Lab 10

EOSC 511

Christopher Rodell

Problem One

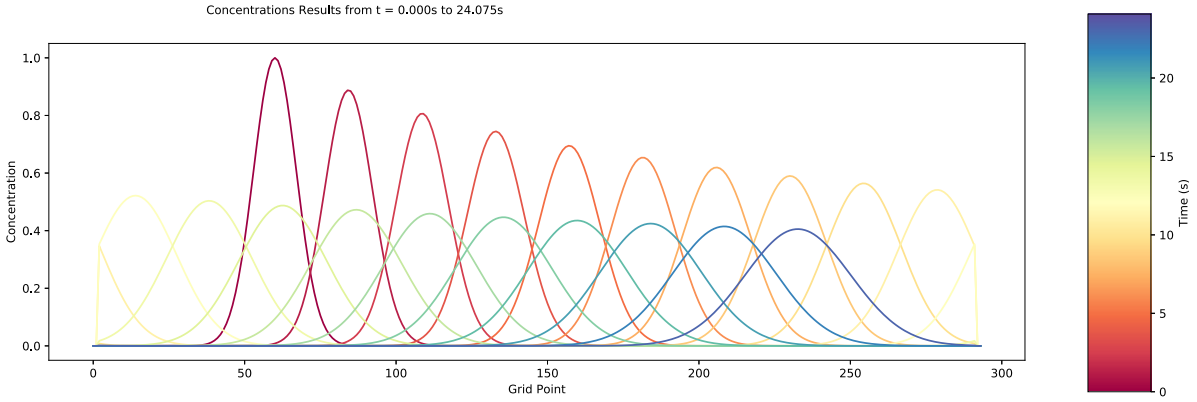
Using the scheme presented in this section, modify advection3.py to solve the following advection problem: The wind is moving along the x-axis with speed u=20 m/s. The initial distribution curve is 290 km in width. Use your program to approximate the curve during 24 hours.

- a) Run your program for different orders of approximating polynomials (up to 4). Compare the accuracy of approximation for different orders. Do you see better results with increasing order? Is this true for all orders from 0 to 4? Is there any particularity to odd and even order polynomials?

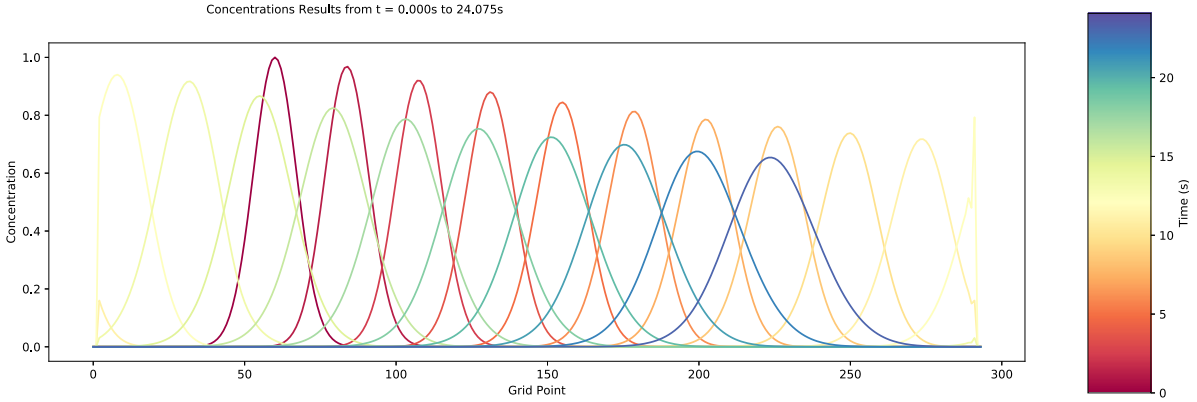
  If we choose ℓ=0, we will end up with the upstream method. In other words, the upstream method assumes that c is constant in each grid box. This poor representation of c results in strong numerical diffusion. Experiments have shown that generally if we use higher-order polynomials (where $\ell \leq 4$), we can significantly suppress numerical diffusion. Thus the higher-order produces better results. The even order scheme shows much higher rates of diffusion as compared to the odd order scheme. This is explained here: "Note that for even order polynomials, an odd number of$c$c values including the grid point jj are needed to calculate the coefficients $a_{j,k}$aj,k. This means the same number of grid points to the left and right of $x_j$xj are used in the calculation of $a_{j,k}$aj,k. If on the other hand, we choose odd order polynomials, there will be one extra point used to either side of $x_j$xj, thus resulting in 2 different representations of $a_{j,k}$aj,k." All this mean is that the even-order are basically centered difference and that the odd-order maintian the upstream method. Which is important for numeric stbaility. *(SEE PLOTS BELOW AS SUPORT FOR MY ANSWER)*

In [1]:
```
import context
import numpy as np
import advection_funs as adv
import matplotlib.pyplot as plt


order = np.arange(0,5)
for od in order:
    adv.advection3(1070,od)
    print("The Order:   ", od)
    plt.show('all')
```

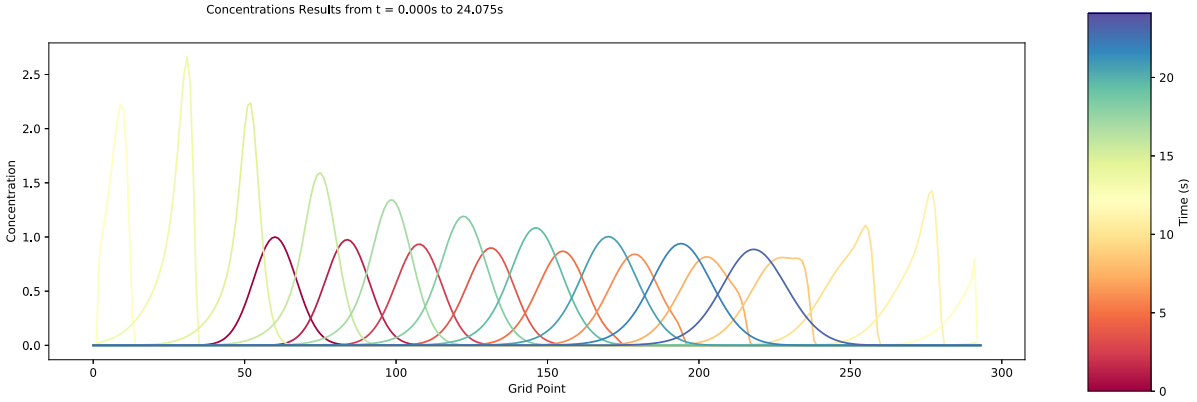
```
*****************************
context imported. Front of path:
/Users/rodell/repos/numeric_students
back of path: /Users/rodell/.ipython
*****************************

through /Users/rodell/repos/numeric_students/numeric_notebooks/labs_7b_
8_10/context.py
The Order:    0
```



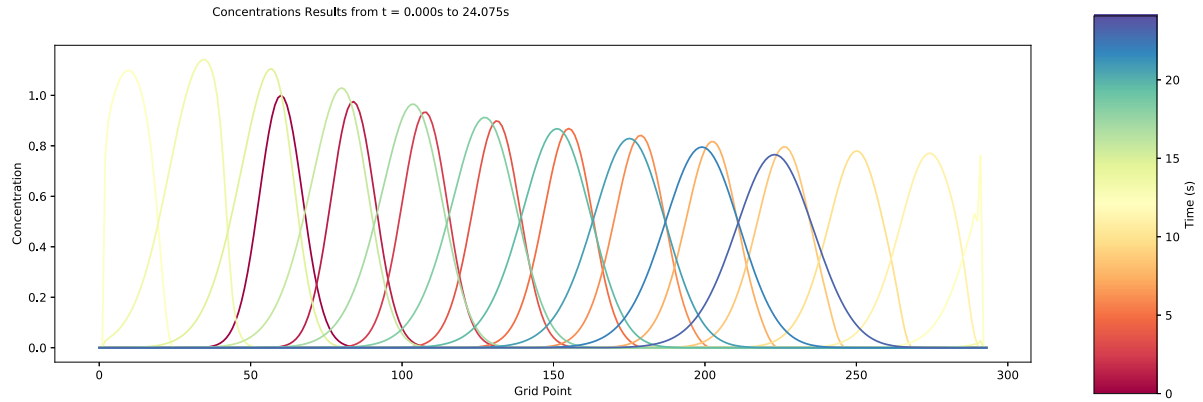

```
The Order:    1
```



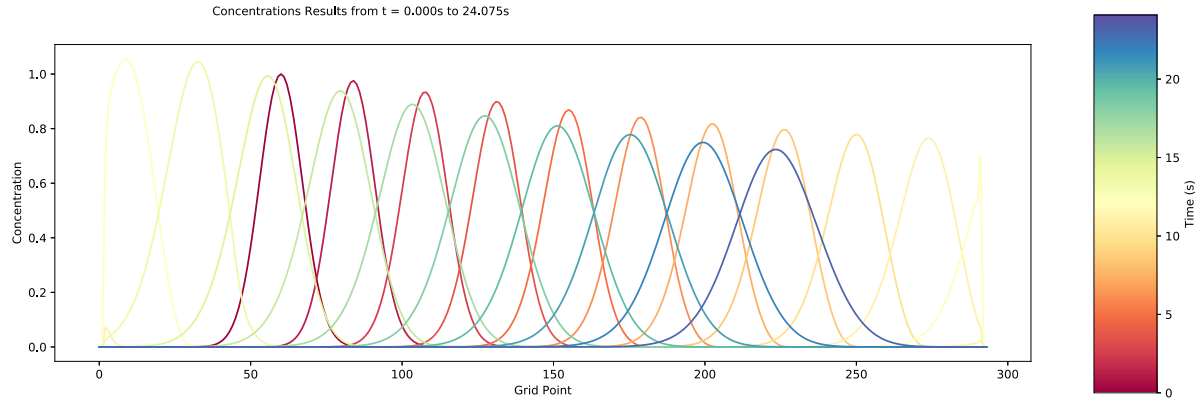

```
The Order:    2
```




```
The Order:    3
```




```
The Order:    4
```



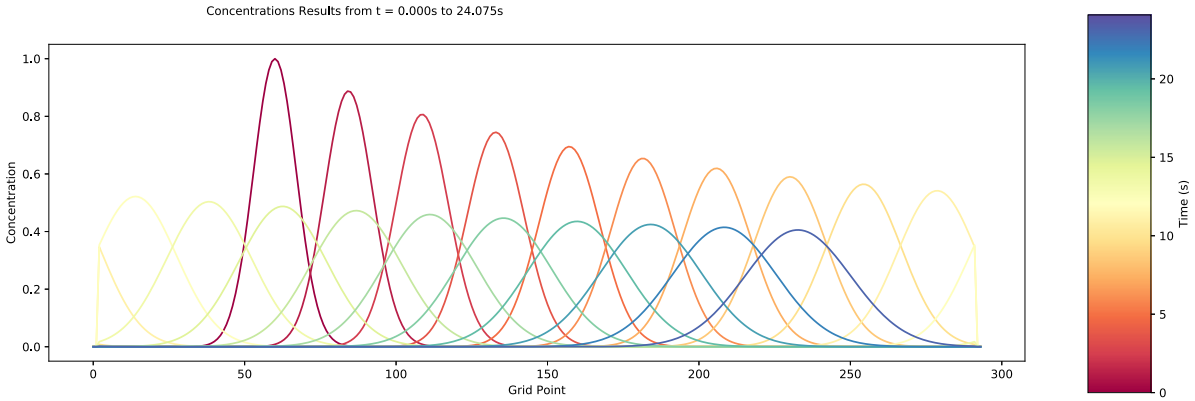
- b) For odd ordered polynomials, advection3.py uses the representation of aj,k that involves an extra point to the right of the centre grid point. Modify the table of coefficients for odd ordered polynomials (Table :math:ell=1 <#tab:ell1> **)and (Table :math:ell=3 <#tab:ell3>**) to use the extra point to the left of the centre grid point. Run your program again and compare the results of 2 different representations of aj,k for order 1 and 3, respectively. Is one representation better than the other, or about the same, or does each have its own problem? How, do you think the different representation affects the result? **FOR THIS QUESTION I AM USING advection_funs_cr.py AS IS CALLSES IN MY MODIFIED TABLES FROM FOLDER Tables_cr**

  By modifying the tables to add the extra point to the left of the center grid point for odd ordered polynomials, we find higher rates of diffusion rates than we did before. This goes back to the point I made early that we are not treating the upstream method to be a centered difference schme. The centered differencing scheme does not work well for the advection process because it is not positive definite, i.e., it generates negative values that are impossible in real life.

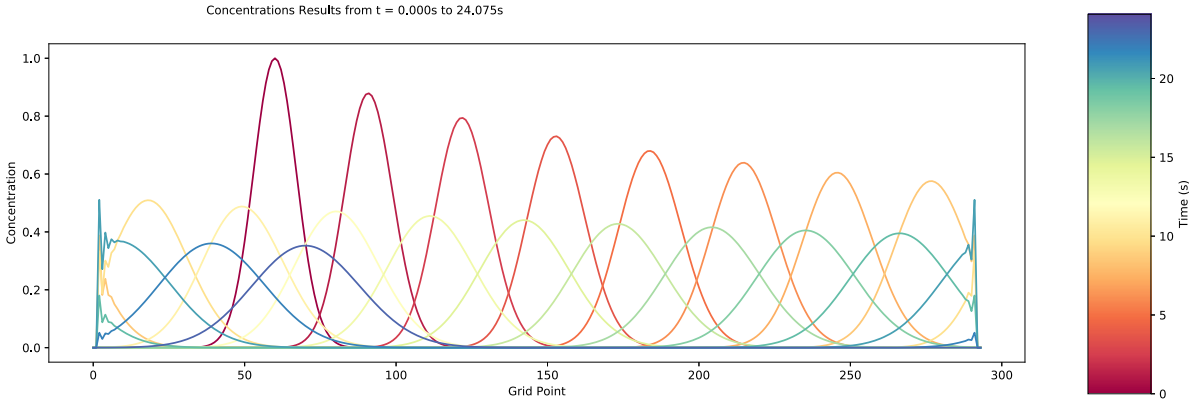
  (SEE CODE BLOCK BELOW FOR PLOTS TO SUPPORT MY ANSWER)

In [2]:
```
import advection_funs_cr as adv_cr

order = np.arange(0,5)
for od in order:
    adv_cr.advection3(1070,od)
    print("The Order:   ", od)
    plt.show('all')
```

