Scientific Computing - Exercise Sheet 5

Jonathan Hellwig, Jule Schütt, Mika Tode, Giuliano Taccogna

June 4, 2021

1 Exercise 1

(a) Sketch of the mesh and the computational pattern:

$$C = \begin{bmatrix} 1 & 17 & 33 & 49 & ... \\ 2 & 18 & 34 & 50 & ... \\ 3 & 19 & 35 & 51 & ... \\ 4 & \mathbf{20} & 36 & 52 & ... \\ \mathbf{5} \rightarrow & |\underset{\uparrow}{\overset{\downarrow}{\underline{21}}}| & \leftarrow \mathbf{37} & 53 & ... \\ 6 & \mathbf{22} & 38 & 54 & ... \\ 7 & 23 & 39 & 55 & ... \\ 8 & 24 & 40 & 56 & ... \\ 9 & 25 & 41 & 57 & ... \\ 10 & 26 & 42 & 58 & ... \\ 11 & 27 & 43 & 59 & ... \\ 12 & 28 & 44 & 60 & ... \\ 13 & 29 & 45 & 61 & ... \\ 14 & 30 & 46 & 62 & ... \\ 15 & 31 & 47 & 63 & ... \\ 16 & 32 & 48 & 64 & ... \end{bmatrix}$$

The computational pattern for grid point 21 is given by the fat numbers.

(b) We will describe the average cache load for computing the next Laplace smoothing. The cache line length is 8 and we assume that if one data has been accessed, the following 7 entries were loaded instantly. That means if we load the third item, we loaded item 3 to item 10 in our cache.
We will describe 3 different item positions

- Interior items:
  If we look on the sketch and choose one interior item m of the mesh, we see that if we load the item above $(m-1)$, then we always have load the item below $(m+1)$, too. But the left item has the number $m-16$ and the right item has the number $m+16$. Thus, we have to load them separately.

- Corner items:
  If we look on an corner item, we have one neighbour to the left or to the right and one neighbour above or below. with the same explanation as above it follows that we have to load two caches for the computation.
- Boundary items that were no corner items:
  If we look at an item on the side and not on the corner, there exists either 2 side neighbours (if we are at the top or at the bottom) and one neighbour above or below such that we have to load 3 caches.
  If the item is on the left or on the right side, there are two neighbours above and below and one neighbour either to the left or to the right. Then we have to load 2 caches.

There are 14 times 14 interior points, 4 corner points, 14 left side points as well as 14 right/bottom/top side point. All in all the average number of cache loads per computation is:

$$\frac{14 \cdot 14 \cdot 3 + 4 \cdot 2 + 14 \cdot 3 + 14 \cdot 3 + 14 \cdot 2 + 14 \cdot 2}{16 \cdot 16} = \frac{23}{8} = 2,875$$

(c) Sketch of the mesh which results from the Cuthill-McKee algorithm:

$$C = \begin{bmatrix} ... & ... & ... & ... & ... & ... & ... & ... \\ 27 & ... & ... & ... & ... & ... & ... & ... \\ 20 & 26 & ... & ... & ... & ... & ... & ... \\ 14 & 19 & 25 & ... & ... & ... & ... & ... \\ 9 & 13 & 18 & 24 & ... & ... & ... & ... \\ 5 & 8 & 12 & 17 & 23 & ... & ... & ... \\ 2 & 4 & 7 & 11 & 16 & 22 & ... & ... \\ 0 & 1 & 3 & 6 & 10 & 15 & 21 & ... \end{bmatrix}$$

(d) As one can see from the sketch if we look at interior item $x_{i,j}$, we need one cache line load to access te data value corrseponding to grid point $x_{i,j-1}$ (below), this load gives the processor also access to the data value corresponding to grid point $x_{i-1,j}$ (on the left). The same holds for the grid points $x_{i+1,j}$ (on the right) and $x_{i,j+1}$ (above). That means 2 cache line loads are necessary for the 14 times 14 inner grid points.
For the 60 boundary and corner grid points at most 2 cache line loads are needed. But for example for the point 0 or 1 only one cache line load is necessary.
That means the average number of cache line loads is smaller than, but close to 2.

2 Exercise 2

(a) The connectivity matrix $C \in \mathbb{N}^{10\times 10}$ of the mesh is the following:

$$C = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

(b) The initial partition has the following form:

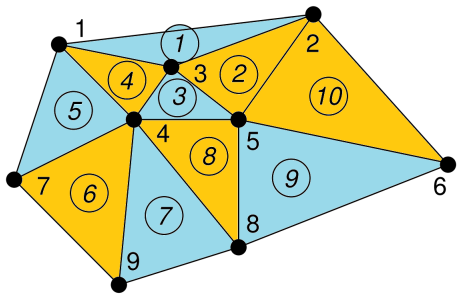

Figure 1: Inital Partition

Notation: i,j in the tables below, describes the edge, which seperates Element i and Element j. We write 1, if i is not in the same partition as j and 0 if i and j are in the same partition. c.w. stands for cut weight, i.c. stands for interchange. That means in the i.c column, we state which elements are interchanged w.r.t the partition stated in the first row.

1,4	1,2	4,5	3,4	2,3	2,10	5,6	3,8	9,10	6,7	7,8	8,9	i.c	c.w
1	1	1	1	1	0	1	1	1	1	1	1	Ini. Part.	11
1	0	0	0	1	0	1	1	1	1	1	1	1,4	8
0	1	1	1	0	1	1	1	1	1	1	1	1,2	10
0	1	1	0	1	0	0	1	1	1	1	1	4,5	8
0	1	0	1	0	0	1	0	1	1	1	1	3,4	**7**
0	0	1	0	1	1	1	0	1	1	1	1	2,3	8
1	1	0	1	1	0	1	1	1	0	1	1	5,6	9
1	1	1	0	0	0	1	1	1	1	0	0	3,8	**7**
1	1	1	1	1	1	1	1	1	1	1	0	9,10	11
1	1	1	1	1	0	0	1	1	1	0	1	6,7	9
1	1	1	1	1	0	1	0	1	0	1	0	7,8	8
1	1	1	1	1	0	1	0	0	1	0	1	8,9	8

Interchange Element 3 and 4:

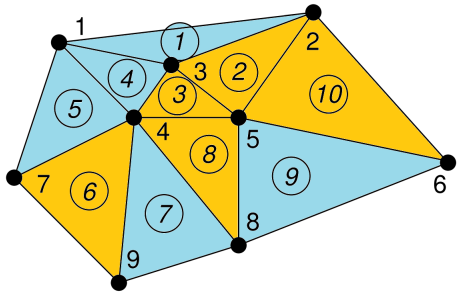

Figure 2: Second Partition

1,4	1,2	4,5	3,4	2,3	2,10	5,6	3,8	9,10	6,7	7,8	8,9	i.c	c.w
0	1	0	1	0	0	1	0	1	1	1	1	2. Part.	7
1	1	0	1	1	1	1	0	1	1	1	1	1,2	10
1	1	1	1	1	0	1	1	1	1	1	1	3,4	11
0	1	1	1	0	0	1	0	1	0	1	1	5,6	7
0	1	0	1	0	0	0	0	1	1	0	1	9,10	7
0	1	0	1	0	0	0	0	1	1	0	1	6,7	**5**
0	1	0	1	0	0	1	1	1	0	1	0	7,8	6
0	1	0	1	0	0	1	1	1	0	1	0	8,9	6

Interchange element 6 and 7:

1,4	1,2	4,5	3,4	2,3	2,10	5,6	3,8	9,10	6,7	7,8	8,9	i.c	c.w
0	0	1	0	1	1	1	0	1	1	1	1	3. Part.	**5**
1	1	0	1	1	1	1	0	1	0	1	0	1,2	8
1	1	1	1	1	0	1	1	1	0	1	0	3,4	9
0	1	0	1	0	1	0	0	1	1	0	0	9,10	**5**
0	1	0	1	0	0	1	0	1	1	0	1	6,7	6
0	1	0	1	0	0	0	1	0	1	1	1	8,9	6

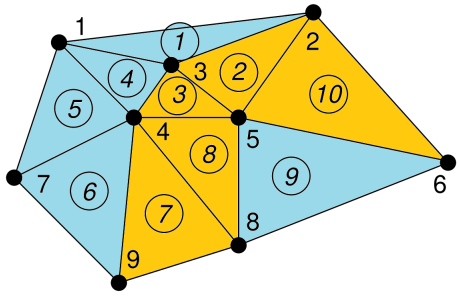

Figure 3: 3. Partition

No further improvement, the algorithm terminates.

The last picture shows, that the algorithm is not creating the best partition because one can see, if we change for example 7 and 9 (no direct neighbours), we get an improved partition. Unfortunately, the algorithm does just recognize improvements if one interchanges direct neighbours.