

**Computional Methods**

**C++**

**Assignment**

***Author:***

*Wiktor Bednarek*

***Supervisors:***

*Dr Irene Moulitsas*

*Dr Peter Sherar*

December 17, 2016

**Abstract**

Thanks to growing branch of science which is Computational fluid dynamics there are many numerical methods provided.

This paper will discuss mathematical description and results of 4 schemes, following schemes will be considered:

* Explicit Upwind
* Implicit Upwind
* Lax-Wendroff
* Richtmyer multi-step

Together with this paper C++ program has been created. It gives as output results of computation of 4 above mentioned methods. It will be estimated which gives the most exact results. Also will be explained why there is an problem to choose one best method since each of them is works better for different inputs. Crucial variable is Courant–Friedrichs–Lewy (CFL) condition which is responsible for output qualty. For all schemes there are different stability conditions, that aspect will be covered as well.

Code is written in C++ and is object-oriented. Also it uses modern C++ methods and features to make it easy to manage and remaining high performance. UML diagram shows connections and dependencies between program classes.

Spis treści

[**1.** **Introduction** 4](#_Toc469772088)

[**2.** **Used methods** 5](#_Toc469772089)

# **Introduction**

Numerical methods allows to solve problems which are not described by any function.

Using C++ language it is possible to perform this task effectively. First significant publication about Computational methods was created by Richard Courant, Kurt Friedrichs, and Hans Lewy in 1928. This scientist’s paper brought CFL (Courant–Friedrichs–Lewy) condition which is one of the most important values in C++ project created to present results in following paper. Numerical methods have many applications one essential is solving differential equations and differential systems equations. Those methods also helps to find approximation of unknown function and find zero points in polynomials when it’s degree is greater than two.

Methods used in this report have widely usage in CFD (Computational fluid dynamics). This branch of science describes fluid flow problem. Numerical methods in CFD helps to approximate various fluid parameters like: pressure, temperature, flow velocity or density. Those parameters are calculated using functions of space and time.

There are number of factors that are responsible for quality of results. Probably most common are precision errors. This error is result of way in which computers stores floating point values, because limited precision, practically in all cases of computation (especially in performing operations like multiplication and division) numbers with fractional parts ≠ 0 will occur errors . Second type of error is truncation error. Computers are not performing computations in continuous domain. Mathematical formulas need to be discretized what brings error because data are lost during that process. No matter how small is value between two next points in discretized function there will be always error as far function is not continuous.

# **Used methods**

* 1. **Initial conditions**

There is given advection equation or wave equation (one dimensional):

(1)

Above equation is used for motion of one-dimesional description, u stands for speed.

Direction of wave depend on u value:

(2)

Equation (1) linear advection (one-dimensional), when x is direction and u is speed. When u is greater than 0 so wave moves in space axis direction (upstream direction). In this report this case is considered.

To solve all schemes it is needed to have initial boundaries conditions. Two sets of conditions are considered:

First set

This boundary condition is based on sign function which is described as follow:

)

Function in space and time is considered ,

where x is space point and t is time point.

When time is equals 0 following boundaries equations are given:

()

Second set

Space initialization function is based on exponential function:

)

)

()

For each initial boundary conditions set analytical solutions are given respectively:

For first set:

()

For second set:

)

Also there is given speed condition:

)

And space domain:

()

* 1. **Explicit Upwind Scheme**

Upwind Schemes is one of numerical discretization methods. Upwind schemes is used for solving hyperbolic partial differential equations.

First order Upwind scheme is described by following equations:

)

()

For program purposed and since u > 0 in this report case only first equation was used to computation. From above scheme equation it is noticeable that method subdivides time and space respectively by and .

.

For stability reasons Courant number is important. Method gives stable results for following condition:

()

Where

)

Explicit Upwind Scheme gives the best results for Courant number which is closest to 1.

* 1. **Implicit Upwind Scheme**

The essential fact about Implicit Upwind Scheme is unconditional stability of this method. It also allows bigger time steps, so to get results less computations are needed comparing to other solutions. Implicit Upwind Scheme is first order accurate as well in space and in time.

Below equation describes that scheme:

)

After transformations described in Appendix following for was used for implementation:

()

Since algorithm is based on previous values the same as Explicit upwind scheme boundaries initialization are necessary for computation.

* 1. **Lax-Wendroff scheme**

Lax-Wendroff scheme is useful when it comes to approximate solutions of hyperbolic partial differential equations. It is second order accurate in time and space either. It’s current instant depends on only previous step value as in other differential schemes.

This Scheme is described by following equation:

)

* 1. **Richtmyer multi-step scheme**

Richtmyer is also called two-step Lax–Wendroff method[4]. In the first step Richtmyer method values for f(u(x, t)) at half time steps (half time steps) and (half grid points are calculated. Also Richtmyer is considered as Jacobian free method. General approach to that method is to calculate it in two steps.

For Richtmyer multi-step scheme stability (Equation 21) condition described by Counant umber is following:

(21)

The same as in previous methods because of it’s predictive character initial boundaries conditions are required to perform computation.

* 1. **Conditions and norms**
     1. **Courant–Friedrichs–Lewy (CFL) condition**

In 1928 Richard Courant, Kurt Friedrichs, and Hans Lewy introduced their paper about computational methods *On the partial difference equations of mathematical physics*. CFL value was introduced in that publication.

Considering one-dimensional case CLFhas it’s general form is as following;

()

**Where**

– time step

– step in space domain

Cmax depends on actual scheme type there are different Cmax values for Explicit Upwind Scheme and for Richtmyer multi-step scheme. The usual value for Cmax is 1, while in Richtmyer multi-step scheme this value equals 2.

