

MATH 612

Numerical Methods for Partial Differential Equations

Spring Term 2004

Instructor: Georg Dolzmann

Solutions to Homework Set 10

Problem 1: The five-point formula leads to a discretization of second order for Laplace's equation. The goal of this problem is to find a formula of fourth order for the discretization of this equation.

(a) Identify the coefficients in the approximation of the second derivatives of a smooth function given by

$$u''(x) \sim D_h^2 u(x) = \frac{1}{h^2} \sum_{\nu=-2}^2 c_\nu u(x + \nu h)$$

in such a way that the approximation is (at least) of order four, i.e.,

$$\sum_{\nu=-2}^2 c_\nu u(x + \nu h) = u''(x) + \mathcal{O}(h^4).$$

(b) Use your scheme in (a) to write down a generalized star for the discretization of Laplace's equation. Why is this star not practical?

Solution: (a) We use Taylor series expansions to find

$$\begin{aligned} u(x - 2h) &= u(x) - 2hu'(x) + 2h^2u''(x) - \frac{4}{3}h^3u^{(3)}(x) + \frac{2}{3}h^4u^{(4)}(x) + \dots, \\ u(x - h) &= u(x) - hu'(x) + \frac{1}{2}h^2u''(x) - \frac{1}{6}h^3u^{(3)}(x) + \frac{1}{24}h^4u^{(4)}(x) + \dots, \\ u(x) &= u(x), \\ u(x + h) &= u(x) + hu'(x) + \frac{1}{2}h^2u''(x) + \frac{1}{6}h^3u^{(3)}(x) + \frac{1}{24}h^4u^{(4)}(x) + \dots, \\ u(x + 2h) &= u(x) + 2hu'(x) + 2h^2u''(x) + \frac{4}{3}h^3u^{(3)}(x) + \frac{2}{3}h^4u^{(4)}(x) + \dots \end{aligned}$$

This leads to the following linear system:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 2 & \frac{1}{2} & 0 & \frac{1}{2} & 2 \\ -\frac{4}{3} & -\frac{1}{6} & 0 & \frac{1}{6} & \frac{4}{3} \\ \frac{2}{3} & \frac{1}{24} & 0 & \frac{1}{24} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} c_{-2} \\ c_{-1} \\ c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

The fourth and the fifth equation imply immediately the symmetry $c_{-2} = c_2$ and $c_{-1} = c_1$. This allows us to reduce the system to

$$\begin{pmatrix} 2 & 2 & 1 \\ 4 & 1 & 0 \\ \frac{4}{3} & \frac{1}{12} & 0 \end{pmatrix} \begin{pmatrix} c_{-2} \\ c_{-1} \\ c_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

The solution of this system is given by

$$c_{-2} = c_2 = -\frac{1}{12}, \quad c_{-1} = c_1 = \frac{4}{3}, \quad c_0 = -\frac{5}{2}.$$

We conclude that

$$u''(x) \sim \frac{1}{12h^2} (-u(x-2h) + 16u(x-h) - 30u(x) + 16u(x+h) - u(x+2h)).$$

(b) The star would be given by

$$\frac{1}{12h^2} \begin{bmatrix} & & -1 & & \\ & & 16 & & \\ -1 & 16 & -60 & 16 & -1 \\ & & 16 & & \\ & & -1 & & \end{bmatrix}.$$

The difficulty with this star is that it needs four neighboring points in each direction. If we write down this star at the point (h, h) then we need two boundary values and the values of two nodes that are not contained in $\Omega \cup \Gamma$.

Problem 2: In this problem we experiment with the two ways to solve the Neumann problem for Laplace's equation on the unit square with the reduced system, that is, the versions with and without the corrections for the mismatch in the discrete compatibility condition. We want to solve

$$-\Delta u = f \quad \text{in } \Omega, \quad \frac{\partial u}{\partial n} = \phi \quad \text{on } \partial\Omega$$

with two sets of data,

$$u_1(x, y) = \frac{1}{2\pi^2} \sin(\pi x) \sin(\pi y) + \frac{1}{5\pi^2} \sin(\pi x) \sin(2\pi y)$$

with

$$(1) \quad f_1(x, y) = \sin(\pi x) \sin(\pi y) + \sin(\pi x) \sin(2\pi y),$$

and

$$(2) \quad u_2(x, y) = x^2 + y^2, \quad f_2(x, y) = -4.$$

We choose the Neumann data ϕ from the solution so that the (continuous) compatibility condition is always satisfied.

