Wednesday 04/11/12:

HW submission: HW2_Team3.pdf
Mention names of all team mates in body of e-mail (in Latin letters) as well as in PDF file
Cc all team members!

Time conflict Th 11-13: Move "open exercise" with Stefan Kopecz to Monday 15-17 in 2421.
→ Notice: Stefan supports both Meister and Gobbert ⇒ You can ask any questions in all seminars.

We now want to develop a Matlab solution to this problem.

Advantage: - fast to code (setup using high-level / vectorized commands; linear solver available)

Disadvantage: - memory usage high
- not parallel
- maybe linear solver not the fastest

Basic code:
Use backslash operator \ to solve $A\vec{u} = \vec{b}$ by saying $u = A \backslash b$. So, our task is simply to set up matrix A and vector b!

So, simply try to write code:

Given N for mesh resolution ($\rightarrow$ input to function or hardcode)

$h = 1 / (N+1)$

Need $x_i = ih$, $i = 1, \ldots, N$.

```
for i = 1 : N
    x(i) = i * h
end
```

Really want vector $x \Rightarrow$ there must be a vectorized thinking

$$x = [\, h : h : 1-h \,]$$

"vector from h to $1-h$ in steps of h"

$y = x$ to make same

Set up b vector $b \in \mathbb{R}^{N^2}$:

Want $u(x,y) = \sin^2(\pi x) \sin^2(\pi y)$ on $\Omega = (0,1)^2$

$\Rightarrow f(x,y) = -\Delta u = -u_{xx} - u_{yy} =$

$(-2\pi^2)\left[\cos 2\pi x \sin^2 \pi y + \sin^2 \pi x \cos 2\pi y\right]$

We want $F = f(x,y)$ with above vectors $x, y \in \mathbb{R}^N$

If this worked like this — in vectorized form — then $F \in \mathbb{R}^N$ probably.

$\Rightarrow$ This cannot be correct since we need $f_{ij} = f(x_i, y_j)$ at all N^2 mesh points.

The key is the realization that we need all possible combinations of x_i with y_j $\Rightarrow$

MatLab provides a function for this idea

```
[X, Y] = ndgrid(x, y)
F = f(X, Y)
```

Here, $X, Y \in \mathbb{R}^{N \times N}$ such that each row of X is x and each column of Y is y.

Then $F = f(X, Y) \Leftrightarrow$

$F(i,j) = f(X(i,j), Y(i,j)) = f(x(i), y(j))$

This assumes that $f(x,y)$ is coded with elemental (= componentwise) operations:

```
F = (-2*pi^2) *
    ( cos(2*pi*X) .* sin(pi*Y) .^2
    + ..... )
```

$F \in \mathbb{R}^{N \times N}$

```
b = h^2 * reshape(F, [N^2 1])
```

$b \in \mathbb{R}^{N^2}$

makes $b \in \mathbb{R}^{N^2 \times 1} = \mathbb{R}^{N^2}$

or shorthand for this `b = h^2 * F(:)`

This makes `F(:)` a column vector.

Setup of A :

Basic idea: A is a sum of two Kronecker products of $I \in \mathbb{R}^{N \times N}$ and $T = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{N \times N}$

$$A = I \otimes T + T \otimes I$$

$$\Leftrightarrow A = \text{kron}(I, T) + \text{kron}(T, I)$$

$I = \text{eye}(N) = \text{eye}(N, N)$

This is a dense or full matrix that stores all 0's in I.

better $I = \text{speye}(N)$

sparse function

$$T = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & \ddots \\ & & \ddots & -1 \\ & & & 2 \end{bmatrix} = \text{band matrix}$$

$T = \text{spdiags}(B, d, \underbrace{N, N}_{\text{dimensions}})$

such that $B(:, k) = k\text{-th}$ column of B is on diagonal $d(k)$

$\Rightarrow$ $e = \text{ones}(N, 1) \in \mathbb{R}^N$ vector of 1's

$B = [-e, 2*e, -e] \in \mathbb{R}^{N \times 3}$

$d = [-1 \quad 0 \quad 1]$

$T = \text{spdiags}(B, d, N, N)$

Then $u = A \backslash b \in \mathbb{R}^{N^2}$

$U = \text{reshape}(u, [N\ N]) \in \mathbb{R}^{N \times N}$

mesh(X, Y, U) → Fig. 3.1 (a) in HPCF-2010-2

Other Matlab commands for HW:

xlabel, ylabel, zlabel, title for the plot

max and abs are useful for norm

$$\|u - u_h\|_{L^\infty(\Omega)} = \sup_{(x,y) \in \Omega} |u(x,y) - u_h(x,y)|$$

E = U - Utrue with Utrue = sin(pi*X).^2 .* sin(pi*Y).^2

max(abs(E(:)))

Time code by tic – toc :

```
tic
u = A\b
tsec = toc
```

Due to fill-in in Gaussian elimination, A\b will eventually run out of memory for large N.

For large N, we need to use iterative methods like conjugate gradients (CG)

In Matlab, CG is in function pcg = preconditioned CG, which in CG if we input empty arrays [] in place of the preconditioning matrices

Replace u = A\b by

```
u = zeros(N^2,1)
tol = 1.0e-6 ; maxit = 50000
[u, flag, relres, iter] = pcg(A, b, tol, maxit,
                              [], [], u)
```

↑ initial guess