Introduction to Spectral Methods

Numerical Solution of Poisson's Equation

Pablo Brubeck[1]

[1]Department of Physics
Tecnologico de Monterrey

September 17, 2016

Outline

One dimension

Two Dimensions

Poisson's Equation

We are interested to find the function $u(x, y, z)$ that satisfies the boundary value problem

$$\nabla^2 u = f$$

Subject to boundary conditions $u(C) = g$.

Where the laplacian operator is $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$.

When $f = 0$, we have Laplace's equation as a special case.

Poisson's Equation has many important applications.

- Electrostatics $\nabla^2 V = -\rho/\varepsilon_0$
- Newtonian gravity $\nabla^2 \phi = 4\pi G \rho$
- Thermal equilibrium $\nabla^2 T = 0$
- Incompresible fluid flow $\nabla^2 v = 0, \nabla^2 p = f(\nu, V)$
- Mechanical Engineering
- Image processing
- Minimal area surfaces
- Surface reconstruction

Outline

One dimension

Two Dimensions

Poisson's Equation in one dimension

$$\frac{d^2 u}{dx^2} = f(x)$$

Subject to $u(a) = g_1$ and $u(b) = g_2$

We must translate this continuous problem into a discrete one.

For the space coordinate $x \in [a, b]$, we choose n nodes $\{x_i\}_{i=1}^{n}$ on which we would like to know the value of $u(x_i) = u_i$.

Input u_1, u_n, $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, and $\vec{f} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$; output $\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$.

The discrete solution must solve the continuous problem.

Is there a way to express $u(x)$ in terms of u_i?

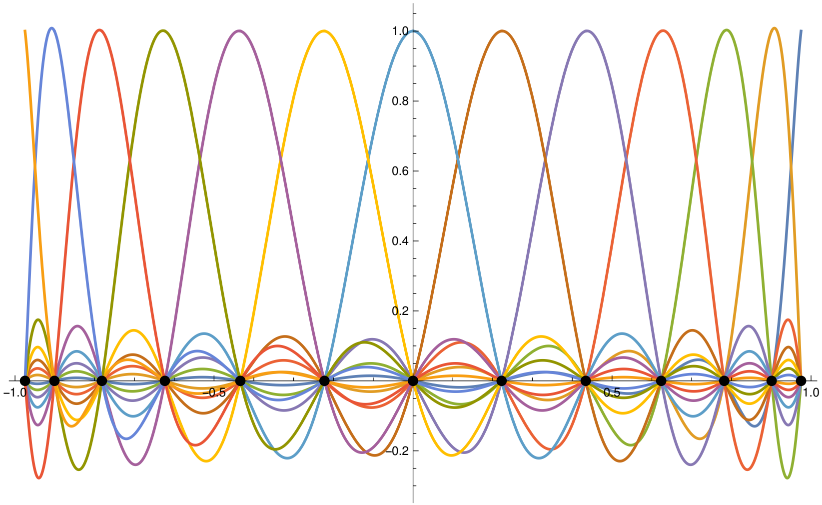
$$u(x) = \sum_j u_j \delta_j(x)$$

We introduce the cardinal functions $\delta_j(x)$.

$$u(x_i) = \sum_j u_j \delta_j(x_i) = u_i$$

$$\delta_j(x_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

This is how the cardinal functions look like.



Now differentiate the expansion in cardinal functions.

$$u'(x_i) = \sum_j u_j \delta_j'(x_i)$$

We have found a matrix $D_{ij} = \delta_j'(x_i)$ that operates as a derivative.

$$\frac{d}{dx}\vec{u} = D\vec{u}$$

$$\frac{d^2}{dx^2}\vec{u} = D^2\vec{u} = \vec{f}$$

Solve the system of linear equations.

$$D^2\vec{u} = \vec{f}$$

Recall that u_1 and u_n were given as boundary conditions.

$$\begin{bmatrix} D^2_{11} & D^2_{1j} & D^2_{1n} \\ D^2_{i1} & D^2_{ij} & D^2_{in} \\ D^2_{n1} & D^2_{nj} & D^2_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_j \\ u_n \end{bmatrix} = \begin{bmatrix} D^2_{11} \\ D^2_{i1} \\ D^2_{n1} \end{bmatrix} u_1 + \begin{bmatrix} D^2_{1j} \\ D^2_{ij} \\ D^2_{nj} \end{bmatrix} \left[u_j\right] + \begin{bmatrix} D^2_{1n} \\ D^2_{in} \\ D^2_{nn} \end{bmatrix} u_n = \begin{bmatrix} f_1 \\ f_i \\ f_n \end{bmatrix}$$

With i and j ranging from 2 to $n-1$.

We have $n-2$ unknowns, but still n equations.

Discard the first and last rows.

$\nabla^2 u = f$ will not hold true at the boundary, but we will ensure the boundary conditions.

$$\begin{aligned} \left[D^2_{ij}\right]\left[u_j\right] &= \left[f_i\right] - u_1\left[D^2_{i1}\right] - u_n\left[D^2_{in}\right] \\ \left[u_j\right] &= \left[D^2_{ij}\right]^{-1}\left[f_i - u_1 D^2_{i1} - u_n D^2_{in}\right] \end{aligned}$$

With i and j ranging from 2 to $n-1$.

Cardinal functions are obtained from a mother function.

Let $\Psi_n(x)$ be a function with n roots on $\{x_i\}_{i=1}^{n}$, i. e. $\Psi_n(x_i) = 0$. By expanding $\Psi_n(x)$ around its roots we can find a expression for $\delta_j(x)$.

$$\Psi_n(x) = 0 + (x - x_j)\Psi_n'(x_j) + (x - x_j)^2\Psi_j''(x) + O((x - x_j)^3)$$

$$\delta_j(x) = \frac{\Psi_n(x)}{(x - x_j)\Psi_n'(x_j)} = 1 + (x - x_j)\frac{\Psi_j''(x)}{\Psi_n'(x_j)} + O((x - x_j)^2)$$

Differentiate to obtain the elements of D.

$$D_{ij} = \delta_j'(x_i) = \frac{1}{x_i - x_j} \frac{\Psi_n'(x_i)}{\Psi_n'(x_j)}, \qquad i \neq j$$

The diagonal is a bit tricky.

$$D_{jj} = \delta_j'(x_j) = \frac{\Psi_n''(x_j)}{\Psi_n'(x_j)}$$

Let's code!

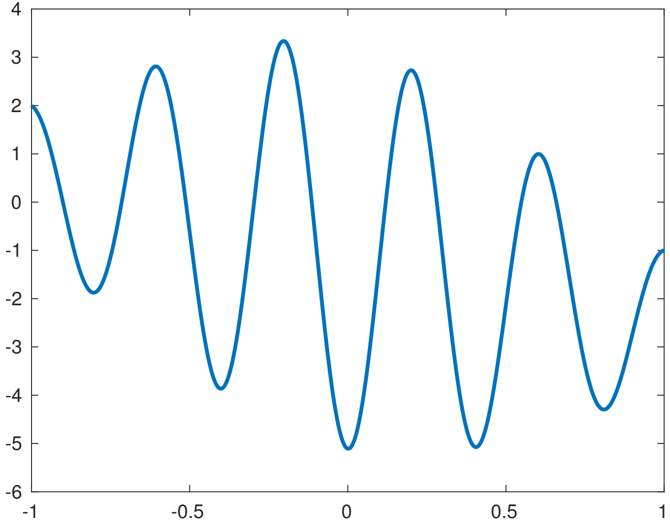
Solve Poisson's equation

$$\frac{d^2u}{dx^2} = 1000\cos(5\pi x)e^{-x^2}$$

with boundary conditions $u(-1) = 2$, $u(1) = -1$.

```
[D,x]=chebD(n); D2=D*D;    % Differentiation matrix and nodes
f=1000*cos(5*pi*x).*exp(-x.^2); % Source term
u=zeros(n,1); u([1,n])=[-1,2];  % Impose boundary conditions
% Solve and plot
u(2:n-1)=D2(2:n-1,2:n-1)\(f(2:n-1)-D2(2:n-1,[1,n])*u([1,n]));
plot(x,u);
```

Here is our solution.



Outline

Poisson's Equation in two dimensions

Our domain of solution will be the rectangle $[a, b] \times [c, d]$

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

Subject to boundary conditions

$$u(x, y) = \begin{cases} g_1(y), & x = a \\ g_2(y), & x = b \\ h_1(x), & y = c \\ h_2(x), & y = d \end{cases}$$

In 2D we use matrices for u and f.

$$u(x_i, y_j) = u_{ij}$$
$$f(x_i, y_j) = f_{ij}$$

$$u(x, y) = \sum_{k,l} u_{kl}\delta_k(x)\delta_l(y)$$

Multiplying times D does the job of partial derivatives.

$$\frac{\partial u}{\partial x} = DU = \begin{bmatrix} Du_{i1} & Du_{ij} & Du_{in} \end{bmatrix}$$

$$\frac{\partial u}{\partial y} = UD^{\mathsf{T}} = (DU^{\mathsf{T}})^{\mathsf{T}} = \begin{bmatrix} (Du_{1i})^{\mathsf{T}} \\ (Du_{ji})^{\mathsf{T}} \\ (Du_{ni})^{\mathsf{T}} \end{bmatrix}$$

$$\nabla^2 u = D^2 U + U(D^2)^{\mathsf{T}}$$

Now solve the system of linear equations.

$$D^2 U + U(D^2)^{\mathsf{T}} = F$$

Boundary conditions are the tricky part.

$$U = \tilde{U} + U_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & u_{ij} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} u_{11} & u_{1j} & u_{1n} \\ u_{i1} & 0 & u_{in} \\ u_{n1} & u_{nj} & u_{nn} \end{bmatrix}$$

$$D^2 \tilde{U} + \tilde{U}(D^2)^{\mathsf{T}} = F - D^2 U_B - U_B(D^2)^{\mathsf{T}} = \tilde{F}$$

Discard first and last rows and columns.

$\nabla^2 u = f$ will not hold true at the boundary, but we will ensure the boundary conditions.

$$(D^2_{ij})\tilde{U}_{ij} + \tilde{U}_{ij}(D^2_{ij})^{\mathsf{T}} = \tilde{F}_{ij}$$

With i and j ranging from 2 to $n-1$.

Now we just have to solve $(n-2)^2 \times (n-2)^2$ system of linear equations.

Matrix inversion does not help.

A matrix equation of the form $AX + XB = C$ is known as a Sylvester equation.

There's a built-in MATLAB function that solves them.

```
X=sylvester(A,B,C)
```

This algorithm uses the Schur decomposition to solve a block-triangular matrix very efficiently.

Another example

Solve Laplace's equation on $(x, y) \in [-1, 1] \times [-1, 1]$

$$\nabla^2 u = 0$$

with boundary conditions

$$u(x, y) = \begin{cases} \sin^4(\pi x), & y = 1 \text{ and } -1 < x < 0, \\ \frac{1}{5}\sin(3\pi y), & x = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Let's code again!

```
[D,x]=chebD(n); D2=D*D; y=x';
[xx, yy]=ndgrid(x);

% Boundary conditions
g=[0.2*sin(3*pi*y); 0*y];
h=[(x<0).*sin(pi*x).^4, 0*x];
uu=zeros(n);
uu([1 n],:)=g;
uu(:,[1 n])=h;

% Solve Laplace's equation
F=zeros(n);
RHS=F-D2(:,[1 n])*g-h*D2(:,[1 n])';
uu(2:n-1, 2:n-1)=sylvester(D2(2:n-1, 2:n-1), ...
D2(2:n-1, 2:n-1)', RHS(2:n-1, 2:n-1));

surfl(xx,yy,uu,'light'); colormap(jet(256));
shading interp; axis square;
```

Here is our solution.