(a) Write a MATLAB code that solves the Neumann problem with the reduced system with and without the correction for the mismatch in the discrete compatibility condition

$$-h^2 \sum_{x \in \Omega_h} f(x) = h \sum_{x \in \Gamma'_h} \phi(x).$$

(b) Solve the two test examples with $J = 8, 16, 32, 64$, find the L^∞ errors, and determine the experimental rate of convergence. Here we define the L^∞ error as

$$\|u - u_h\|_\infty = \max_{x \in \Omega_h \cup \Gamma'_h} (u(x) - u_h(x)) - \min_{x \in \Omega_h \cup \Gamma'_h} (u(x) - u_h(x)).$$

The reason for this choice is that the solution is only determined up to a constant and therefore we need to factor out this arbitrary constant. Plot the error for $J = 32$ for both schemes. You can either plot the error only on the interior nodes Ω_h or you can define the solution u_h in the corners by interpolation and plot the error on $\Omega_h \cup \Gamma_h$. This can lead to artificial singularities of the error in the corners. Make also a double-logarithmic plot that shows both the error in the scheme without and the scheme with the correction.

Some tips: If you define the boundary conditions, the right-hand side, and the exact solution as `mfiles` or as inline functions, remember to make them “array smart”. Use the commands `@phi` and `feval(phi,x,y)` to pass `phi` as an argument to a subroutine if `phi` is an `mfile`. If you choose a point which you eliminate from the system, make sure that the point lies always in Ω_h and not on Γ_h . Check that the scaling of L_h and of the terms in the right-hand side q_h that correspond to f and ϕ are correct. Use the discretization of the Neumann data find the values of the solution on Γ'_h .

You may want to solve this problem with various choices of data, for example with the following functions,

$$(3) \quad u_3(x, y) = \frac{1}{2\pi^2} \cos(\pi x) \cos(\pi y), \quad f_3(x, y) = \cos(\pi x) \cos(\pi y).$$

The advantage of first working with (3) is that the Neumann conditions are zero and this allows us to test the part of the code that involves only the right-hand side f , the solution of the linear system and the representation of the solution. Also the effect of the discrete mismatch is minimal.

Solution: Here is the code for the solution without the correction:

```
%
% function [u, einf] = nr(J,f,phi,ux,pl)
%
% solution of -Delta u = f on [0,1]^2
%
% with Neumann data using the reduced system
% without correction
% f, phi, ux = data and solution
% pl = plot level, controles the output
%
% J = 1/h = number of points on the axes
% f=f(x,y)= right-hand side of the equation
% phi=phi(x,y)= Neumann data
%
function [u, einf] = nr(J,f,phi,ux,pl)
%
% generate the block structure of the matrix
%
I=eye(J-1);
```

```

T=diag([3;4*ones(J-3,1);3]);
T=T+diag(-ones(J-2,1),1);
T=T+diag(-ones(J-2,1),-1);
%
% define the big system matrix
%
N=(J-1)*(J-1);
L=zeros(N,N);
%
% fill in the block structure
% the first and the last line
%
L(1:J-1,1:2*(J-1))=[T-I,-I];
L(N-(J-1)+1:N,N-2*(J-1)+1:N)=[-I,T-I];
%
% the regular blocks with three entries
%
for i=2:J-2,
    L((i-1)*(J-1)+1:i*(J-1),(i-2)*(J-1)+1:(i+1)*(J-1))=[-I,T,-I];
end
%
% correct scaling of L
%
h=1/J;
L=L/h^2;
if pl>=2,
    figure
    spy(L);
end;
%
% evaluation of RHS b:
% f(x,y)=f(i*h,j*j)
% Neumann data for points close to the boundary
% allow some tolerance in the test on (x,y)
% being close to the boudnary
%
b=zeros(N,1);
for i=1:J-1,
    for j=1:J-1
        x=i*h;
        y=j*h;
        bval=f(x,y);
        if x<1.5*h bval=bval+feval(phi,0,y)/h; end;
        if x>1-1.5*h bval=bval+feval(phi,1,y)/h; end;
        if y<1.5*h bval=bval+feval(phi,x,0)/h; end;
        if y>1-1.5*h bval=bval+feval(phi,x,1)/h; end;
        b((j-1)*(J-1)+i)=bval;
    end
end
%
% deleting of a row and a column at x_0
%
ix0=J/2;

