Math 567 **Homework 5** **Due Nov. 15, 2020**
Grady Wright

1. *Conjugate gradients*

   (a) Implement the conjugate gradient (CG) method given on page 87 of the textbook. Replace the termination criteria "if $\|r_k\|$ is less than some tolerance then stop" with "if $\|r_k\|_2 < tol\|f\|_2$ then stop", where *tol* is a user input parameter. Your code should be written as a function that can called with any symmetric positive definite matrix A, right hand side $\mathbf{b}$, and tolerance. Turn in a listing of your code.

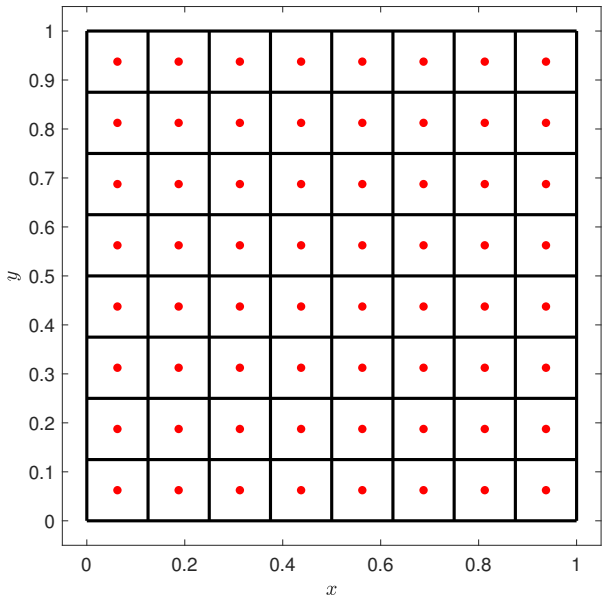
   (b) Use your code from part (a) to (approximately) solve the model problem from problem 2 of homework 4,

$$\begin{aligned} \nabla^2 u(x,y) &= 10\pi^2 e^{\sin(2\pi(x+2y))}(-2\sin(2\pi(x+2y)) + \cos(4\pi(x+2y)) + 1) \\ u(x,y) &= e^{\sin(2\pi(x+2y))} \quad \text{for } (x,y) \in \partial\Omega, \end{aligned} \tag{1}$$

   using the second-order accurate finite difference method. Use $m = 2^7 - 1$ and $tol = 10^{-8}$ in your CG code and plot the error in the solution you obtain to the model problem. Report the number of iterations your code takes to converge, the time it takes to run, and the norm of the final residual when the solution converged. Compare the timing of your code to the SOR method from the previous homework (run with the same tolerance).

   (c) Extra credit (5 points): Implement a specific CG algorithm for the model problem that does not involve constructing the A matrix.

2. *Cell-centered Poisson solver* So far we have focused on so called vertex-centered finite difference methods for solving boundary value problems like the Poisson equation. For the unit square $\Omega = [0,1] \times [0,1]$, these methods use the grid $(x_j, y_k) = (jh, kh)$, $j,k = 0,1,\ldots,m+1$, with $h = 1/(m+1)$. Another strategy that is often used (especially in finite volume methods) is to use a *cell-centered* grid. For these methods the grid over the unit square is $(x_j, y_k) = ((j-1/2)h, (k-1/2)h)$, $j,k = 1,\ldots,m$, with $h = 1/m$. The figure below illustrates this grid for the case of $m = 8$. The solid red dots mark the grid locations where the unknowns u_{jk} are to be determined.



One big difference between the vertex-centered and cell-centered approaches is that for the latter there are no grid points on the boundary of the domain, which means Dirichlet boundary conditions have to be handled differently.

Consider the Poisson problem with *mixed boundary conditions*

$$\begin{aligned} \nabla^2 u = -34\pi^2 \cos(5\pi x)\sin(3\pi y), \qquad & (x,y) \in \Omega = [0,1]\times[0,1] \\ u(x,0) = 0 \qquad & 0 \le x \le 1, \\ u(x,1) = 0 \qquad & 0 \le x \le 1, \\ \frac{\partial}{\partial x} u(x,y)\big|_{x=0} = 0 \qquad & 0 < y < 1, \\ \frac{\partial}{\partial x} u(x,y)\big|_{x=1} = 0 \qquad & 0 < y < 1. \end{aligned} \tag{2}$$

In this problem you will develop a second-order accurate finite difference method for solving this problem. Note that the exact solution to this Poisson problem is $u(x,y) = \cos(5\pi x)\sin(3\pi y)$.

(a) Suppose you want to solve $u''(y) = f(y)$, $0 \le y \le 1$, with $u(0) = \alpha_1$, $u(1) = \alpha_2$ on a cell-centered grid $y_k = (k-1/2)h$, $k = 1,\ldots,m$, $h = 1/m$. Use the *fictitious point* method (see homework 3) to develop a second-order accurate approximation to this problem. State the discretization of the boundary value problem at the interior points y_k, $k = 2,\ldots,m-1$, and at the boundary points y_1 and y_m. You don't need to write any code here.

(b) Suppose you want to solve $u''(x) = f(x)$, $0 \le x \le 1$, with $u'(0) = \beta_1$, $u'(1) = \beta_2$ on a cell-centered grid $x_k = (k-1/2)h$, $k = 1,\ldots,m$, $h = 1/m$. Similar to part (a), develop a second-order accurate approximation to this problem using the *fictitious point* method. Again, state the discretization of the boundary value problem at the interior points x_k, $k = 2,\ldots,m-1$, and at the boundary points x_1 and x_m. You don't need to write any code here.

