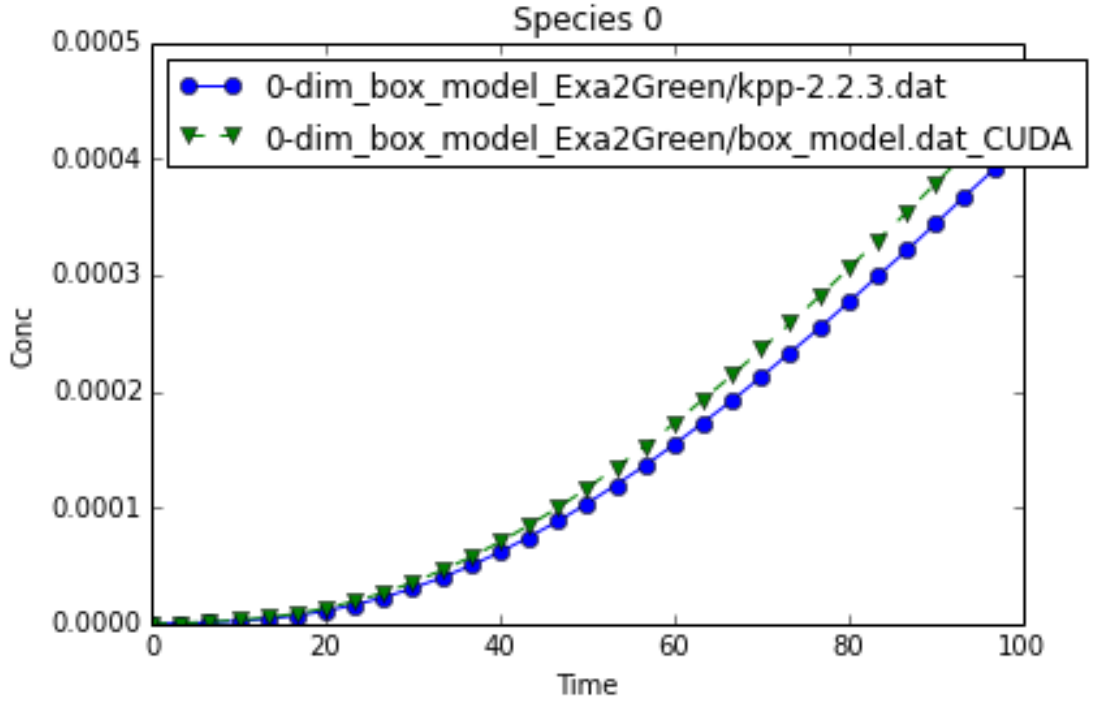
In [28]:
```
kpp_dat_1 = read_datfile(kpp_file_1, 0, 1)
kpp_dat_2 = read_datfile(kpp_file_2, 0, 1)
```

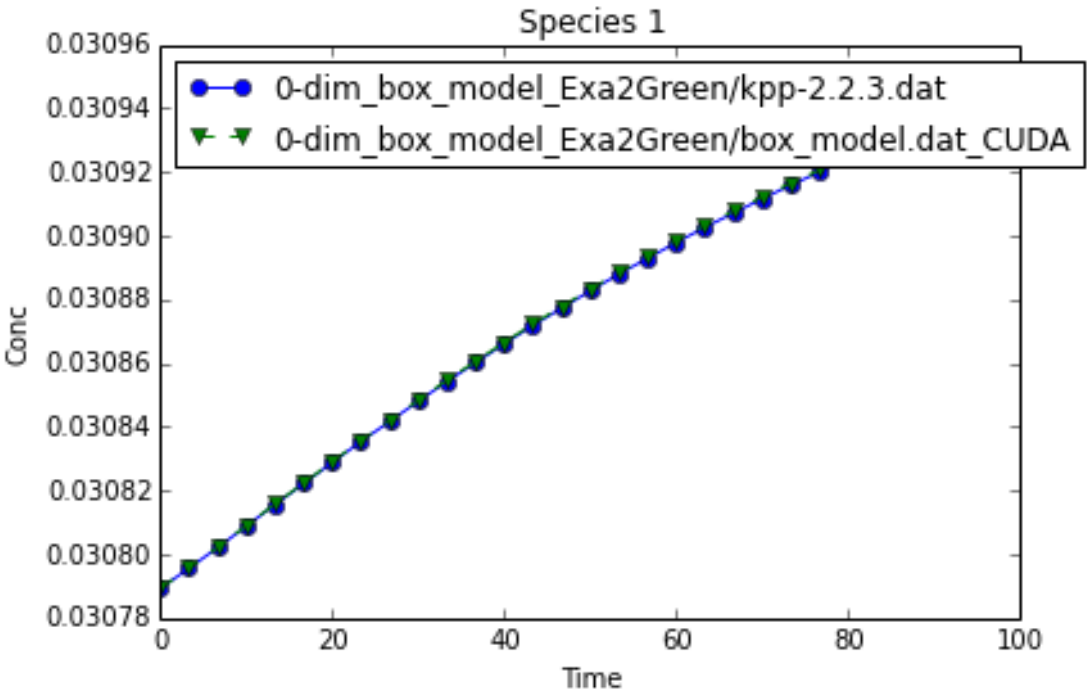
In [29]:
```
err_dat = calc_err(kpp_dat_1, kpp_dat_2)
```