```

```

iy0=J/2;
j0=(iy0-1)*(J-1)+ix0;
L=L([1:j0-1,j0+1:N],[1:j0-1,j0+1:N]);
b=b([1:j0-1,j0+1:N]);

v=L\b;
%
% adding u(x_0)=0 to the system
%
v=[v(1:j0-1);0;v(j0:N-1)];
%
% produce a matrix as output
% in a format compatible with meshgrid
u(i,j)=u((j-1)*h,(i-1)*h)
%
u=zeros(J+1);
for j=1:J-1,
    for i=1:J-1
        u(j+1,i+1)=v((j-1)*(J-1)+i);
    end
end
%
% fill in the values on the boundary by
% use of the Neumann conditions
% recall that i=1 corresponds to x=0
%
for i=2:J,
    u(1,i)=u(2,i)+h*feval(phi,(i-1)*h,0);
    u(J+1,i)=u(J,i)+h*feval(phi,(i-1)*h,1);
end;

for j=2:J,
    u(j,1)=u(j,2)+h*feval(phi,0,(j-1)*h);
    u(j,J+1)=u(j,J)+h*feval(phi,1,(j-1)*h);
end;
%
% "artificial" interpolation at the corner points
% this helps for plots, but introduces big errors
%
u(1,1)=(u(1,2)+u(2,1))/2;
u(1,J+1)=(u(2,J+1)+u(1,J))/2;
u(J+1,J+1)=(u(J,J+1)+u(J+1,J))/2;
u(J+1,1)=(u(J,1)+u(J+1,2))/2;
%
% generating some plots and some data
% we add the exact solution at the point
% that was eliminated to find comparable
% solutions (this is arbitrary)
%
u=u+ux(ix0*h,iy0*h);
x=0:1/J:1;
y=0:1/J:1;
[X,Y]=meshgrid(x,y);

```

```

if pl>=2,
    figure
    surf(X,Y,u);
    title('the discrete solution','FontSize',18);
    figure
    surf(X,Y,ux(X,Y));
    title('the exact solution','FontSize',18);
end;
eh=u-ux(X,Y);
if pl>=1,
    figure
    surf(X,Y,eh);
    t=strcat('J=', num2str(J));
    t=strcat(t,': the error on \Omega\cup\Gamma without correction');
    title(t,'FontSize',18);
end;
%
% find the maximum error on
% \Omega_h and \Gamma_h^prime
% delete the corners
%
corner=[1 J+1 J*(J+1)+1 (J+1)^2];
ehp=reshape(eh',(J+1)*(J+1),1);
ix=setdiff(1:length(ehp),corner);
einf = max(ehp(ix))-min(ehp(ix));

```

The code for the solution with the correction is identical, but includes the following statements that adjust the right-hand side of the linear system.

```

%
% correction of the system for the mismatch
%
b=b-sum(b)/length(b);

```

It is convenient to write a routine that accepts the data and the solution of the problem as input parameters and that solves the system and finds the errors and the rates of convergence. Here is one possible version.

```

%
% solve Laplace's equation with Neumann data
%
% function u = sln(f,phi,ux)
%
% f = right-hand side
% phi = Neumann boundary data
% ux = solution of the PDE
%
function u = sln(f,phi,ux)
%
% definition of the refinement levels
%

