

Reference

Farid Dshabarow

May 25, 2007

Contents

| | | |
|----------|---------------------------------|-----------|
| 1 | Method of Lines | 3 |
| 2 | Finite Volume Method | 5 |
| 2.1 | Introduction | 5 |
| 2.2 | Unstable Method | 7 |
| 2.3 | Lax-Friedrichs Method | 7 |
| 2.4 | Implementation | 8 |
| 3 | Flux Limiter | 9 |
| 4 | Bibliography | 11 |

1 Method of Lines

The first step in constructing PDE is to drag the *WorldModel* block in the diagram. The *WorldModel* block contains general information about the problem, such as the number of grid points the user wish to use to solve the problem. These information are then "propagated" to all the blocks in the diagram that need them.

The heart of the Method of Lines Package are the integrator and derivative blocks. Once the user know the problem it is easy to implement it in *Dymola*. Typically the problem consists of a PDE and initial and boundary conditions. The initial condition can be passed to the *IC* input of the integrator block. For the boundary conditions we must additionally specify some parameters in the *WorldModel* block:

- *bcl* and *bcr* specify whether there is a boundary condition at the left and at the right respectively (0: no, 1: yes).
- *vb* and *ve* specify the first and last unknown variable respectively. The same applies to *icb* and *ice*, which specify the first and last variable for which an initial condition must be specified.

The information in the second point is redundant and could be deduced from the first point. However, for better understanding and use this choice was taken. In the *WorldModel* block the number of grid points can be specified in the *n* parameter. Per default $n = 10$. Let now turn our attention to the integrator block. This block implements PDEs of the form

$$\frac{\partial u}{\partial t} = R \quad (1)$$

where u is the unknown function and R is the right part of the equation that can contain space derivatives, constant values and so on (See Examples). This means that if we do not have the PDE in this form we must first convert it to this form before passing it to the integrator. Once this step is achieved we can build the right part of the equation and at the end pass it to the R input of the integrator block.

Now say that one of the component of the right part of the equation is a derivative of unknown function with respect to space. In this case we can use the derivative blocks that the Package provide. For example for the first-order space derivative use the d/dx block. It is important to note here that in the case of the boundary condition of type

$$\frac{\partial u}{\partial x} = 0 \quad (2)$$

say at the left part of the domain, we must set the *bcl* parameter in the corresponding block to -1 which tells that we are in front of the *symmetry* boundary condition at the left part of the domain. The same applies for the right part of the domain, in which case we must set the parameter *bcr* to -1 . Another important point is the accuracy of the derivative computation. For the computation of the first order space derivative

for example, the second, fourth and sixth order central difference approximations were provided. The user can choose which one to use by specifying the value of the u_x parameter in the *WorldModel* block.

2 Finite Volume Method

Before going into details of FVM Package we need to explain some theory. In the following a short introduction to the finite volume method is made. This short introduction is based on book of Randall Leveque, "Finite Volume Methods for Hyperbolic Problems".

2.1 Introduction

In one dimension, the finite volume method consists in subdividing the spatial domain into intervals, "finite volumes" (or cells), and approximate the integral of the function q over each of these volumes at each time step. Denote the i -th finite volume by

$$C_i = (x_{i-1/2}, x_{i+1/2}) \quad (3)$$

Then the approximation to the average of q in the cell C_i at time t , which we denote with Q_i^t , is

$$Q_i^t \approx \frac{1}{\Delta x} \int_{C_i} q(x, t) dx \quad (4)$$

Remains the question of how to find this approximation. If we think about conservation law, we note that the average within the cell can only changes due to the fluxes at the boundaries (if we assume that no source or sink is present in the cell). The integral form of conservation law is

$$\frac{d}{dt} \int_{C_i} q(x, t) dx = f(q(x_{i-1/2}, t)) - f(q(x_{i+1/2}, t)) \quad (5)$$