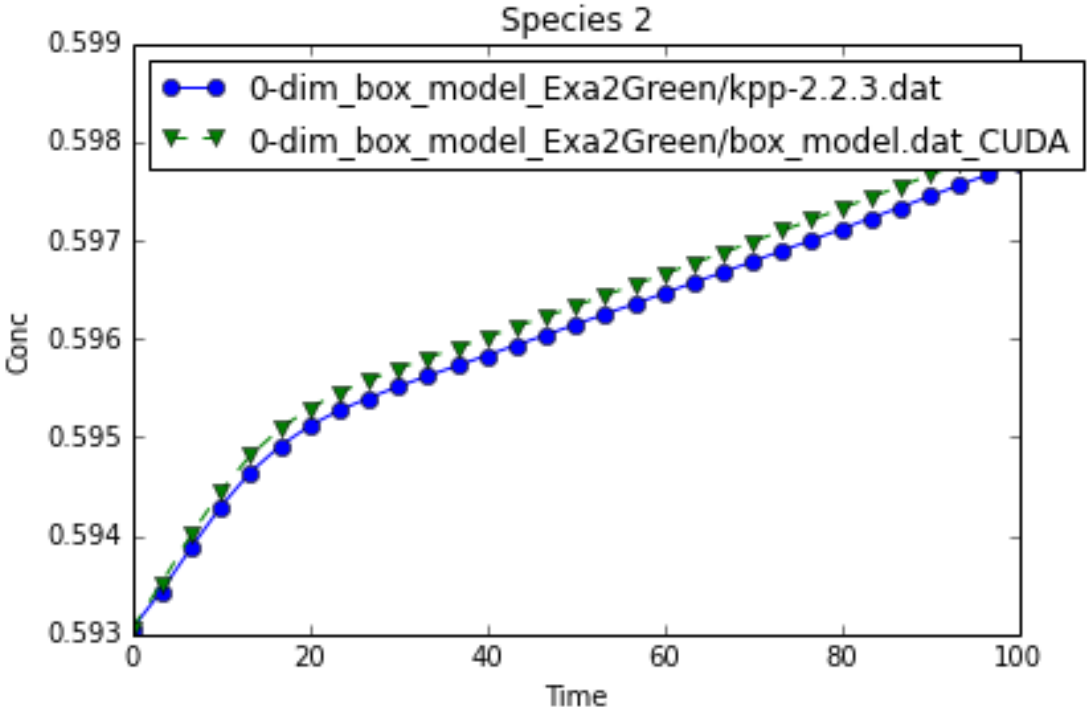
```
0.361757 > 0.01: 3.47136e-07, 5.43892e-07
0.0500139 > 0.01: 2.58455e-06, 2.45528e-06
0.317047 > 0.01: 1.02877e-06, 1.50635e-06
0.0515577 > 0.01: 1.38377e-06, 1.31242e-06
0.270037 > 0.01: 2.19895e-06, 3.01242e-06
0.0513771 > 0.01: 8.57723e-07, 8.13656e-07
0.231043 > 0.01: 4.09773e-06, 5.32894e-06
0.0523446 > 0.01: 6.25009e-07, 5.92293e-07
0.200784 > 0.01: 6.9504e-06, 8.69651e-06
0.0527517 > 0.01: 5.20819e-07, 4.93345e-07
0.178726 > 0.01: 1.0933e-05, 1.33123e-05
0.052588 > 0.01: 4.70499e-07, 4.45757e-07
0.0114677 > 0.01: 0.00118458, 0.00117099
0.162418 > 0.01: 1.61581e-05, 1.92914e-05
0.0529799 > 0.01: 4.42936e-07, 4.1947e-07
0.0178991 > 0.01: 0.000441011, 0.000433117
0.15027 > 0.01: 2.26787e-05, 2.66893e-05
0.0530034 > 0.01: 4.24064e-07, 4.01588e-07
0.026886 > 0.01: 0.000162478, 0.000158109
0.141005 > 0.01: 3.0508e-05, 3.55159e-05
0.0531093 > 0.01: 4.08404e-07, 3.86714e-07
0.0314442 > 0.01: 6.20304e-05, 6.00799e-05
0.133753 > 0.01: 3.96327e-05, 4.57521e-05
0.0531921 > 0.01: 3.93743e-07, 3.72799e-07
0.0315556 > 0.01: 2.77849e-05, 2.69081e-05
0.127923 > 0.01: 5.00239e-05, 5.73618e-05
0.0533362 > 0.01: 3.79387e-07, 3.59152e-07
0.0207285 > 0.01: 1.67887e-05, 1.64407e-05
0.123121 > 0.01: 6.16424e-05, 7.02975e-05
0.0534569 > 0.01: 3.6513e-07, 3.45611e-07
0.0108523 > 0.01: 1.3687e-05, 1.35385e-05
0.119076 > 0.01: 7.44419e-05, 8.45043e-05
0.0536226 > 0.01: 3.51022e-07, 3.32199e-07
0.115601 > 0.01: 8.83707e-05, 9.99218e-05
0.0537533 > 0.01: 3.37118e-07, 3.18997e-07
0.112566 > 0.01: 0.000103373, 0.000116485
0.0539258 > 0.01: 3.23543e-07, 3.06095e-07
0.109874 > 0.01: 0.000119389, 0.000134126
0.0540592 > 0.01: 3.10353e-07, 2.93576e-07
0.107457 > 0.01: 0.000136358, 0.000152774
0.0542338 > 0.01: 2.9765e-07, 2.81508e-07
0.105264 > 0.01: 0.000154215, 0.000172358
```

