CME 211 Software Development for Scientists and Engineers - Final Project

Leon H. Kloker

12/12/2022

1 Summary

This project revolves around solving the steady-state heat equation in 2 dimensions on a rectangular domain via central finite differences with C++. The domain has periodic boundaries in the lateral directions and Dirichlet boundaries at the top and bottom. The arising sparse linear equation system is solved iteratively with the Conjugate Gradient algorithm. The iterate solutions are written to files and can be read with a python postprocessing script in order to visualize the solution.

2 CG solver

In order to implement the CG solver in C++, we have to write our own library of linear algebra functions in order to perform matrix-vector or vector-vector operations. In the file `matvecops.cpp`, the following functions were implemented in order to avoid duplicate code in the CG algorithm:

1. `vector_addition` - This function adds to vectors together elementwise.
2. `vector_subtraction` - This function subtracts two vectors from each other.
3. `scalar_multiplication` - This function multiplies every element in a vector by the given scalar.
4. `scalar_product` - This function calculates the scalar product of two vectors.
5. `l2_norm` - This function calculates the L2 norm of a vector.

Moreover, in `sparse.cpp`, a SparseMatrix class is implemented which comprises the following function:

1. `AddEntry` - This function adds a new entry to the sparse matrix.

2. `ConvertToCSR` - This function converts the matrix to CSR format.

3. `MulVec` - This function multiplies the matrix by a given vector.

Finally, the CG solver method itself is implemented in `CGSolver.cpp`, where the `CGSolver` method takes a SparseMatrix, right-hand side vector, initial solution guess and a tolerance. Using the methods previously described, the implementation of the CG solver pseudo-code in algorithm 1 is straightforward.

Algorithm 1 CG solver

Initialize x_0
$r_0 = b - Ax_0$
L2normr0 = $||r_0||_2$
$p_0 = r_0$
$n = 0$
while $n <$ nmax **do**
    $\alpha = r_n^T r_n / p_n^T A p_n$
    $x_{n+1} = x_n + \alpha p_n$
    $r_{n+1} = r_n - \alpha A p_n$
    L2normr = $||r_{n+1}||_2$
    if L2normr / L2normr0 $<$ threshold **then**
        break
    end if
    $\beta = r_{n+1}^T r_{n+1} / r_n^T r_n$
    $p_{n+1} = r_{n+1} + \beta p_n$
    $n = n + 1$
end while

3 Exemplary Usage

Build the executable file:

```
$make
g++ -std=c++11 -Wall -Wconversion -Wextra -Wpedantic -c main.cpp
g++ -std=c++11 -Wall -Wconversion -Wextra -Wpedantic -c CGSolver.cpp
g++ -std=c++11 -Wall -Wconversion -Wextra -Wpedantic -c COO2CSR.cpp
g++ -std=c++11 -Wall -Wconversion -Wextra -Wpedantic -c matvecops.cpp
g++ -std=c++11 -Wall -Wconversion -Wextra -Wpedantic -c heat.cpp
g++ -std=c++11 -Wall -Wconversion -Wextra -Wpedantic -c sparse.cpp
g++ -O3 -std=c++11 -Wall -Wconversion -Wextra -Wpedantic main.o CGSolver.o
heat.o sparse.o COO2CSR.o matvecops.o -o main
```

Solve the steady-state heat equation for the parameters given in the input file with the CG solver and write the iterate solutions to sol files:

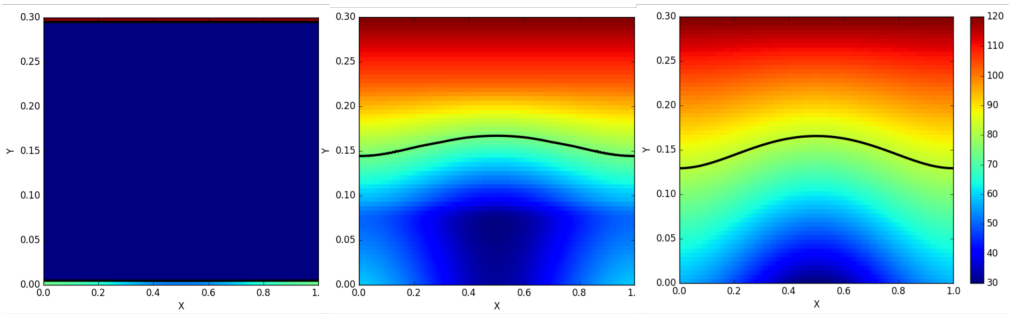

Figure 1: Solution iterates of the CG method from left to right: Initial guess, solution after 80 iterations, converged solution after 157 iterations. The black line is the isoline of the average temperature in the rod.

```
$ ./main input2.txt sol
SUCCESS: CG solver converged in 157 iterations.
```

Visualize the solution in file sol.txt of the problem specified in input.txt:

```
$ python3 postprocess.py input2.txt sol157.txt
Input file processed:  input2.txt
Mean Temperature:  81.83170
```

4 Visualization

The `postprocessing.py` script also saves the created visualizations to output files. Figure 1 shows solutions iterates for a given problem setting.

5 List of files

1. `CGSolver.cpp, CGSolver.hpp` [Canvas, 2022a]
2. `COO2CSR.cpp, COO2CSR.hpp` [Canvas, 2022b]
3. `heat.cpp, heat.hpp` [Canvas, 2022a]
4. `matvecops.cpp, matvecops.hpp`
5. `sparse.cpp, sparse.hpp` [Canvas, 2022a]
6. `main.cpp` [Canvas, 2022a]
7. `makefile`
8. `postprocess.py`
9. `input0.txt, input1.txt, input2.txt` [Canvas, 2022b]

References

Canvas. Partially (method signature) provided file. *CME 211, Stanford University*, 2022a.

Canvas. Fully provided file. *CME 211, Stanford University*, 2022b.