Scientific Software Report: Assignment 2

Daniel Loscos Barroso
<daniel.loscosbarroso@student.kuleuven.be>

November 10, 2018

`Time spent:  21 hours`

Changes

To code from assignment 1

- Removed the hard-coded dimensions for matrixes.
- Changed `gauss` and `gauss_pivot` to avoid passing the dimension of the matrix as an argument.
- Re-coded `gauss` and `gauss_pivot` with an external re-escalation function that is called after the LU decomposition is obtained.
- Procedure `gauss_pivot` now outputs the permutation vector to make it more flexible (should have coded it as an optional argument but did not do this on time).
- Code now uses heap memory allocation, it did not previously.
- Introduced my own eigenvalue module using Lapack with a **precision independent `eig()` function**.

EXTRA: To obtain compiler independence

The original code for this assignment was developed using `gfortran` as reference compiler.

I was able to compile the same code in the first try with `nagfor` the code for assignment 2 run with no problems. However, while running the code from the first assignment I run into a run-time divide by 0 exception. The executable generated by `gfortran` was able to prevent this error automatically by using `NaN` for these values. This result is good enough for me, but adding the compiler option `-ieee=full` allows us to run the code flawlessly.

Finally I tested my code using `ifort` compiler. It compiled on the first try and run with no errors. The conclusion is: **no changes were needed to make the code compiler independent**.

Discussion of the results

Gauss vs. Schur

The results that I am going to discuss in this section are those of `sprec.txt` and `dprec.txt`. These where compiled without any optimization level. For a discussion of the performance results with different optimization levels, go to the next chapter.

Performance wise, the Gauss-pivot method seems to have both higher accuracy and shorter run times than the Schur-complement method for both systems. The accuracy difference can be explained by the number of operations and memory storages needed to obtain every component of the result vector. The process is much simpler in the Gauss-pivot method and fewer rounding errors are accumulated. Regarding the differences in the run time the difference in the big system is much less noticeable than in the small one. My take on these results is that the Schur-complement method has better asymptotic performance but the two systems are just too small to make the difference.

To make a comparison in the memory consumption of the two methods we run Valgrind on the double precision version of the code for each of the methods applied to the two systems. Even if it was a small difference, Gauss-pivot required more memory allocation than the new method. Gauss-pivot needs to allocate the full KKT matrix, while Shchur-complement only needs to store A and G but requires some internal variables to be time efficient by storing matrix multiplications and two different LU decompositions (which are much smaller). The determining factor here is the dimension of A. Let $Dim(A) = n \times m$, then Gauss-pivot needs to store roughly $n^2 + m^2 + 2nm + n + m$ values and Schur-complement needs to store roughly $2n^2 + 2nm + n + m$ values. Therefore, the bigger the ratio $\frac{m}{n}$, the bigger the difference.

EXTRA: Optimization

To easily compare different optimization options I wrote the bash script `optimization_results.sh` which dumps the outputs of the program compiled with four different `-Olevel` options and their respective Valgrind logs.

Higher O levels lead to shorter run times and did not increase the errors. Therefor we can safely assume that the compiler is working as intended. The only exception to this trend is `O3` performing worse than `O2` for the small system. This may be due to some optimization technique that aims to obtain better asymptotic performance but performs worse for small calculations.

Another interesting observation is that at `O2` and `O3`, the Schur-complement method outperforms the Gauss-pivot method for the big system. I do not know which is causing this but I consider it a relevant result for future reference. In fact, at this optimization levels the run time of the Schur-complement method improves while the run time of the Gauss-pivot method doubles. At first I thought this could be due to someone else remotely using the computer but running the tests several times showed consistency in these results.

Regarding the memory usage, Valgrind does not show any changes in the heap usage of the differently optimized programs. My conclusion is that the heap memory usage in the source code is coded as efficiently as it could be.

Raw Results

All the raw results can be found in the folder `outputs`.