(c) Using the results from part (a) and (b), develop a second-order accurate approximation for solving the Poisson problem (2). Write a code to solve this problem and demonstrate that your code produces a second-order accurate approximation by some means. Produce a plot of the error in your solution for $m = 128$. You can use whatever method you want to solve the linear system that arises for solving this boundary value problem (e.g. SOR, Conjugate Gradient, Direct sparse solver, FFT based solver, block Thomas, Bartels-Stewart, etc.)

3. *Linear multi-step methods* The following are six suggestions for linear multistep methods for solving the initial value problem $y' = f(t,y(t))$, $a \le t \le b$, $y(a) = y_0$:

(a) $y^{n+1} = \frac{1}{2}\left[y^n + y^{n-1}\right] + 2kf^n$

(b) $y^{n+1} = y^n$

(c) $y^{n+1} = y^{n-3} + \frac{4}{3}k\left[f^n + f^{n-1} + f^{n-2}\right]$

(d) $y^{n+1} = y^{n-1} + \frac{1}{3}k\left[7f^n - 2f^{n-1} + f^{n-2}\right]$

(e) $y^{n+1} = \frac{8}{19}\left[y^n - y^{n-2}\right] + y^{n-3} + \frac{6}{19}k\left[f^{n+1} + 4f^n + 4f^{n-2} + f^{n-3}\right]$

(f) $y^{n+1} = -y^n + y^{n-1} + y^{n-2} + 2h\left[f^n + f^{n-1}\right]$

Here k is the time-step, i.e. $k = (b-a)/N$ for some positive integer N.

The (incomplete) table below summarizes their properties. Complete the missing entries of this table (you need not supply any derivations), draw the corresponding stencil, state the generating polynomials $\rho(w)$ and $\sigma(w)$, and state the number of steps r.

case	char. eq	roots	stability	accuracy	consistency	leading error term	convergence
(a)	$w^2 - \frac{1}{2}w - \frac{1}{2} = 0$	$1, -\frac{1}{2}$	Yes	0		$-\frac{1}{2}kf'(\xi)$	
(b)				0	No		No
(c)					Yes		Yes
(d)				3		$\frac{1}{3}k^4 f^{(4)}(\xi)$	Yes
(e)	$w^4 - \frac{8}{19}w^3 + \frac{8}{19}w - 1 = 0$	$\pm 1, \frac{4}{19} \pm \frac{\sqrt{345}}{19}i$		6		$-\frac{6}{665}k^7 f^{(7)}(\xi)$	
(f)				2	Yes	$\frac{2}{3}k^3 f^{(3)}(\xi)$	

4. *Predictor-Corrector method*

   (a) Implement a function in the language of your choosing for numerically solving the general *vector-valued* IVP

   $$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad a \le t \le b, \quad \mathbf{y}(a) = \mathbf{y}_0 \ \ (\mathbf{y} \in \mathbb{R}^n),$$

   using the *fourth-order*, predictor-corrector method of AB4 (4-step method) and AM4 (3-step method):

   $$\begin{aligned} \mathbf{y}^* &= \mathbf{y}^n + \frac{k}{24}[55\mathbf{f}(t_n, \mathbf{y}_j) - 59\mathbf{f}(t_{n-1}, \mathbf{y}^{n-1}) + 37\mathbf{f}(t_{n-2}, \mathbf{y}^{n-2}) - 9\mathbf{f}(t_{n-3}, \mathbf{y}^{n-3})] \ \text{(AB4 predictor)} \\ \mathbf{y}^{n+1} &= \mathbf{y}^n + \frac{k}{24}[9\mathbf{f}(t_{n+1}, \mathbf{y}^*) + 19\mathbf{f}(t_n, \mathbf{y}^n) - 5\mathbf{f}(t_{n-1}, \mathbf{y}^{n-1}) + \mathbf{f}(t_{n-2}, \mathbf{y}^{n-2})] \ \text{(AM4 corrector)} \\ & \quad j = 3, 4, \ldots, N-1 \ . \end{aligned}$$

   Your function should be called `abm4`, and take as input a function that represents the vector-valued function $\mathbf{f}(t, \mathbf{y})$, a, b, the initial condition $\mathbf{y}_0$, and the number of steps to take N. The output of your function should be a vector `t` containing all of the time-steps, and a matrix `y` containing the numerical solution of all the components of the system $\mathbf{y}$ at each time-step.

   You will need to obtain the three values $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$ to get your `abm4` function started. These values should be obtained with a fourth-order accurate method to keep the scheme consistent. The easiest approach is to use the standard fourth-order Runge-Kutta method (see Example 5.13 on p. 126 of the book).

   Turn in a printed listing and e-mail me your program(s).

   > **Note:** Your function must be entirely general. This means that I should be able to create my own function for any $f(t, \mathbf{y})$, the values a, b, and $\mathbf{y}_0$, and use your program `abm4` to try and solve the problem. Furthermore, your program should minimize the number of times the function $f(t, \mathbf{y})$ needs to be called.

   (b) Consider the initial value problem

   $$y' = 1 + \frac{y}{t} + \left(\frac{y}{t}\right)^2, \quad 1 \le t \le 3, \quad y(1) = 0 \ ,$$

   where the exact solution is $y(t) = t\tan(\log(t))$. Use your abm4 method from part (a) to numerically solve this IVP for $1 \le t \le 3$ for the different N values [50,100,200,400]. Compute the error in the numerical solution at $t = 3$ and make either a table or a graph clearly showing fourth-order convergence of the solution. On a single graph, plot the solution for $N = 100$ over the interval $1 \le t \le 3$.