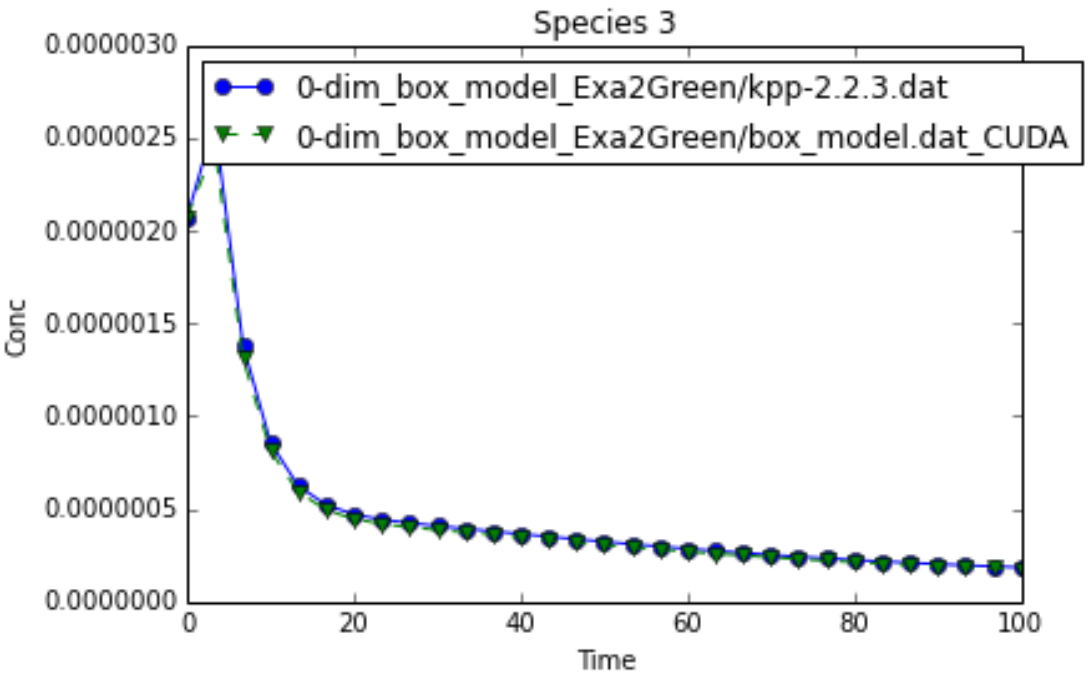
```
0.0543685 > 0.01: 2.85465e-07, 2.69945e-07
0.103255 > 0.01: 0.000172895, 0.000192803
0.0545433 > 0.01: 2.73864e-07, 2.58927e-07
0.101402 > 0.01: 0.000192334, 0.000214037
0.0546787 > 0.01: 2.62852e-07, 2.4848e-07
0.0996805 > 0.01: 0.000212464, 0.000235987
0.0548522 > 0.01: 2.52466e-07, 2.38618e-07
0.0980734 > 0.01: 0.000233221, 0.000258581
0.0549876 > 0.01: 2.42688e-07, 2.29343e-07
0.0965663 > 0.01: 0.00025454, 0.000281747
0.0551586 > 0.01: 2.33529e-07, 2.20648e-07
0.0951477 > 0.01: 0.000276357, 0.000305417
0.0552933 > 0.01: 2.24955e-07, 2.12517e-07
0.0938081 > 0.01: 0.000298612, 0.000329525
0.0554606 > 0.01: 2.1696e-07, 2.04927e-07
0.09254 > 0.01: 0.000321245, 0.000354004
0.0555936 > 0.01: 2.09499e-07, 1.97852e-07
0.0913369 > 0.01: 0.000344196, 0.000378794
0.0557564 > 0.01: 2.02554e-07, 1.9126e-07
0.0901936 > 0.01: 0.000367412, 0.000403835
0.0558866 > 0.01: 1.96076e-07, 1.85118e-07
0.0891054 > 0.01: 0.000390837, 0.00042907
0.0560443 > 0.01: 1.90039e-07, 1.79389e-07
0.0880685 > 0.01: 0.000414422, 0.000454444
0.056171 > 0.01: 1.84396e-07, 1.74038e-07
SNR: 195.358743db
67 samples with relative error > 0.01
```

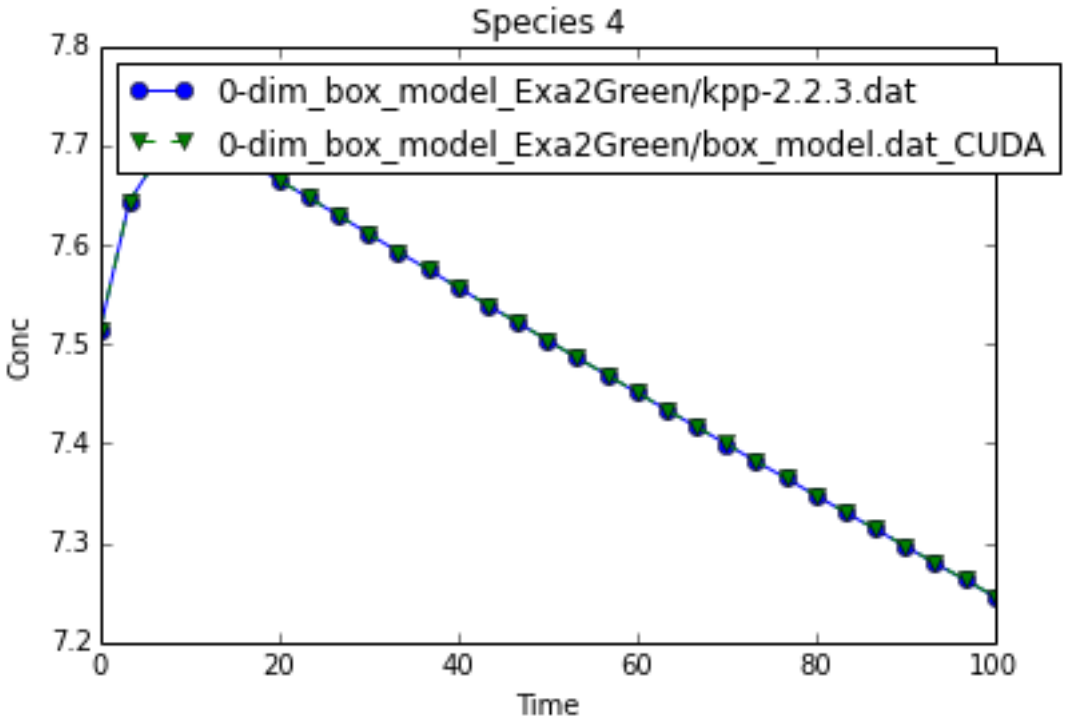
In [30]:
```
plot_dat([kpp_dat_1, kpp_dat_2], names=[kpp_file_1, kpp_file_2], titles=None)
```