Single Precision

```
 System Number 1:
Condition number of KKT matrix= 216.780746
Gauss Pivot method:
Relative 2-norm of the forward error = 3.59564137E-06
Time taken by gauss pivot call is 2.5568999999999999E-005 s.
Schur complement method:
Relative 2-norm of the forward error = 5.82195980E-06
```

```
Time taken by shur-comp.  call is 4.2225999999999999E-005 s.
System Number 2:
Condition number of KKT matrix= 790.295288
Gauss Pivot method:
Relative 2-norm of the forward error = 2.86331706E-05
Time taken by gauss pivot call is 2.7792470000000000E-002 s.
Schur complement method:
Relative 2-norm of the forward error = 2.10997998E-04
Time taken by shur-comp.  call is 2.6715089000000001E-002 s.
```

Single Precision Memory

```
 ==5746== Memcheck, a memory error detector
==5746== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5746== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5746== Command:  ./main
==5746== Parent PID: 31451
==5746==
==5746==
==5746== HEAP SUMMARY:
==5746== in use at exit:  0 bytes in 0 blocks
==5746== total heap usage:  122,959 allocs, 122,959 frees, 44,601,402 bytes allocated
==5746==
==5746== All heap blocks were freed -- no leaks are possible
==5746==
==5746== For counts of detected and suppressed errors, rerun with:  -v
==5746== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Double Precision

```
 System Number 1:
Condition number of KKT matrix= 216.74757033898985
Gauss Pivot method:
Relative 2-norm of the forward error = 7.8594346472140982E-015
Time taken by gauss pivot call is 2.5863999999999998E-005 s.
Schur complement method:
Relative 2-norm of the forward error = 1.6500802319526260E-014
Time taken by shur-comp.  call is 4.5605999999999997E-005 s.
System Number 2:
```

```
Condition number of KKT matrix= 786.63037347928559
Gauss Pivot method:
Relative 2-norm of the forward error = 1.0686833205356464E-013
Time taken by gauss pivot call is 3.3476751999999999E-002 s.
Schur complement method:
Relative 2-norm of the forward error = 6.1633835226386937E-013
Time taken by shur-comp.  call is 3.1107840000000001E-002 s.
```

Double Precision Memory

```
 ==5339== Memcheck, a memory error detector
==5339== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5339== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5339== Command:  ./main
==5339== Parent PID: 31451
==5339==
==5339==
==5339== HEAP SUMMARY:
==5339== in use at exit:  0 bytes in 0 blocks
==5339== total heap usage:  122,959 allocs, 122,959 frees, 45,868,762 bytes allocated
==5339==
==5339== All heap blocks were freed -- no leaks are possible
==5339==
==5339== For counts of detected and suppressed errors, rerun with:  -v
==5339== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Double Precision Gauss Memory

```
 ==6017== Memcheck, a memory error detector
==6017== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==6017== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==6017== Command:  ./main
==6017== Parent PID: 31451
==6017==
==6017==
==6017== HEAP SUMMARY:
==6017== in use at exit:  0 bytes in 0 blocks
==6017== total heap usage:  61,768 allocs, 61,768 frees, 23,432,354 bytes allocated
==6017==
```

```
==6017== All heap blocks were freed -- no leaks are possible
==6017==
==6017== For counts of detected and suppressed errors, rerun with:  -v
==6017== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Double Precision Schur Memory

```
 ==6439== Memcheck, a memory error detector
==6439== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==6439== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==6439== Command:  ./main
==6439== Parent PID: 31451
==6439==
==6439==
==6439== HEAP SUMMARY:
==6439== in use at exit:  0 bytes in 0 blocks
==6439== total heap usage:  61,668 allocs, 61,668 frees, 22,698,114 bytes allocated
==6439==
==6439== All heap blocks were freed -- no leaks are possible
==6439==
==6439== For counts of detected and suppressed errors, rerun with:  -v
==6439== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

O0

```
 System Number 1:
Condition number of KKT matrix= 216.74757033898985
Gauss Pivot method:
Relative 2-norm of the forward error = 7.8594346472140982E-015
Time taken by gauss pivot call is 2.6100000000000001E-005 s.
Schur complement method:
Relative 2-norm of the forward error = 1.6500802319526260E-014
Time taken by shur-comp.  call is 4.1875999999999997E-005 s.
System Number 2:
Condition number of KKT matrix= 786.63037347928559
Gauss Pivot method:
Relative 2-norm of the forward error = 1.0686833205356464E-013
Time taken by gauss pivot call is 1.8315934999999998E-002 s.
Schur complement method:
```

```
Relative 2-norm of the forward error = 6.1633835226386937E-013
Time taken by shur-comp.  call is 3.8396765999999999E-002 s.
```

