Assignment-2

Shivanshu Shekhar

November 9, 2023

1 Assignments

1. Study cython to understand how to interface Python to C. Look at `http://cython.readthedocs.io/en/latest/src/tutorial/external.html` for a short introduction to this. Try to create an interface to spline.c.

2. We require a 6 digit accurate method to compute the function

$$f(x) = \sqrt{\frac{x^{1+J_0(x)}}{(1+100x^2)(1-x)}} = \sqrt{\frac{x^{1+J_0(x)}}{1-x+100x^2-100x^3}} \quad (1)$$

between 0 and 0.9. The function is known exactly (to 15 digits) at certain locations, $x_k = x_0 + kdx$,$k = 0, ..., n$ where n is the order of interpolation required.

   (a) Convert the function to a table, spaced 0.05 apart, sampling it from 0.1 to 0.9.

   (b) n python, plot this function and determine its general behavior. Is it analytic in that region? What is the radius of convergence at x = 0 and at x = 0.9?

   (c) Use spline interpolation on the function in Eq. (1). Use $y'' = 0$ at the boundaries and obtain the spline coefficients and hence interpolate as above. How does the error vary with the number of uniformly spaced spline points? How many points are required to achieve six digit accuracy? Use the C code above with weave.

   Note: What you vary are the number of points in the table. You test by interpolating on many more points and generating an error profile. Plot the same for different cases on same plot. Find the maximum error in each case, and study the scaling (how does maximum error vary with spacing of points in your table?)

(d) Revise the C code to implement the not-a-knot boundary condition. Repeat Q3 and see how the error profiles and the maximum errors vary.

(e) Use Eq. 1 to analytically compute the derivatives at the endpoints. Note that $J_0'(x)$ can be written in terms in $J_1(x)$ and hence computed. Use that information to obtain the spline coefficients. How does the error in the interpolated value compare with Q3 and Q4? This is an important issue for an experimentalist since he usually does not know y' at x_0 and x_n but can use either of Q3 or Q4 to generate his spline fit.

(f) Use hundred times the actual derivitive as your boundary condition. How does the spline fit change? Plot the points near the edge and show the way that the errors decay. Obtain a functional dependence of error vs distance from edge. This is what you need to explain in the theory assignment.

(g) Can a non-uniform spacing of spline points help reduce the number of spline points required? Try to determine the optimal spacing at which er- ror appears to be uniform across the domain. **Develop a theory for the optimal spacing to use assuming error is similar to cubic interpolation.**

2 Spline Interpolation

Spline interpolation is a mathematical technique used to construct piecewise polynomial functions that approximate a continuous curve passing through a given set of data points. The most common form of spline interpolation is cubic spline interpolation, which constructs a set of cubic polynomials that connect the data points smoothly. These cubic splines are continuous at each data point and ensure the continuity of their first and second derivatives.

The key mathematical steps in cubic spline interpolation are as follows:

1. **Data Collection:** Let (x_i, y_i) be the data points, where x_i represents the independent variable, and y_i represents the dependent variable. For $i = 0, 1, 2, \ldots, n$, where n is the total number of data points.

2. **Knot Selection:** Define the knots as $x_0, x_1, x_2, \ldots, x_n$, which correspond to the data points themselves.

3. **Polynomial Fitting:** Within each interval $[x_i, x_{i+1}]$, for $i = 0, 1, 2, \ldots, n-1$, fit a cubic polynomial of the form:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

These cubic polynomials should satisfy the following conditions:

−Interpolation Condition: $S_i(x_i) = y_i$ for all i.
−Continuity Condition: $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$ for all i.
−First Derivative Continuity: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ for all i.
−Second Derivative Continuity: $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$ for all i.

4. **Constructing the Spline:** By fitting cubic polynomials to all intervals, you obtain a set of piecewise cubic functions $\{S_0(x), S_1(x), \ldots, S_{n-1}(x)\}$. These functions, when connected end-to-end, create the cubic spline that approximates the underlying data.

5. **Interpolation:** To estimate values between data points, evaluate the appropriate cubic polynomial for any given input within the range of the data. This cubic spline provides a continuous and differentiable approximation of the data throughout its entire domain.

The cubic spline interpolation method possesses mathematical advantages such as smoothness, continuity, and differentiability. This makes it particularly suitable for applications in scientific and engineering fields where the accurate modeling of smooth curves is essential. It ensures that the interpolated curve maintains its visual and analytical properties while accurately representing the underlying data.

3 Solutions

3.1 2.1 Create a table of values

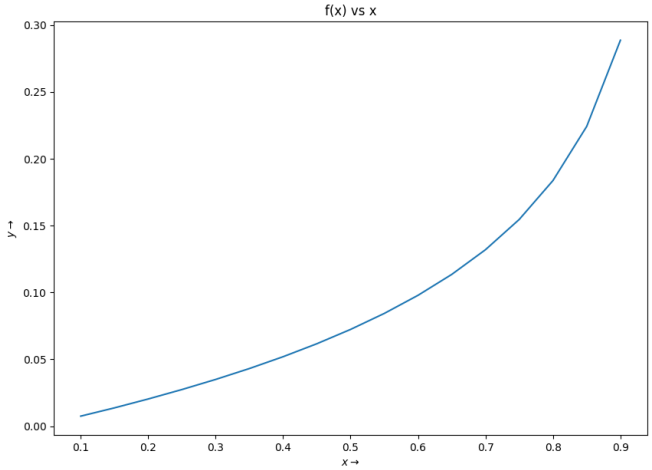
We used the following the line of code to define the function in python

```
def f(x):
return x**(1+sc.j0(x))/np.sqrt((1+x**2*100)*(1-x))
```

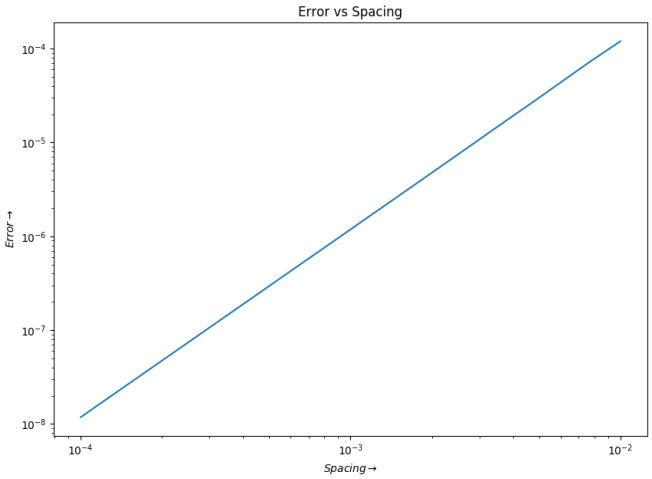
We choose x's at intervals of 0.05 between 0.1 and 0.90 both inclusive and got the corresponding function values.

3.2 2.2 Plot the function

We used matplotlib to plot the tabulated function values.



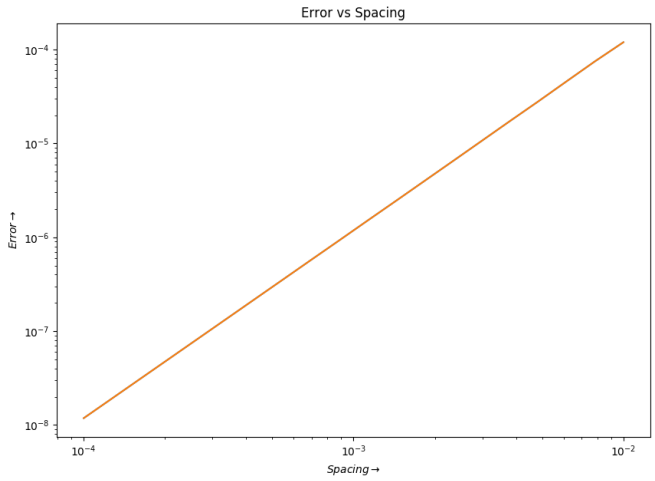
3.3 2.3 Error and Spacing



From the above plot, we can see that on a log-log scale the error linearly increases with the spacing. Hence, larger the number of sampled points the greater the accuracy. From this graph we see that for the error to be below $5 \cdot 10^{-7}$ we need a spacing of around 10^{-4} or N=8000.

3.4 2.4 Implement not a knot and repeat 2.3

We implement the code for not a knot in a different C file and interface it with python using weave. Below is the error vs spacing plot for the same.



3.5 2.5 Calculating Derivative

We start by defining:

$$f_1(x) = x^{1+J_0(x)} \tag{2}$$

$$f_2(x) = \sqrt{(1-x)(1+100x^2)} \tag{3}$$

Calculating derivatives of $f_1(x)$ and $f_2(x)$:

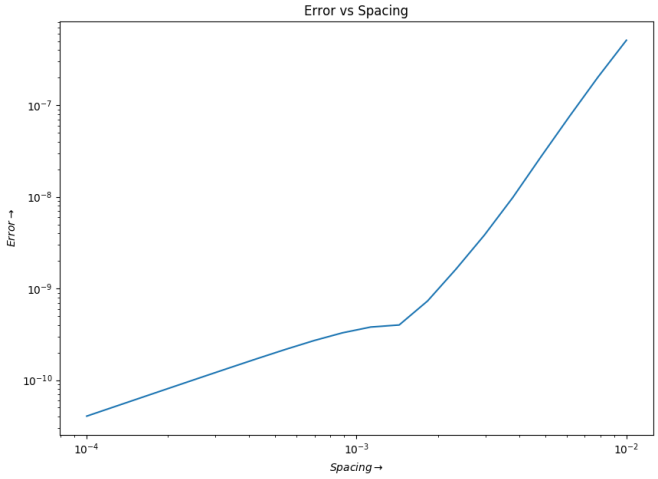
$$\frac{\mathrm{d}f_1(x)}{\mathrm{d}x} = x^{1+J_0(x)}\frac{1+J_0(x)}{x} - J_1(x)\log(x)x^{1+J_0(x)} \tag{4}$$

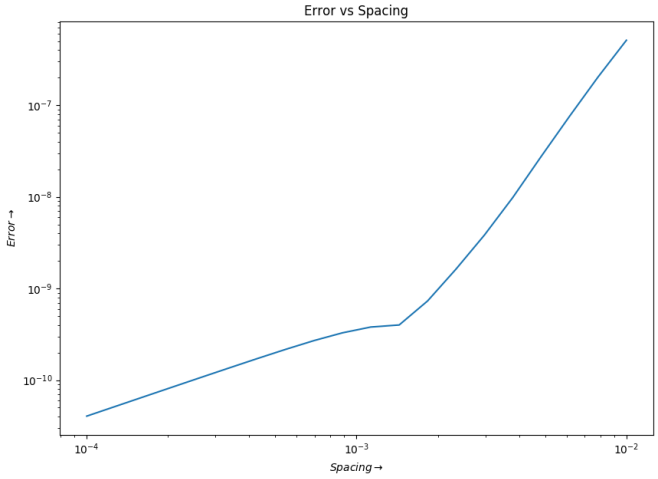
$$\frac{\mathrm{d}f_2(x)}{\mathrm{d}x} = (200x - 300x^2 - 1)(2\sqrt{(1-x)(1+100x^2)}) \tag{5}$$

Since, $f(x) = f_1(x)/f_2(x)$ we get:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \frac{f_2\frac{\mathrm{d}f_1}{\mathrm{d}x} - f_1\frac{\mathrm{d}f_2}{\mathrm{d}x}}{f_2(x)^2} \tag{6}$$

