Using Compact Finite Difference C++ Operators

Andrew Carroll

June 14, 2024

1 Introduction

This document outlines the process of implementing compact finite difference operators in C++, utilizing the LAPACK library for matrix operations, and visualizing the results with Python. The workflow includes compiling a C++ script, installing LAPACK, and running a Python script to generate plots.

2 C++ Implementation

2.1 Installing LAPACK

LAPACK (Linear Algebra PACKage) is a comprehensive library for numerical linear algebra. To install LAPACK on a Linux system, use the following commands:

```
sudo apt-get update
sudo apt-get install liblapacke-dev liblapack-dev libblas-dev
```

For other operating systems, refer to the official LAPACK documentation or use the respective package manager.

2.2 Compiling and Running the C++ Code

The provided C++ script calculates matrix operations using LAPACK and generates data files for further analysis. Follow these steps to compile and run the C++ code:

1. Save the source code in a file named `CFD.cpp`.
2. Compile the C++ code using the following command in your terminal:

   ```
   g++ -o CFD CFD.cpp -llapacke -lcblas -lm
   ```

   This command compiles `CFD.cpp` and links it with the LAPACK, CBLAS, and math libraries.
3. Execute the compiled program to generate the results:

   ```
   ./CFD
   ```

   This will create data files named `result_n.dat` and `error_n.dat` for each matrix size n, along with a `timing.dat` file that records the execution times.

2.3 Adding New Matrices

Currently, the matrix operators are defined in the header file `kim.h`. The matrices $\mathbf{P}$ and $\mathbf{Q}$ are structured as follows:

$$\mathbf{P} = \begin{pmatrix} 1 & \gamma_{01} & \gamma_{02} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \gamma_{10} & 1 & \gamma_{12} & \gamma_{13} & 0 & \cdots & \cdots & \cdots & 0 \\ \gamma_{20} & \gamma_{21} & 1 & \gamma_{23} & \gamma_{24} & 0 & \cdots & \cdots & 0 \\ 0 & \beta & \alpha & 1 & \alpha & \beta & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \beta & \alpha & 1 & \alpha & \beta & 0 \\ 0 & \cdots & \cdots & 0 & \gamma_{24} & \gamma_{23} & 1 & \gamma_{21} & \gamma_{20} \\ 0 & \cdots & \cdots & \cdots & 0 & \gamma_{13} & \gamma_{12} & 1 & \gamma_{10} \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & \gamma_{02} & \gamma_{01} & 1 \end{pmatrix}$$

$$\mathbf{Q} = \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} & b_{04} & b_{05} & b_{06} & 0 & \cdots & \cdots & 0 \\ b_{10} & b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & 0 & \cdots & \cdots & 0 \\ b_{20} & b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} & 0 & \cdots & \cdots & 0 \\ -a_3 & -a_2 & -a_1 & 0 & a_1 & a_2 & a_3 & 0 & \cdots & \cdots & 0 \\ 0 & -a_3 & -a_2 & -a_1 & 0 & a_1 & a_2 & a_3 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -a_3 & -a_2 & -a_1 & 0 & a_1 & a_2 & a_3 & 0 \\ 0 & \cdots & \cdots & 0 & -a_3 & -a_2 & -a_1 & 0 & a_1 & a_2 & a_3 \\ 0 & \cdots & \cdots & 0 & -b_{26} & -b_{25} & -b_{24} & -b_{23} & -b_{22} & -b_{21} & -b_{20} \\ 0 & \cdots & \cdots & 0 & -b_{16} & -b_{15} & -b_{14} & -b_{13} & -b_{12} & -b_{11} & -b_{10} \\ 0 & \cdots & \cdots & 0 & -b_{06} & -b_{05} & -b_{04} & -b_{03} & -b_{02} & -b_{01} & -b_{00} \end{pmatrix},$$

The equation we solve is:

$$Pf' = \frac{1}{\Delta x} Qf$$

or

$$f' = \frac{1}{\Delta x} QP^{-1} f$$

To add your own matrix, follow the structure in the provided code.

3 Visualizing with Python

To visualize the results, use Python with the `matplotlib` and `numpy` libraries.

3.1 Python Environment Setup

Ensure you have the necessary Python libraries installed:

```
pip install numpy matplotlib
```

3.2 Running the Python Plotter

Save the provided Python script into a file named `plot.py`. This script reads the generated data files, plots the errors, and performs a convergence test.

Set the order of expected convergence in `plot.py`:

```
# Order parameter
O = 4.0  # You can set this to any value, here it's set to 4.0 for illustration
```

Execute the script with the following command:

```
python3 plot.py
```

This will produce plots comparing the errors for different matrix sizes and a convergence test plot. The plots will be saved as PNG files named `error_comparison_log_scale.png` and `convergence_test_log_scale.png`.

Currently, the convergence test is set for 4th order. If it converges at the expected order, the lines should roughly align, as seen in the example plot:

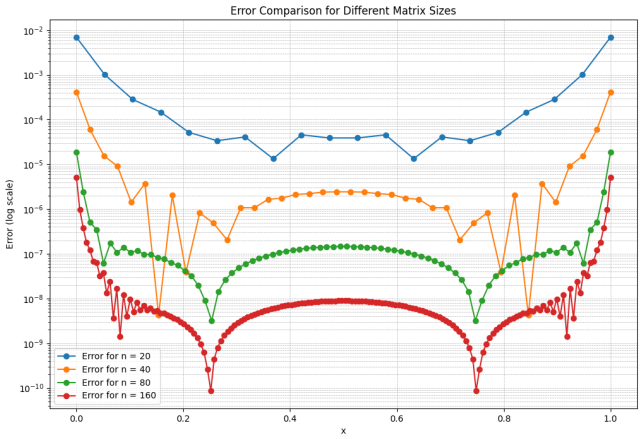

Figure 1: Comparison of errors on a logarithmic scale

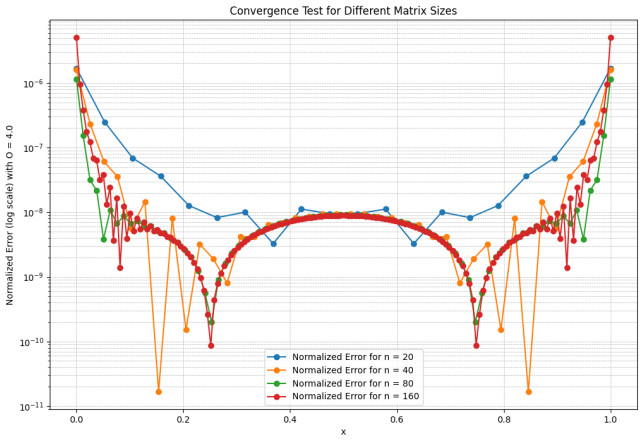

Figure 2: Convergence test on a logarithmic scale