```

```

jvals=[8 16 32 64];
%
% data structures to store and evaluate the errors
% nr = Neumann problem with reduced system
% nrc = Neumann problem with reduced system&correccion
%
linf_error_nr=zeros(length(jvals),1);
conv_linf_nr=zeros(length(jvals),1);

linf_error_nrc=zeros(length(jvals),1);
conv_linf_nrc=zeros(length(jvals),1);

for i=1:length(jvals)
    [u einf] = nr(jvals(i),f,phi,ux,1);
    linf_error_nr(i)=einf;
    [u einf] = nrc(jvals(i),f,phi,ux,1);
    linf_error_nrc(i)=einf;
end;

for j=2:length(jvals),
    conv_linf_nr(j)= -log(linf_error_nr(j)/linf_error_nr(j-1))/log(2);
    conv_linf_nrc(j)= -log(linf_error_nrc(j)/linf_error_nrc(j-1))/log(2);
end

disp('the infinity error without correction')
for i=1:length(jvals),
    disp(sprintf(' J=%2i, eh=%8.6f, EOC=%8.6f', ...
        jvals(i), linf_error_nr(i) ,conv_linf_nr(i)));
end

disp('the infinity error with correction')
for i=1:length(jvals),
    disp(sprintf(' J=%2i, eh=%8.6f, EOC=%8.6f', ...
        jvals(i), linf_error_nrc(i), conv_linf_nrc(i)));
end

figure
hold on
plot(log(1./jvals), log(linf_error_nr),'ro-')
plot(log(1./jvals), log(linf_error_nrc),'go-')

```

The first example We check that

$$\begin{aligned} \nabla \left(\frac{1}{2\pi^2} \sin(\pi x) \sin(\pi y) + \frac{1}{5\pi^2} \sin(\pi x) \sin(2\pi y) \right) \\ = \left(\frac{1}{2\pi} \cos(\pi x) \sin(\pi y) + \frac{1}{5\pi} \cos(\pi x) \sin(2\pi y) \right). \end{aligned}$$

We implement the Neumann data as an mfile:

```

%
% the boundary condition phi in HW10

```

J	L^∞ -error (no correction)	EOC	L^∞ -error (correction)	EOC
8	0.006195		0.006113	
16	0.001659	1.900674	0.001443	2.082573
32	0.000446	1.894447	0.000351	2.040927
64	0.000120	1.895071	0.000086	2.020213

TABLE 1. The L^∞ error in the calculation of the solution of Poisson's equation for example (1). The rate of convergence is better than the predicted linear convergence rate.

```
% phi is only defined on the unit square
%
function p = phi1(x,y)
%
% fix the sign for the normal derivative with s(ign)
%
if (x==0)|| (x==1)
    s=(-1)^x;
    p=s*(1/2/pi*cos(pi*x).*sin(pi*y)+1/5/pi*cos(pi*x).*sin(2*pi*y));
else
    s=(-1)^y;
    p=s*(1/2/pi*sin(pi*x).*cos(pi*y)+2/5/pi*sin(pi*x).*cos(2*pi*y));
end;
%
% in case that the sign trick fails...
%
if imag(p)~=0
    disp('catastrophic failure of boundary values')
    p=0;
end
```

We use the master routine to perform the computation for the first example:

```
clear all;
close all;

f1 =inline('sin(pi*x).*sin(pi*y)+sin(pi*x).*sin(2*pi*y)', 'x', 'y');
ux1 =inline('1/2/pi^2*sin(pi*x).*sin(pi*y)+1/5/pi^2*sin(pi*x).*sin(2*pi*y)', 'x', 'y');

sln(f1,@phi1,ux1);
```

We summarize the errors in the approximation in Table 1 and Figure 1.

The second example We now take the second data set, $u(x, y) = x^2 + y^2$. Then

$$\nabla f(x, y) = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

and the normal is equal to zero on $x = 0$ and $y = 0$ and equal to one for $x = 1$ and $y = 1$. Here is the implementation of these boundary data:

```
%
% the boundary condition phi in HW10
```

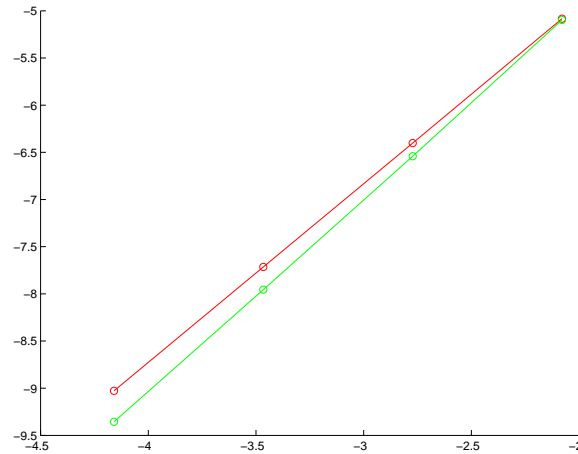


FIGURE 1. Double logarithmic plot of the L^∞ error for the solution of the Neumann problem for example (1) without (red curve) and with (green curve) correction for the mismatch in the discrete compatibility condition.

J	L^∞ -error (no correction)	EOC	L^∞ -error (correction)	EOC
8	0.236779		0.055804	
16	0.154806	0.613074	0.029427	0.923213
32	0.093920	0.720960	0.015152	0.957592
64	0.054655	0.781085	0.007692	0.978053

TABLE 2. The L^∞ error in the calculation of the solution of Poisson's equation in the second example. Here the convergence is only linear as predicted, and the effect of the singularity is clearly visible

```
% phi is only defined on the unit square
%
function p = phi2(x,y)

if (x==0)|| (y==0), p=0;
elseif (x==1)|| (y==1), p=1;
else disp('catastrophic error in boundary data');
end
```

Here is the call of the master routine:

```
f2 =inline('-4','x','y');
ux2 =inline('x.^2+y.^2','x','y');

sln(f2,@phi2,ux2);
```

We summarize the errors in the approximation in Table 2 and Figure 3.

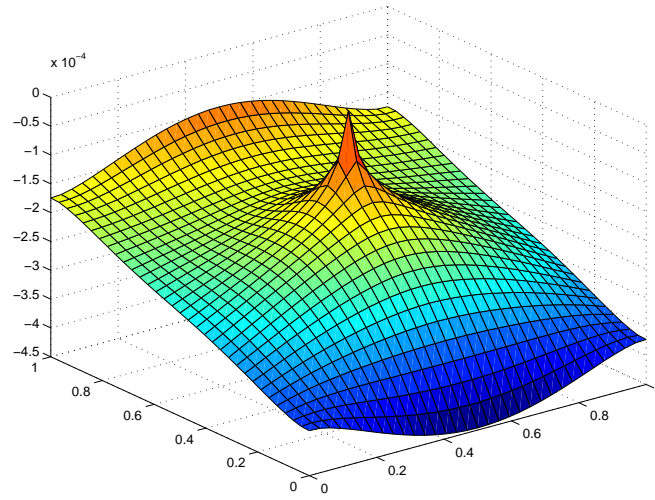
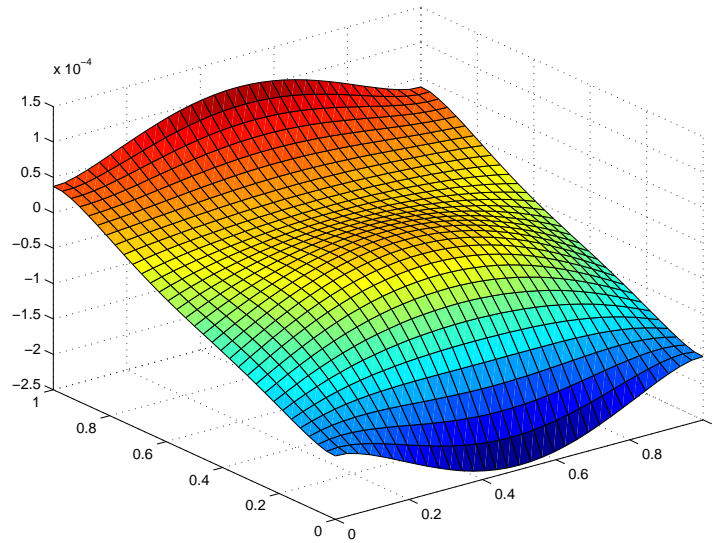
J=32: the error on $\Omega \cup \Gamma$ without correctionJ=32: the error on $\Omega \cup \Gamma$ with correction

FIGURE 2. The error on $\bar{\Omega}$ for the solution of the Neumann problem for the data (1). The effect of lumping the mismatch to one point leads to a singularity of the solution at this point.