If we integrate this expression in time from t to $t + \Delta t$, we obtain

$$\int_{C_i} q(x, t + \Delta t) dx - \int_{C_i} q(x, t) dx = \int_t^{t+\Delta t} f(q(x_{i-1/2}, t)) dt - \int_t^{t+\Delta t} f(q(x_{i+1/2}, t)) dt \quad (6)$$

and dividing by Δx we reach the form

$$\frac{1}{\Delta x} \int_{C_i} q(x, t + \Delta t) dx = \frac{1}{\Delta x} \int_{C_i} q(x, t) dx - \frac{1}{\Delta x} \left(\int_t^{t+\Delta t} f(q(x_{i+1/2}, t)) dt - \int_t^{t+\Delta t} f(q(x_{i-1/2}, t)) dt \right) \quad (7)$$

which gives us an explicit time marching algorithm. This is more clearly seen if we rewrite the expression using the notation we introduced above:

$$Q_i^{t+\Delta t} = Q_i^t - \frac{\Delta t}{\Delta x} (F_{i+1/2}^t - F_{i-1/2}^t) \quad (8)$$

where $F_{i-1/2}^t$ approximates the average flux along the interface $x_{i-1/2}$:

$$F_{i-1/2}^t \approx \frac{1}{\Delta t} \int_t^{t+\Delta t} f(q(x_{i-1/2}, t)) dt \quad (9)$$

As can be seen from the equation (??), in order to find the average at the next time step we need to find the fluxes at the interfaces. The flux at the interface $x_{i-1/2}$ for example, depends on $q(x_{i-1/2}, t)$, which changes with time along the interface and for which we do not know the analytical solution. For this reason we need to find some approximation to this fluxes in order to calculate the averages at the next time step. Let us now see some simple flux approximations.

Examples of fluxes:

Advection equation

Consider the advection equation $q_t + \bar{u}q_x = 0$, where \bar{u} is the fluid velocity. We have seen in the previous chapters, that the flux of the contaminant at some point x , at some time t , could be written as $\bar{u}q(x, t)$. Consider now the flux through the interface $x_{i-1/2}$. If $\bar{u} > 0$ then the flux will be $\bar{u}Q_{i-1}$, otherwise, if $\bar{u} < 0$, the flux will be $\bar{u}Q_i$. Inserting it into the average update rule, we obtain the finite volume method for the advection equation:

$$Q_i^{t+\Delta t} = Q_i^t - \frac{\bar{u}\Delta t}{\Delta x}(Q_i^t - Q_{i-1}^t) \quad (10)$$

if $\bar{u} > 0$, and

$$Q_i^{t+\Delta t} = Q_i^t - \frac{\bar{u}\Delta t}{\Delta x}(Q_{i+1}^t - Q_i^t) \quad (11)$$

if $\bar{u} < 0$.

Diffusion equation

In the advection equation, the flux depends on q : $f(q) = \bar{u}q$. The flux in the diffusion equation depends on the derivative of q :

$$f(q_x) = -\beta q_x \quad (12)$$

where β is the conductivity. If β is space dependent then the flux will depend on space too ($f(x, q_x) = -\beta(x)q_x$). In the following we will assume for simplicity that β is constant.

Now remains the question of how to approximate numerically the diffusion flux. One possibility were:

$$F_{i-1/2} = -\beta \left(\frac{Q_i - Q_{i-1}}{\Delta x} \right) \quad (13)$$

By inserting this flux approximation into the average update rule (??), we obtain:

$$Q_i^{t+\Delta t} = Q_i^t + \frac{\Delta t}{\Delta x^2} \beta (Q_{i-1}^t - 2Q_i^t + Q_{i+1}^t) \quad (14)$$

It is interesting to note, that after some algebraic manipulations, we can write the average update rule in the form

$$\frac{Q_i^{t+\Delta t} - Q_i^t}{\Delta t} = -\frac{1}{\Delta x} (F_{i+1/2}^t - F_{i-1/2}^t) \quad (15)$$

which is equivalent to the finite difference discretization of the conservation law equation $q_t + f(q)_x = 0$. As said in (LeVeque): Many methods can be equally well viewed as finite difference approximations to this equation or as finite volume methods.

Another form of the average update rule is

$$\frac{d}{dt} Q_i = -\frac{1}{\Delta x} (F_{i+1/2}^t - F_{i-1/2}^t) \quad (16)$$

which give us an ODE for each average cell. This form is more suitable for the implementation in Dymola and all Finite Volume Method blocks are based on this form of update rule.

2.2 Unstable Method

The unstable flux just take the arithmetic average of the fluxes based on either side of the interface $x_{i-1/2}$:

$$F_{i-1/2}^t = \frac{1}{2} (f(Q_{i-1}^t) + f(Q_i^t)) \quad (17)$$

By using this flux, the average update rule becomes:

$$Q_i^{t+1} = Q_i^t - \frac{\Delta t}{2\Delta x} (f(Q_{i+1}^t) - f(Q_{i-1}^t)) \quad (18)$$

This method is generally unstable.

2.3 Lax-Friedrichs Method

The Lax-Friedrichs flux is defined as:

$$F_{i-1/2}^t = \frac{1}{2} (f(Q_{i-1}^t) + f(Q_i^t)) - \frac{\Delta t}{2\Delta x} (Q_i^t - Q_{i-1}^t) \quad (19)$$

inserting it into the average update rule, we obtain the Lax-Friedrichs method:

$$Q_i^{t+\Delta t} = \frac{1}{2}(Q_{i-1}^t + Q_{i+1}^t) - \frac{\Delta t}{2\Delta x}(f(Q_{i+1}^t) - f(Q_{i-1}^t)) \quad (20)$$

If we take a closer look at the Lax-Friedrichs flux, we notice that it is similar to the unstable flux, but with the addition of some correction term. This correction term looks like the diffusion flux (??) with

$$\beta = \frac{(\Delta x)^2}{2\Delta t} \quad (21)$$

The Lax-Friedrichs flux can thus be interpreted as the unstable flux plus a numerical diffusion. This numerical diffusion damps the instabilities that arise in the unstable method, however, it damps it too much. Later we will see another method, the Lax-Wendroff method, that add less diffusion.

2.4 Implementation

As already seen, we obtain by starting from the integral conservation law the cell average update rule

$$\frac{d}{dt}Q_i = -\frac{1}{\Delta x}(F_{i+1/2}^t - F_{i-1/2}^t) \quad (22)$$

This ODE is implemented in *FVMIntegrator* block. The use of this block is similar to the use of the integrator block in the *MOL* package. We must specify the initial and boundary conditions and give them to the inputs *IC* and *gcl*, *gcr* respectively. The only difference here is that instead of writing special formulas for the boundary conditions like in the *MOL* package, we extend the domain with additional cells, called *ghost cells*, and fill them with values. This way, the first and the last cell of the domain for example, can use the same formula as all other cells in the domain. Finally, we must pass the *flux* vector *F* to the *F* input of the integrator, which can now use the cell average update rule with the information just provided and compute the averages for the next time step. See *Examples* for better understanding.

3 Flux Limiter

Given a hyperbolic system of m equations we want to solve it with the flux limiter method. We can specify the parameter m as well as the number of cell averages n that we wish in the *WorldModel* block.