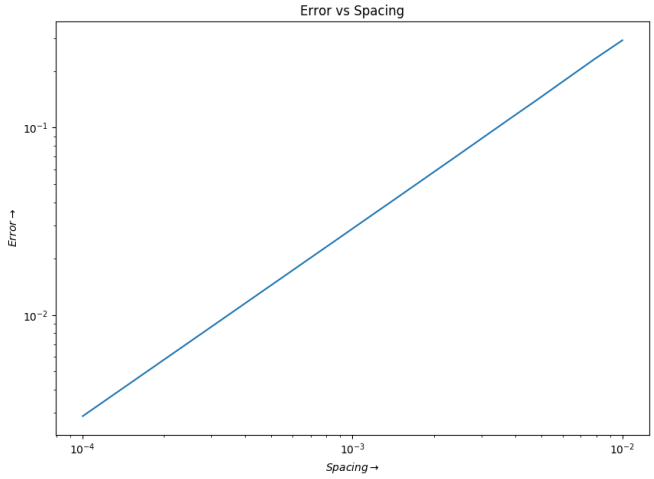
Using this as the boundary condition for the spline we get the following graphs for not a knot respectively:

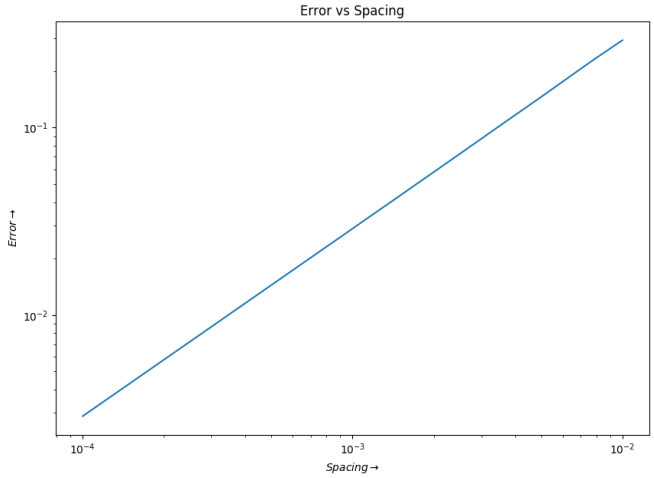




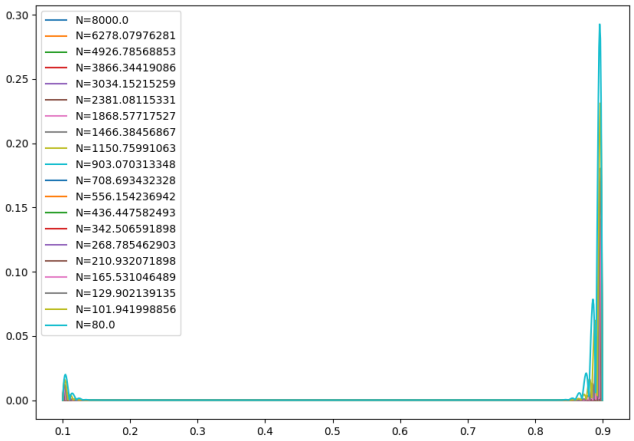
3.6 2.6 Using 100x the actual derivative values

Using 100x the actual derivative values we get the following graphs:





Clearly, the errors have increased by nearly an order of 8. This shows that the magnitude of the derivative at the boundaries plays an important role in obtaining the spline fit coefficients. The points near the edge have the maximum error, as can be seen from the plot below:



3.7 2.7

Non-uniform spacing of spline points can indeed impact the number of points required for an accurate representation. The idea is to concentrate more points in regions where the function being interpolated varies rapidly and fewer points in smoother regions. This way, you allocate computational resources where they are most needed.

The optimal spacing depends on the nature of the function being approximated. If we assume that the error is similar to cubic interpolation, you'd want to have more points in regions with higher curvature or faster changes in the function.

One way to approach this is to consider the second derivative of the function. Higher second derivatives indicate regions of higher curvature and faster changes. So, you might want to distribute more points in areas where the second derivative is larger.

In mathematical terms, if $f''(x)$ represents the second derivative of your function, you could choose spline points x_i such that $|f''(x_i)|$ is larger in regions where you want more points.

However, keep in mind that finding the exact optimal spacing can be a complex task and might involve numerical methods or adaptive algorithms. It's also worth experimenting with different spacings and checking the resulting error to fine-tune the approach for your specific function.

4 Conclusion

- We studied the spline interpolation technique and saw how does the boundary condition affects the error in interpolation.

- We also how the sampling could be done to minimize error.