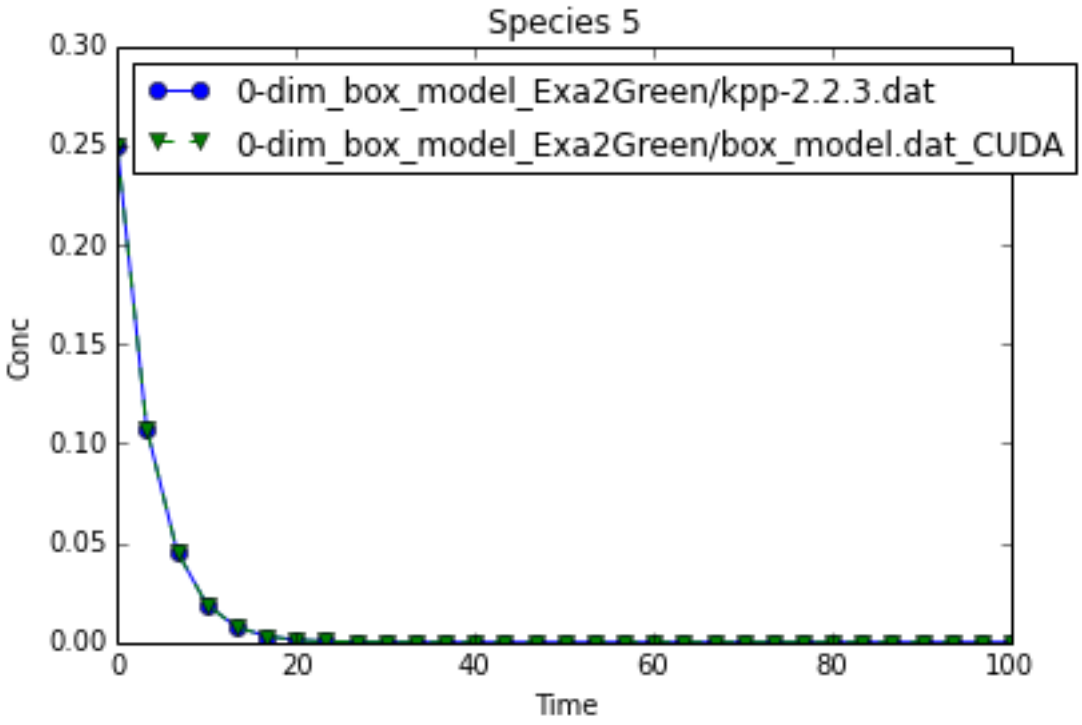


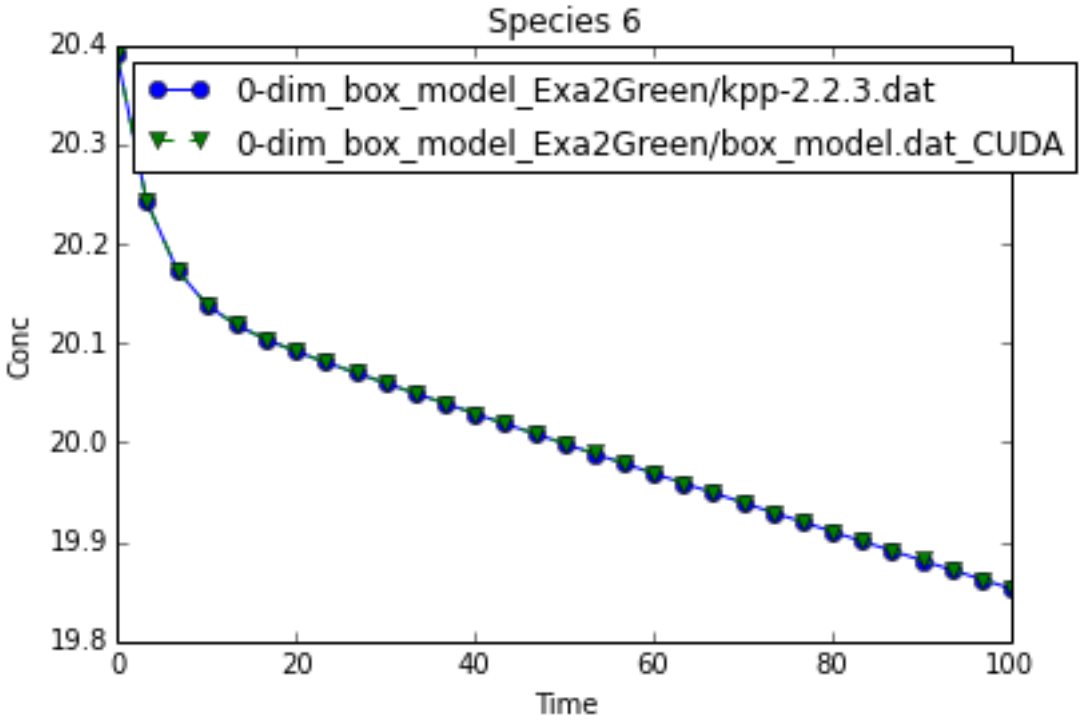

Species 1
0-dim_box_model_Exa2Green/kpp-2.2.3.dat
0-dim_box_model_Exa2Green/box_model.dat_CUDA
Conc
Time
0.03096
0.03094
0.03092
0.03090
0.03088
0.03086
0.03084
0.03082
0.03080
0.03078
0
20
40
60
80
100

