مجله مهندسی مکانیک مدرس، اسفند 1396، دوره 17، شماره 12، ص‌ص 400-408

ماهنامه علمی پژوهشی

مهندسی مکانیک مدرس

mme.modares.ac.ir

توسعه یک چارچوب نرم‌افزاری متن‌باز برای روش‌های تفاضل محدود با قابلیت پردازش موازی

حسین محمودی‌داریان*

استادیار، مهندسی مکانیک، دانشگاه تهران، تهران

تهران، صندوق پستی 11155-4563، hmahmoodi@ut.ac.ir

اطلاعات مقاله

مقاله پژوهشی کامل

دریافت: 31 شهریور 1396

پذیرش: 05 آذر 1396

ارائه در سایت: 24 آذر 1396

کلید واژگان:

نرم‌افزار متن‌باز

تفاضل محدود

شبکه‌های باسازمان

پردازش موازی

فرابرنامه‌نویسی با الگوها

چکیده

در این مقاله یک چارچوب نرم‌افزاری متن‌باز با نام «چشمه» برای حل عددی معادلات سیال با روش تفاضل محدود روی شبکه‌های باسازمان معرفی می‌گردد. طراحی ساختار داده در نرم‌افزار به گونه‌ای است که چارچوب نرم‌افزاری از شبکه‌های باسازمان با ابعاد فضایی دلخواه پشتیبانی می‌نماید. نرم‌افزار قابلیت تقسیم شبکه عددی به چندین شبکه کوچک‌تر جهت پردازش موازی را دارد. بعلاوه با استفاده از توابعی، پیچیدگی‌های برنامه‌نویسی پردازش موازی برای کاربر بسیار تسهیل شده است. نرم‌افزار با استفاده از قابلیت‌های جدید زبان سی‌پلاس‌پلاس، از جمله قابلیت فرابرنامه‌نویسی با الگوها، توسعه یافته است طوری که امکان محاسبه کارآمد عبارات حسابی و تفاضل محدود را به نحو ساده‌ای برای متغیرهای میدان فراهم می‌آورد. علاوه بر روش‌های تفاضل محدود خطی که به سادگی پیاده‌سازی می‌شوند، روش‌های غیرخطی نظیر روش‌های تسخیر شوک ضرورتاً غیرنوسانی وزن‌دار پیاده‌سازی شده‌اند. همچنین امکان استفاده از روش‌های تفاضل محدود فشرده که منجر به دستگاه معادلات سه‌قطری می‌شوند، در نرم‌افزار وجود دارد. تعریف و اعمال شرایط مرزی مختلف در نرم‌افزار پیش‌بینی گردیده است. تمهیداتی نیز برای ورودی و خروجی از فایل در نظرگرفته شده است. با استفاده از چندین آزمون از جریان‌های تراکم‌ناپذیر و تراکم‌پذیر و نیز از جریان‌های لزج و غیرلزج قابلیت نرم‌افزار نشان داده می‌شود.

Development of an open-source software framework for finite difference schemes with parallel processing feature

Hossein Mahmoodi Darian

Faculty of Engineering Science, University of Tehran, Tehran, Iran.
P.O.B. 11155-4563, Tehran, Iran, hmahmoodi@ut.ac.ir

ARTICLE INFORMATION

Original Research Paper
Received 22 September 2017
Accepted 26 November 2017
Available Online 15 December 2017

Keywords:
Open-Source Software
Finite Difference
Structured Grids
Parallel Processing
Template Metaprogramming

ABSTRACT

In this paper, an open-source software framework named “Chesmeh” for numerical solution of the fluid dynamics equations is introduced. The data structure is designed in a way that the software framework supports structured grids on arbitrary number of spatial dimensions. The software has the ability to decompose the numerical grid into several smaller grids for parallel processing. Furthermore, using some functions, the complexity of the parallel programming is considerably made easier for the user. The software is developed using the new features of the C++ programming language, specially the template metaprogramming feature. In addition to the linear finite difference schemes, which can be simply implemented, the nonlinear schemes such as essentially non-oscillatory shock capturing schemes are implemented. Using the software, it is also possible to use compact finite difference schemes, which lead to a tridiagonal system of equations. Defining and applying different kinds of boundary conditions are also predicted in the software. In addition, utilities are considered for file input and output. Using several test cases of compressible and incompressible flows and viscous and inviscid flows, the capabilities of the software are demonstrated.

1- مقدمه

نرم‌افزارهای متن‌باز امروزه در میان محققان برای شبیه‌سازهای عددی بسیار رایج شده است. علاوه بر رایگان بودن، اصلی‌ترین مزیت این نرم‌افزارها متن‌باز بودن آنها می‌باشد که کاربران آن را با توجه به نیازهای خاص خود تنظیم و یا تغییر می‌دهند، نیازهایی که ممکن است در نرم افزارهای تجاری موجود در بازار پیش‌بینی نشده باشد.

غالب نرم‌افزارهای تجاری و متن‌باز موجود برای شبیه‌سازهای عددی بر مبنای روش حجم محدود[1] یا اجزای محدود[2] می‌باشند. در زمینه دینامیک سیالات محاسباتی می‌توان به نرم‌افزارهایی همانند اُپن‌فوم[3] [2,1]، اِس‌یو[4] [4,3] و کیوا[5] [5] اشاره کرد. برخی نظیر کیوا با زبان فُرترن و برخی دیگر نظیر اُپن‌فوم و اِس‌یو با استفاده از قابلیت شیءگرایی[6] با زبان سی‌پلاس‌پلاس

[1] Finite Volume
[2] Finite Element
[3] OpenFOAM
[4] SU2
[5] KIVA
[6] Object Oriented Programming

Please cite this article using:

برای ارجاع به این مقاله از عبارت ذیل استفاده نمایید:

H. Mahmoodi Darian, Development of an open-source software framework for finite difference schemes with parallel processing feature, *Modares Mechanical Engineering*, Vol. 17, No. 12, pp. 400-408, 2018 (in Persian)

توسعه یافته‌اند. هرچند روش تفاضل محدود بر خلاف روش‌های حجم محدود و اجزای محدود برای هندسه‌های پیچیده به سادگی قابل استفاده نیست، اما برای هندسه‌های ساده کارایی بهتری دارد طوری که می‌توان روش‌های مرتبه بالا با سرعت اجرای مناسب را به سادگی استفاده نمود.

در این مقاله یک چارچوب نرم‌افزاری متن‌باز بومی با نام چشمه (چارچوب شبیه‌سازی متن‌باز شاره‌ها) برای حل معادلات سیالات با شبکه‌های باسازمان با استفاده از روش‌های تفاضل محدود معرفی می‌شود. چارچوب نرم‌افزاری چشمه با استفاده از قابلیت شیءگرایی در زبان سی‌پلاس‌پلاس و نیز با استفاده از قابلیت‌های جدید در این زبان خصوصاً فرابرنامه‌نویسی با الگوها[1]، توسعه یافته است [6-8]. به طور خلاصه قابلیت‌های چارچوب نرم‌افزاری چشمه به شرح ذیل می‌باشد:

1) کاربر روش (یا عملگر) تفاضل محدود خود را می‌تواند با استفاده از توابعی به راحتی تعریف و استفاده می‌نماید.
2) کاربر می‌تواند نقاطی را انتخاب نماید و محاسبات خاصی را برای آن نقاط انجام داد.
3) عبارات محاسباتی، عبارات شرطی، توابع استاندارد ریاضی و توابع تعریف شده توسط کاربر همگی تحت یک طراحی واحد در چشمه پیاده‌سازی شده‌اند.
4) با استفاده از فرابرنامه نویسی با الگوها و الگوهای متغیر امکان تعریف توابع با تعداد آرگومان دلخواه برای اعمال به متغیرهای میدان و نیز امکان تعریف روابط تفاضل محدود غیرخطی وجود دارد.
5) امکان تعریف روش‌های تسخیر شوک[2] از جمله روش‌های ضرورتاً غیرنوسانی وزن‌دار[3] وجود دارد. علاوه بر آن امکان استفاده از روش‌های تفاضل محدود فشرده[4] که منجر به حل دستگاه معادلات سه‌قطری می‌شوند در نرم‌افزار با قابلیت پردازش موازی وجود دارد.
6) یک طراحی واحد برای هر تعداد بُعد فضایی صورت گرفته است و حل مسائل با هر تعداد بُعد فضای (حتی بالاتر از سه‌بُعدی) امکان‌پذیر است.
7) تعریف و اعمال شرایط مرزی به سادگی امکان‌پذیر می‌باشد.

موارد 1، 3، 4 و 6 را می‌توان به عنوان نوآوری در چارچوب نرم‌افزاری چشمه به شمار آورد. همچنین (تا جایی که نویسنده اطلاع دارد) مورد 5 نیز در سایر کُدهای متن‌باز وجود ندارد.

2- شرح چارچوب نرم‌افزاری چشمه

برای حل یک مسأله در چارچوب نرم‌افزاری چشمه در حالت کلی گام‌های زیر نیاز می‌باشد:

1) تعریف پارامترهای مسأله و مقداردهی آنها در داخل برنامه یا از روی یک فایل
2) تعریف یک شبکه عددی
3) تقسیم شبکه به شبکه‌های کوچک‌تر
4) تعریف متغیرهای میدان روی شبکه تعریف شده
5) تعریف وصله‌ها[5] برای نقاط درون میدان و نقاط مرزی
6) تعریف عملگرهای تفاضل محدود
7) ایجاد حلقه برای تکرار یا پیمایش زمانی
8) اعمال عملگرهای تفاضل محدود مرتبط برای نقاط درون میدان یا اعمال شرایط مرزی
9) انتقال داده‌ها برای بروزسازی متغیرهای میدان میان شبکه‌ها
10) خروجی گرفتن و ذخیره‌سازی نتایج روی یک فایل

برای تمامی مراحل فوق در کتابخانه چشمه ابزارهای مورد نیاز در نظر گرفته شده است. در ادامه برای یک مسأله ساده یک برنامه کامل که شامل تمام مراحل فوق می‌باشد شرح داده می‌شود.

2-1- مسأله نمونه، معادله موج خطی دوبُعدی

معادله موج خطی دوبُعدی به همراه شرایط اولیه و مرزی داده شده طبق روابط (1) در نظر گرفته می‌شوند:

$$\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}=0,\quad 0\le x\le 1,\quad 0\le y\le 1$$
$$u(x,y,t)=e^{-10((x-0.5)^2+(y-0.5)^2)}\quad t=0$$
$$u(x,0,t)=0\quad u(0,y,t)=0 \tag{1}$$

گسسته معادله فوق به روش پایادسوی مرتبه اول به صورت رابطه (2) می‌باشد:

$$u_{i,j}^{n+1}=u_{i,j}^{n}-\frac{\Delta t}{\Delta x}(u_{i,j}^{n}-u_{i-1,j}^{n})-\frac{\Delta t}{\Delta y}(u_{i,j}^{n}-u_{i,j-1}^{n}) \tag{2}$$

برای حل این معادله با استفاده از چشمه برنامه زیر نوشته شده است.

```
#include"FrameWork.h"

const real pi = acos(-1.0);

struct init : public MultiVarFun < init, real >
{
    static real apply(real x, real y)
    {
        return exp(-10 * (pow(x - 0.5, 2) + pow(y - 0.5, 2)));
    }
};

int main(int argc, char *argv[])
{
    int rank = 0, size = 1;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

1   Multi<StructuredGrid> multiGrid;
2   GridDistributor distributor(size, rank, multiGrid);

    real L1 = 1.0, L2 = 1.0;

    int imax[] = { 101, 101 };
    int ghost[] = { 1, 1 };
    real length[] = { L1, L2 };
    real dx[] =
        { length[0] / (imax[0] - 1), length[1] / (imax[1] - 1) };
    real dx2[] = { dx[0] * dx[0], dx[1] * dx[1] };

3   distributor.Decompose(2, imax, ghost, length);

4   VectorField<RealField> X("x", "position", 2);
5   ScalarField<RealField> u("u", "dependent variable", 2);
6   ScalarField<RealField>
        un("un", "dependent variable new time step", 2);

    real dt = 0.5*dx[0];

7   X.Allocate(multiGrid);

8   distributor.SetGeometry(X);
```

[1] Template Metaprogramming
[2] Shock capturing
[3] Weighted Essentially Non-Oscillatory (WENO) schemes
[4] Compact finite difference schemes
[5] Patch

```
        u.Allocate(multiGrid);
        un.Allocate(multiGrid);

9       Patch interior("interior", multiGrid);
10      Patch bottom("bottom", multiGrid);
11      Patch top("top", multiGrid);
12      Patch left("left", multiGrid);
13      Patch right("right", multiGrid);

        RealField &x = X(0);
        RealField &y = X(1);

14      interior.MakeInteriorPatch();
15      left.MakeBoundaryPatch(0);
16      right.MakeBoundaryPatch(1, y > 0.0);
17      bottom.MakeBoundaryPatch(2);
18      top.MakeBoundaryPatch(3, x > 0.0);

        // Initial condition

        u() = init::fun(x, y);
        un() = u();

        // stencils

19      Stencil<2> backward = CreateStencil(-1, -1.0)(0, 1.0);

20      MultiDimStencil<2> backwardx =
            CreateStencil(backward, 0, dt / dx[0]);
21      MultiDimStencil<2> backwardy =
            CreateStencil(backward, 1, dt / dx[1]);

        int itermax = 50;
        real time = 0.0;
22      for (int iter = 0; iter < itermax; iter++)
        {
            time = dt*iter;

24          un()(interior) =
                u() - backwardx.D(u()) - backwardy.D(u());
25          un()(left) = init::fun(x - time, y - time);
26          un()(bottom) = init::fun(x - time, y - time);
27          un()(top) =
                u() - backwardx.D(u()) - backwardy.D(u());
28          un()(right) =
                u() - backwardx.D(u()) - backwardy.D(u());

29          distributor.UpdateHaloNodes(un());

            u() = un();
        }
        char fname[] = "case/wave/field";

        IOController out(rank, size, fname, multiGrid, X, u);
        out.Write();
30
31      MPI_Finalize();

    }
```

در اولین خط برنامه فایل FrameWork.h فراخوانی شده است که در حقیقت شامل تمام کتابخانه چشمه می‌باشد. برای توضیح برنامه برخی خطوط شماره‌گذاری شده‌اند. ابتدا در خط 1 یک شبکه باسازمان تعریف می‌شود. کلاس StructuredGrid دارای تمامی خواص مورد نیاز برای یک شبکه با سازمان می‌باشد. از آنجا که برای پردازش موازی شبکه عددی اصلی به چند شبکه کوچک‌تر تقسیم می‌شود لذا نیاز به چندین شبکه با سازمان داریم که با استفاده از کلاس Multi<DataType> انجام شده است. کلاس Multi کلاسی است که آرایه‌ای از متغیرها از نوع DataType را در خود دارد. در خط 2 این شبکه برای تقسیم‌بندی به یک تقسیم‌کننده داده شده است. تقسیم‌کننده شیئی از نوع کلاس GridDistributor می‌باشد که این کلاس توابع لازم جهت تقسیم شبکه را دارا می‌باشد. برای تقسیم‌بندی شبکه متغیرهای size و rank را کتابخانه ام‌پی‌آی مقداردهی می‌کند. متغیر اول تعداد هسته‌های پردازنده و متغیر دوم شماره منحصربفرد هر هسته می‌باشد که از 0 تا size-1 شماره‌گذاری شده‌اند.

در خط 3 پس از تعریف ابعاد و اندازه شبکه در خطوط قبل، تقسیم‌بندی شبکه با توجه به تعداد پردازنده‌ها با استفاده از تابع Decompose صورت می‌گیرد. برای نمونه اگر تعداد هسته‌ها برابر 4 باشد، شبکه با اندازه 101×101 در راستای افقی به یک زیرشبکه 101×26 و سه زیرشبکه 101×25 (در مجموع 4 شبکه) تقسیم می‌شود. شایان ذکر است برای پردازش موازی نیاز است که همانند شکل 1 نقاط حفره (دایره‌های توخالی) برای شبکه در نظر گرفته شود. نیاز به نقاط حفره را اینگونه می‌توان بیان کرد که برای هر رابطه تفاضل محدود در نقطه i نیاز به چندین نقطه همسایه آن وجود دارد. وقتی نقطه i در مرز زیرشبکه باشد برخی نقاط همسایه آن در زیرشبکه دیگر (در حافظه یک پردازنده دیگر) قرار دارند. برای همین به تعداد نیاز باید نقاط حفره تعریف نمود که پیش از انجام محاسبات با استفاده از دستورات پردازش موازی (کتابخانه اِم‌پی‌آی) مقدار نقاط همسایه در زیرشبکه دیگر به این نقاط حفره منتقل و ذخیره شوند و سپس مقدار رابطه تفاضل محدود محاسبه شود. در حقیقت نقاط حفره متناظر نقاط همسایه برای نقاط مرزی زیرشبکه‌ها می‌باشد که در حافظه یک پردازنده دیگر است و باید هم‌مقدار آنها باشند. در شکل 1 پیکان‌های خمیده نشان می‌دهند که داده‌ها چگونه از نقاط همسایه در زیرشبکه همسایه (دایره‌های توپُر) به نقاط حفره (دایره‌های توخالی) منتقل می‌شوند.

