Compulsory Assignment 3

Shared memory parallelization using OpenMP

Anton Sjöberg, Alessandro Piccolo

Task 1

A 12x12 matrix was created, where the inner 10x10 matrix was set to ones, and the boundary to zeroes. The inner element (3,4) was set to 5.

```
celsius> ./task1 10 10
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 5.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
celsius>
```

Figure 1: Runned the program task1 for a 10 by 10 matrix.

Task 2

A serial solution to the PDE problem was implemented, using the steps given in the assignment. The normalization occurs every 10th iteration and the max iteration is set to 1000. As one can see from the results the residual norm decreases. The relaxation parameter was set to 1, 1.7 and 1,9 for the following figures. The function was set to $f(x,y) = sin(2\pi x)$. The max error L2 norm was set to 0.008.

```
celsius> ./poisson_serial 10 10 1
...: Poisson serial calculation :...
Iteration: 1, residual norm = 0.749826
Iteration: 2, residual norm = 0.236389
Iteration: 3, residual norm = 0.111823
Iteration: 4, residual norm = 0.072331
Iteration: 5, residual norm = 0.056618
Iteration: 6, residual norm = 0.048937
Iteration: 7, residual norm = 0.044738
Iteration: 8, residual norm = 0.042312
Iteration: 9, residual norm = 0.040800
...: Normalized around zero :...
Iteration: 10, residual norm = 0.039785
Iteration: 11, residual norm = 0.039059
Iteration: 12, residual norm = 0.038513
Iteration: 13, residual norm = 0.038089
Iteration: 14, residual norm = 0.037753
Iteration: 15, residual norm = 0.037482
Iteration: 16, residual norm = 0.037259
```

```
Iteration: 17, residual norm = 0.037074
Iteration: 18, residual norm = 0.036919
Iteration: 19, residual norm = 0.036788
.
.
.
...: Normalized around zero :...
Iteration: 990, residual norm = 0.036266
Iteration: 991, residual norm = 0.036266
Iteration: 992, residual norm = 0.036266
Iteration: 993, residual norm = 0.036266
Iteration: 994, residual norm = 0.036266
Iteration: 995, residual norm = 0.036266
Iteration: 996, residual norm = 0.036266
Iteration: 997, residual norm = 0.036266
Iteration: 998, residual norm = 0.036266
Iteration: 999, residual norm = 0.036266

Total number of iterations 1000, final residual norm 0.036266
```
gives the following figure,

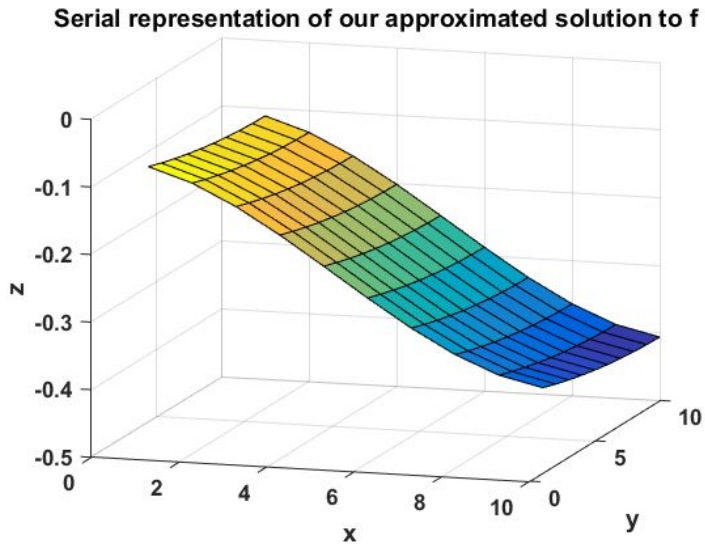

Figure 2: Serial representation of our approximated solution after 1000 iterations on a 10x10 sized matrix with relaxation parameter set to one. The final residual norm is 0.036266.

```
fries> ./poisson_serial 10 10 1.7
Total number of iterations 1000, final residual norm 0.036266
```
gives the following figure,