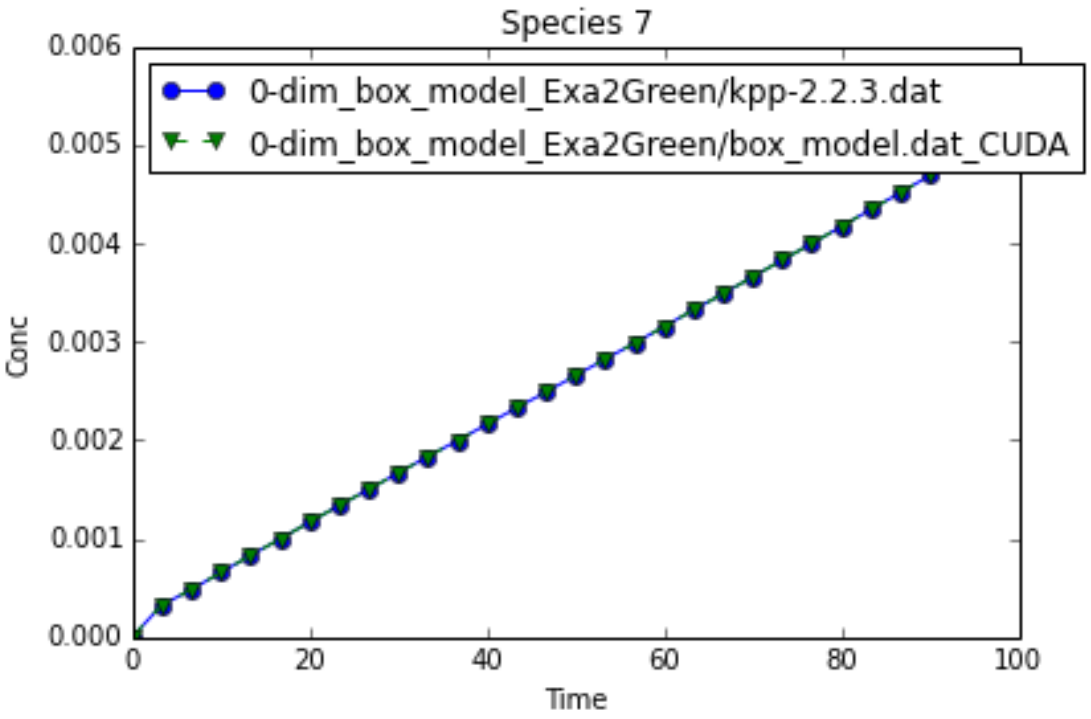

Species 2
0-dim_box_model_Exa2Green/kpp-2.2.3.dat
0-dim_box_model_Exa2Green/box_model.dat_CUDA
Conc
Time
0.599
0.598
0.597
0.596
0.595
0.594
0.593
0
20
40
60
80
100


Species 3
0-dim_box_model_Exa2Green/kpp-2.2.3.dat
0-dim_box_model_Exa2Green/box_model.dat_CUDA
Conc
Time
0.0000030
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
0
20
40
60
80
100

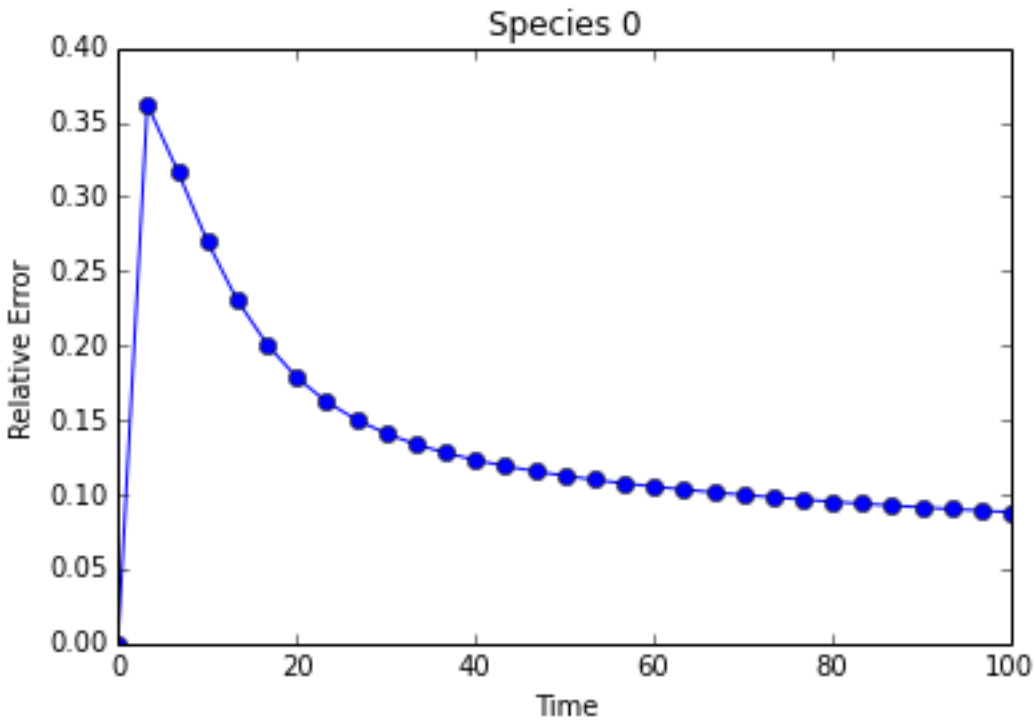

Species 4
0-dim_box_model_Exa2Green/kpp-2.2.3.dat
0-dim_box_model_Exa2Green/box_model.dat_CUDA
Conc
Time
7.8
7.7
7.6
7.5
7.4
7.3
7.2
0
20
40
60
80
100

