Kppa Verification using KPP-generated code

```
Created: 2 June 2014
John C. Linford <jlinford@paratools.com>
Copyright (c) ParaTools, Inc.
```

Description

Shows results from `box_model` as generated by KPP and Kppa. The Kppa code has been modified to match KPP's integration limits, use Rodas3, and use a constant temperature of 270K.

Instructions

1. Compile and run the `box_model` and `box_model_kpp221` codes
2. Execute all cells in this notebook.

In [5]:
```
kppa_file = 'box_model/box_model.dat'
kpp_file = 'box_model_kpp221/box_model.dat'
```

Processing Script

Skip down for results

In [6]:
```
%matplotlib inline
import re
from itertools import cycle
from pylab import *
from matplotlib.markers import MarkerStyle
import matplotlib.pyplot as plt

ATOL = 1.0e-3
RTOL = 1.0e-4
EPS = 2.220446049250313E-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 be
ginning in field cstart.
    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 [7]:
```
kppa_dat = read_datfile(kppa_file, 0, 3)
kpp_dat = read_datfile(kpp_file, 1, 1)
```

Results: Relative Error and SNR

The following cell shows the relative error per concentration for each time step written to the `.dat` files. A signal-to-noise ratio and relative error count are shown below. As a rule of thumb, SNR >= 250 indicates solution match to five decimal places. This is a **relative error calculation**, so concentrations close to zero are likely to have high relative error but low absolute error.

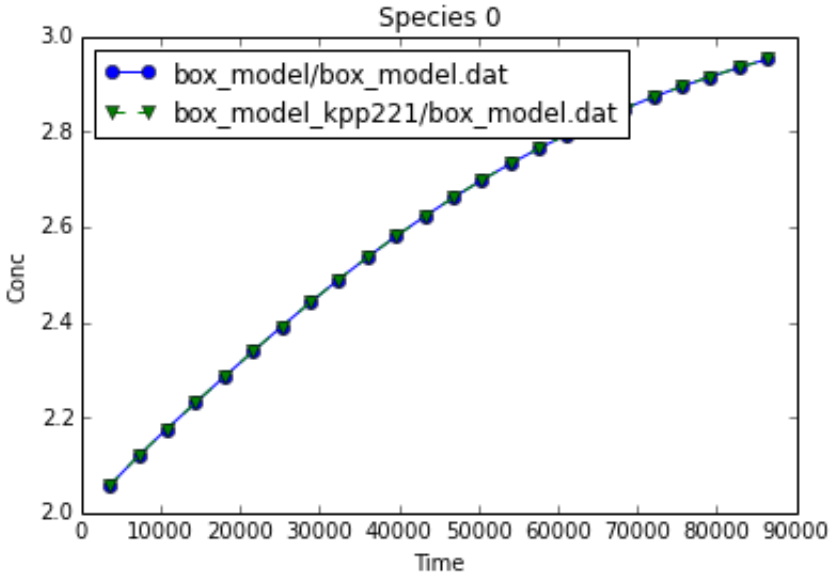
In [8]:
```
err_dat = calc_err(kpp_dat, kppa_dat)
```

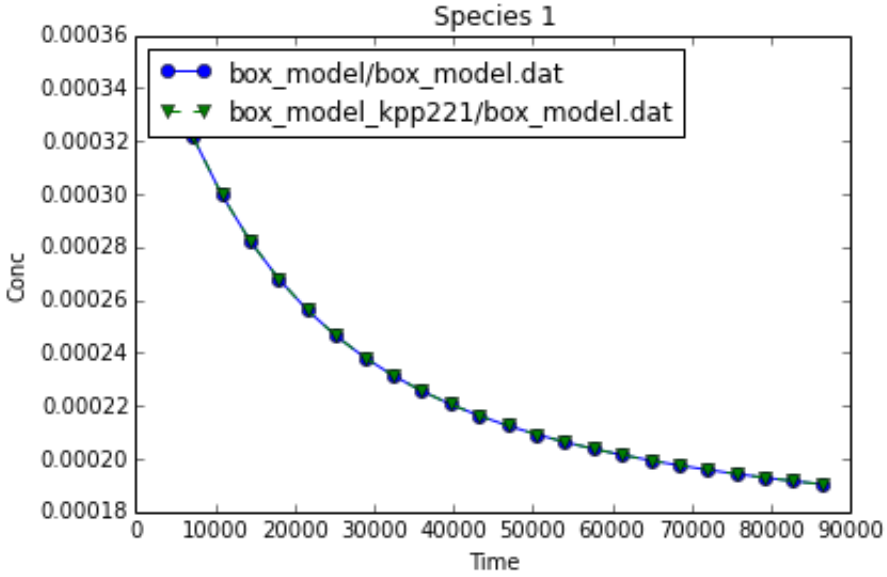
```
SNR: 256.508769db
```

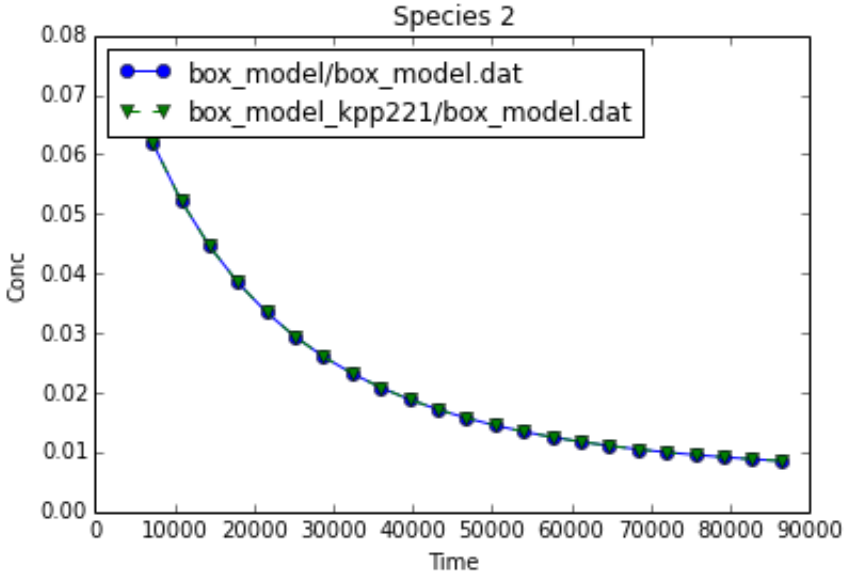
Results: Concentration vs. Time

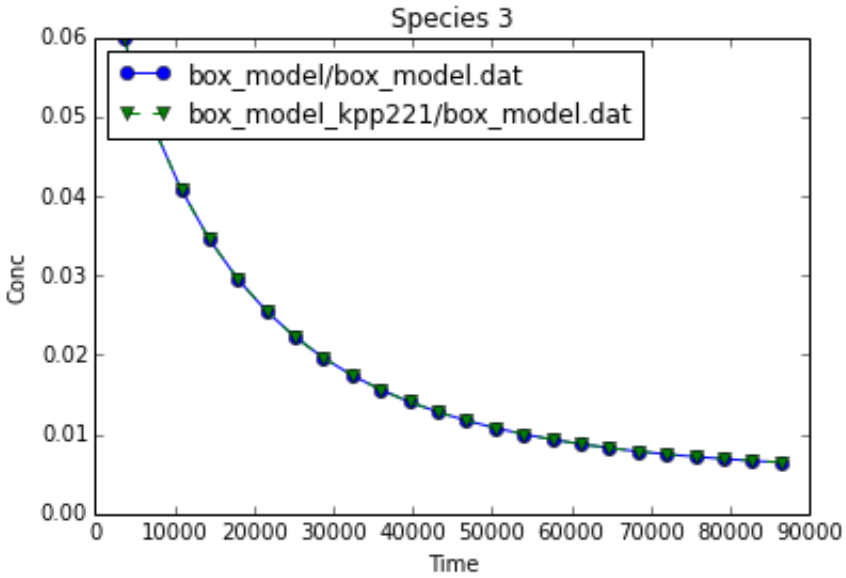
The following charts show concentrations from both files vs. time. The species name is not recored in the `.dat` file, but you can replace `None` in the `plot_dat` call with an ordered list of names to set the plot titles.

