Assignment #1. Spectral approximations and operations.

In this assignment, you will develop fundamental routines for a spectral methods toolbox in Python/Matlab/Julia that will be useful for later solving differential equations. A partial goal is to develop your ability to perform 'computational thinking' through mastering fundamental concepts and linking theory to practical implementations to be able to explain gaps.

Fourier Spectral Methods (Estimated effort: 2 weeks)

a) Consider the function $u(x) = \frac{1}{2-\cos(x\pi)}$ on $x \in [0, 2]$. Determine the magnitude and asymptotic rate of decay of the continuous Fourier coefficients, i.e. do not resort to approximations and determine instead the exact coefficients. Illustrate convergence behaviour for the truncation error when the function is approximated using a truncated Fourier series. (HINT: symbolic tools may have difficulty to determine the analytic expression, however, it is possible! For example, use complex analysis, method of induction, or other means that you may know). If you get stuck with this exercise, you may still carry on with the next exercises.

b) Carry out a small experiment here. Compute the discrete (approximate) coefficients of the complex Fourier expansion using the Discrete Fourier Transform Method for $u(x)$ for $N = 4, 8, 16, 32, 64$ and compare to the analytical results obtained in a). Notice patterns in the results that you may spot when visualising the results. Discuss your expectations on convergence and properties of trigonometric polynomials based on, respectively, continuous/discrete Fourier coefficients, etc.

c) Show that the discrete trigonometric polynomial can be represented in terms of a nodal expansion and derive the following closed-form expression for the Lagrange polynomials $h_j(x) = \frac{1}{N}\sin\left(\frac{N}{2}(x - x_j)\right)\cot\left(\frac{1}{2}(x - x_j)\right)$ valid for $j = 0, 1, ..., N-1$ for N even on $x \in [0, 2\pi]$ based on a uniform grid $x_j = \frac{2\pi}{N}j,\ j = 0, 1, ..., N-1$. Visualize the Lagrange polynomials on a grid where $N = 6$ and notice how the polynomials take the value of either 1 or 0 at the nodes chosen, i.e. $h_j(x_i) = \delta_{ij}$. Use the expression for the Lagrange polynomials to derive a closed-form expression for a first-order Fourier Differentiation matrix $\mathcal{D}$ on a grid $x \in [0, 2\pi]$ which can be used on a uniform grid to compute first order derivatives.

d) Implement a routine which takes as input N and returns as the output a Fourier differentiation matrix $\mathcal{D}$ for arbitrary N. Verify that the implementation is correct by computing the discrete derivative of $v(x) = \exp(\sin(\pi x))$ on $x \in [0, 2]$ (notice the size of interval !!) and measure the errors for different N. What convergence rate is found for increasing N?

e) Consider the sequence of functions

$$w^0(x) = \begin{cases} -\cos(\pi x) & , -2 \le x < 0 \\ \cos(\pi x) & , 0 \le x \le 2 \end{cases} \quad , \quad w^i(x) = \frac{dw^{i+1}}{dx}, \quad i = 0, 1, 2, ...$$

Compute estimates of the L^2-error by discrete L^2-norms of the discrete first derivative of $w^i(x)$ for $i = 1, 2, 3$. Recall, the definition of the L^2-norm is $||u||^2_{L^2} = \int_\Omega u(x)^2 dx$. Determine $w^i(x)$ via analytic integration such that $w^{i+1}(x) \in C^i([-2, 2])$ and visualize each function $w^i(x)$, $i = 0, 1, 2, 3$. Comment on expected and measured convergence rates for increasing N. (HINT: define the L^2-norm and propose a way to find an approximation to these expression using an appropriate quadrature rule that is accurate)

f) Use Python / Matlab's Fast Fourier Transform (FFT) to compute the derivative of $v(x)$ from d). Make a performance study where timings for discrete differentiation via either a differentiation matrix vs. a FFT is compared. Find the break-even point and confirm expected asymptotic convergence rates for $N \to \infty$. Discuss implications of your results. (HINT: check `fftw` prior to using `fft` in Matlab. Use SciPy in Python)

Polynomial Methods (Estimated effort: 2 weeks)

This part comes with Matlab scripts for computing the Jacobi-Gauss-Lobatto nodes (`JacobiGL.m`) and Jacobi quadrature rules (`JacobiGQ.m`). Two additional Matlab scripts for evaluating Jacobi Polynomials (`JacobiP.m`) and the first derivatives hereof (GradJacobiP.m) will be needed.

h) Implement a Python/Matlab routine in a script with the header below. The routine will be useful and should be based on evaluating Jacobi polynomials via a three-term recurrence at a set of collocation points $\mathbf{x}$ in order to return $P_n^{(\alpha,\beta)}(\mathbf{x})$. Verify your implementation and visualize the first six polynomials of respectively, Chebyshev and Legendre polynomials.
[P] = JacobiP(x,alpha,beta,n)

i) Now, carry out some numerical experiments. Compute the first 200 discrete coefficients of the polynomial expansion for $u(x)$ defined in exercise a) using a Discrete Transform Method based on Legendre polynomial basis functions for $N = 10, 40, 80, 100, 200$ quadrature nodes. Explain and comment on the decay of computed coefficients, errors and explain noticeable patterns in the coefficients of the expansions.

j) Construct the generalized Vandermonde matrix $\mathcal{V}$ that enables transformation between modal and nodal coefficients for a polynomial expansion based on Legendre polynomials. Use this transformation matrix to evaluate each of the interpolating Lagrange polynomial based on a Legendre Gauss Lobatto grid with 6 nodes. Visualize the Lagrange polynomials in a single figure with 100 nodes on a uniform grid for $x \in [-1, 1]$. This can be done by constructing a version of Vandermonde matrix that help do the interpolation. Can you see how? Also, try and approximate a function $v(x) = \sin(\pi x)$ for $x \in [-1, 1]$ using a polynomial basis and do a convergence test. Is it possible to extrapolate the polynomial epresentation to outside this domain? Use a vandermonde matrix to do this and discuss the results.

k) Implement a Python/Matlab routine in a script with the header below. The routine will be useful for computing first derivatives of Jacobi polynomials. Construct the generalized Vandermonde matrix $\mathcal{V}_x$ based on Legendre polynomials as in j). This matrix enables transformation between modal and nodal coefficients for a polynomial expansion for the first derivative of a function. Use $\mathcal{V}_x$ and $\mathcal{V}$ to construct the first order differentiation matrix based on Legendre Gauss Lobatto nodes with N nodes. Then, test convergence for $N \to \infty$ for the discrete derivative of $v(x)$ from e) and use the generalized Vandermonde matrix $\mathcal{V}$ to compute the L^2-norm via matrix-based integration (quadrature) using $\mathcal{V}$. Discuss expectations for convergence and present results in a plot with comments.
[P] = GradJacobiP(x,alpha,beta,n)

l) Construct a mass matrix $M = (VV^T)^{-1}$ from a Vandermonde matrix that make it possible to do integration on any interval $x \in [a, b]$. Use the mass matrix to compute the L^2-norm of the functions $u(x) = 1$ and $u(x) = \sin(x)$ when $a = 0$ and $b = 2$ and compare with the analytical result. When is such matrix-based integration exact? (HINT: how is the matrix derived?)

Communicate your results in a written report. Make sure to describe and comment on important details and findings. Include sufficiently details for the reader to both reproduce and understand the reported results and conclusions.

Enjoy! Allan P. Engsig-Karup

Deadline for this assignment is Sunday, 29th Sep 2024, 23:59.
Hand in online.