در چارچوب چشمه کاربر نیاز به آشنایی با دستورات پردازش موازی ندارد و تنها با استفاده از تابع UpdateHaloNodes انتقال داده‌ها را انجام می‌دهد. مشابه خط 29 هر متغیر میدان که نیاز باشد به عنوان آرگومان به این تابع داده می‌شود تا انتقال داده‌ها برای آن متغیر صورت گیرد.

در خطوط 4 تا 6 متغیرهای میدان تعریف شده‌اند. در خط 7 تخصیص حافظه برای متغیر مکان روی شبکه تعریف شده صورت می‌گیرد و سپس در خط 8 این متغیر مقدار دهی می‌شود. شایان ذکر است تابع فراخوانی شده در خط 8 مقدار دهی را با توجه به اندازه مستطیل ناحیه حل، اندازه شبکه و تعداد نقاط حفره (که در خط 3 مشخص شده‌اند)، انجام می‌دهد. تخصیص

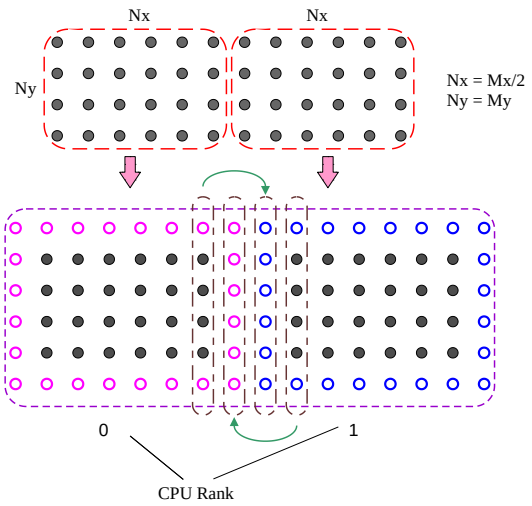

Fig 1 Adding halo points to sub-grids for parallel processing

شکل 1 افزودن نقاط حفره به زیرشبکه‌ها برای پردازش موازی

حافظه برای سایر متغیرها نیز به نحو مشابه صورت می‌گیرد. شایان ذکر است متغیرهای میدان می‌توانند اسکالر، بردار و تانسور باشند. البته امکان افزودن سایر انواع دیگر متغیر میدان به راحتی در چارچوب وجود دارد. برای دسترسی به یک درایه (مؤلفه) از متغیر از پرانتز استفاده می‌شود. برای مثال برای دسترسی به درایه فشار (اسکالر)، سرعت (بردار) و تنش (تانسور) از پرانتز استفاده می‌شود:

```
p(),    v(0),    tau(0,1)
```

شایان ذکر است هماهنگ با اندیس‌گذاری در زبان سی‌پلاس‌پلاس درایه اول (مؤلفه x) با عدد صفر، درایه دوم (مؤلفه y) با عدد یک و ... مشخص می‌شوند. برای مثال X(0) و X(1) به ترتیب بیانگر x و y می‌باشند.

با استفاده از الگوهای عبارت[1] [7] عملیات حسابی و اعمال توابع ریاضی به راحتی صورت می‌گیرد:

```
v(0) = sqrt ( sin(X(0)) + cos(X(1) )
```

در حقیقت رابطه فوق برای تمام نقاط شبکه اجرا می‌شود. همچنین با استفاده از روابط شرطی و نیز وصله‌ها می‌توان محاسبات را محدود به نقاط خاصی نمود. برای مثال می‌توان برای محدود ساختن محاسبات به نقاطی که مختصات عمودی آنها مثبت است ($y > 0$)، رابطه را به صورت زیر نوشت:

```
v(0)( X(1) > 0 ) = sqrt ( sin(X(0)) + cos(X(1) )
```

روش الگوهای عبارت این اجازه را می‌دهد تا از قابلیت شیءگرایی برای بازتعریف (سربارگذاری[2]) عملگرهای حسابی و توابع ریاضی استفاده نمود بدون آن که کارایی محاسبات کاهش یابد. برای جزئیات این روش خواننده به مرجع [8] رجوع داده می‌شود.

برای تعریف توابع جدید، برای اینکه با الگوهای عبارت چارچوب چشمه هماهنگ باشد، کاربر تابع خود را (برای مثال $\sin(x) + \cos(y)$) باید به صورت زیر تعریف نماید:

```
struct FunctionName : public MultiVarFun < FunctionName, real >
{
static real apply(real x, real y)
{
    return sin(x) + cos(y);
}
};
```

و سپس برای استفاده از تابع برای متغیرهای میدان، به نحو زیر عمل می‌شود:

```
v(0) = FunctionName::fun(X(0),X(1));
```

با استفاده از این روش در ابتدای فایل برنامه یک تابع برای شرایط اولیه در رابطه (1) تعریف و در خطوط 25 و 26 استفاده گردیده است.