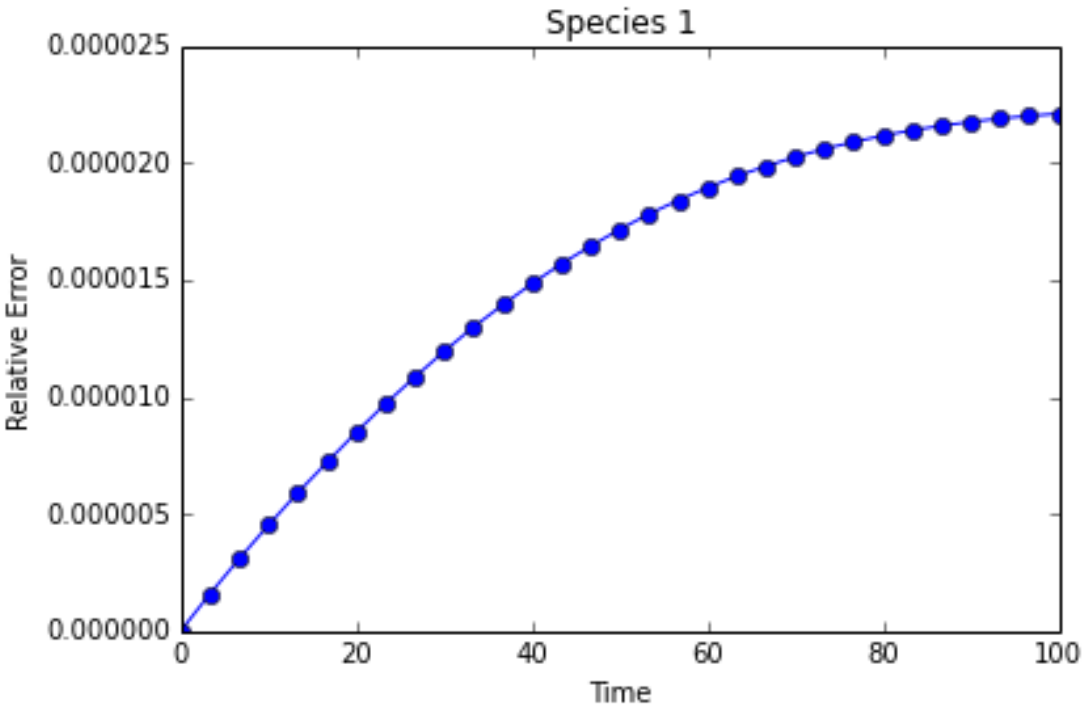

Species 5
0-dim_box_model_Exa2Green/kpp-2.2.3.dat
0-dim_box_model_Exa2Green/box_model.dat_CUDA
Conc
Time
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100

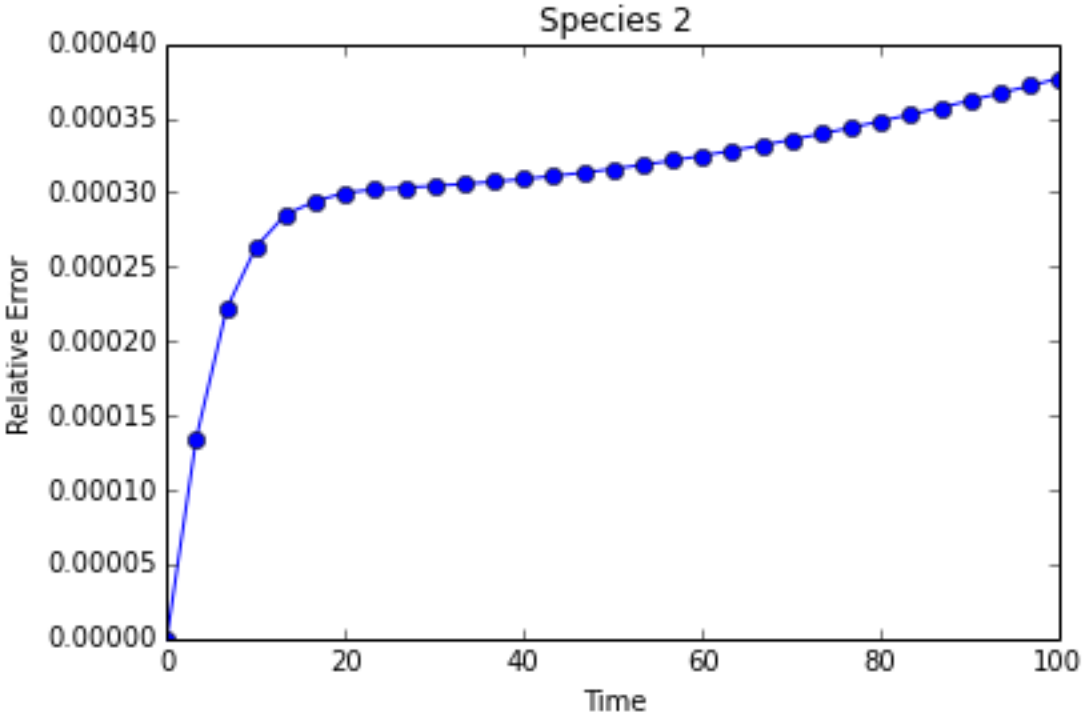

Species 6
0-dim_box_model_Exa2Green/kpp-2.2.3.dat
0-dim_box_model_Exa2Green/box_model.dat_CUDA
Conc
Time
20.4
20.3
20.2
20.1
20.0
19.9
19.8
0
20
40
60
80
100

