C1 - Assignment 1 Report: Sparse Matrices.

Student Number: 1894945

November 4, 2018

Contents

Abstract

1 Introduction

1.1 Well-posed, direct problems

The problems that will be addressed in the following are always representable in the form:

$$F(x, d) = 0 \tag{1}$$

where x represents the unknown, d the set of data from which the solution depends on and F the functional relation between x and d. Such types of problem are called *direct problems* ([1]).

Definition 1. Let D be the set of admissible data, i.e. the set of data for which problem (1) admits a unique solution x. Let $d \in D$ and denote by δd a perturbation such that $d + \delta d \in D$ and by δx the corresponding change in the solution, in such a way that

$$F(d + \delta d, x + \delta x) = 0$$

Then the solution x depends continuously on the data d if

$$\exists \eta_0(d),\ \exists K_0(d)$$

such that:

$$||\delta d|| \leq \eta_0(d) \implies ||\delta x|| \leq K_0(d)||\delta d||$$

If d is admissible for (1) and if the same problem admits a unique solution x continuously depending on the data d, then the problem is said to be *well-posed* or *stable*. Whenever the aforementioned properties are not satisfied, the problem is said to be *ill-posed*.

1.2 Numerical Methods

In the following, it will always be assumed that problem (1) is well-posed. A numerical method for the approximate solution of the aforementioned equation consists in a sequence of approximate problems:

$$F_n(x_n, d_n) = 0 \quad n \geq 1 \tag{2}$$

with the underlying expectation that $x_n \to x$ as $n \to \infty$, i.e. the approximate solution converges to the exact one.

Definition 2. Consider the problem

$$F_n(x_n, d_n) = 0, \quad n \in \mathbb{N}$$

and denote D_n the set of admissible data for this problem. Then, the numerical method F_n is stable if its solution x_n depends continuously on the the data d_n, for all admissible data $d_n \in D_n$.

Definition 3. The numerical method (2) is convergent iff

$$\forall \epsilon > 0,\ \exists n_\epsilon,\ \exists \delta(n_\epsilon) \quad | \quad \forall n > n_\epsilon,\ \forall \delta d_n : ||x(d) - x_n(d + \delta d_n)|| < \epsilon$$

where δd_n a perturbation of d_n, d_n is an admissible datum for the n^{th} approximate problem, $x_n(d + \delta d_n)$ the corresponding solution of it and $x(d)$ the solution of the exact problem.

1.3 Linear systems

Consider the following linear system:

$$Ax = b$$

where $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$. It is evident that such a problem can be represented in the form (1) as follows:

$$F(x, (A, b)) = 0$$

Assuming A to be non singular, in order to obtain the solution x of the given problem, one should generally invert A, so that:

$$x = A^{-1}b$$

The standard *direct* method for inverting a generic dense matrix is given by computing its LU decomposition [2]. The general computational cost for such a procedure is $\mathcal{O}(n^3)$: the method becomes impractical if A is large or sparse, as it happens in the present assignment.
A naive implementation of a direct method to invert a sparse matrix is a computational waste, both in memory and compute time. One would rather not neither store the zeros in a sparse matrix, nor multiply by them. Certainly, a direct method can be adapted to do neither of these things, leading to the *sparse direct methods*. Alternitavely, one can rely on the so called *iterative methods*. Iterative methods are a class of matrix inversion techniques depending only on the calculation of matrix–vector multiplications, which are typically relatively easy to implement in the case of sparse matrices [2].
The performance of an iterative method is generally evaluated by the number of iterations required to converge to a sufficiently accurate approximation of the actual solution. A generic sparse-matrix–vector multiplication costs $\mathcal{O}(kn)$, k being the average number of non-zero elements in each. Hence, ideally, one desires an iterative method whose number of iterations required to converge is either independent of n or scales sublinearly with respect to n, in order to be competitive with direct methods, without relying on the sparsity of the matrix.

1.3.1 Splitting Methods

In order to implement an iterative method, it is necessary to generate a sequence of approximations $\{x^{(k)}\}$ to the solution. In the case of a splitting method for the problem $Ax = b$, the sequence is obtained as follows:

$$Px^{(k+1)} = Nx^{(k)} + b$$

or, equivalently:

$$x^{(k+1)} = x^{(k)} + P^{-1}r^{(k)}$$

where $A = P - N$ is the *splitting matrix*, P is the preconditioner matrix and $r^{(k)} = b - Ax^{(k)}$. In particular, the inversion of P, which has to be non singular, has not cost more than $\mathcal{O}(n^2)$ operations, in order to have $\mathcal{O}(n^2)$ operations for the whole method (the left multiply with N is always this much).

1.3.2 Gauss-Seidel Method

Let the decomposition of A be redefined as $A = D - (E + F)$, where D is the diagonal of A, $-E$ and $-F$ are the upper-triangular and lower-triangular components of $A - D$.
The Guass-Seidel method is defined by setting $P_G = D - E$ and $N_G = F$. Clearly, inverting P_G costs $\mathcal{O}(n^2)$. It can be proven (see [2], [3]) that, if $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, then the Guass-Seidel

iteration converges for any $x^{(0)}$.[1]
Taking advantage of the triangular form of P, the updating of the i^{th} component of x at the $(k+1)^{th}$ iteration can be written in the following form:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] \quad i = 1, \cdots, n$$

Hence, in the Gauss-Seidel method, at the $(k+1)^{th}$ step, the available values of $x^{(k+1)}$ are used to update the solution.
The algorithm is clearly terminated on the residual $r^{(k)}$, meaning that the iterations stop when

$$||r^{(k)}|| < TOL$$

TOL being some error tolerance. It is, however, not clear with respect to which norm the previous condition has to be satisfied. Even though the choice a norm is completely subjective, L_∞ and L_2 norms are the ones most commonly used (see [2]).

2 Problem setup

2.1 Performed tests

3 Conclusive remarks

Controlling menory leaks.

```
$ valgrind --leak-check=yes ./sparsematrix

==4200== Memcheck, a memory error detector
==4200== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4200== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4200== Command: ./sparsematrix
==4200==
==4200==
==4200== HEAP SUMMARY:
==4200==     in use at exit: 0 bytes in 0 blocks
==4200==   total heap usage: 1,252,428 allocs, 1,252,428 frees, 989,616,540 bytes allocated
==4200==
==4200== All heap blocks were freed -- no leaks are possible
==4200==
==4200== For counts of detected and suppressed errors, rerun with: -v
==4200== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

No memory lost. Everything deleted correctly.
Also did:

```
$ valgrind --tool=callgrind ./sparsematrix
$ kcachegrind
```

First command analyses the perfomances in the terms of load distrubution. kachegrind to visualise the visualise the load distribution results.
INCOLLA FOTO

[1] Clearly, in order to improve convergence of the algorithm, an optimal choice of the *initial guess* $x^{(0)}$, has to be made.

References

[1] A. Quateroni, R. Sacco, F. Saleri; *Numerical Mathematics*, Vol.37, Springer Verlag, (2007).

[2] T. Grafke; *Scientific Computing*, Lecture Notes, University of Warwick, (2018).

[3] W. Hackbush; *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, (1994).