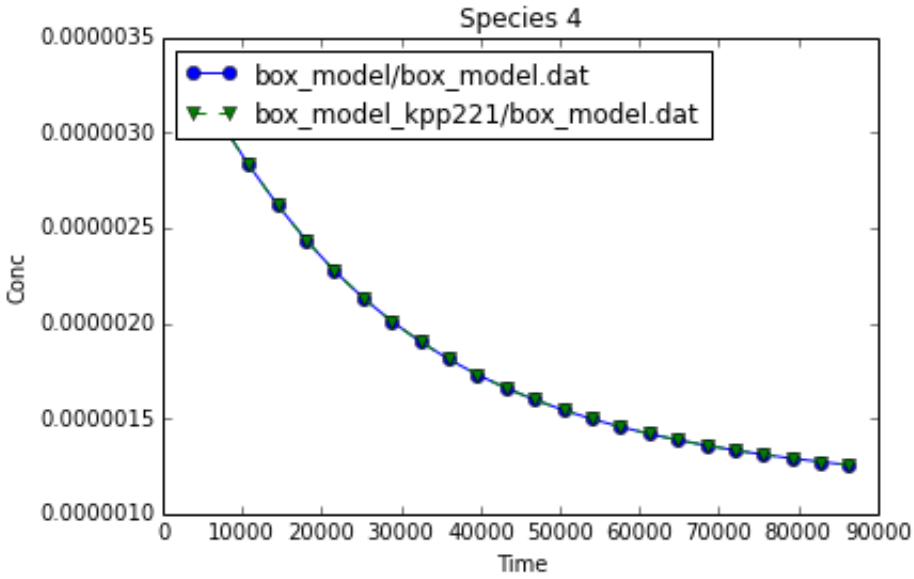
In [9]:
```
plot_dat([kppa_dat, kpp_dat], names=[kppa_file, kpp_file], titles=None)
```




Species 1
box_model/box_model.dat
box_model_kpp221/box_model.dat
Conc
Time
0.00036
0.00034
0.00032
0.00030
0.00028
0.00026
0.00024
0.00022
0.00020
0.00018
0
10000
20000
30000
40000
50000
60000
70000
80000
90000


Species 2
box_model/box_model.dat
box_model_kpp221/box_model.dat
Conc
Time