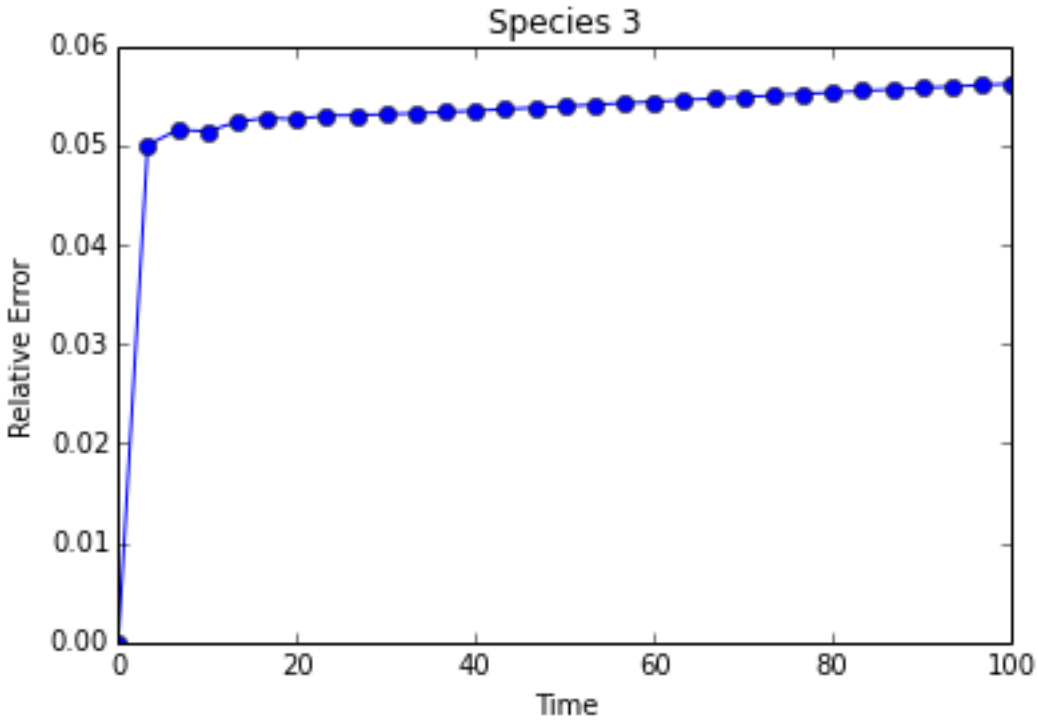


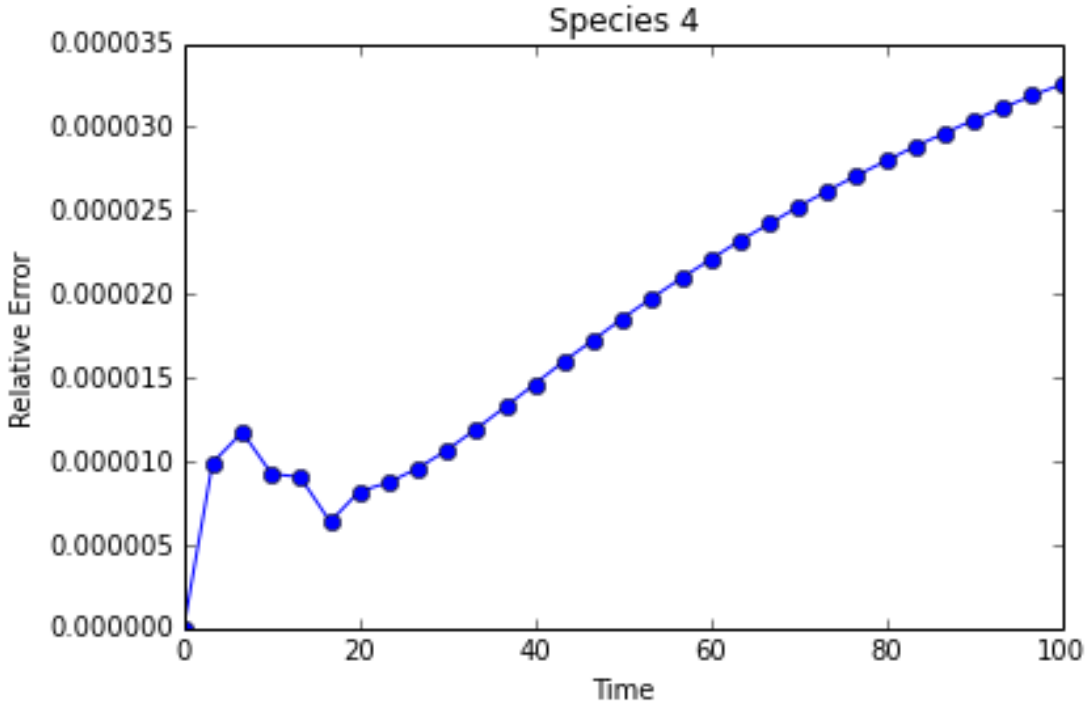
In [31]:
```
plot_dat([err_dat], ylabel='Relative Error')
```