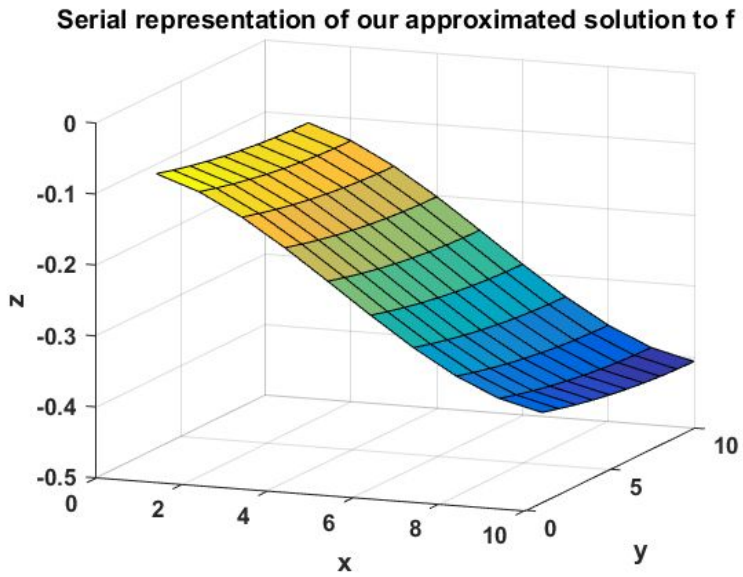

Figure 3: Serial representation of our approximated solution after 1000 iterations on a 10x10 sized matrix with relaxation parameter set to 1.7. The final residual norm is 0.036266.

```
fries> ./poisson_serial 10 10 1.9
Total number of iterations 1000, final residual norm 0.036266
```

gives the following figure,

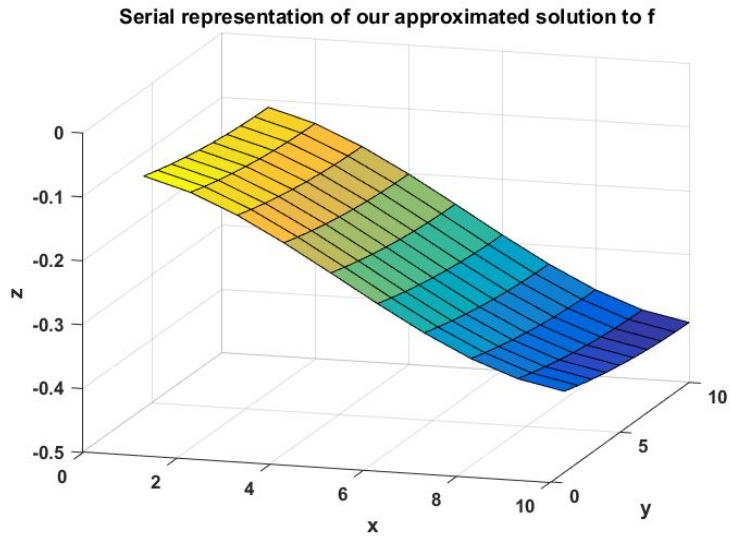

Figure 4: Serial representation of our approximated solution after 1000 iterations on a 10x10 sized matrix with relaxation parameter set to 1.9. The final residual norm is 0.036266.

```
fries> ./poisson_serial 40 50 1
Total number of iterations 1000, final residual norm 0.377310
```

gives the following figure:

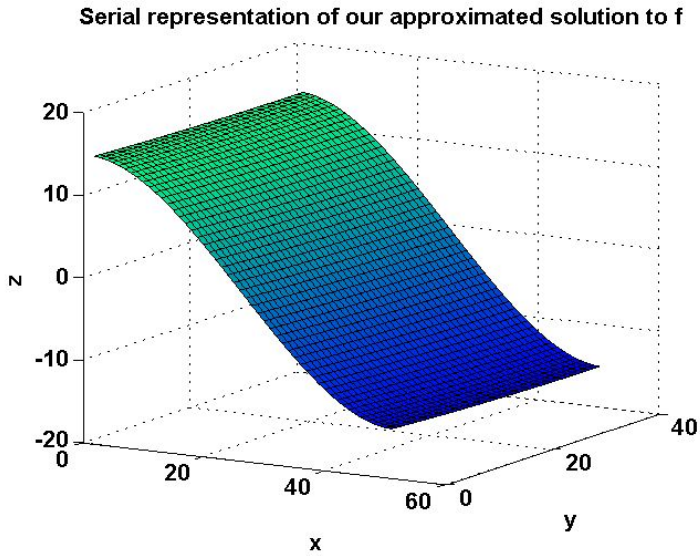

Figure 5: Serial representation of our approximated solution after 1000 iterations on a 40x50 sized matrix with the relaxation parameter set to 1. The final residual norm is 0.377310.

```
fries> ./poisson_serial 40 50 1.7
Total number of iterations 1000, final residual norm 0.377310
```
gives the following figure:

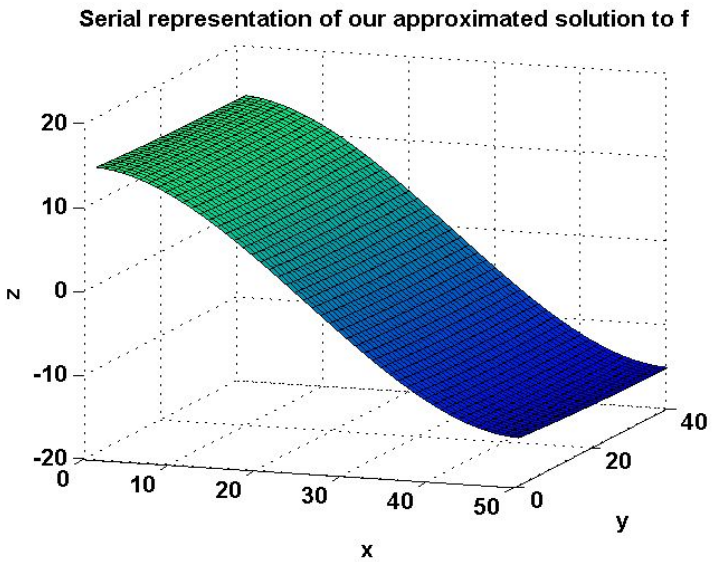

Figure 6: Serial representation of our approximated solution after 1000 iterations on a 40x50 sized matrix with the relaxation parameter set to 1.7. The final residual norm is 0.377310.

```
fries> ./poisson_serial 40 50 1.9
Total number of iterations 1000, final residual norm 0.377310
```
gives the following figure:

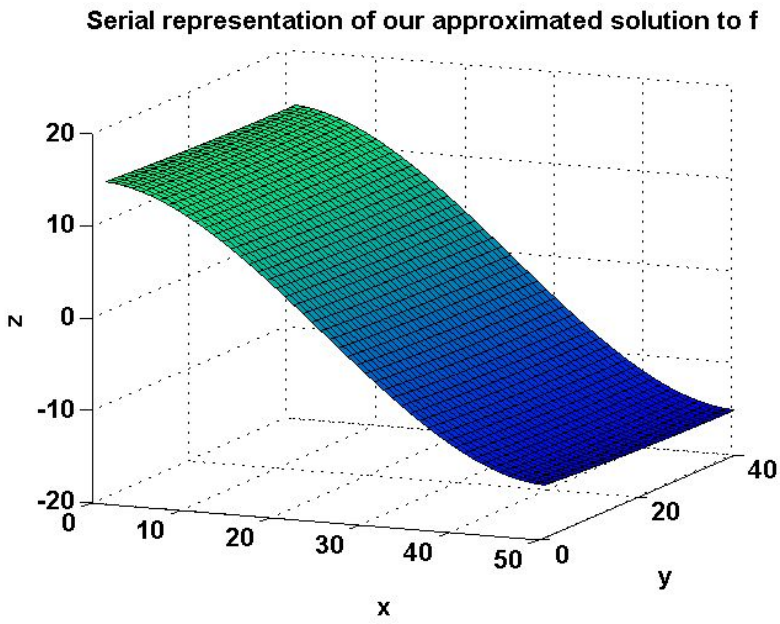

Figure 7: Serial representation of our approximated solution after 1000 iterations on a 40x50 sized matrix with the relaxation parameter set to 1.9. The final residual norm is 0.377310.

Task 3

The following figures were achieved with a 10 by 10 matrix and 1000 by 1000 matrix with relaxation parameter of 1.7 and on 8 threads.

```
celsius> ./poisson_parallel 10 10 1.7 8
...: Poisson serial calculation :...

Elapsed time: 0.712233
Total number of iterations 1000, final residual norm 0.046379
```

gives the following figure,

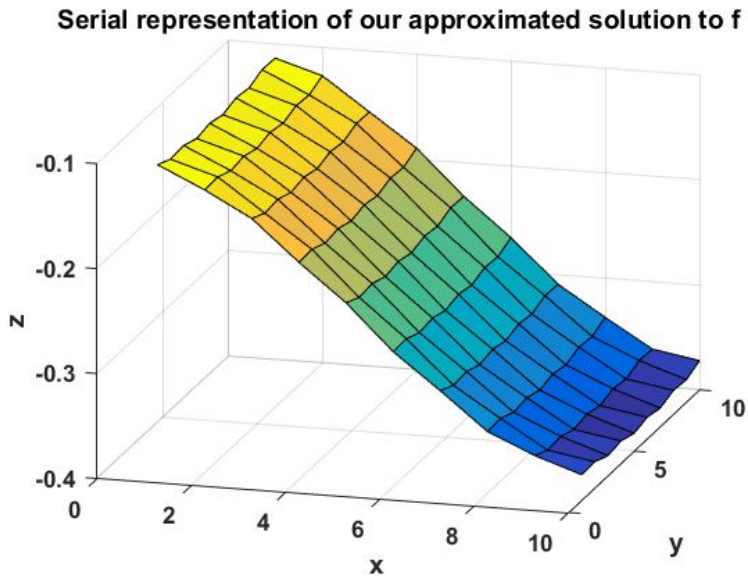

Figure 8: Parallel programs representation of our approximated solution after 999 iterations on a 10x10 sized matrix with the relaxation parameter set to 1.7. The final residual norm is 0.046379.

```
celsius> ./poisson_parallel 40 50 1.7 8
...: Poisson parallel calculation :...

Elapsed time: 0.722594
Total number of iterations 1000, final residual norm 0.518317
```

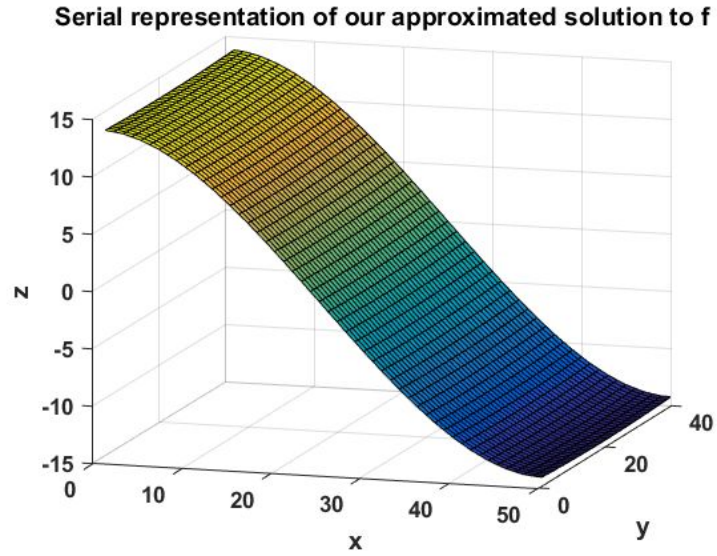

Figure 9: Parallel programs representation of our approximated solution after 999 iterations on a 40x50 sized matrix with the relaxation parameter set to 1.7. The final residual norm is 22.090153.

A relaxation parameter of 1.7 was used with a maximum of 999 iterations and maximum error norm of 0.008. The parallelized program was runned three times on each thread ranging from one to eight. Running the program yields the following result,

```
celsius> ./poisson_parallel 1000 1000 1.7 1
...: Poisson parallel calculation :...
```

```
Elapsed time: 16.423573
Total number of iterations 999, final residual norm 22.090153
```

Which gives the following speedup plot.

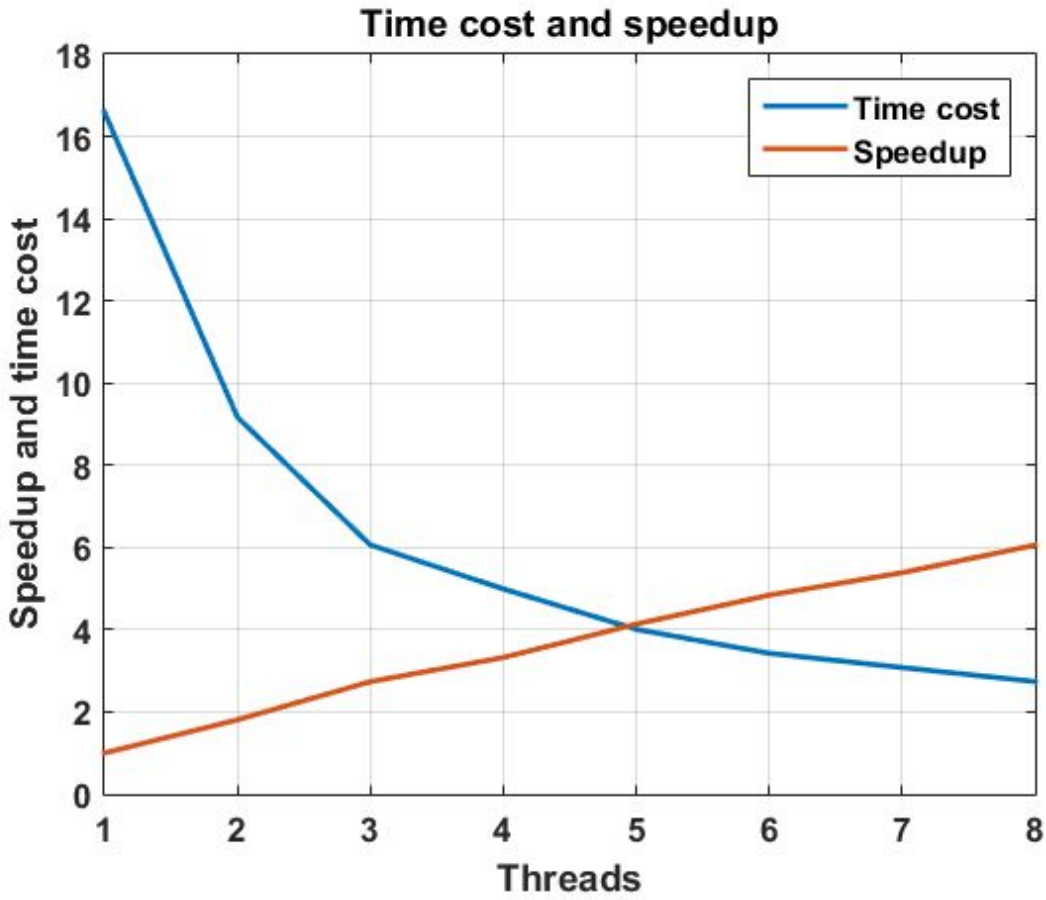

Figure 10: Speedup versus time cost for a matrix of 1000 by 1000 with relaxation parameter of 1.7 and one to eight threads was used.

Table 1: Time cost in seconds for different number of threads used for program poisson_parallel.c.

1 thread	**2 threads**	**3 threads**	**4 threads**	**5 threads**	**6 threads**	**7 threads**	**8 threads**
16,83271	9,232522	6,162515	4,867531	4,037279	3,427179	3,101819	2,75812
16,42357	9,025833	5,991352	5,032488	4,005653	3,426717	3,091921	2,71791
16,51308	9,254303	6,031612	5,077150	3,984197	3,426052	3,044845	2,73926

Discussion

A lower relaxation parameter yields a general higher residual norm. As one can see the figures under task 2 represents a sinus curve, as the inner grid becomes larger this is more obvious. In figure 5, 6 and 7 one can observe that there is maximum peak at z = 10 and minimum peak at -10.

The parallelisation was made with openMP and the code was optimised as best as we could removing unnecessary barriers with nowait. There is a clear speedup of the code when increasing the number of threads. However there are some barriers that could potentially be removed and the cache could possibly be exploited better.

Appendix

The following files were attached to this document in studentportalen,
Makefile
task1.c
poisson_serial.c
poisson_parallel.c

poisson_serial_plot.m

```
% Assigments 3, surf plot of the solution u of the poisson eq
% By Alessandro Piccolo and Anton Sjöberg
clear all; close all;

filename = 'output.txt';
delimiterIn = ' ';
[u,delimiterOut] = importdata(filename)

figure
colormap parula
%mesh(u,'FaceColor','interp','EdgeColor','none','FaceLighting','gouraud')
%figure
%mesh(u,'FaceLighting','gouraud','LineWidth',0.3)
%colormap(parula(5))
surf(u,'FaceLighting','gouraud','LineWidth',0.3)
set(gca,'FontSize',13, 'FontWeight', 'bold');
title('Serial representation of our approximated solution to f')
xlabel('x')
ylabel('y')
zlabel('z')
```

speedupVStime.m

```
% speedup
clear all; clc; close all;

% #threads
threads = [1 2 3 4 5 6 7 8];

% Experimental data

% poisson; 1 thread; 2 threads; 3 thread; 4 thread; 5 thread; 6 thread; 7, thread, thread 8;
```

```
poisson_time = [16.423573 9.232522 6.162515 4.867531 4.037279 3.427179 3.101819 2.758126
                16.832716 9.025833 5.991352 5.032488 4.005653 3.426717 3.091921 2.717910
                16.513080 9.254303 6.031612 5.077150 3.984197 3.426052 3.044845 2.739263];
poisson_time_mean = mean(poisson_time);

% Calculating speedup, S = Tinit/Tnew
speedup = poisson_time_mean(1)./poisson_time_mean(:);

figure
[ax,p1,p2] = plotyy(threads,poisson_time_mean,threads,speedup,
'semilogy','plot', 'LineWidth',2);
hold(ax(1), 'on')
hold(ax(2), 'on')
plot(ax(1),threads,poisson_time_mean,'sb')
plot(ax(2),threads,speedup,'sr')
hold(ax(1), 'off')
hold(ax(2), 'off')
set(ax,'FontSize',14) %, 'FontWeight', 'bold');
title(['Time cost and speedup for Poisson with f = sin(2xpi)']);
ylabel(ax(2),'Speedup') % label left y-axis
ylabel(ax(1),'Time cost [s]') % label right y-axis
xlabel(ax(1),'Threads') % label x-axis
grid(ax(2),'on')
```