The first thing we need is the *integrator* block which will give us the $m \times n + gcl + gcr$ average matrix Q as output with which we can start then the construction. Having matrix Q we can compute the jumps

$$\Delta Q_i = Q_i - Q_{i-1} \quad (23)$$

at each interface $i = 1, \dots, n + gcl + gcr - 1$. The *deltaQ* block achieves this task. The next step is to solve the Riemann problem

$$\Delta Q = R\alpha \quad (24)$$

This can be accomplished by passing ΔQ and $m \times m$ eigenvalue matrix R to the *Riemann* block which will give us the $m \times n + gcl + gcr - 1$ matrix α as output. It is important to note that the eigenvalue matrix R as well as the eigenvalues λ_i must be provided by the user. In the case of a constant coefficient linear hyperbolic system these do not change with time. The next step is to use the α matrix just obtained together with the i -th eigenvalue λ_i to calculate θ_i . The θ_i matrix has the θ_i values in the i -th row and zeros elsewhere. With θ_i matrix at hand we can use for instance *Beam Warming* block to compute $\phi(\theta)$, which is just θ in the Beam Warming method. Beam Warming together with many other methods is implemented in *FluxSolver* block. The user can choose which method to use in the *WorldModel* block by giving the corresponding value to the *method* variable. Here is a list of methods with their corresponding values:

- Upwind method: method = 1
- Lax-Wendroff method: method = 2
- Beam-Warming method: method = 3
- Fromm method: method = 4
- van Leer method: method = 5

The next step is to pass the α and R matrix to the *pWave* block, that will calculate the p -th Wave matrix

$$W_{i-1/2}^p = \alpha_{i-1/2}^p r^p \quad (25)$$

and give the $m \times n + 1$ matrix as output. Each column p in this matrix contains the wave $W_{i-1/2}^p$. By using the same block but giving this time $\tilde{\alpha}$ as input instead of α we can

compute the limited wave \widetilde{W}

$$\widetilde{W}_{i-1/2}^p = \widetilde{\alpha}_{i-1/2}^p r^p \quad (26)$$

The limited $\widetilde{\alpha}$ can be computed by using the block *limitedalpha* which need $\phi(\theta)$ and α matrices as input. With the eigenvalues λ_p and the waves matrices W , \widetilde{W} we can compute the fluxes and fluctuations. This is done by blocks *FluxLimited* and *Fluctuation*. The *FluxLimited* block computes

$$\frac{1}{2}|\lambda^p|(1 - \frac{\Delta t}{\Delta x}|\lambda^p|)\widetilde{W}_{i-1/2}^p \quad (27)$$

If we have m equations in the system then we must use m *FluxLimited* block to compute each term of the sum

$$\frac{1}{2}\sum_{p=1}^m|\lambda^p|(1 - \frac{\Delta t}{\Delta x}|\lambda^p|)\widetilde{W}_{i-1/2}^p \quad (28)$$

and then sum all the outputs of the blocks and pass it to *Flux* input of the integrator. The same applies to the *Fluctuation* block, only that here we must sum the $+$ outputs of the blocks and $-$ outputs of the blocks separately and at the end pass them to the $+$ and $-$ inputs of the integrator respectively. Finally we specify the initial and boundary conditions and connect them to the *IC* and *gcl*, *gcr* inputs respectively.

4 Bibliography

References

- [1] Francois E. Cellier, Ernesto Kofman: Continuous System Simulation, Springer
- [2] Randall J. Leveque: Finite Volume Methods for Hyperbolic Problems, Cambridge Texts in Applied Mathematics
- [3] Bruce R. Munson, Donald F. Young, Theodore H. Okiishi: Fundamentals of Fluid Mechanics, Wiley International edition
- [4] Michael Heath: Scientific Computing
- [5] Stanley J. Farlow: Partial Differential Equations for Scientists and Engineers
- [6] Jose Diaz Lopez: Shock Wave Modeling for Modelica.Fluid library using Oscillation-free Logarithmic Reconstruction, The Modelica Association, 2006, September
- [7] Robert Artebrant, H. Joachim Schroll: Limiter-Free Third Order Logarithmic Reconstruction, SIAM, Vol.28, No. 1, pp. 359-381
- [8] Dmitri Kuzmin: Introduction to Computational Fluid Dynamics Lecture, www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/cfd.html