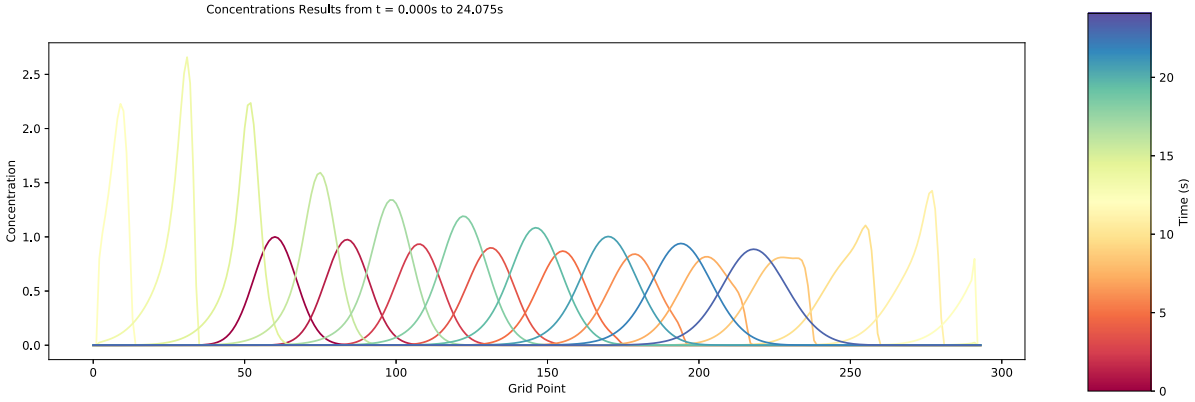
```
The Order:    0
```



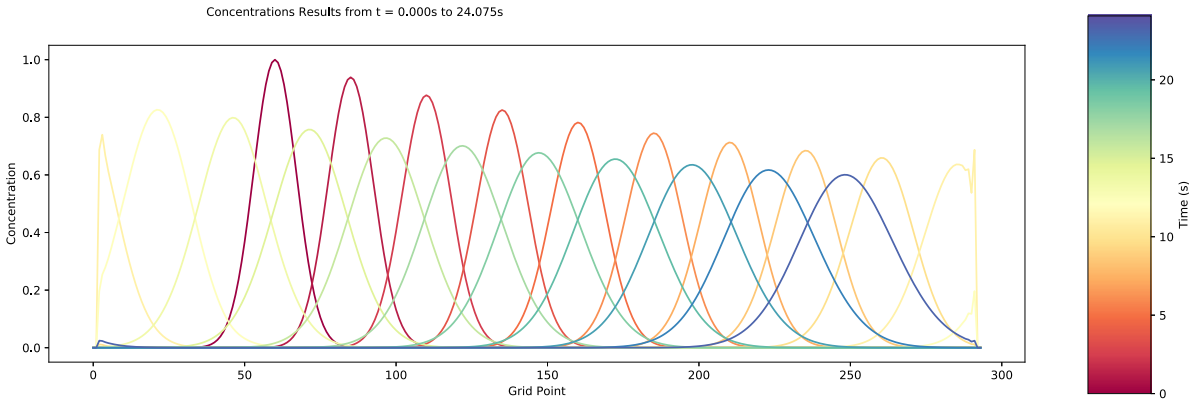

```
The Order:    1
```



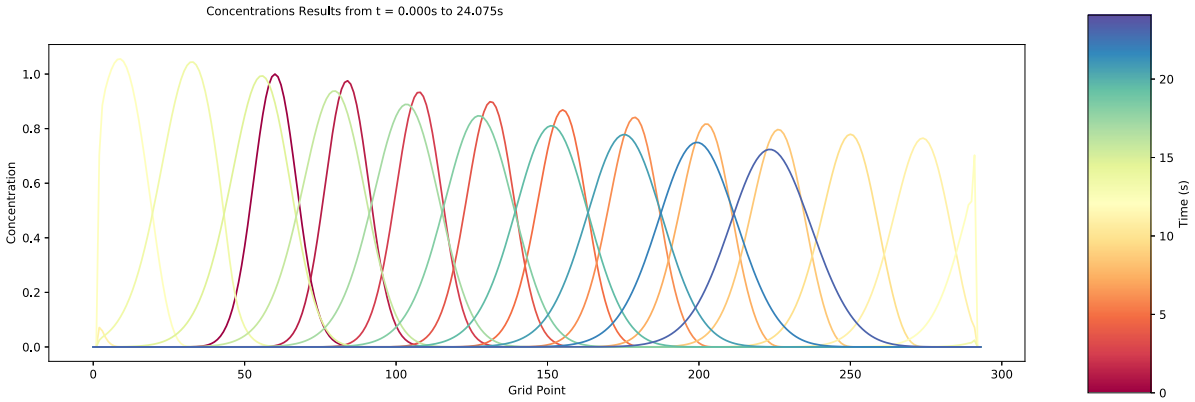

```
The Order:    2
```




```
The Order:    3
```


Concentrations Results from t = 0.000s to 24.075s
Concentration
Grid Point
Time (s)

```
The Order:    4
```


Concentrations Results from t = 0.000s to 24.075s
Concentration
Grid Point
Time (s)