Numerical Algorithms for Visual Computing II

Assignment 9

Jan Hendrik Dithmar
2031259

Andreas Steinel
2029984

9.1 Getting Symmetric

We have to derive symmetric versions of Gauss-Seidel (Part a) and SOR (Part b):

a) Symmetric Gauss Seidel ($A = L + D + U$):

$$\begin{aligned}
x^{m+1} &= -(D+U)^{-1}L\left[\underbrace{-(D+L)^{-1}Ux^m + (D+L)^{-1}b}_{\text{ordinary Gauss-Seidel}}\right] + (D+U)^{-1}b \\
&= (D+U)^{-1}L(D+L)^{-1}Ux^m - (D+U)^{-1}L(D+L)^{-1}b + (D+U)^{-1}b \\
&= \underbrace{(D+U)^{-1}L(D+L)^{-1}U}_{M_{SGS}}\,x^m + \underbrace{\left((D+U)^{-1} - (D+U)^{-1}L(D+L)^{-1}\right)}_{N_{SGS}} b
\end{aligned}$$

For the case that A spd, this boils down to:

$$\begin{aligned}
x^{m+1} &= (D+L)^{-1}L(D+L)^{-1}Lx^m + \left((D+L)^{-1} - (D+L)^{-1}L(D+L)^{-1}\right) b \\
&= \left((D+L)^{-1}L\right)^2 x_m + \left((D+L)^{-1} - (D+L)^{-1}L(D+L)^{-1}\right) b
\end{aligned}$$

b) Symmetric SOR:

$$\begin{aligned}
x^{m+1} &= (D+\omega U)^{-1}[(1-\omega)D - \omega L]\left(\underbrace{(D+\omega L)^{-1}[(1-\omega)D-\omega U]x^m + \omega(D+\omega L)^{-1}b}_{\text{ordinary SOR}}\right) + \omega(D+\omega U)^{-1}b \\
&= \underbrace{(D+\omega U)^{-1}[(1-\omega)D-\omega L](D+\omega L)^{-1}[(1-\omega)D-\omega U]}_{M_{SSOR}}\,x^m + \underbrace{\left((D+\omega U)^{-1}[(1-\omega)D-\omega L]\omega(D+\omega L)^{-1} + \omega(D+\omega U)^{-1}\right)}_{N_{SSOR}} b
\end{aligned}$$

As a further proof, lets assume $\omega = 1$:

$$\begin{aligned}
x^{m+1} &= \underbrace{(D+U)^{-1}L(D+L)^{-1}U}_{M_{SSOR,\omega=1}}\,x^m + \underbrace{\left((D+U)^{-1}(-L)(D+L)^{-1} + (D+U)^{-1}\right)}_{N_{SSOR,\omega=1}} b \\
&= M_{SGS}x^m + N_{SGS}b
\end{aligned}$$

So, we see that for $\omega = 1$, the SSOR method boils down to the SGS method.

9.2 Preposterous in Hilbert's space

In this exercise, we investigate the Hilbert Matrix with corresponding Hilber vector as given by:

$$\begin{aligned}
a_{i,j} &= \frac{1}{i+j-1} \\
H &= (a_{i,j})_{i,j=1}^n \\
h_i &= \sum_{j=1}^n a_{i,j}
\end{aligned}$$

This problem is a standard problem for numerical analysis which can shows the limitations of numerical methods.

a. We have to compute the real (in the mathematical sense) solution of

$$Hx = h$$

which is for row i defined as follows:

$$\sum_{j=1}^{n} a_{i,j}\, x_i = \sum_{j=1}^{n} a_{i,j}$$

So, one can see that the solution is fullfilled by $x_i = 1$. Therefore, $x = \vec{1}$.

b. One can easily see that the trace of the matrix is not significantly increasing by the addition of one dimension, e.g.,

$$\begin{aligned} \operatorname{tr}(H_{n+1}) &= \sum_{i=1}^{n+1} a_{ii} \\ &= \operatorname{tr}(H_n) + \frac{1}{2\,(n+1)-1} \end{aligned}$$

For $n \to \infty$, this series converges, which can easily be shown with the ratio test (it is always positive, so we omit the absolute values):

$$\begin{aligned} \frac{a_{n+1}}{a_n} &= \frac{\frac{1}{2(n+1)-1}}{\frac{1}{2n-1}} \\ &= \frac{2n-1}{2n+1} < C \end{aligned}$$

Therefore, $\operatorname{tr}(H) = \sum_{i=1}^{n} \lambda_i$ is bound to C which is also less than 1 and therefore it converges absolutely.

So, we know that the trace represents the sum of all eigenvalues. In worst case, we have the same eigenvalues and the smallest eigenvalue is therefore $\frac{\sum_{i=1}^{n} a_{ii}}{n}$, which is also decreasing by increasing n ($n \geq a_{ii} \forall i$. If we have different eigenvalues, the smallest eigenvalue is even smaller and also becomes even smaller.

This leads to the fact that the smallest eigenvalue becomes even smaller with increasing dimension n. In addition, the condition number increases tremendously:

```
dimension        condition number
  2 ->                         19
  3 ->                        524
  5 ->                     476607
 10 ->             16024717801185
 15 ->         398078708382037888
 25 ->         577430030307699507 2
 50 ->       9297756495440922214 4
100 ->      14103599270058093772 8
```

Due to this big condition numbers, an interative solver have problems in determining the true error (compare to chapter 17, especially Theorem 17.4).

c. The CG-error in the maximum norm in comparison to the correct solution $\vec{1}$:

```
dimension       absolute error
  2 -> 0.00000000000000288658
  3 -> 0.00000000000069633188
  5 -> 0.00356480305521222895
 10 -> 0.00008775740479349814
 15 -> 0.00007187035314881740
 25 -> 0.00002137496845033837
 50 -> 0.00001437576758900327
100 -> 0.00049619274308021488
```

d. The Preconditioned CG-error has been computed as

```
dimension       absolute error
  2 -> 0.00000000000000011102
  3 -> 0.00000000000003508305
  5 -> 0.00000000438746028397
 10 -> 0.05328292484288166264
 15 -> 0.05922950956662620392
 25 -> 0.04165658976247055278
 50 -> 0.02999276484401813825
100 -> 0.02835902228564224359
```