When it comes to implementation calculation of CFL is important to know that it is based on Courant number (C, left part of equation).

There are couple conditions for CFL and it is necessary to define following quantities:

1. Time – it shows how system behaves in time
2. Spatial coordinate – defines points in space for problem
3. Spatial problem dimension - represents number n of spatial dimensions
   * 1. **Von Neumann stability**

This approach determines maximum value which provides stable solution. It is defined by dividing two next time values and when result of that operation is greater than stability condition (which is 1) . Below equation represents stability calculation:

()

Stability requirement is as follow:

)

There are many cases when above solution may give false results, for instance G factor is complex number. In that case result from above (25) may be false, that’s why it is often more reasonable to use squared absolute value such cases.

)

* + 1. **Errors and norms**

There are many ways to cause mistake in program. One of the most common is programmer mistake which is often result of inattention. Really often occur mistakes like exceed scope of vector, wrong reference to object, wrong reference to specific element (it happens when programmer tries to access element out of array/vector scope) and typical logic mistakes.

Next errors source is limitation of variables precision. All floating point primitive types (such as float or double) have limited number of values after comma (fractional part). Especially when a lot of multiplications or divisions are performed bigger error will occur.

Considering computer computation there is always discretization error. All operations which computer performs is discrete that why we are losing data each time when we discretizing some continuous function. Also infinite value in computation is result of specific agreement, there are always limited resources for example memory. That king of inconvenience occurs often when functions that’s operate on infinite domain are considered. One of example is Fourier transform because of limited domain undesirable effect like aliasing occurs.

There are plenty of methods that helps to compare two or more data sets. On o them is for instance correlation and convolution. But in program which was made to this report really useful indicators are vector/matrix norms. They’re help to estimate solution quality.

For this exercise purposes three types of norm will be presented:

Uniform norm:

)

Two norm:

)

Second norm is defined as sum of vector’s elements absolute values.

Three norm:

)

Third norm is defined as sum of vector’s squared absolute values.

# **Results**

* 1. **Analytical solution**

Here analytical solution results are presented.

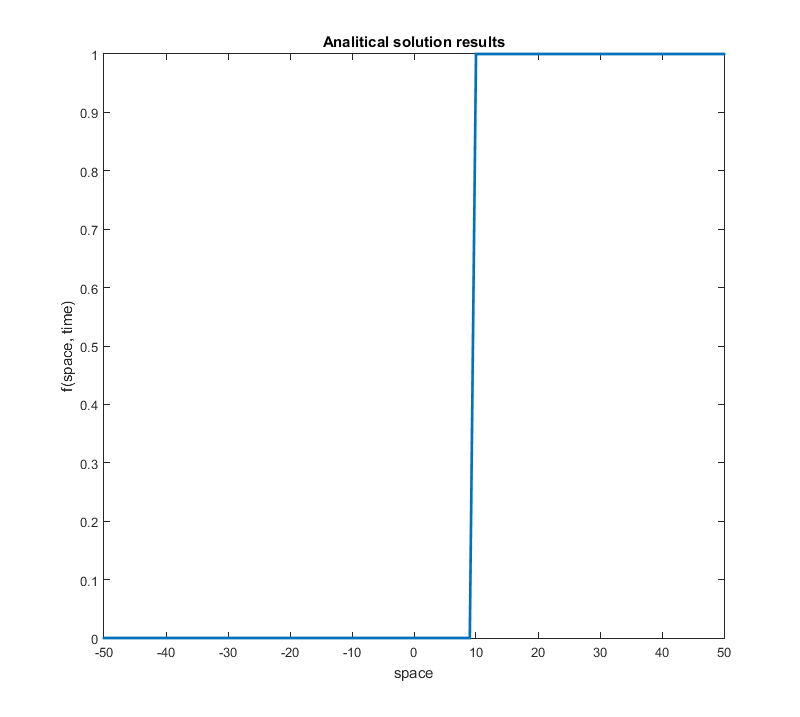


Fig. 1 Analytical solution results for sign type of initial boundary t=5, CFL = 0.999, number of points = 100

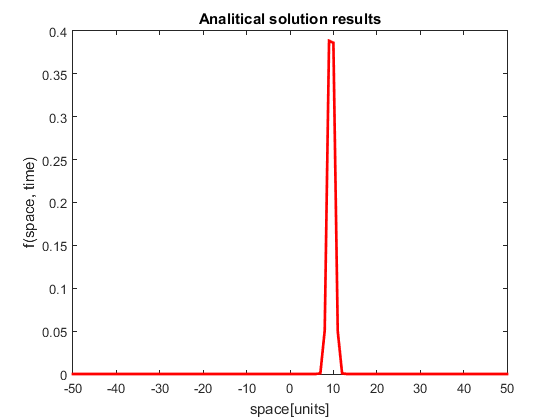


Fig. 2 Analytical solution results for exponential type of initial boundary t=5, CFL = 0.999, number of points = 100

As can be observed output of analytical solution function look very different form analytical solution despite of function is the same in both cases. That means the input data (initial boundaries in this case) naturally dictating output of function.

On the other hand there is significant similarity between those two outputs behavior. Functions soar about point 10 of space in sign and exponential boundary type. Nevertheless in first case from that point it remain at the same level until the end of space scope, but in second graph we can observe that function quickly returns to its initial state after point 10.

* 1. **Explicit Upwind Scheme**