Math 567 **Homework 1** **Due 11-Sept-2020**
Grady Wright

1. **(Order of accuracy)**. The following three tables of errors were produced from different finite difference approximations to $u'(x)$, for some function $u(x)$. The first column of each table shows the size h used in the difference approximation, and the second column is the absolute error, $e(h)$, in that approximation, as a function of h.

   Assume that the error can be modeled as

$$e(h) \approx Ch^p,$$

   where p is an integer. Approximate the order of accuracy p for each sequence. You can find the data for these sequences in the file `errors.txt` on the Blackboard website, under the homework 1 link.

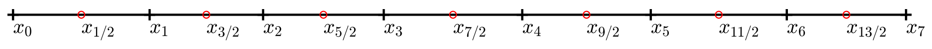
h	err(h)
7.8125000e-03	2.0844e-02
3.9062500e-03	1.1118e-02
1.9531250e-03	5.3455e-03
9.7656250e-04	2.7049e-03
4.8828125e-04	1.3469e-03

h	err(h)
7.8125000e-03	1.9059e-03
3.9062500e-03	4.3086e-04
1.9531250e-03	1.0318e-04
9.7656250e-04	2.6007e-05
4.8828125e-04	6.5716e-06

h	err(h)
1.00000e+00	1.3829e-02
5.00000e-01	1.8805e-03
2.50000e-01	1.3742e-04
1.25000e-01	8.9170e-06
6.25000e-02	5.6252e-07
3.12500e-02	3.5239e-08
1.56250e-02	2.2037e-09
7.81250e-03	1.3775e-10
3.90625e-03	8.6098e-12

2. **(Finite-volume scheme)** An alternative way to approximate the second derivative than those discussed in the class and Chapter 1 of the book is to use the Divergence Theorem (or, in 1D, the Fundamental Theorem of Calculus). The technique is fundamental to finite-volume methods where cell-centered grids are used.

   One convention for a cell-centered grid is to let integer indices indicate the cell boundaries and half-integer indices indicate the center of the cell, where the function (or it's average value) is given. This is illustrated in the figure below, where the circle at $x_{j+1/2}$ is the center of the cell $[x_j, x_{j+1}]$, i.e. $x_{j+1/2} = \frac{x_j + x_{j+1}}{2}$:



The second derivative of some function u at $x_{j+1/2}$ can be approximated as

$$u''(x_{j+1/2}) \approx \frac{1}{x_{j+1}-x_j}\int_{x_j}^{x_{j+1}} u''(x)\,dx = \frac{u'(x_{j+1})-u'(x_j)}{x_{j+1}-x_j}.$$

Here we have approximated u'' at $x_{j+1/2}$ by its *average* value over the cell $[x_j, x_{j+1}]$ and then applied the Divergence Theorem. If we approximate the first derivative as

$$u'(x_j) \approx \frac{u(x_{j+1/2})-u(x_{j-1/2})}{x_{j+1/2}-x_{j-1/2}} \text{ and } u'(x_{j+1}) \approx \frac{u(x_{j+3/2})-u(x_{j+1/2})}{x_{j+3/2}-x_{j+1/2}}$$

we obtain the discretization

$$u''(x_{j+1/2}) \approx \frac{1}{x_{j+1}-x_j}\left[\frac{u(x_{j+3/2})-u(x_{j+1/2})}{x_{j+3/2}-x_{j+1/2}} - \frac{u(x_{j+1/2})-u(x_{j-1/2})}{x_{j+1/2}-x_{j-1/2}}\right]. \tag{1}$$

(a) Show that if we assume a uniform grid, i.e. $x_j = x_0 + jh$, $h = 1/n$, the stencil you obtain for the approximation to $u''(x_{j+1/2})$ at the cell centers $x_{j+1/2}$ is essentially the same second order approximation we obtained in class for $u''(x_j)$ on a vertex centered grid.

(b) Show that on a non-equispaced grid, however, this approximation differs from the technique we discussed in class for approximating the second derivative (i.e. using Lagrange interpolation or method of undetermined coefficients).