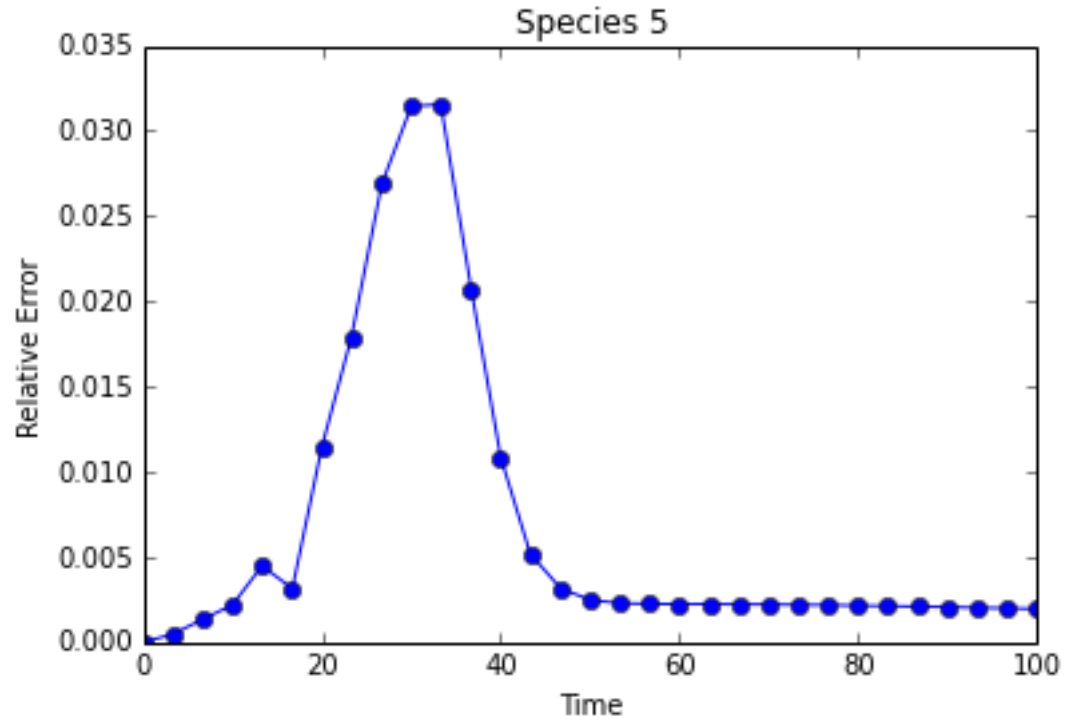


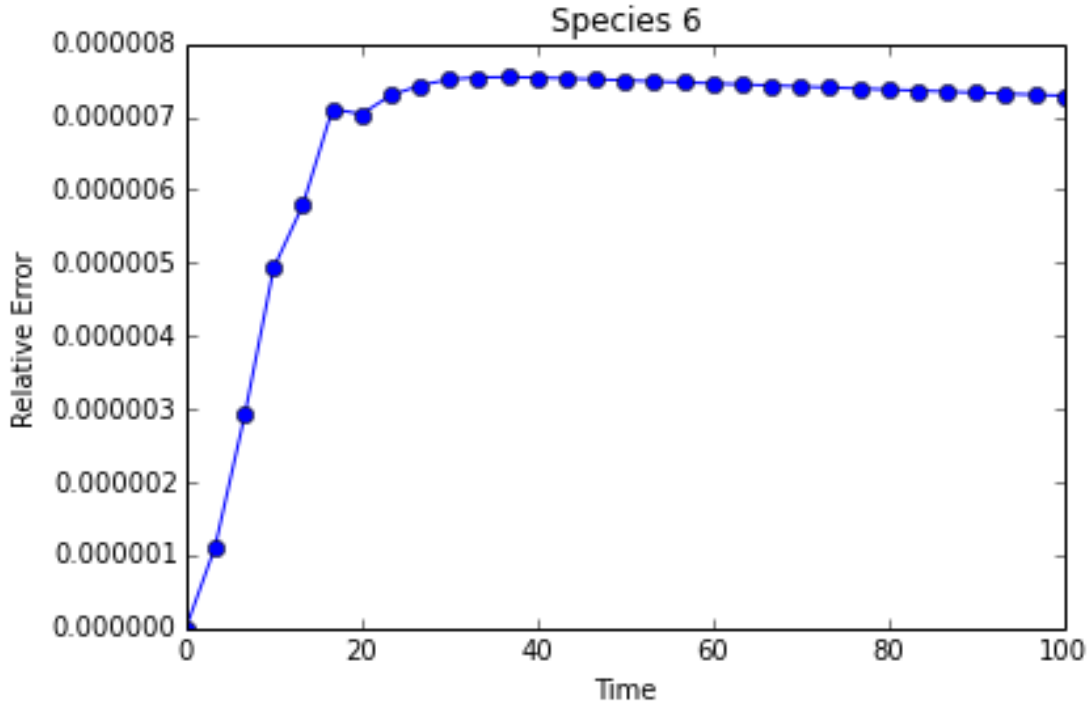

Species 1
Relative Error
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000
0
20
40
60
80
100
Time

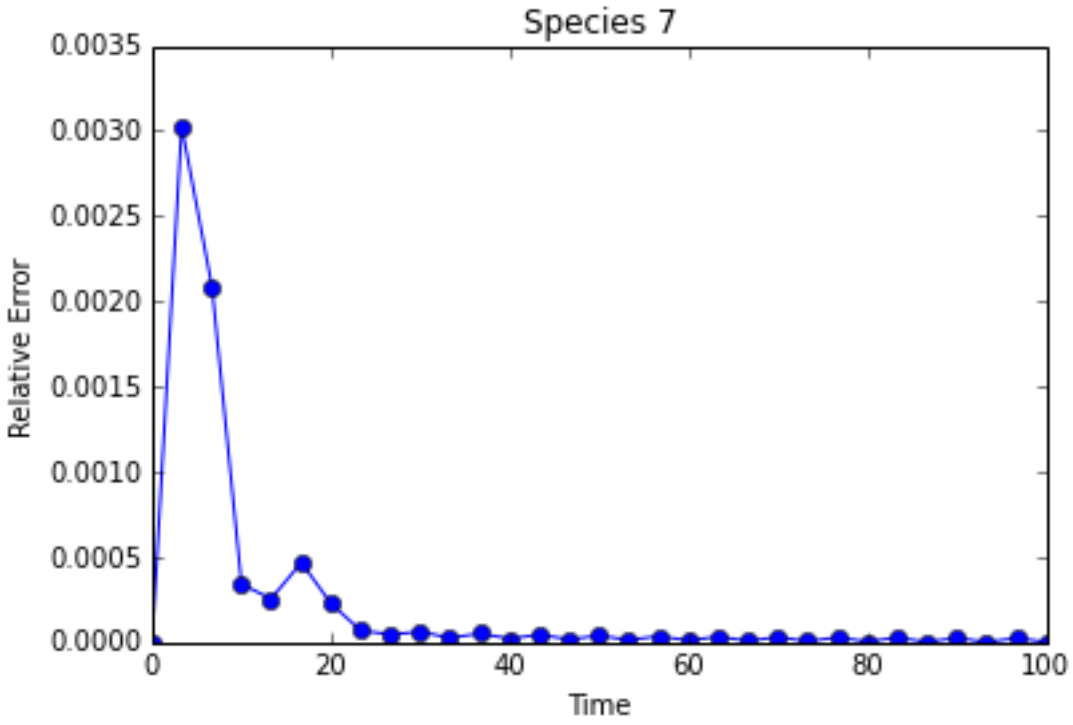

Species 2
Relative Error
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
20
40
60
80
100
Time


Species 3
Relative Error
Time
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100


Species 4
Relative Error
Time
0.000035
0.000030
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000
0
20
40
60
80
100


Species 5
Relative Error
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
Time


Species 6
Relative Error
0.000008
0.000007
0.000006
0.000005
0.000004
0.000003
0.000002
0.000001
0.000000
0
20
40
60
80
100
Time


Species 7
Relative Error
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
20
40
60
80
100
Time

In [31]: