Partial Differential Equations (PDE's)

Huge topic in science and engineering. ANYONE with a master degree in engineering should know

- **something about** PDE's
- **something about** how to formulate boundary conditions for PDE's
- **something about** how to solve PDE's numerically including **clever** usage of available software.

In this course only 2nd order PDE's in two variables. Classification into

Hyperbolic PDE's with the 1D wave equation as the model example:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = v^2 \frac{\partial^2 u(x,t)}{\partial x^2}$$

Elliptic PDE's with the 2D Poisson's equation as the model example:

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

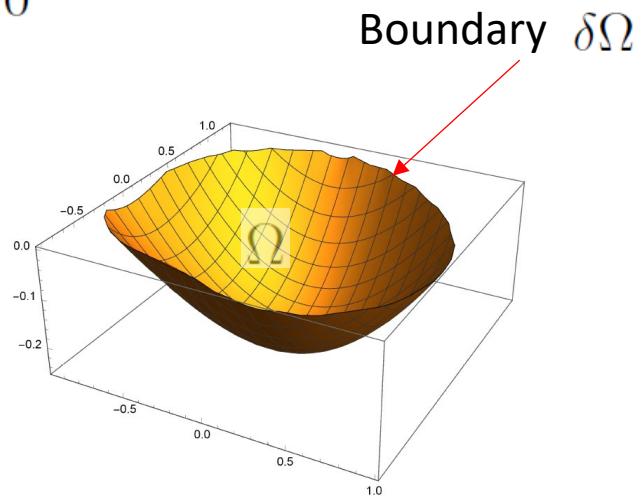
Parabolic PDE's with the 1D diffusion equation as the model example:

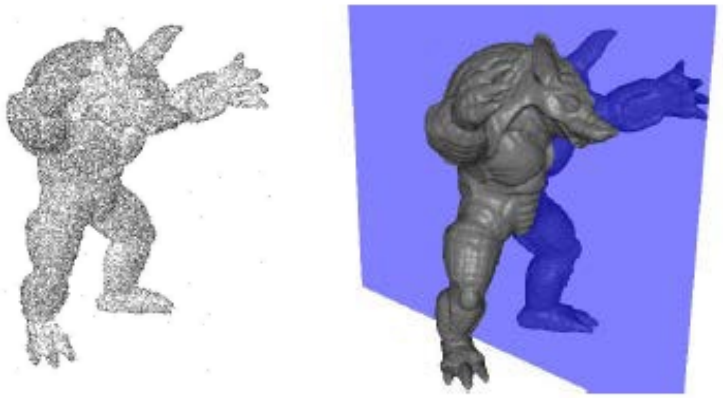
$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t)$$

Poissons equation:

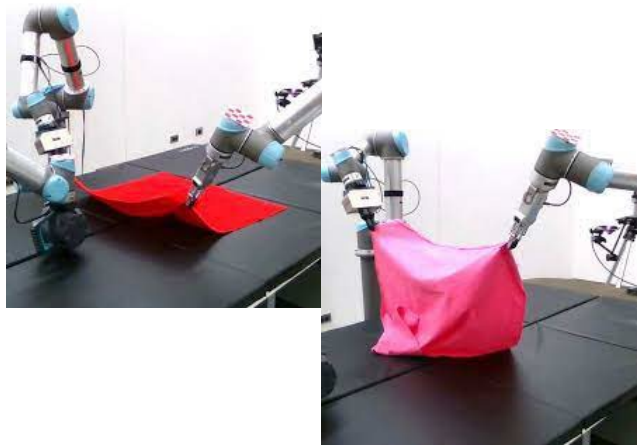
$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)+\lambda u(x,y)=f(x,y);\quad (x,y)\in\Omega\subseteq\mathbb{R}^2;\quad \lambda\geq 0$$

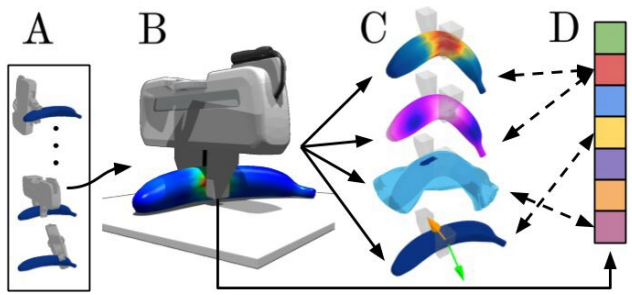
Condition on $u(x,y)$ on the boundary $\delta\Omega$ (discussed on next slide)





Poisson Reconstruction of surfaces from point clouds





Model equation for shapes of handled deformable materials

Poissons equation:

$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)+\lambda u(x,y)=f(x,y);\quad (x,y)\in\Omega\subseteq\mathbb{R}^2;\quad \lambda\geq 0$$

On the boundary of Ω, written as $\delta\Omega$, we have either that $u(x,y)$ is specified

$$u(x,y)=\phi_D(x,y);\quad (x,y)\in\delta\Omega$$

called *Dirichlet boundary conditions.*

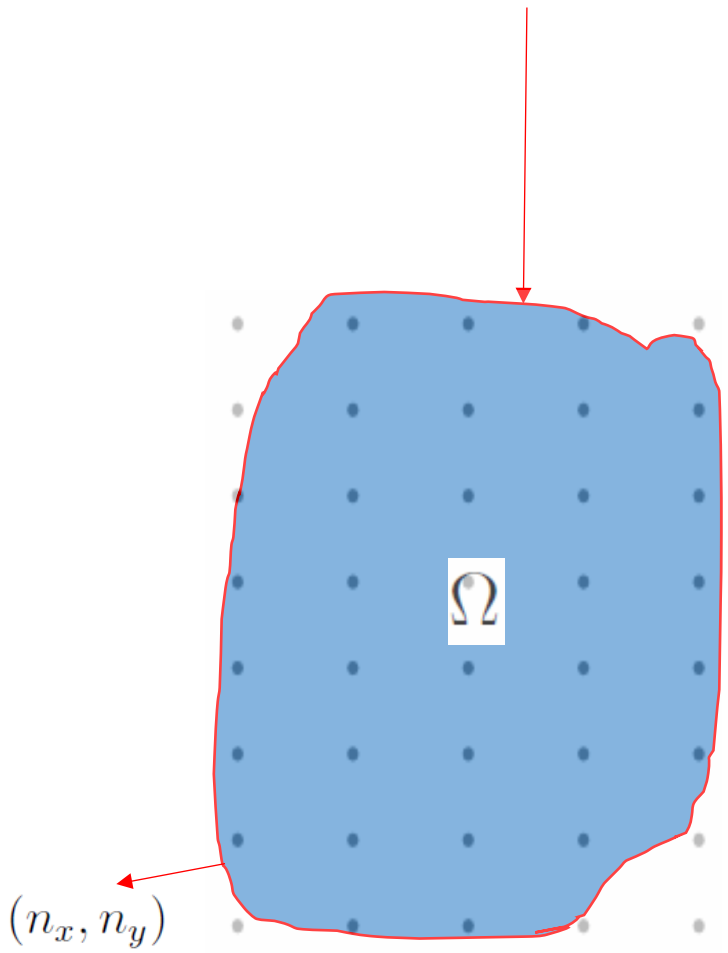
Or the derivative is specified

$$\left(\frac{\partial u(x,y)}{\partial x},\frac{\partial u(x,y)}{\partial y}\right)\cdot(n_x,n_y)=\phi_N(x,y);\quad (x,y)\in\delta\Omega$$

where (n_x,n_y) is a vector perpendicular to the boundary at (x,y). These are called *Neumann boundary conditions.* Often $\phi_N(x,y)=0$, which we call *free Neumann boundary conditions.* It can also be a mixture of the two conditions along the boundary.

Discretization: Grid size h.

$$x_j=x_0+jh;\quad y_k=y_0+kh;\quad u_{j,k}\simeq u(x_j,y_k);\quad f_{j,k}\equiv f(x_j,y_k)$$



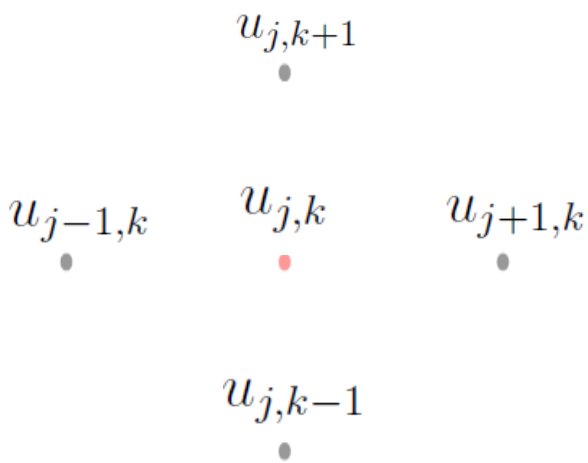
Poissons equation:

$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)+\lambda u(x,y)=f(x,y);\quad (x,y)\in\Omega\subseteq\mathbb{R}^2;\quad \lambda\geq 0$$

$$x_j=x_0+ih;\quad y_k=y_0+kh;\quad u_{j,k}\simeq u(x_j,y_k);\quad f_{j,k}\equiv f(x_j,y_k)$$

We then get

$$\begin{aligned}\frac{\partial^2 u(x_j,y_k)}{\partial x^2} &= \frac{u_{j+1,k}-2u_{j,k}+u_{j-1,k}}{h^2}+\mathcal{O}(h^2)\\ \frac{\partial^2 u(x_j,y_k)}{\partial y^2} &= \frac{u_{j,k+1}-2u_{j,k}+u_{j,k-1}}{h^2}+\mathcal{O}(h^2)\end{aligned}$$



Inserting into Poisson's equation yields the following equation for every interior point (none of the four neighbours are at or exceeds the boundary of Ω:

$$-u_{j-1,k}-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j+1,k}-u_{j,k+1}=h^2 f_{j,k}$$

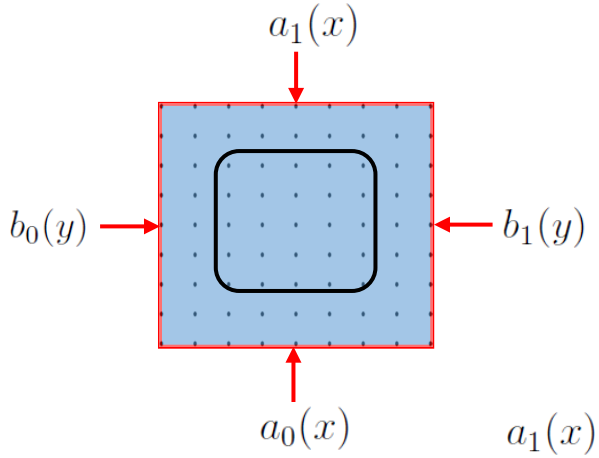
$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)+\lambda u(x,y)=f(x,y);\quad 0\le x,y\le 1;\quad \lambda\ge 0$$

$$u(x,0)=a_0(x);\quad u(x,1)=a_1(x);\quad u(0,y)=b_0(y);\quad u(1,y)=b_1(y)$$

$$-u_{j-1,k}-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j+1,k}-u_{j,k+1}=h^2 f_{j,k}$$

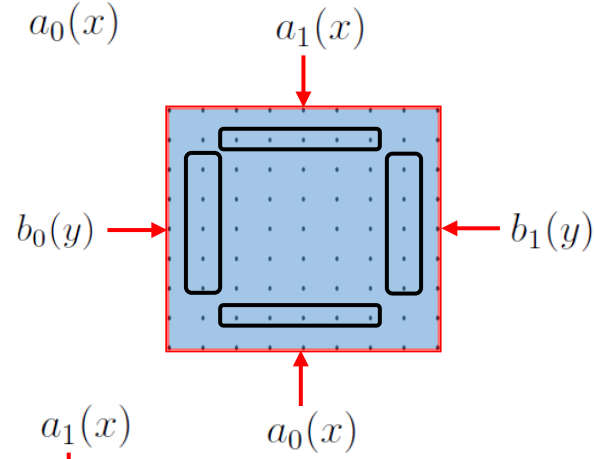
We then get the equation for the interior points

$$-u_{j-1,k}-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j+1,k}-u_{j,k+1} = h^2 f_{j,k} \quad 2\le j,k\le N-2$$



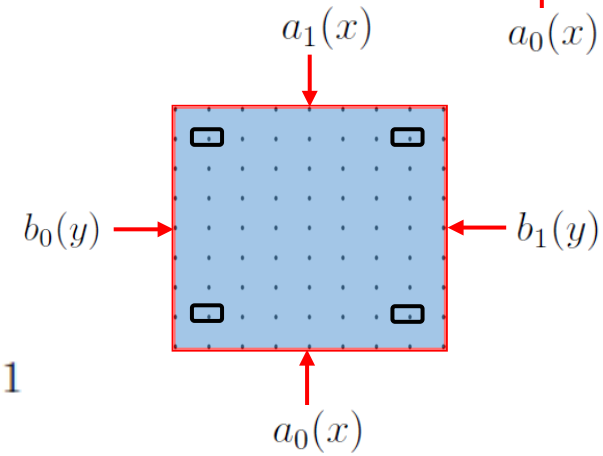
For the edge points, we get

$$\begin{aligned}
-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j+1,k}-u_{j,k+1} &= h^2 f_{j,k}+b_0(y_k) \quad j=1;\quad 2\le k\le N-2\\
-u_{j-1,k}+(4+h^2\lambda)u_{j,k}-u_{j+1,k}-u_{j,k+1} &= h^2 f_{j,k}+a_0(x_j) \quad k=1;\quad 2\le j\le N-2\\
-u_{j-1,k}-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j,k+1} &= h^2 f_{j,k}+b_1(y_k) \quad j=N-1;\quad 2\le k\le N-2\\
-u_{j-1,k}-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j+1,k} &= h^2 f_{j,k}+a_1(x_j) \quad k=N-1;\quad 2\le j\le N-2
\end{aligned}$$



and for the corner points:

$$\begin{aligned}
(4+h^2\lambda)u_{j,k}-u_{j+1,k}-u_{j,k+1} &= h^2 f_{j,k}+a_0(x_j)+b_0(y_k) \quad j=1;\quad k=1\\
-u_{j-1,k}+(4+h^2\lambda)u_{j,k}-u_{j,k+1} &= h^2 f_{j,k}+a_0(x_j)+b_1(y_k) \quad j=N-1;\quad k=1\\
-u_{j,k-1}+(4+h^2\lambda)u_{j,k}-u_{j+1,k} &= h^2 f_{j,k}+a_1(x_j)+b_0(y_k) \quad j=1;\quad k=N-1\\
-u_{j-1,k}-u_{j,k-1}+(4+h^2\lambda)u_{j,k} &= h^2 f_{j,k}+a_1(x_j)+b_1(y_k) \quad j=N-1;\quad k=N-1
\end{aligned}$$



We now define an index function $\text{inx}(j,k) = (N-1)(k-1)+j$. We then get a system of $(N-1)^2$ linear equations in $(N-1)^2$ unknowns $u_{i,j} \quad 1 \le i,j \le N-1$. With $i = inx(j,k)$, we write $w_i \equiv u_{j,k}$. We then write the system of linear equations as $Aw = \phi$. **Notice: Subtract 1 from inx(j,k) if indices run from zero**

For the interior points, we get with $i = \text{inx}(j,k)$

$$A_{i,s} = \begin{cases} -1 & s \in \{i-(N-1), i-1, i+1, i+(N-1)\} \\ 4+h^2\lambda & s = i \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_i = h^2 f_{j,k}$$

For the edge points, we get with $i = \text{inx}(N-1,k)$ (the other edges are similar)

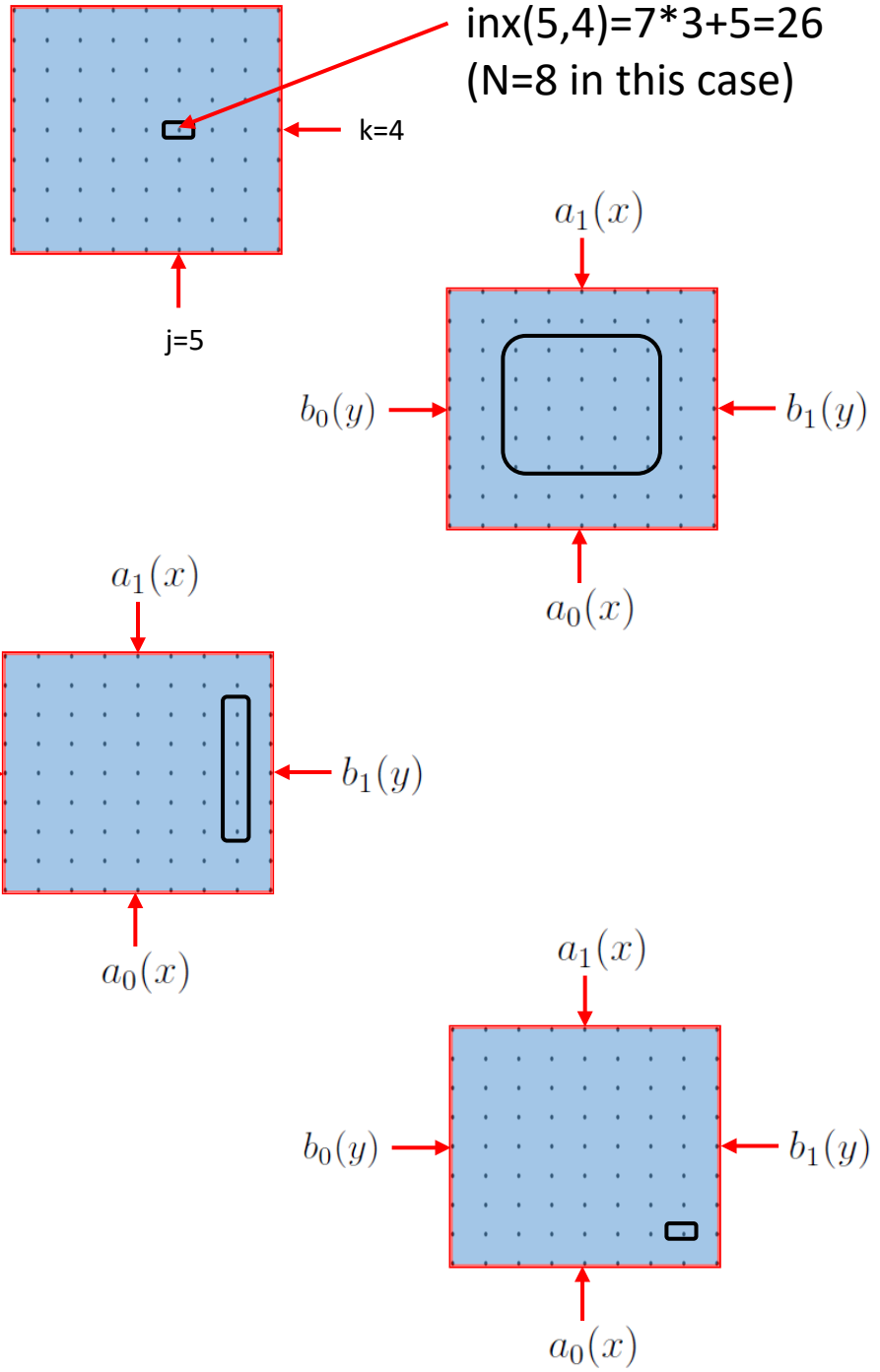
$$A_{i,s} = \begin{cases} -1 & s \in \{i-(N-1), i-1, i+(N-1)\} \\ 4+h^2\lambda & s = i \\ 0 & \text{otherwise} \end{cases}$$

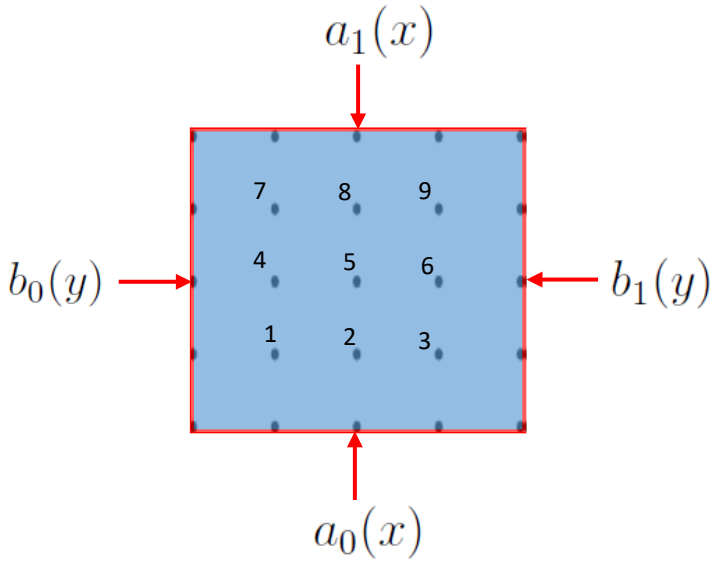
$$\phi_i = h^2 f_{N-1,k} + b_1(y_k)$$

For the corner points, we get with $i = \text{inx}(N-1,1)$ (the other corners are similar)

$$A_{i,s} = \begin{cases} -1 & s \in \{i-1, i+(N-1)\} \\ 4+h^2\lambda & s = i \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_i = h^2 f_{N-1,1} + a_0(x_{N-1}) + b_1(y_1)$$





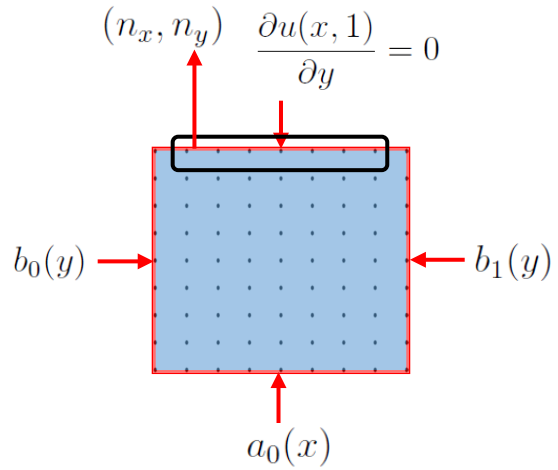
To summarize, we provide the matrix and the right hand side for $N = 4$:

$$\begin{pmatrix} X & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & X & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & X & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & X & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & X & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & X & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & X & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & X & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & X \end{pmatrix}; \quad \phi = \begin{pmatrix} h^2 f_{1,1} + a_0(x_1) + b_0(y_1) \\ h^2 f_{2,1} + a_0(x_2) \\ h^2 f_{3,1} + a_0(x_3) + b_1(y_1) \\ h^2 f_{1,2} + b_0(y_2) \\ h^2 f_{2,2} \\ h^2 f_{3,2} + b_1(y_2) \\ h^2 f_{1,3} + a_1(x_1) + b_0(y_3) \\ h^2 f_{2,3} + a_1(x_2) \\ h^2 f_{3,3} + a_1(x_3) + b_1(y_3) \end{pmatrix} \quad \text{where } X = 4 + h^2\lambda.$$

Let us now consider a model problem with combined fixed and free boundary conditions where we have free boundary conditions at $y = 1$:

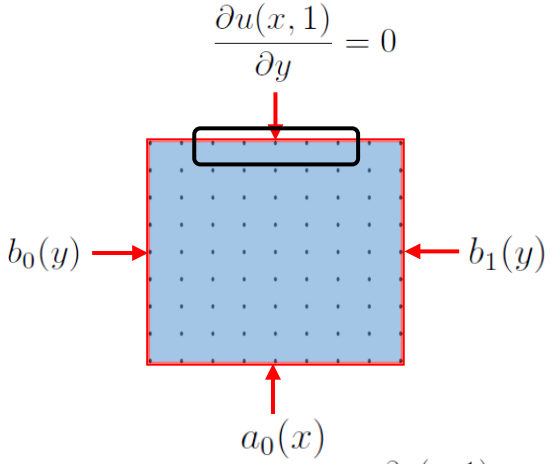
$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2}\right) + \lambda u(x,y) = f(x,y); \quad 0 \leq x, y \leq 1; \quad \lambda \geq 0$$

$$\boxed{\frac{\partial u(x,1)}{\partial y} = 0;} \quad u(x,0) = a_0(x); \quad u(0,y) = b_0(y); \quad u(1,y) = b_1(y)$$



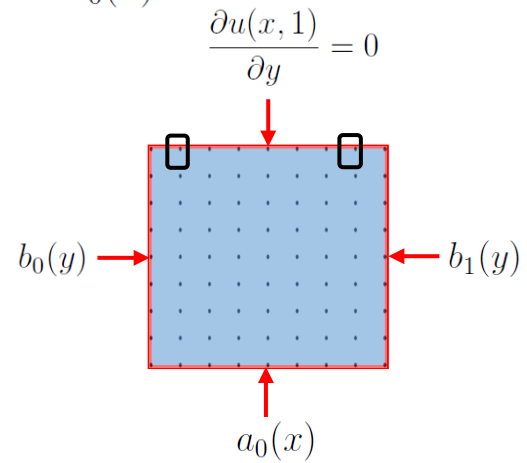
We implement the free boundary conditions by introducing numerical Poissons equation at the free boundary. For the edge at $y = 1$ corresponding to $k = N$, we get

$$-u_{j-1,N} - u_{j,N-1} + (4 + h^2\lambda)u_{j,N} - u_{j+1,N} - \boxed{u_{j,N+1}} = h^2 f_{j,N} \quad 2 \leq j \leq N-2$$



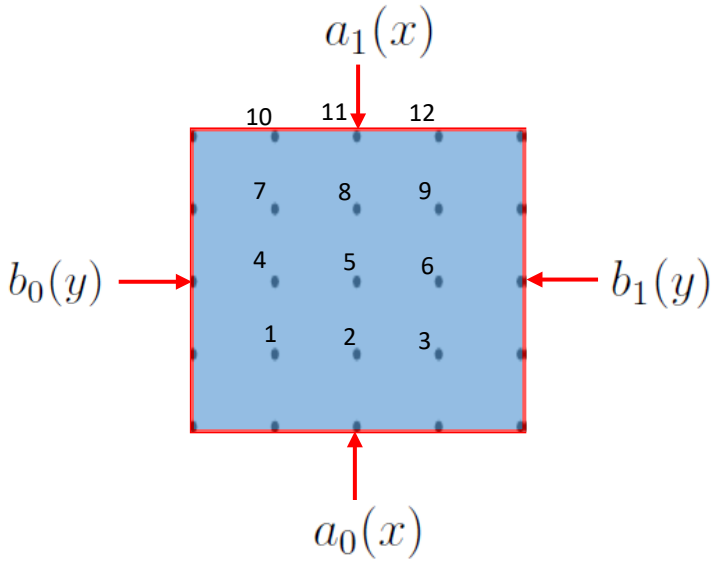
We now implement the boundary condition $\frac{\partial u(x,1)}{\partial y} = 0$ by setting the "phantom point" $\boxed{u_{j,N+1} \equiv u_{j,N-1}}$. We then get

$$-u_{j-1,N} - 2u_{j,N-1} + (4 + h^2\lambda)u_{j,0} - u_{j+1,N} = h^2 f_{j,N} \quad 2 \leq j \leq N-2$$



For the two corner points at the free edge, we get

$$\begin{aligned} -2u_{1,N-1} + (4 + h^2\lambda)u_{1,N} - u_{2,N} &= h^2 f_{1,N} + b_0(y_N) \\ -u_{N-2,N} - 2u_{N-1,N-1} + (4 + h^2\lambda)u_{N-1,N} &= h^2 f_{N-1,N} + b_1(y_N) \end{aligned}$$



Notice that you will typically need to modify the index function inx(j,k) with free boundary conditions as more points are included (this example being the exception). Also non-square shapes lead to different index functions. Hence, when implementing:

1) Define carefully the index function inx(j,k)
2) Use the index function to retrieve the nonzero elements in the matrix and the right hand side

We now get $(N-1)N$ equations in $(N-1)N$ unknowns. The coefficient matrix and right hand side looks as follows for $N = 4$:

$$\begin{pmatrix}
X & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & X & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & X & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & X & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & X & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & -1 & X & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & X & -1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & X & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & X & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & X & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & X & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & X
\end{pmatrix}$$

$$\phi = \begin{pmatrix}
h^2 f_{1,1} + a_0(x_1) + b_0(y_1) \\
h^2 f_{2,1} + a_0(x_2) \\
h^2 f_{3,1} + a_0(x_3) + b_1(y_1) \\
h^2 f_{1,2} + b_0(y_2) \\
h^2 f_{2,2} \\
h^2 f_{3,2} + b_1(y_2) \\
h^2 f_{1,3} + b_0(y_3) \\
h^2 f_{2,3} \\
h^2 f_{3,3} + b_1(y_3) \\
h^2 f_{1,4} + b_0(y_4) \\
h^2 f_{2,4} \\
h^2 f_{3,4} + b_1(y_4)
\end{pmatrix}$$

Band matrix

- In all cases, we have a band-width not larger than N from the diagonal. Hence the algorithm “Bandec” in Sec. 2.4.2 p. 58-61 in NR can be applied.
- Complexity will then be O(N^4) instead of O(N^6) for LU decomposition and O(N^3) instead of O(N^4) for forward reduction and back substitution
- Similar band structures occur for arbitrarily shaped regions however with a bandwidth that varies from row to row.
- Hence, it is a numerical crime not to use this computational advantage.

nonzero elements we now call x's, y's, and z's. We call the modified right-hand sides q. The transformed problem is now

$$\begin{bmatrix} b_0 & & & c_0 & & \\ & b_2 & & a_2 & c_2 & \\ & & b_4 & & a_4 & c_4 \\ & & & b_6 & & a_6 \\ & & & y_0 & z_0 & \\ & & & x_1 & y_1 & z_1 \\ & & & & x_2 & y_2 \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ u_2 \\ u_4 \\ u_6 \\ u_1 \\ u_3 \\ u_5 \end{bmatrix} = \begin{bmatrix} r_0 \\ r_2 \\ r_4 \\ r_6 \\ q_0 \\ q_1 \\ q_2 \end{bmatrix} \tag{2.4.5}$$

Notice that the last three rows form a new, smaller, tridiagonal problem, which we can solve simply by recursing. Once its solution is known, the first four rows can be solved by a simple, parallelizable, substitution. For discussion of this and related methods for parallelizing tridiagonal systems, and references to the literature, see [2].

2.4.2 Band-Diagonal Systems

Where tridiagonal systems have nonzero elements only on the diagonal plus or minus one, band-diagonal systems are slightly more general and have (say) $m_1 \geq 0$ nonzero elements immediately to the left of (below) the diagonal and $m_2 \geq 0$ nonzero elements immediately to its right (above it). Of course, this is only a useful classification if m_1 and m_2 are both $\ll N$. In that case, the solution of the linear system by LU decomposition can be accomplished much faster, and in much less storage, than for the general $N \times N$ case.

The precise definition of a band-diagonal matrix with elements a_{ij} is that

$$a_{ij} = 0 \quad \text{when} \quad j > i + m_2 \quad \text{or} \quad i > j + m_1 \tag{2.4.6}$$

Band-diagonal matrices are stored and manipulated in a so-called compact form, which results if the matrix is tilted 45° clockwise, so that its nonzero elements lie in a long, narrow matrix with $m_1 + 1 + m_2$ columns and N rows. This is best illustrated by an example: The band-diagonal matrix

$$\begin{pmatrix} 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 5 & 0 & 0 & 0 & 0 \\ 9 & 2 & 6 & 5 & 0 & 0 & 0 \\ 0 & 3 & 5 & 8 & 9 & 0 & 0 \\ 0 & 0 & 7 & 9 & 3 & 2 & 0 \\ 0 & 0 & 0 & 3 & 8 & 4 & 6 \\ 0 & 0 & 0 & 0 & 2 & 4 & 4 \end{pmatrix} \tag{2.4.7}$$

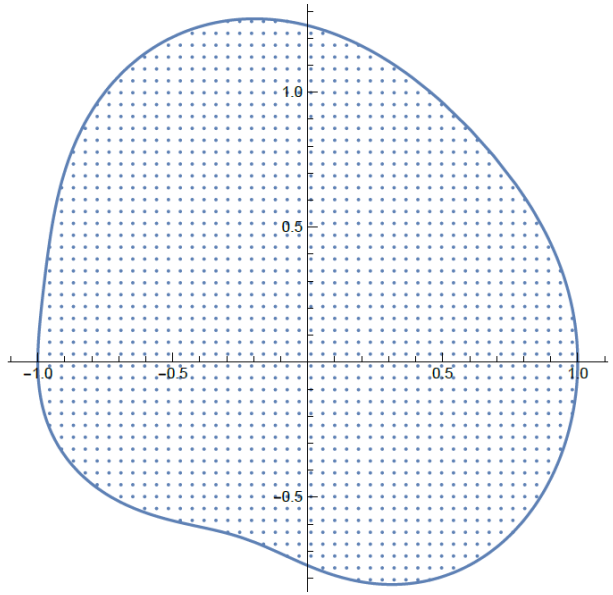
which has $N = 7$, $m_1 = 2$, and $m_2 = 1$, is stored compactly as the 7×4 matrix,

$$\begin{pmatrix} x & x & 3 & 1 \\ x & 4 & 1 & 5 \\ 9 & 2 & 6 & 5 \\ 3 & 5 & 8 & 9 \\ 7 & 9 & 3 & 2 \\ 3 & 8 & 4 & 6 \\ 2 & 4 & 4 & x \end{pmatrix} \tag{2.4.8}$$

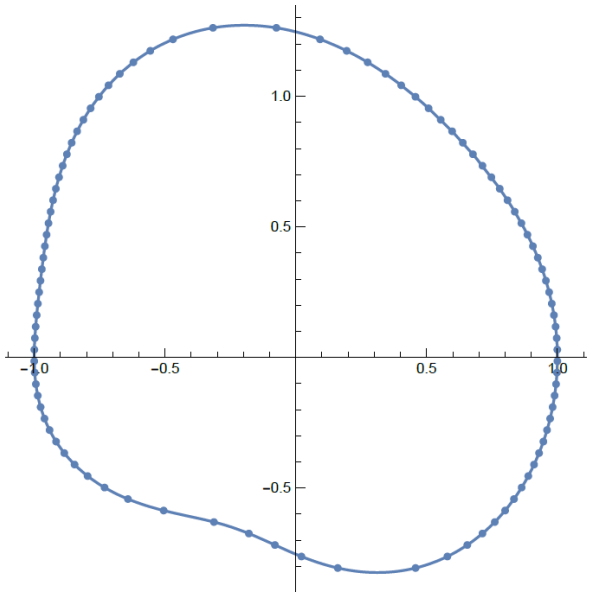
Here x denotes elements that are wasted space in the compact format; these will not be referenced by any manipulations and can have arbitrary values. Notice that the diagonal of the original matrix appears in column m_1, with subdiagonal elements to its left and superdiagonal elements to its right.

The simplest manipulation of a band-diagonal matrix, stored compactly, is to multiply it by a vector to its right. Although this is algorithmically trivial, you might want to study the following routine as an example of how to pull nonzero elements a_{ij} out of the compact storage format in an orderly fashion.

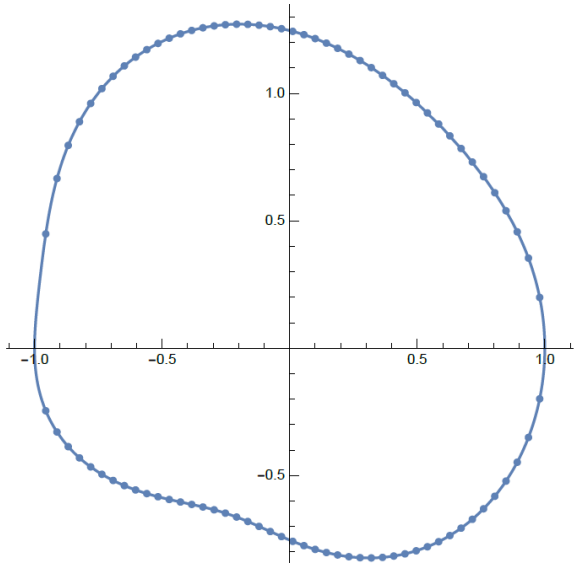
General 2D shapes



Interior grid points

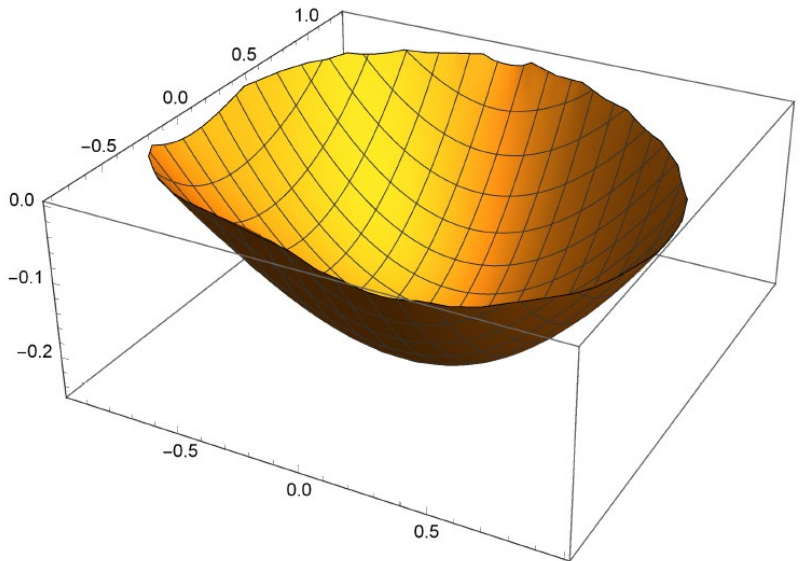


Row boundary points

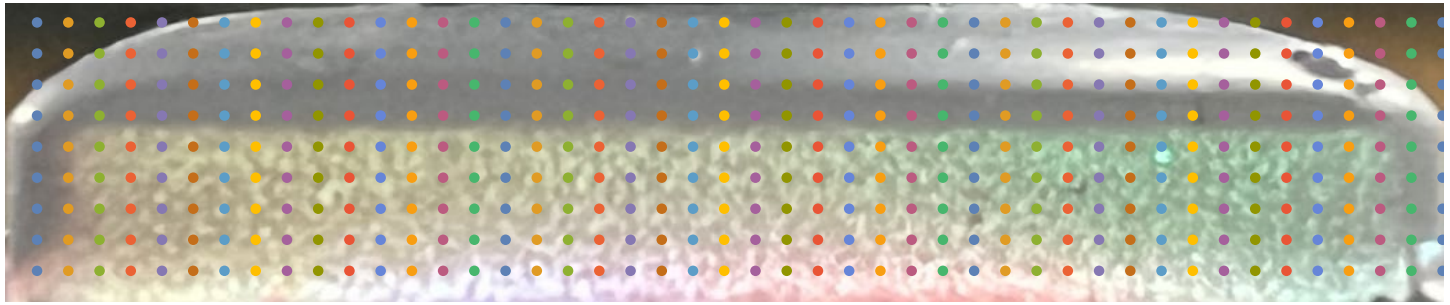


Column boundary points

Implement boundary conditions using phantom points immediately outside the boundary and interpolate (here same idea for Dirichlet and Neumann conditions)



A little discussion about extensions to 3D. Modeling tactile sensor shapes



$$(\lambda+\mu)\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+\mu\left(\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial y^2}+\frac{\partial^2 u_x}{\partial z^2}\right)+F_x=0$$

Following the same procedure for the y-direction and z-direction we have

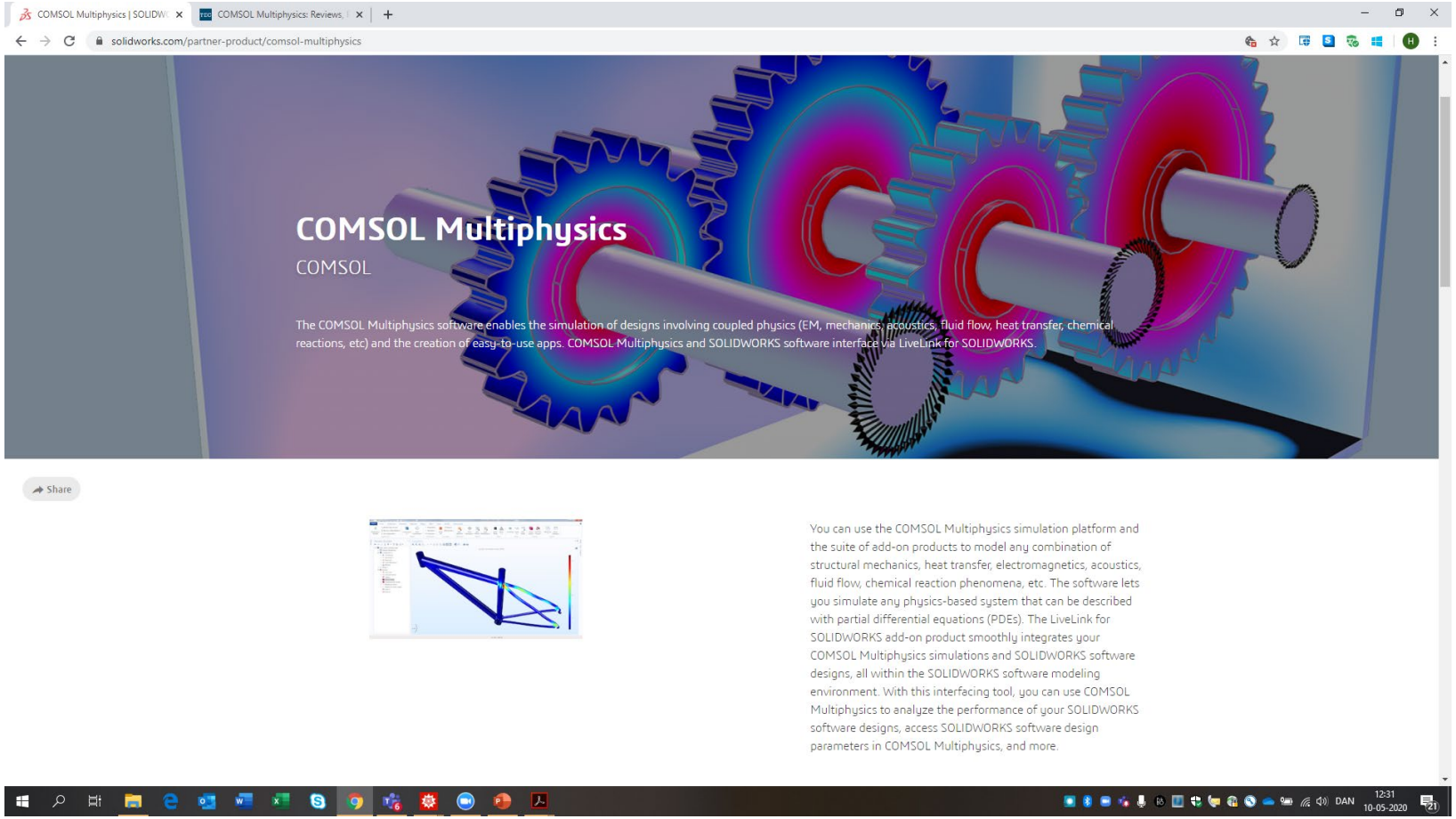
$$(\lambda+\mu)\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+\mu\left(\frac{\partial^2 u_y}{\partial x^2}+\frac{\partial^2 u_y}{\partial y^2}+\frac{\partial^2 u_y}{\partial z^2}\right)+F_y=0$$

$$(\lambda+\mu)\frac{\partial}{\partial z}\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+\mu\left(\frac{\partial^2 u_z}{\partial x^2}+\frac{\partial^2 u_z}{\partial y^2}+\frac{\partial^2 u_z}{\partial z^2}\right)+F_z=0$$

These last 3 equations are the steady Navier–Cauchy equations, which can be also expressed in vector notation as

$$(\lambda+\mu)\nabla(\nabla\cdot\mathbf{u})+\mu\nabla^2\mathbf{u}+\mathbf{F}=\mathbf{0}$$

Doing this for complex PDE's and going to PDE's in 3 dimensions is by the many considered to be too complicated. People therefore often blindly use software packages that are commercially available (expensive) (typically finite element based, but ideas are the same). If you use these packages at some stage, be aware to use them correctly and to estimate errors. Buildings have crashed because of wrong use.



Exercise:

Consider the elliptic partial differential equation

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=1+x+y \qquad \text{for} \qquad (x,y)\in\Omega$$

where $\Omega=\{(x,y)|0<x<1,0<y<1\}$ and $u(x,y)=0$ for $(x,y)\in\partial\Omega$

- Set up the system of linear equations for N = 4,8,16,... and solve
- Perform Richardson extrapolation and error estimation for $u(0.5,0.5)$