Numeric simulation using finite elements

Sparse but not scarce

Charlotte Vavourakis, Martin Fasser

University of Innsbruck

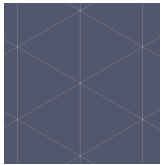
June 22, 2023

Original implementation: dense matrix

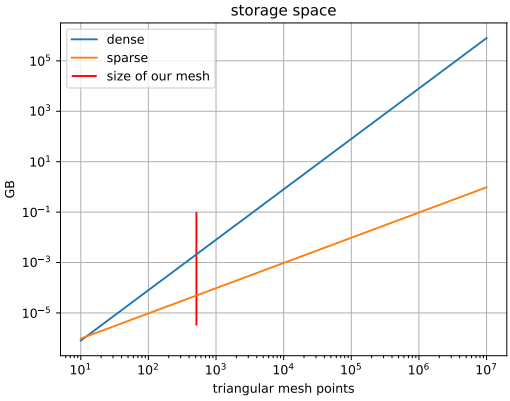
Show recording (or t0/tend) of the solution

Why Sparse?

Storage space



1. $N =$ number of mesh points
2. $\approx 6 \cdot N$ -nonzero elements in matrix
3. storage for dense matrix ($\propto N^2$): 2.11 MB
4. storage for sparse matrix ($\propto N$): 0.05 MB



Assignment: Sparse but not scarce

- Convert PDE solver matrix to sparse format and compare the performance with your dense matrix implementation.
- Use library implementation and compare it with a self-implemented CSR format.

Solution: Sparse but not scarce

- two self-implemented classes for sparse matrices: non-CSR and CSR format.

  Explain a little?

- Eigen library:

  SparseMatrix class, with SparseMatrix::makeCompressed() conversion to CSR

- Bonus, Forward Euler written using matrix multiplication:

  SparseMatrix B * VectorXd u

Different Sparse Formats

- sparse - manual
  - $\text{values} = \begin{bmatrix} 5.3, & 1.5, & 4.2, & 3.1, & 2, & 2.2, & 1.9 \end{bmatrix}$
  - $\text{pos} = \begin{bmatrix} 1, & 4, & 5, & 7, & 9, & 18, & 20 \end{bmatrix}$
- sparse - coo (coordinate list)
  - $\text{values} = \begin{bmatrix} 5.3, & 1.5, & 4.2, & 3.1, & 2, & 2.2, & 1.9 \end{bmatrix}$
  - $\text{col\_ind} = \begin{bmatrix} 1, & 4, & 0, & 2, & 4, & 3, & 0 \end{bmatrix}$
  - $\text{row\_ind} = \begin{bmatrix} 0, & 0, & 1, & 1, & 1, & 3, & 4 \end{bmatrix}$
- sparse - csr (compressed row storage)
  - $\text{values} = \begin{bmatrix} 5.3, & 1.5, & 4.2, & 3.1, & 2, & 2.2, & 1.9 \end{bmatrix}$
  - $\text{col\_ind} = \begin{bmatrix} 1, & 4, & 0, & 2, & 4, & 3, & 0 \end{bmatrix}$
  - $\text{row\_ptr} = \begin{bmatrix} 0, & 2, & 5, & 5, & 6, & 7 \end{bmatrix}$

$$\begin{pmatrix} 0 & 5.3 & 0 & 0 & 1.5 \\ 4.2 & 0 & 3.1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.2 & 0 \\ 1.9 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrix assembly

Conversion coo to csr

coo format:

$\text{row\_ind} = \begin{bmatrix} 0, & 0, & 1, & 1, & 1, & 3, & 4 \end{bmatrix}$

```
for (int i = 0; i < row_ind.size(); i++) {
    row_ptr[row_ind[i] + 1]++;
}
```

how many elements in each row:

$\text{row\_step} = \begin{bmatrix} 0, & 2, & 3, & 0, & 1, & 1 \end{bmatrix}$

```
for (int i = 0; i < dim; i++) {
    row_ptr[i + 1] += row_ptr[i];
}
```

csr format:

$\text{row\_ptr} = \begin{bmatrix} 0, & 2, & 5, & 5, & 6, & 7 \end{bmatrix}$

$$\begin{pmatrix} 0 & 5.3 & 0 & 0 & 1.5 \\ 4.2 & 0 & 3.1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.2 & 0 \\ 1.9 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Sparse - manual

- sparse - manual
  - $\text{values} = \begin{bmatrix} 5.3, & 1.5, & 4.2, & 3.1, & 2, & 2.2, & 1.9 \end{bmatrix}$
  - $\text{pos} = \begin{bmatrix} 1, & 4, & 5, & 7, & 9, & 18, & 20 \end{bmatrix}$
- matrix-vector multiplication

```
for (int k = 0; k < _count; k++) {
    const int i = _pos[k] / _dim;
    const int j = _pos[k] % _dim;
    res[i] +=  _val[k] * u[j];
}
```

$$\begin{pmatrix} 0 & 5.3 & 0 & 0 & 1.5 \\ 4.2 & 0 & 3.1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.2 & 0 \\ 1.9 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Sparse - CSR

- sparse - csr (compressed row storage)
  - $\text{values} = \begin{bmatrix} 5.3, & 1.5, & 4.2, & 3.1, & 2, & 2.2, & 1.9 \end{bmatrix}$
  - $\text{col\_ind} = \begin{bmatrix} 1, & 4, & 0, & 2, & 4, & 3, & 0 \end{bmatrix}$
  - $\text{row\_ptr} = \begin{bmatrix} 0, & 2, & 5, & 5, & 6, & 7 \end{bmatrix}$
- matrix-vector multiplication

```
for (int i=0; i<u.size(); i++) {
    res[i]=0;
    for (int j= _row_ptr[i]; j<_row_ptr[i+1]; j++) {
        res[i]+= _val[j]*u[_col_ind[j]];
    }
}
```

$$\begin{pmatrix} 0 & 5.3 & 0 & 0 & 1.5 \\ 4.2 & 0 & 3.1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.2 & 0 \\ 1.9 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Google Benchmark

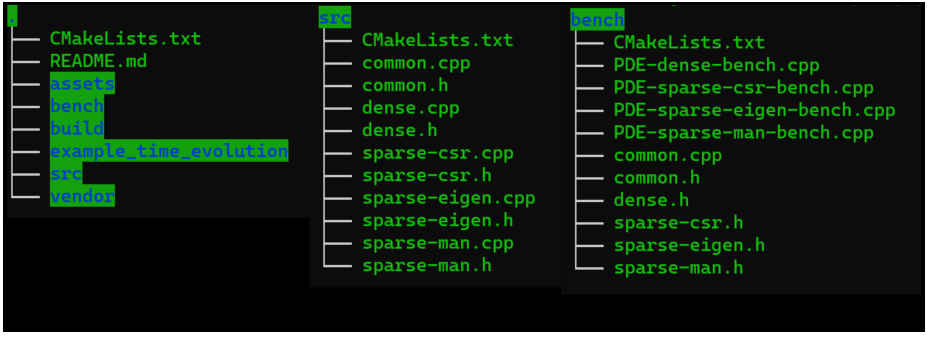
- github.com/google/benchmark
- a library to benchmark code snippets (functions)
- added to to repository as a git submodule

```cpp
#include <benchmark/benchmark.h>

static void BM_SomeFunction(benchmark::State& state) {
  // Perform setup here
  for (auto _ : state) {
    // This code gets timed
    SomeFunction();
  }
}
// Register the function as a benchmark
BENCHMARK(BM_SomeFunction);
// Run the benchmark
BENCHMARK_MAIN();
```

Code organization

- Cmake build 8 targets, common + unique code
- 4 versions of PDE solver
- corresponding benchmarks, testing functions unique to each implementation



Benchmark assembly matrix B

Example output

Benchmark assembly matrix B

Graph or table comparing 4 implementations

Benchmark time evolution

Graph or table comparing 4 implementations

Benchmark assembly B + time evolution for different dt

Graph or table comparing 4 implementations

Conclusions

- Using the sparse format for B resulted in a 10-fold time speed up
- With Eigen even more speed up and code simplification using matrix product expressions

Questions?

Blocks of Highlighted Text

In this slide, some important text will be highlighted because it's important. Please, don't abuse it.

Block

Sample text

Alertblock

Sample text in red box

Examples

Sample text in green box. The title of the block is "Examples".

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Table

Treatments	Response 1	Response 2
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table: Table caption

Theorem

Theorem (Mass–energy equivalence)

$E = mc^2$

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References


John Smith (2012)
Title of the publication
Journal Name 12(3), 45 – 678.