O0 Memory

```
 ==5820== Memcheck, a memory error detector
==5820== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5820== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5820== Command:  ./main
==5820== Parent PID: 5790
==5820==
==5820==
==5820== HEAP SUMMARY:
==5820== in use at exit:  0 bytes in 0 blocks
==5820== total heap usage:  122,959 allocs, 122,959 frees, 45,868,762 bytes allocated
==5820==
==5820== All heap blocks were freed -- no leaks are possible
==5820==
==5820== For counts of detected and suppressed errors, rerun with:  -v
==5820== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

O1

```
 System Number 1:
Condition number of KKT matrix= 216.74757033898985
Gauss Pivot method:
Relative 2-norm of the forward error = 7.8594346472140982E-015
Time taken by gauss pivot call is 1.2106999999999999E-005 s.
Schur complement method:
Relative 2-norm of the forward error = 1.6500802319526260E-014
Time taken by shur-comp.  call is 2.4073000000000001E-005 s.
System Number 2:
Condition number of KKT matrix= 786.63037347928559
Gauss Pivot method:
Relative 2-norm of the forward error = 1.0686833205356464E-013
Time taken by gauss pivot call is 9.0397320000000000E-003 s.
Schur complement method:
Relative 2-norm of the forward error = 6.1633835226386937E-013
Time taken by shur-comp.  call is 2.0927918000000000E-002 s.
```

O1 Memory

```
 ==5861== Memcheck, a memory error detector
==5861== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5861== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5861== Command:  ./main
==5861== Parent PID: 5790
==5861==
==5861==
==5861== HEAP SUMMARY:
==5861== in use at exit:  0 bytes in 0 blocks
==5861== total heap usage:  122,959 allocs, 122,959 frees, 45,868,762 bytes allocated
==5861==
==5861== All heap blocks were freed -- no leaks are possible
==5861==
==5861== For counts of detected and suppressed errors, rerun with:  -v
==5861== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

O2

```
 System Number 1:
Condition number of KKT matrix= 216.74757033898985
Gauss Pivot method:
Relative 2-norm of the forward error = 7.8594346472140982E-015
Time taken by gauss pivot call is 1.0001000000000000E-005 s.
Schur complement method:
Relative 2-norm of the forward error = 1.6500802319526260E-014
Time taken by shur-comp.  call is 2.1033999999999999E-005 s.
System Number 2:
Condition number of KKT matrix= 786.63037347928559
Gauss Pivot method:
Relative 2-norm of the forward error = 1.0686833205356464E-013
Time taken by gauss pivot call is 1.5310195000000000E-002 s.
Schur complement method:
Relative 2-norm of the forward error = 6.1633835226386937E-013
Time taken by shur-comp.  call is 8.0537479999999995E-003 s.
```

O2 Memory

```
 ==5899== Memcheck, a memory error detector
```

```
==5899== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5899== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5899== Command:  ./main
==5899== Parent PID: 5790
==5899==
==5899==
==5899== HEAP SUMMARY:
==5899== in use at exit:  0 bytes in 0 blocks
==5899== total heap usage:  122,959 allocs, 122,959 frees, 45,868,762 bytes allocated
==5899==
==5899== All heap blocks were freed -- no leaks are possible
==5899==
==5899== For counts of detected and suppressed errors, rerun with:  -v
==5899== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

O3

```
 System Number 1:
Condition number of KKT matrix= 216.74757033898985
Gauss Pivot method:
Relative 2-norm of the forward error = 7.8594346472140982E-015
Time taken by gauss pivot call is 1.0064999999999999E-005 s.
Schur complement method:
Relative 2-norm of the forward error = 1.6500802319526260E-014
Time taken by shur-comp.  call is 1.9242999999999999E-005 s.
System Number 2:
Condition number of KKT matrix= 786.63037347928559
Gauss Pivot method:
Relative 2-norm of the forward error = 1.0686833205356464E-013
Time taken by gauss pivot call is 1.5684936000000000E-002 s.
Schur complement method:
Relative 2-norm of the forward error = 6.1633835226386937E-013
Time taken by shur-comp.  call is 8.1227710000000009E-003 s.
```

O3 Memory

```
 ==5959== Memcheck, a memory error detector
==5959== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5959== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
```

```
==5959== Command:  ./main
==5959== Parent PID: 5790
==5959==
==5959==
==5959== HEAP SUMMARY:
==5959== in use at exit:  0 bytes in 0 blocks
==5959== total heap usage:  122,959 allocs, 122,959 frees, 45,868,762 bytes allocated
==5959==
==5959== All heap blocks were freed -- no leaks are possible
==5959==
==5959== For counts of detected and suppressed errors, rerun with:  -v
==5959== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```