AMATH 585 Homework 5

Cade Ballew

February 23, 2022

1 Problem 1

Considering the problem

$$u_{xx} + u_{yy} = f(x, y), \quad 0 < x, y < 1$$

$$u(x, 0) = u(x, 1) = u(0, y) = u(1, y) = 1.$$

with $f(x, y) = x^2 + y^2$, we use the following MATLAB code to solve with a 5 point Laplacian stencil and various stepsizes for our square uniform grid. Note that a portion of the code for assembling the matrix comes directly from that which is given on page 68 of the text.

```
func = @(x,y) x.^2+y.^2;

pow = 10; %power of 2 we use as true solution
m = 2^pow-1; h=1/(m+1);

x = linspace(0,1,m+2);
y = linspace(0,1,m+2);
[X,Y] = meshgrid(x,y);
x = x(2:end-1); y = y(2:end-1); %eliminate boundary


a = 1; %boundary condition

utrue = fiveptpois(func,a,x,y);

fprintf(['   h          L2 error          L2 ratio      ' ...
    '   Inf error        Inf ratio\n'])
for i = 2:6
m = 2^i-1; h=1/(m+1);
x = linspace(0,1,m+2);
y = linspace(0,1,m+2);
[X,Y] = meshgrid(x,y);
x = x(2:end-1); y = y(2:end-1); %eliminate boundary
```

```
u = fiveptpois(func,a,x,y);

utrue_adj = utrue(2^(pow-i):2^(pow-i):end,2^(pow-i):2^(pow-i):end);
err = u-utrue_adj;
err_L2 = h*norm(err(:),2);
err_inf = norm(err(:),'inf');
constvec = err_inf/h^2;
constvec2 = err_L2/h^2;
fprintf('%1.4e %.10e %.10e %.10e %.10e\n',h,err_L2, constvec2, err_inf, constvec)
end



function [u,uBC] = fiveptpois(func,a,x,y)
m = length(x); h = 1/(m+1);
%From pg. 68 of course text
I = eye(m);
e = ones(m,1);
T = spdiags([e -4*e e],[-1 0 1],m,m);
S = spdiags([e e],[-1 1],m,m);
A = (kron(I,T) + kron(S,I))/h^2;

f = func(x',y);
%implement boundary condition
f(1,:) = f(1,:) - a/h^2;
f(m,:) = f(m,:) - a/h^2;
f(:,1) = f(:,1) - a/h^2;
f(:,m) = f(:,m) - a/h^2;

f = f(:);

u = A\f;
u = reshape(u,m,m);
uBC = a*ones(m+2);
uBC(2:end-1,2:end-1)=u; %add back boundary condition

end
```

Note that we incorporate our boundary condition by observing that points bordering the boundary have a $1/h^2$ term in their stencil that is not accounted for (corner points have two of these terms), so we scale these by our boundary condition (1 here so no scaling is actually done) and subtract from their respective entries in the RHS vector f.
For our "true solution," we consider $m = 2^{10} - 1$ interior nodes as a fine grid to compare our approximate solutions to. The following table presents our computed errors for various values of h.

h	$\|\|u-\hat{u}\|\|_{L_2}$	$\|\|u-\hat{u}\|\|_{L_2}/h^2$	$\|\|u-\hat{u}\|\|_{\infty}$	$\|\|u-\hat{u}\|\|_{\infty}/h^2$
2.5000e-01	1.9909307757e-03	3.1854892412e-02	3.3587979110e-03	5.3740766576e-02
1.2500e-01	5.4215474511e-04	3.4697903687e-02	9.6067228767e-04	6.1483026411e-02
6.2500e-02	1.3882728012e-04	3.5539783712e-02	2.4409110291e-04	6.2487322346e-02
3.1250e-02	3.4926843229e-05	3.5765087466e-02	6.1234296804e-05	6.2703919927e-02
1.5625e-02	8.7238998765e-06	3.5733093894e-02	1.5279784326e-05	6.2585996598e-02

We can see that our error in both the L_2 and infinity norm appears roughly constant when divided by h^2, meaning that our convergence appears to be second order as anticipated.