The additional text example We check that

$$\nabla \left(\frac{1}{2\pi^2} \cos(\pi x) \cos(\pi y) \right) = \frac{1}{2\pi} \begin{pmatrix} \sin(\pi x) \cos(\pi y) \\ \cos(\pi x) \sin(\pi y) \end{pmatrix}$$

and hence $\partial u / \partial n = 0$ on the boundary of the unit square. We implement the boundary conditions as an mfile

```
%
% the zero function for the first test problem
```

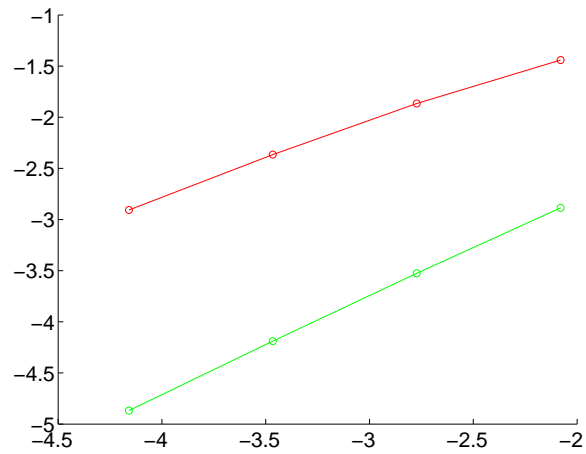


FIGURE 3. Double logarithmic plot of the L^∞ error for the solution of the Neumann problem for the data (2) without (red curve) and with (green curve) correction for the mismatch in the discrete compatibility condition.

```
%  
function p = phi0(x,y)  
  
p=0;
```

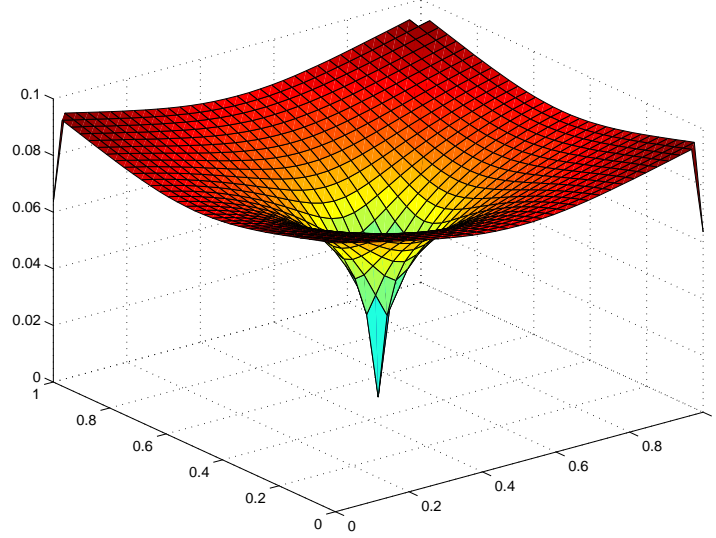
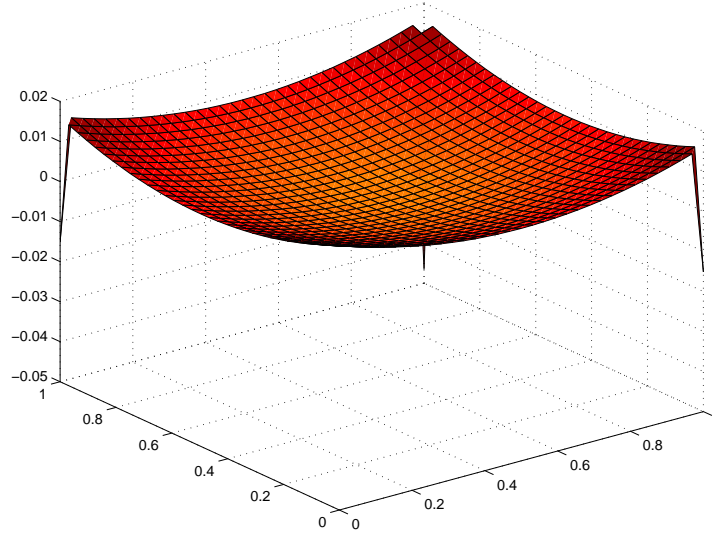
J=32: the error on $\Omega \cup \Gamma$ without correctionJ=32: the error on $\Omega \cup \Gamma$ with correction

FIGURE 4. The error on $\bar{\Omega}$ for the solution of the Neumann problem for the data (2). Note that the errors in the corners are due to the fact that the values are not computed in the scheme but added by interpolation. The effect of lumping the mismatch to one point leads to a singularity of the solution at this point.