In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from scipy.interpolate import interp1d
```

1.1 Following equation 1-2 (without the buffer effect), build a two-box model to compute the atmospheric CO2 level in ppm (parts per million) from 1987 to 2004.

In [3]:
```
df_gamma = pd.read_csv('global.1751_2014.csv')
df_gamma
```

Out[3]:

	Year	Total carbon emissions from fossil fuel consumption and cement production (million metric tons of C)	Carbon emissions from gas fuel consumption	Carbon emissions from liquid fuel consumption	Carbon emissions from solid fuel consumption	Carbon emissions from cement production	Carbo emission from ga flarin
0	Source: Tom Boden (Oak Ridge National Laboratory)	Gregg Marland (Appalachian State University)	and Bob Andres (Oak Ridge National Laboratory)	NaN	NaN	NaN	Na
1	1751	3	0	0.0	3.0	0.0	0.
2	1752	3	0	0.0	3.0	0.0	0.
3	1753	3	0	0.0	3.0	0.0	0.
4	1754	3	0	0.0	3.0	0.0	0.
...	...	...	...	...	...	...	.
260	2010	9128	1696	3107.0	3812.0	446.0	67.
261	2011	9503	1756	3134.0	4055.0	494.0	64.
262	2012	9673	1783	3200.0	4106.0	519.0	65.
263	2013	9773	1806	3220.0	4126.0	554.0	68.
264	2014	9855	1823	3280.0	4117.0	568.0	68.

265 rows × 8 columns

In [4]:
```
df_gamma['Year'] = pd.to_numeric(df_gamma['Year'], errors='coerce')
df_gamma['Total carbon emissions from fossil fuel consumption and cement prod
```

In [5]:
```
df_gamma = df_gamma[df_gamma['Year'] < 2005]
```

In [6]:
```
gamma = interp1d(df_gamma['Year'], df_gamma['Total carbon emissions from foss
```

In [7]:
```
def nobufferModel (N, t, Args):
    N1, N2 = N
    k12, k21, gamma = Args
    gamma2 = gamma(t)*0.001
    dN1dt = -k12*N1+k21*N2+gamma2
    dN2dt = k12*N1-k21*N2
    return np.array([dN1dt, dN2dt])
```

1.2 Following equation 3-4 (with the buffer effect), build a two-box model to compute the atmospheric CO2 level in ppm from 1987 to 2004.

In [8]:
```
def withbufferModel (N, t, Args):
    N1, N2 = N
    k12, k21, gamma, s, N20 = Args
    gamma2 = gamma(t)*0.001
    dN1dt = -k12*N1+k21*(s*(N2-N20)+N20)+gamma2
    dN2dt = k12*N1-k21*(s*(N2-N20)+N20)
    return np.array([dN1dt, dN2dt])
```

In [ ]:
```

```