Finite Differences

Peter Hertel

University of Osnabrück, Germany

Lecture presented at APS, Nankai University, China

http://www.home.uni-osnabrueck.de/phertel

Spring 2012

Overview

- boundary value problems
- approximate differential quotient by difference quotient
- one-dimensionals example
- sparse matrices
- Laplacian in two dimensions
- domain of definition
- setting up the matrix
- solve einvalue problem
- various ways to visualize 2D fields

Initial value problems

- think of a second order ODE
  $y'' = f(x, y, y')$
- itegrate it from x_0 to x_1 (x-span)
- you must specify two initial conditions
  $y(x_0) = y_0 \text{ and } y'(x_0) = y'_0$
- in general, there is a unique solution
- calculate it by one of the ODE solvers

Boundary value problems

- think of a second order ODE
  $y'' = f(x, y, y')$
- itegrate it from x_0 to x_1 (x-span)
- you must specify two conditions
- boundary values
  $y(x_0) = y_0 \text{ and } y(x_1) = y_1$
- cannot easily be solved by ODE solvers

Finite differences

- derivative of f defined by

$$f'(x) = \lim_{h\to 0} \frac{f(x+h/2) - f(x-h/2)}{h}$$

- replace the limes by a small, but finite h
- second derivative

$$f''(x) = \frac{f'(x+h/2) - f'(x-h/2)}{h}$$

- that is

$$f''(x) = \frac{f(x+h) - 2f(0) + f(x-h)}{h^2}$$

- for $x_j = jh$ and $f_j = f(x_j)$

$$(f'')_j = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2}$$

Setting up the matrix

- the variable x is approximated by a vector $x_j = jh$
- the function $f = f(x)$ is approximated by a vecor f_j
- the second derivative is approximated by a matrix L_{jk}
- setup this matrix

$$(f'')_j = \sum_k L_{jk} f_k = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2}$$

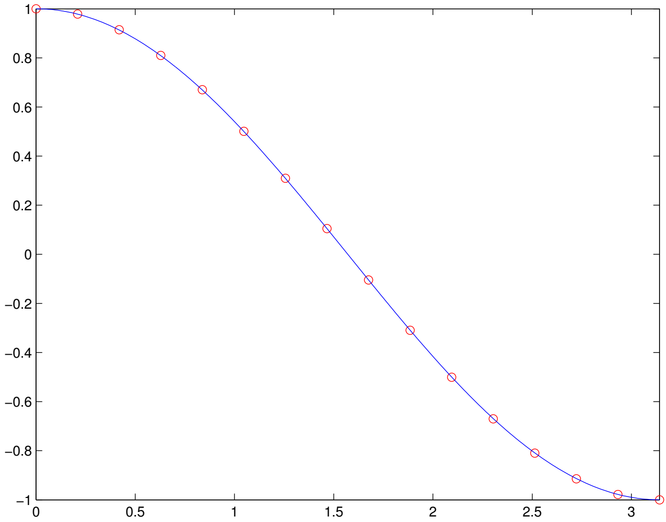
- if j runs from 1 to N, the first and the last equation are exceptions
- because f_0 and f_{N+1} are given boundary values, not unknowns

Example $f'' + f = 0$

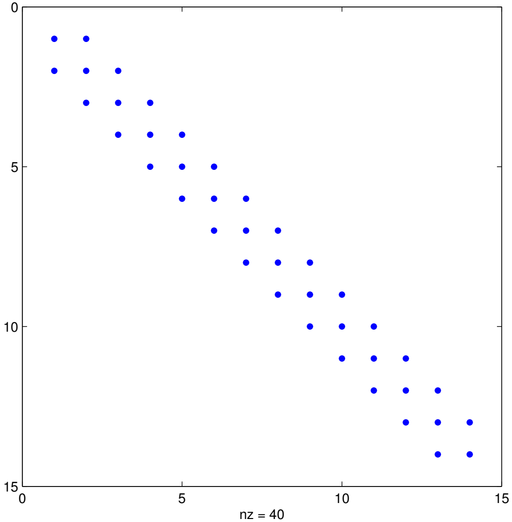
- ```
  b   u   u   --------   u   u   b
  0   1   2   --------  N-1  N  N+1
  ```
- first equation
  $$\frac{f_2 - 2f_1}{h^2} + f_1 = -\frac{f_0}{h^2}$$
- in between
  $$\frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} + f_j = 0$$
- last equation
  $$\frac{-2f_N + f_{N-1}}{h^2} + f_N = -\frac{f_{N+1}}{h^2}$$
- the matrix has one main and two side diagonals
  ```

```
function [x,f]=od_fdm(xlo,flo,xhi,fhi,NX)
% solve f"+f=0 on x=linspace(xlo,xhi,NX)
% with boundary values flo and fhi, resp.
% NX must be 3 or larger
N=NX-2; % number of unknowns
x=linspace(xlo,xhi,NX);
h=x(2)-x(1);
main=(-2/h^2+1)*ones(1,N);
next=(1/h^2)*ones(1,N-1);
DE=diag(next,-1)+diag(main,0)+diag(next,1);
BV=zeros(N,1);
BV(1)=-flo/h^2;
BV(N)=-fhi/h^2;
sol=DE\BV;
f=[flo,sol',fhi];
end % od_fdm

>> [x,f]=od_fdm(0,1,pi,-1,16);
>> xx=linspace(0,pi,256);
>> plot(x,f,'ro',xx,cos(xx),'b-');
>> axis tight
>> print -depsc od_fdm
```



The boundary problem $y'' + y = 0$ was solved by the finite difference method. $x \in [0, \pi]$ and $f(0) = 1$, $f(\pi) = -1$.



Non-vanishing elements of matrix `DE`, as produced by

```
>> spy(DE)
```

Sparse matrices

- for larger matrices, the percentage of non-zeroes becomes smaller and smaller
- for a 100 x 100 base region, there are 10,000 unknowns
- the Laplacian then has 10^8 matrix elements
- requiring 10^9 Bytes, i.e. 1 GB
- mostly zeroes
- sparse matrix technology
- list of $\{$i,k,value$\}$ entries for non-vanishing values
- iterative techniques for solving systems of linear equations
- only a few eigenvalues and eigenvectors make sense

Laplacian in two dimensions

- for simplicity, assume same spacing h along x and y
- mesh points $(x_i, y_k) = (ih, kh)$ with integer indexes i, k
- field $u = u(x, y)$ represented by unknowns $u_{ik} = u(x_i, y_k)$
- Laplacian

$$(\Delta u)(x,y) = \frac{\partial u(x,y)}{\partial x^2} + \frac{\partial u(x,y)}{\partial y^2}$$

- is represented by

$$(\Delta u)_{i,k} = \frac{u_{i+1,k} + u_{i-1,k} + u_{i,k+1} + u_{i,k-1} - 4u_{i,k}}{h^2}$$

- defined on a region Ω
- values at the boundary $\partial\Omega$ are given

The MATLAB logo

- long-standing test problem for computational solutions of partial differential equations (PDE)
- solve $-\Delta u = \Lambda u$ on an L-shaped region
- vibration of a thin membrane
- describe the domain Ω
- work out the Laplacian, a sparse matrix
- solve the eigenvalue problem for the smallest eigenvalue
- visualize the solution

```
function d=domain(N)
d.x=linspace(-1,1,N);
d.y=linspace(-1,1,N);
[X,Y]=meshgrid(d.x,d.y);
d.omega=(abs(X)<1)&(abs(Y)<1)&((X>0)|(Y>0));
r=0;
d.rr=zeros(N,N);
for i=1:N
    for k=1:N
        if d.omega(i,k)
            r=r+1;
            d.ii(r)=i;
            d.kk(r)=k;
            d.rr(i,k)=r;
        end;
    end;
end;
d.NU=r;
end % setup domain
```

domain

- the output is collected into a record `d`
- `x` and `y` are x and y axis of the mesh
- `omega(i,k)` is 1 if a mesh point `i,k` is an unknown (interior), 0 otherwise
- `r` is a running index for the unknowns
- `ii(r)` is the x-index of unknown `r`
- `kk(r)` likewise
- `rr(i,k)` is the running index of `i,k` or 0
- forward and backward mapping from double to single indexes
- `NU` is the number of unknowns

```
function d=laplace(d)

function neighbor(di,dk)
    if d.omega(i+di,k+dk)
        d.L(r,d.rr(i+di,k+dk))=1;
    end
end % neighbor

d.L=sparse(d.NU);
for r=1:d.NU
    d.L(r,r)=-4;
    i=d.ii(r);
    k=d.kk(r);
    neighbor(1,0);
    neighbor(-1,0);
    neighbor(0,1);
    neighbor(0,-1);
end
h=d.x(2)-d.x(1);
d.L=d.L/h^2;
end % laplace
```

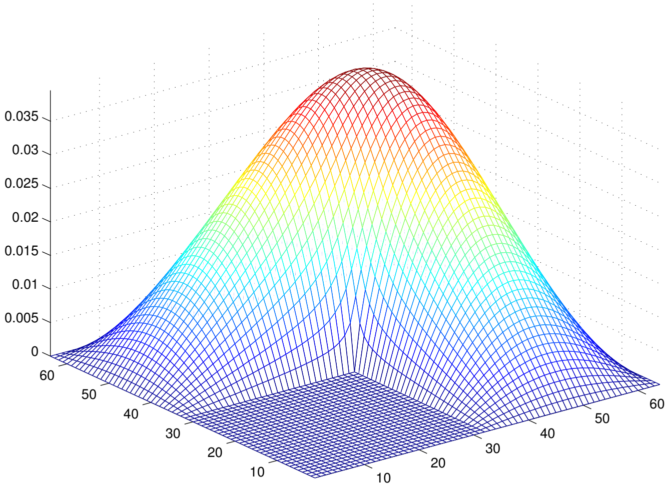
Laplacian

- create a sparse `NU x NU` matrix `L`
- set diagonal elements to -4
- inspect neighbors to the north, south, east and west
- if neighbor is an interior point, set `L` matrix element to $+1$
- for this, use a private function
- it has access to local variables
- finally, divide by h^2
- Laplacian added to record `d`

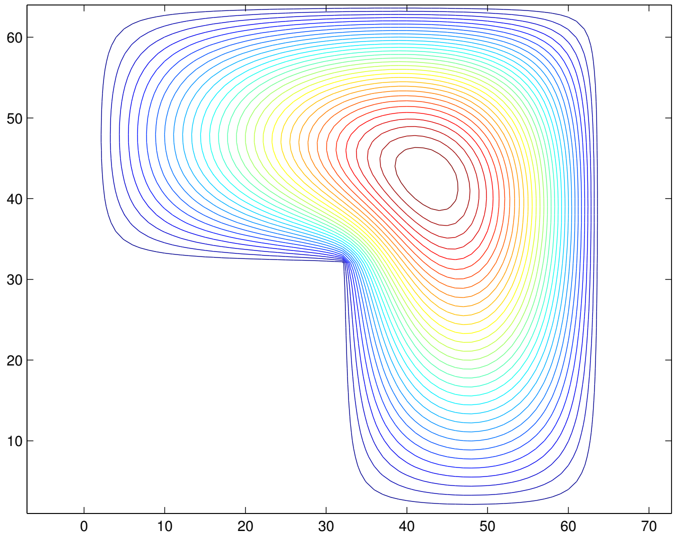
```
function basemode(d)
[evec,eval]=eigs(-d.L,1,'sm');
s=sign(sum(evec));
field=zeros(size(d.omega));
for r=1:d.NU
    field(d.ii(r),d.kk(r))=s*evec(r);
end
mesh(field);
axis off
print -depsc ml_logo_m.eps
contour(field, 32);
axis equal
print -depsc ml_logo_c.eps
imagesc(field);
axis equal
print -depsc ml_logo_i.eps
```

Base mode

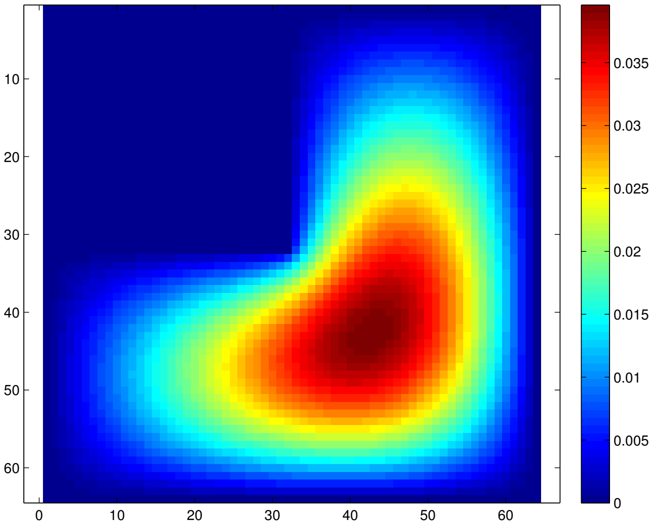
- eigenvalues of $-\Delta$ are positive
- calculate eigenfunction to smallest eigenvalue
- iterative algorithm!
- transform running index to field indexes
- plot it by the `mesh` method
- also: contour plot
- also: image plot



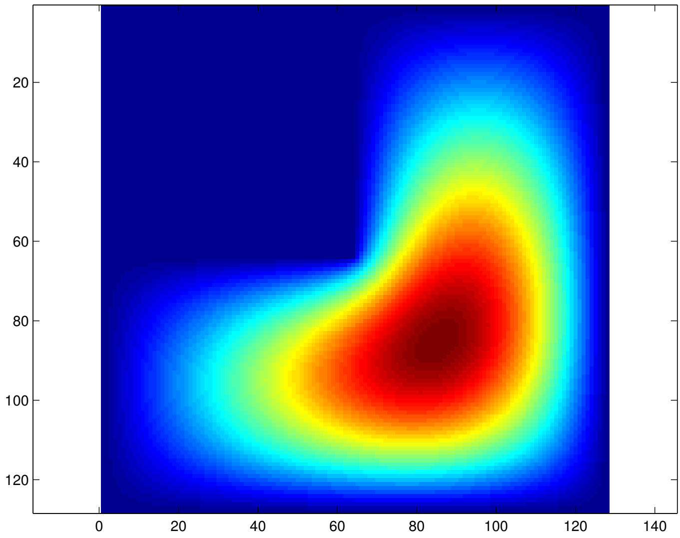
Base mode of Laplacian on an L-shaped domain. Plotted by `mesh`. 2883 unknowns.



Same as before, but plotted by `contour`.



Same as before, but plotted by `imagesc` (scaled image).



Same as before, but higher resolution. 11907 unknowns.