1 abstract

In this project, a C++ implementation of Extrapolated Stabilized Explicit Runge-Kutta methods, in particular ESERK4 and ESERK5, originally introduced by J.Martín-Vaquero and A.Kleefeld, is developed. In addition to that, a C++ template which serves as an unified framework for various ESERK schemes is created. Further, in order to improve performance, manual optimizations are performed. Lastly, an attempt was made to parallelize the C++ implementation using OpenMP.

2 introduction

ifjrgnorgoi

3 Source code translation

There is already Fortran90 codes exist for ESERK4 (fourth order) and ESERK5 (fifth order) which is developed by J.Martín-Vaquero and A.Kleefeld. The first task is to translate these source codes of ESERK4 and ESERK5 from Fortran90 to C++.

The idea of ESERK is as follows: one chooses initial conditions, boundary conditions, discretization schemes, initial time step, final time step, and desired accuracy for the problem to be solved. Based on those inputs, the program first calculates eigenvalue in absolute value of Jacobian of the function. Depending on the eigenvalue stages, internal stages, and initial step size are calculated. Then the stabilized explicit Runge-Kutta (SERK) method calculates the function values at the new position. The SERK function uses a three term recurrence formula and is derived from stability polynomials which in turn constructed based on Chebyshev polynomials. Then Richardson extrapolation is used to calculate the final function value at the new position. After calculating the new function value the algorithm calculates the new step size, number of stages, and internal stages for next iteration. This procedure repeats until the final step is reached.

The ESERK

- C++ feature- templates
-
-
-
-

Algorithms can be included using the commands as shown in algorithm

Algorithm 1 Euclid's algorithm

```
1: procedure EUCLID(a, b)                    ▷ The g.c.d. of a and b
2:     r ← a mod b
3:     while r ≠ 0 do                        ▷ We have the answer if r is 0
4:         a ← b
5:         b ← r
6:         r ← a mod b
7:     return b                              ▷ The gcd is b
```

4 Template creation

Apart from ESERK4 and ESERK5 there exists a family of ESERK algorithms for other orders such ESERK2 and ESERK6. There is also plans for developing further orders in future. This necessitates creation of an unified framework for all ESERK algorithms.

In the previous section the general ESERK algorithm is explained. It is interesting to note that apart from few calculations steps there is only minor differences between ESERK4 and ESERK5.

So the idea is, using C++ template feature, to create a ESERK template in which all order of ESERK algorithm are put together in a single file.

- C++ feature- templates
-
-
-
-

5 Optimization

In the previous section template creation is explained. The initial templated C++ code is run. It is evident from the TABLE that the C++ version is much slower than Fortran and the difference is tied to problem size. That means, when the problem size is very small the difference between two versions is negligible. However when the problem size increases the time differences become very apparent. More importantly these two versions are compiled with GNU compiler with O3 option.

Hence manual optimization is a good choice in order to get good performance.

- Memory access
- Simple substitution
- Avoiding recalculation

6 Parallelization using OpenMP

In the previous section we see the C++ version was performing marginally better than Fortran90 version. However this is not sufficient for larger problem size. Especially when the problem size gets large, for example very fine discretization, computation will take long time to complete.