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
10000
20000
30000
40000
50000
60000
70000
80000
90000

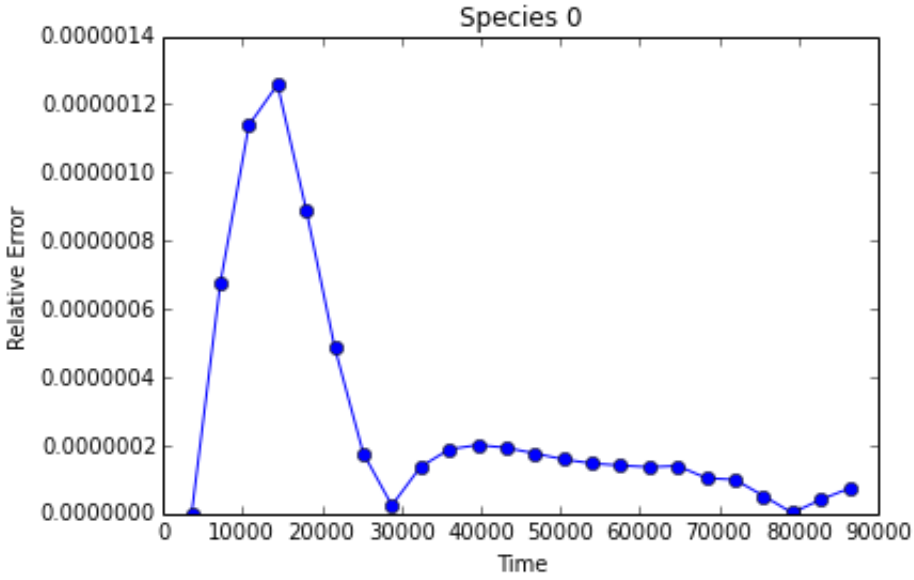

Species 3
box_model/box_model.dat
box_model_kpp221/box_model.dat
Conc
Time
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
10000
20000
30000
40000
50000
60000
70000
80000
90000

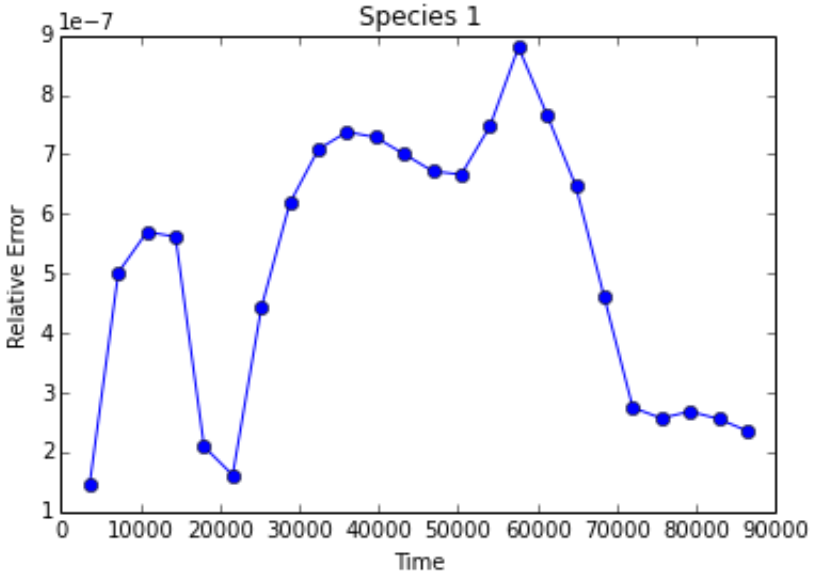


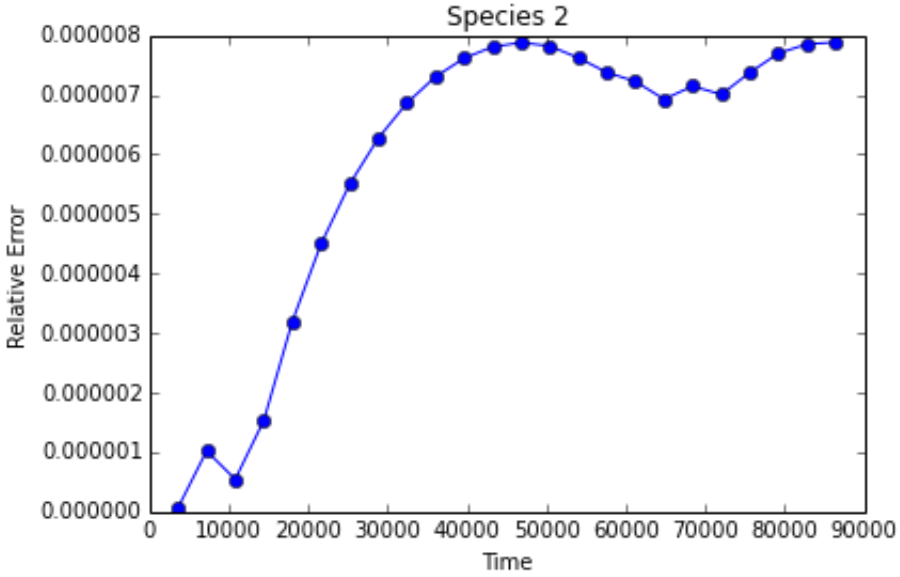
Results: Error vs. Time