2 Problem 2

Now, we solve the same BVP from problem 1 using the 9 point Laplacian with correction term via the following MATLAB code.

```
func = @(x,y) x.^2+y.^2;
lpfunc = @(x,y) 4*ones(length(x),length(y));

pow = 10; %power of 2 we use as true solution
m = 2^pow-1; h=1/(m+1);

x = linspace(0,1,m+2);
y = linspace(0,1,m+2);
[X,Y] = meshgrid(x,y);
x = x(2:end-1); y = y(2:end-1); %eliminate boundary


a = 1; %boundary condition

utrue = nineptpois(func,lpfunc,a,x,y);

fprintf(['   h          L2 error          L2 ratio      ' ...
    '   Inf error        Inf ratio\n'])
for i = 2:6
m = 2^i-1; h=1/(m+1);
x = linspace(0,1,m+2);
y = linspace(0,1,m+2);
[X,Y] = meshgrid(x,y);
x = x(2:end-1); y = y(2:end-1); %eliminate boundary
u = nineptpois(func,lpfunc,a,x,y);

utrue_adj = utrue(2^(pow-i):2^(pow-i):end,2^(pow-i):2^(pow-i):end);
err = u-utrue_adj;
err_L2 = h*norm(err(:),2);
```

```
err_inf = norm(err(:),'inf');
constvec = err_inf/h^2;
constvec2 = err_L2/h^3;
fprintf('%1.4e %.10e %.10e %.10e %.10e\n',h,err_L2, constvec2, err_inf, constvec)
end


function [u,uBC] = nineptpois(func,lpfunc,a,x,y)
m = length(x); h = 1/(m+1);
I = eye(m);
e = ones(m,1);
T = spdiags([4*e -20*e 4*e],[-1 0 1],m,m);
S1 = spdiags([e e],[-1 1],m,m);
S2 = spdiags([e 4*e e],[-1 0 1],m,m);
A = (kron(I,T) + kron(S1,S2))/(6*h^2);

f = func(x',y);
%implement boundary condition
f(1,:) = f(1,:) - a/h^2;
f(m,:) = f(m,:) - a/h^2;
f(:,1) = f(:,1) - a/h^2;
f(:,m) = f(:,m) - a/h^2;
f(1,1) = f(1,1) + a/(6*h^2);
f(1,m) = f(1,m) + a/(6*h^2);
f(m,1) = f(m,1) + a/(6*h^2);
f(m,m) = f(m,m) + a/(6*h^2);

lpf = lpfunc(x',y);
f = f + lpf*h^2/12;

f = f(:);

u = A\f;
u = reshape(u,m,m);
uBC = a*ones(m+2);
uBC(2:end-1,2:end-1)=u; %add back boundary condition

end
```

Note that we now incorporate our boundary condition by noting that points bordering the boundary have terms amounting to $1/(6h^2)+4/(6h^2)+1/(6h^2) = 1/h^2$ which are not accounted for with the exception of the corners which have $1/(6h^2) + 4/(6h^2) + 1/(6h^2) + 4/(6h^2) + 1/(6h^2) = 11/(6h^2)$. As before, we simply subtract these from their respective elements in the RHS vector f. To incorporate our correction term, we note that $\Delta f = 4$, so the correction term can be incorporated as in (3.19) in the text by simply adding $4h^2/12$ to the

RHS vector f in all components.
Again, for our "true solution," we consider $m = 2^{10} - 1$ interior nodes as a fine grid to compare our approximate solutions to. The following table presents our computed errors for various values of h.

h	$\|\|u - \hat{u}\|\|_{L_2}$	$\|\|u - \hat{u}\|\|_{L_2}/h^3$	$\|\|u - \hat{u}\|\|_\infty$	$\|\|u - \hat{u}\|\|_\infty/h^2$
2.5000e-01	7.3824620069e-05	4.7247756844e-03	1.9891575673e-04	3.1826521077e-03
1.2500e-01	8.5371004530e-06	4.3709954319e-03	4.6877938937e-05	3.0001880920e-03
6.2500e-02	1.0409757941e-06	4.2638368525e-03	1.1671052788e-05	2.9877895137e-03
3.1250e-02	1.2925745869e-07	4.2355084062e-03	2.9169807327e-06	2.9869882703e-03
1.5625e-02	1.6129012916e-08	4.2281239618e-03	7.2922359340e-07	2.9868998386e-03

We can see that the error in the L_2 norm is roughly constant when divided by h^3 and the error in the infinity norm is roughly constant when divided by h^2, so our error in the L_2 norm is roughly 3rd order while that in the infinity norm is roughly 2nd order.

Observe that we do not achieve the promised 4th order accuracy from adding this correction term; our square domain, specifically the corners, is the culprit here. On the boundary, an exact solution u to our BVP must be constant and equal to 1. This means that at a corner point, u is constant in both the x and y directions, so $u_{xx} = u_{yy} = 0$ and $\Delta u = 0$. However, at the corner points aside from the origin, $f(x, y) \neq 0$. Because f is continuous and $\Delta u = f$ arbitrarily close to the boundary, we must have that $\Delta u \neq 0$ arbitrarily close to our 3 nonzero corner points but $\Delta u = 0$ at said points. This means that Δu is discontinuous at these 3 points! The true correction term is the Laplacian of this Laplacian, but this cannot be well-defined at the corner points since Δu is discontinuous there. Thus, the correction term we add does little to improve the accuracy near these points and explains why we observe 3rd order accuracy in the L_2 norm and 2nd order in the infinity norm (2nd order errors at the corners appear more prominently in the infinity norm when we have 4th order error elsewhere). Investigating our error matrix in MATLAB, we observe that the largest errors do appear to be near these offending corners.

3 Problem 3

If we wish to adapt our method for solving

$$\triangle u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega,$$

with *homogeneous* Dirichlet boundary conditions to solve

$$\triangle u = f \quad \text{in } \Omega, \quad u = g \quad \text{on } \partial\Omega,$$

where g is some given function, we first relabel our gridpoints to distinguish between interior and boundary points. If we let the number of interior points be given by m_1 and the number of boundary points be given by $m_2 = m - m_1$,

where m is the total number of gridpoints, we say that the indices of the interior points are $1, \ldots, m_1$ and those of the boundary points are $m_1+1, \ldots, m$. Assume that we have a basis of bilinear grid functions φ_i, $i = 1, \ldots, m$ associated with each gridpoint (including those on the boundary) such that $\varphi_i(x_j) = \delta_{i,j}$ where δ is the Kronecker delta (we can just normalize our basis if this is not already true). Then, represent $\hat{u}$ as

$$\hat{u}(x,y) = \sum_{i=1}^{m} c_i \varphi_i(x,y) = \sum_{i=1}^{m_1} c_i \varphi_i(x,y) + \sum_{i=m_1+1}^{m} c_i \varphi_i(x,y)$$

and note that in order to satisfy the boundary conditions, we need that for $j = m_1 + 1, \ldots, m$

$$g(X_j) = \hat{u}(X) = \sum_{i=1}^{m} c_i \varphi_i(X) = c_j$$

where X_j denotes the gridpoint (in x and y) corresponding to index j. Now, we wish to choose $c_1, \ldots, c_{m_1}$ such that

$$\iint_\Omega \Delta \hat{u} \varphi_j dxdy = \iint_\Omega f \varphi_j dxdy$$

for $j = 1, \ldots, m_1$ (i.e. our test space is unchanged from the homogenous case). Applying Green's theorem to the LHS,

$$-\iint_\Omega (\hat{u}_x \varphi_{j_x} + \hat{u}_y \varphi_{j_y}) dxdy + \int_{\partial\Omega} \hat{u}_n \varphi_j d\gamma = \iint_\Omega f \varphi_j dxdy$$

for $j = 1, \ldots, m_1$. Now, note that because we are only considering interior grid functions φ_i (which are zero on the boundary), the second integral on the LHS is zero. Now, we substitute in our expansion for $\hat{u}$ to get

$$-\iint_\Omega \left(\left(\sum_{i=1}^{m_1} c_i \varphi_{i_x} + \sum_{i=m_1+1}^{m} G(x_i) \varphi_{i_x} \right) \varphi_{j_x} + \left(\sum_{i=1}^{m_1} c_i \varphi_{i_y} + \sum_{i=m_1+1}^{m} G(x_i) \varphi_{i_y} \right) \varphi_{j_y} \right) dxdy$$
$$= \iint_\Omega f \varphi_j dxdy$$

for $i = 1, \ldots, m_1$. Now, we can simply break up our integral and move the knowns to the RHS and to get

$$-\sum_{i=1}^{m_1} c_i \iint_\Omega \left(\varphi_{i_x} \varphi_{j_x} + \varphi_{i_y} \varphi_{j_y} \right) dxdy = \iint_\Omega \left(f \varphi_j + \sum_{i=m_1+1}^{m} G(x_i) \left(\varphi_{i_x} \varphi_{j_x} + \varphi_{i_y} \varphi_{j_y} \right) \right) dxdy.$$

for $j = 1, \ldots, m_1$. To write this as a system of equations, we can leave our matrix A unaltered, letting

$$A_{i,j} = -\iint_\Omega \left(\varphi_{i_x} \varphi_{j_x} + \varphi_{i_y} \varphi_{j_y} \right) dxdy$$

and only alter our RHS vector f which now has entries

$$f_j = \iint_\Omega \left(f\varphi_j + \sum_{k=m_1+1}^{m} G(x_i) \left(\varphi_{k_x} \varphi_{j_x} + \varphi_{k_y} \varphi_{j_y} \right) \right) dxdy$$

for $i, j = 1, \ldots, m_1$. Then, if c is the vector of unknown coefficients, we can solve $Ac = f$ to find our approximate solution $\hat{u}$.