You can show this by giving a specific example with non-equally spaced points. For example, let $x_0 = 0$, $x_1 = 1$, $x_2 = 7/4$, and $x_3 = 3$, and compute the stencils for $u''(x_{3/2})$ using (1) and the technique discussed in class (or simply use Fornberg's `weights` algorithm [2]; see course Blackboard page).

Alternatively, you can show that (1) is exact for $u = 1$ and $u = x$, but not $u = x^2$, then argue that the technique we discussed in class has to give the exact value for these functions.

3. **(Compact finite difference formulas)** It is often desirable in certain applications to keep the width (i.e. the number of points) in a finite difference (FD) formula relatively small. This unfortunately directly impacts the order of accuracy we can obtain with the formula. For example, the centered, 3-point FD formulas for the first and second derivative, given, respectively, as

$$u'(\bar{x}) \approx \frac{1}{2h}\left[-u(\bar{x}-h)+u(\bar{x}+h)\right], \tag{2}$$

$$u''(\bar{x}) \approx \frac{1}{h^2}\left[u(\bar{x}-h)-2u(\bar{x})+u(\bar{x}+h)\right], \tag{3}$$

are only second order accurate (i.e. $O(h^2)$). The order of accuracy can often be increased without changing the width of the formula by using an implicit (or compact or Hermite) FD formula, albeit at the cost of solving a sparse and banded linear system [1]. For example, the following are the most popular centered, 3-point implicit FD formulas for the first and second derivative of u at $\bar{x}$:

$$u'(\bar{x}-h)+4u'(\bar{x})+u'(\bar{x}+h) \approx \frac{3}{h}\left[-u(\bar{x}-h)+u(\bar{x}+h)\right], \tag{4}$$

$$u''(\bar{x}-h)+10u''(\bar{x})+u''(\bar{x}+h) \approx \frac{12}{h^2}\left[u(\bar{x}-h)-2u(\bar{x})+u(\bar{x}+h)\right]. \tag{5}$$

(a) Write a code that implements these two implicit formulas for a given function u sampled at $m+2$ equally spaced points x_j over the interval $[a, b]$ (i.e. $x_j = a+jh$, $j = 0, 1, \ldots, m+1$, $h = (b-a)/(m+1)$). In your calculation, you may only use information about $u'(x)$ and $u''(x)$ at the boundary points $x_0 = a$ and $x_{m+1} = b$. This means to get the values of the derivatives at the interior you will need to solve two m-by-m tridiagonal systems of equations—one for the first derivative and one for the second. You can solve this system using any built-in linear solver to the software you are using (e.g. in MATLAB I would use the `spdiags` and 'backslash' functions).

(b) Apply your code to approximate the first and second derivatives of the function $u(x) = x^2 e^{-x}$ over $[0, 1]$. Generate tables and plots showing how the relative two-norm of the errors in the approximations over the interval $[0, 1]$ decreases as the grid spacing h decreases (or N increases). Start with $h = 1/8$ ($m = 7$) and successively reduce h by a factor of 2 until $h = 1/256$ ($m = 255$). The plots should be made on a log-log scale (i.e., logarithmic on by the x and y axes). Determine the orders of accuracy of these two implicit formulas experimentally from the plots and table.

(c) The formal orders of accuracy and exact truncation errors of these implicit FD formulas can be determined in a number of ways. For example, we can use the Taylor series approach discussed in class for the standard FD formulas. Computer Algebra Systems, such as *Mathematica* or *Maple*, make this approach considerably easy. For example, the *Mathematica* code (which can be used directly in WolframAlpha) for deriving the truncation error for the implicit FD formula (4) is given by:

```
s1 = Series[(D[u[x - h] + 4 u[x] + u[x + h], {x, 1}]) - 3/h(-u[x - h] + u[x + h]), {h, 0, 6}]
```

The corresponding *Maple* code is given by:

```
convert(series(diff(u(x-h)+4*u(x)+u(x+h), x)-3/h*(-u(x-h)+u(x+h)), h = 0, 6), diff)
```

Use either of these codes to determine the formal truncation error and order of accuracy of (4) then modify the code and do the same for (5). Do your numerical results from part (a) match theoretical results for the order of accuracy? Explain.

If you have ever studied interpolation with splines then you may recognize the implicit FD formula for the first derivative. It is the same as the one that arises when working out the conditions on continuity of a cubic spline with equally spaced knots. Thus the order of accuracy for computing the derivative with a cubic spline interpolant *at the nodes* x_j is four (assuming the boundaries are handled appropriately), this is one order higher than one might expect based on the accuracy of the interpolant and is sometimes called *super-convergence*.

4. **(Increasing the FD stencil width)** As discussed in class, Fornberg's algorithm [2] can be used to rapidly generate finite difference (FD) weights of any order for arbitrarily spaced points in one dimension. MATLAB , Python, and Julia functions for this algorithm are posted on the course Blackboard page under the name `weights.m`, `weights.py`, and `weights.jl`, respectively (these are similar to the books MATLAB function `fdcoeffF`). You can download any of these functions to perform the tasks listed below.

(a) For the following three node sets (which are known as equispaced, Chebyshev, and Legendre, respectively), use Fornberg's algorithm to compute the FD weights for approximating the first derivative at $x = 0$ and $x = -1 + 3/14$:

(i) $x_j = -1 + \dfrac{2j}{14}$, $j = 0, 1, \ldots, 14$

(ii) $x_j = -\cos\left(\dfrac{j\pi}{14}\right)$, $j = 0, 1, \ldots, 14$

(iii)

(iii)	-0.987992518020485	-0.394151347077563	0.570972172608539
	-0.937273392400706	-0.201194093997435	0.724417731360170
	-0.848206583410427	0	0.848206583410427
	-0.724417731360170	0.201194093997435	0.937273392400706
	-0.570972172608539	0.394151347077563	0.987992518020485.

Ordering the Legendre points in (iii) from smallest (-0.987...) to largest (0.987...) will make things easier in what follows.

You should use all the nodes in the approximations, which means there should be 15 weights for each node set. Plot each set of weights for the $x = 0$ approximation on the same graph (use the index j as the horizontal or independent coordinate). Repeat this for the $x = -1 + 3/14$ approximation. Comment on how the weights differ between node sets and between approximation points.

(b) Use the weights from each of the three node sets from part (a) to approximate the derivative of $u(x) = e^{-\cos(2(x-1/5))}$ at $x = 0$ and $x = -1 + 3/14$. Report the error in these approximations in a nice table and comment on how the errors differ. Which set of points would you prefer to use to approximate derivatives?

Note that in the case of equally spaced and Chebyshev points, analytical expressions can be worked out for the FD weights (see the next problem for an example).

5. **(Trigonometric interpolation)** The trigonometric interpolant at N equally spaced nodes over $[0, 2\pi)$ (i.e. $x_j = \frac{2\pi j}{N}$, $j = 0, 1, \ldots, N-1$) to a function u can be written in *Lagrange* form as

$$p_N(x) = \sum_{j=0}^{N-1} S_N(x - x_j) u(x_j),$$

where

$$S_N(t) = \begin{cases} \dfrac{1}{N} \sin\left(\dfrac{N}{2} t\right) \cot\left(\dfrac{t}{2}\right), & \text{if } N \text{ is even,} \\ \dfrac{1}{N} \sin\left(\dfrac{N}{2} t\right) \csc\left(\dfrac{t}{2}\right), & \text{if } N \text{ is odd.} \end{cases}$$

The function $S_N(t)$ is called the 'periodic sinc function'.

(a) Verify that $p_N(x)$ actually interpolates the data, i.e. show that $p_N(x_k) = u(x_k)$, for $k = 0, 1, \ldots, N-1$. This can be done by simply showing $S_N(x_j - x_k) = 0$ when $k \neq j$ and $S_N(0) = 1$.

(b) For the case of N being even show that

$$S_N'(-x_k) = \begin{cases} 0, & \text{if } k = 0, \\ \dfrac{1}{2}(-1)^k \cot(k\pi/N), & \text{if } k \neq 0. \end{cases}$$

These are the Fourier (or trigonometric) weights for approximating the first derivative of u at $x = 0$ from samples of u at x_k. Since $S_N(t)$ is periodic, they can be shifted appropriately to approximate the derivative of u at any x_k.

References

[1] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.*, 103:16–42, 1992.

[2] B. Fornberg. Calculations of weights in finite difference formulas. *SIAM Rev.*, 40:685–691, 1998.