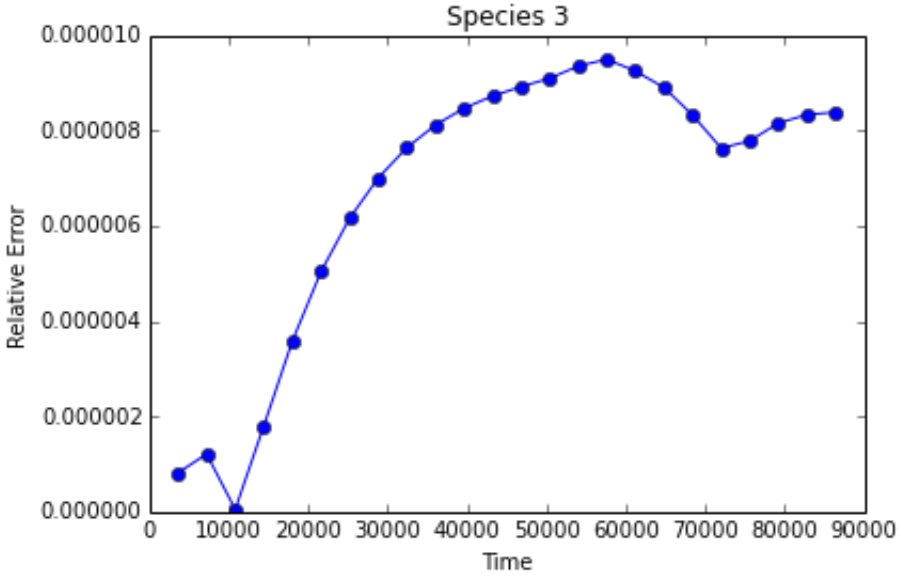
The following charts show relative error between the two data files vs. time.

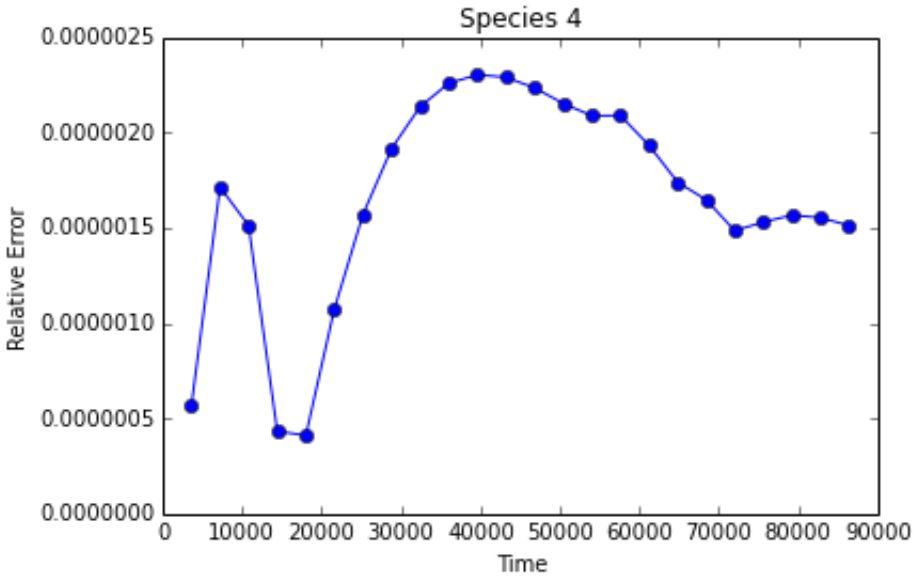
```
In [10]: plot_dat([err_dat], ylabel='Relative Error')
```




Species 1
1e−7
Relative Error
Time


Species 2
Relative Error
Time


Species 3
Relative Error
Time


Species 4
Relative Error
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
Time

In [10]: