Numerical Algorithms for Visual Computing 2008/09
Example Solutions for Assignment 6

Problem 1 (Proving Stuff I)

We should show that

$$F(x) \;=\; \frac{1}{2}(Ax,x)_2 - (b,x)_2$$

is strictly convex and has a unique minimum. First of all, we can write

$$\begin{aligned} F(x) &= \begin{pmatrix} \sum_i a_{1i}x_i,\ x_1 \\ \sum_i a_{2i}x_i,\ x_2 \\ \vdots \\ \sum_i a_{ni}x_i,\ x_n \end{pmatrix} - \begin{pmatrix} b_1, x_1 \\ b_2, x_2 \\ \vdots \\ b_n, x_n \end{pmatrix} \\ &= \frac{1}{2}\sum_i\sum_j a_{ji}x_ix_j - \sum_i x_ib_i. \end{aligned}$$

We can write this as the computed result of the scalar product. As a short reminder, $(x,y)_2 := \langle x,y\rangle := x^\top y$ for $x,y \in \mathbb{R}^n$. Now, we are using some knowledge from multi-dimensional analysis. If the Hessian of a multi-dimensional function is positive definite, then this is a sufficient condition for a strict local minimum.

Let us now compute the Hessian of F step by step. At first, consider the first derivatives of F:

$$\frac{\partial F}{\partial x_k} \;=\; \begin{cases} \frac{1}{2}\sum_i a_{ik}x_k + \frac{1}{2}\sum_i a_{ki}x_k - b_k, & i \neq j \\ \sum_k a_{kk}x_k - b_k, & i = j \end{cases}$$

From this we can compute the second derivatives of F.

$$\frac{\partial^2 F}{\partial x_k \partial x_l} \;=\; \begin{cases} \frac{1}{2}a_{kl} + \frac{1}{2}a_{lk}, & k \neq l \\ a_{kk} & k = l. \end{cases}$$

As the matrix is symmetric, we have that $a_{kl} = a_{lk}$, hence we can see that $H(F(x)) = A$. As A has been given as a symmetric, positive definite matrix, the Hessian of F is therefore also symmetric and positive definite. As we have stated before, the positive definiteness suffices as a condition for a strict local minimum.

Problem 2 (Proving Stuff II)

By use of the function (10.6), we compute

$$\begin{aligned} f_{x,p} = F(x+\lambda p) &= \frac{1}{2}(A(x+\lambda p),(x+\lambda p))_2 - (b, x+\lambda p) \\ &= \underbrace{\frac{1}{2}(Ax,x)_2 - (b,x)_2}_{=F(x)} + \lambda(Ax-b,p)_2 + \frac{1}{2}\lambda^2(Ap,p)_2 \end{aligned}$$

Then, we do the same thing as in exercise 1, i.e. compute the Hessian, or in this case, the second derivative. At first, we compute the first derivative of this function for λ

$$f'_{x,p}(\lambda) = (Ax-b,p)_2 + \lambda(Ap,p)_2$$

and from this

$$f'_{x,p}(\lambda_{\text{opt}}) = (Ax-b,p)_2 + \frac{(b-Ax,p)_2}{(Ap,p)_2}(Ap,p)_2 = 0$$

Now, we again take the second derivative, i.e.

$$f''_{x,p}(\lambda) = (Ap,p)_2 > 0,$$

as A is positive definite and $p \neq 0$. Hence λ_{opt} is a global minimum.

Problem 3 (Scheeming Schemes)

At first, we want to choose B such that we have a splitting $A = B + (A-B)$. By the algorithm 10-1 we compute

$$x_{m+1} = x_m + \lambda_m p_m.$$

By use of (10.9), we set p as $\frac{r}{\|r\|_2}$ for $r \neq 0$ and 0 else. We can plug this in into the definition of lambda in the algorithm, i.e.

$$\lambda_m = \frac{(r_m, r_m)_2}{(Ar_m, r_m)_2}$$

Furthermore, we compute

$$\begin{aligned} x_{m+1} &= x_m + \lambda_m p_m \\ &= x_m + \lambda_m (b - Ax) \\ &= (I - \lambda_m A)x + \lambda_m b \end{aligned}$$

This is our sought iteration function $\phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ that can be described as

$$\phi(x, b) \quad = \quad (I - \lambda(x, b)A)x - \lambda(x, b)b$$

with $\lambda : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ in the form

$$\lambda(x, b) \quad = \quad \begin{cases} \dfrac{\|b - Ax\|_2^2}{(A(b - Ax), b - Ax)_2} & b - Ax \neq 0 \\ 0 & \text{else} \end{cases}$$

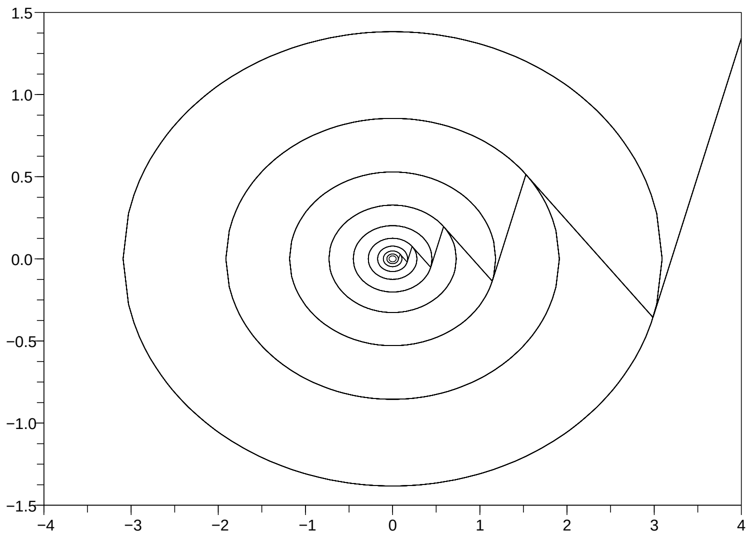
It can be shown by small examples, that the method itself is not linear. For example by considering the results of the next exercise, we can easily see that the changes are not linear.

However, the scheme is consistent, as in the case of $x^* = A^{-1}b$, then there is no longer any change in the solution, as the residuum is $r^* = 0$ and $\lambda(r^*, b) = 0$. Furthermore $x^* = \phi(x^*, b)$ and x^* is a fixed point of the iteration function ϕ.

Problem 4 (A Descent into the Maelstrom)

•

m	$x_{m,1}$	$x_{m,2}$	ε_m
0	4	1.3416408	4.219005
5	0.4368842	-0.0521015	0.439980
10	0.0327105	0.0109714	0.034501
15	0.0035727	-0.0004261	0.003598
20	0.0002675	0.0000897	0.000282
25	0.0000292	-0.0000035	0.000029
30	0.0000022	0.0000007	0.000002
35	0.0000002	0.0000000	0.000000
40	0	0	0



•

One remark: The visualisation of the level lines is a little bit complicated. The Matrix A describes an ellipse with the formula

$$2x^2 + 10y^2 \;=\; f.$$

From this, we can recompute the ellipse on which the calculated iterated value lies on. Now we compute for $x \in [-\sqrt{\frac{f}{2}}, \sqrt{\frac{f}{2}}]$ and

$$y \;=\; \pm\sqrt{\frac{f - 2x^2}{10}}$$

Then we get the desired level lines. From that we can see an interesting property. In case, the diagonal matrix that we used in our example has the same diagonal entries, then the method even converges after the first iteration. By having matrices with different values (and those are arbitrarily large different to each other), then the level lines of F describe stretched ellipses, which may lead to changes of the sign, which may hinder convergence of the method.

Problem 5 (Descending Even Further)

1. By use of $\lambda_{m-1} = \dfrac{\|r_{m-1}\|_2^2}{(Ar_{m-1}, r_{m-1})_2}$ we can follow

$$\begin{aligned}(r_m, r_{m-1})_2 &= (r_{m-1} - \lambda_{m-1} A r_{m-1}, r_{m-1})_2 \\ &= (r_{m-1}, r_{m-1})_2 - \frac{\|r_{m-1}\|_2^2}{(Ar_{m-1}, r_{m-1})_2}(Ar_{m-1}, r_{m-1})_2 \\ &= 0.\end{aligned}$$

   By use of Corollary 10.7 the iterates x_m are optimal. For this result we prove the Corollary.

   For any arbitrary $\xi \in U \setminus \{0\}$ we consider for $\lambda \in \mathbb{R}$

$$f_{x,\xi}(\lambda) = F(x + \lambda\xi).$$

   The function is strictly convex due to the result of exercise 2 and from that we can derive

$$f'_{x,\xi}(\lambda) = (Ax - b, \xi)_2 + \lambda(A\xi, \xi)_2$$

   so that

$$f_{x,\xi}(0) = 0 \quad \Leftrightarrow \quad Ax - b \perp \xi.$$

   which concludes the proofs.

2. The gradient descent method employs in each step an orthogonal projection method with $K = L = \text{span}\{r_{m-1}\}$.

   Ideally, the iterates should be optimal w.r.t. the whole subspace $U = \text{span}\{r_0, \ldots, r_{m-1}\}$, as then it follows for linear independent residual vectors $x_n = x^* = A^{-1}b$. In the gradient descent method however, the approximate solution x_m has only optimality w.r.t. r_{m-1}. As $r \perp p$ is not a transitive operator, it does not necessary hold that with $r_{m-2} \perp r_{m-1}$ and $r_{m-1} \perp r_m$ one could follow $r_{m-2} \perp r_m$.

Problem 6 (A Descent into the Maelstrom with CG)

The cool thing with the CG method is that the overall convergence of the method is partly depending on the size of the system matrix, in our case, we have a 2×2-method, which should converge after two iterations.

m	$x_{m,1}$	$x_{m,2}$	ε_m
0	4	1.3416408	4.219005
1	2.9875519	-0.3562863	3.008722
2	0.0000000	0.0000000	0.000000
3	0	0	0

In general however, one should be careful, as the numerical algorithm may contain some numerical imprecisions due to rounding errors. One should also be cautious, as the method itself relies on the positive definiteness of the matrix A.

Problem 7 (Comparison of different methods)

First of all, let us consider the problem itself. We have been given the problem of a Poisson equation. As we have boundary conditions given us in the form $u(0) = 0$ and $u(1) = 0$, we result in a system

$$\begin{pmatrix} 2 & -1 & & 0 \\ -1 & 2 & -1 & \\ \ddots & \ddots & \ddots & \\ & -1 & 2 & -1 \\ 0 & & -1 & 2 \end{pmatrix} u_h = f_h$$

with $f_h(x) = 2x$ for $x \in \{\frac{1}{8}, \frac{2}{8}, \dots, \frac{7}{8}\}$. The real solution for this problem is the vector $(168, 320, 440, 512, 520, 448, 280)^\top$. In general, a stopping criterion for the CG method would be to check if the L_2-norm is 0. In our case, we want to see what the algorithm does.

For CG:

m	L_2-error	time
1	662.22	0.0
2	386.11	0.0
3	204.36	0.0
4	91.91	0.0
5	33.55	0.0
6	7.81	0.0
7	0	0.0

For Jacobi:

m	L_2-error	time
1	978.52	0.0
10	477.63	0.0
25	145.65	0.0
50	20.12	0.0
75	2.78	0.0
100	0.384	0.0
125	0.05	0.0
150	0.007	0.0
250	0.000003	0.0

For SOR ($\omega_{\text{opt}} = 1.446463$):

m	L_2-error	time
1	825.80	0.0
5	169.12	0.0
10	5.77	0.0
15	0.15	0.0
20	0.00356	0.0
25	0.000079	0.0
30	0.000002	0.0

One can see, that in this setting, the CG-method easily outperforms any of the other methods, however to some extend. In case of larger systems (and therefore larger iteration numbers), the time of the CG-method may become larger than that of the other algorithms, which may need larger iteration times, but take the same time to compute, whereas in the CG-algorithm, the time of a single iteration increases with each step due to the computation of p_{m+1}. In general, one should check whether the given system matrix of a process is better suited for the CG-algorithm or another competitor, such as SOR.