در خطوط 9 تا 13 وصله‌هایی برای نقاط مرزی و نقاط درونی تعریف می‌شوند. وصله‌ها برای انتخاب یک لیست از نقاط می‌باشد (نظیر نقاط مرزی) تا عملیات خاصی روی آنها صورت گیرد. در برنامه فوق، نام وصله‌ها (که در اختیار کاربر است) بیانگر نقاطی است که قرار است در وصله متناظر قرار بگیرند. در خطوط 14 تا 18 این وصله‌ها با استفاده از توابع عضو و گاهی روابط شرطی نقاط مرتبط را یافته و ذخیره می‌نمایند. روابط شرطی نیز (در خطوط 16 و 18) با استفاده از الگوهای عبارت در نرم‌افزار قابل استفاده می‌باشد.

در خطوط 19 تا 21 عملگرهای تفاضلی پسرو در دو جهت عمودی و افقی در دو مرحله تعریف می‌شوند. مرحله اول تعریف عملگر می‌باشد؛ کلاس Stencil<NumPoints> برای ذخیره‌سازی خواص عملگر می‌باشد که آرگومان الگوی آن (NumPoints) بیانگر تعداد نقاط درگیر در رابطه عملگر می‌باشد. هر عملگر تفاضلی را می‌توان با استفاده از زوج‌های اندیس و ضریب مشخص نمود. برای مثال عملگرهای تفاضلی پسرو[3] و مرکزی[4] را که به ترتیب شامل دو و سه نقطه می‌باشند، می‌توان با زوج‌های به صورت رابطه (3) نمایش داد:

$$\begin{aligned} \nabla u_i &= u_i - u_{i-1} \rightarrow (0,+1.0)(-1,1.0) \\ \nabla\Delta u_i &= u_{i+1} - 2u_i + u_{i-1} \rightarrow (1,1.0)(0,-2.0)(1,-1.0) \end{aligned} \quad (3)$$

در هر زوج (داخل هر پرانتز)، عدد اول عددی صحیح است که بیانگر عدد افزوده شده به اندیس i و عدد دوم عددی حقیقی است که بیانگر ضریب جمله متناظر می‌باشد. تابع CreateStencil این زوج‌ها را می‌گیرد و عملگر تفاضلی مورد نظر را به صورت زیر ایجاد می‌نماید:

```
Stencil<2> backward = CreateStencil(-1, -1.0)(0, 1.0);
Stencil<3> central = CreateStencil(1,1.0)(0,-2.0)(1,-1.0);
```

در خط 19 به صورت فوق عملگر تفاضلی پسرو تعریف شده است.

در مرحله دوم باید مشخص شود که عملگر در چه راستایی عمل می‌کند. برای این منظور کلاس MultiDimStencil<NumPoints> در نظر گرفته شده است. تابع CreateStencil با گرفتن سه آرگومان، که به ترتیب عملگر، جهت و یک ضریب می‌باشند، عملگر دارای جهت را می‌سازد. برای مثال عملگرهای رابطه (4) در نظر گرفته می‌شوند:

$$\begin{aligned} \frac{\nabla u_{i,j}}{\Delta x} &= \frac{u_{i,j} - u_{i-1,j}}{\Delta x} \\ \frac{\nabla\Delta u_{i,j}}{\Delta y^2} &= \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} \end{aligned} \quad (4)$$

که به ترتیب عملگر مشتق‌گیری پسرو در جهت x و مشتق‌گیری مرکزی در جهت y می‌باشند، به صورت زیر در چشمه تعریف می‌شوند:

```
MultiDimStencil<2> backward_x =
                CreateStencil(backward, 0, 1.0/dx[0];
MultiDimStencil<3> central_y =
                CreateStencil(central, 1, 1.0/pow(dx[1],2));
```

به همین نحو، در خطوط 20 و 21 عملگر تفاضلی پسرو در دو جهت x و y تعریف شده است. ضرایب $\Delta t/\Delta x$ و $\Delta t/\Delta y$ نیز در عملگر گنجانده شده‌اند تا از محاسبات تکراری صرف نظر شود.

با استفاده از الگوهای عبارت حین اعمال این عملگرها به متغیرهای میدان، به طور خودکار عبارات تفاضل محدود بسط یافته و محاسبه می‌شوند. جهت اعمال عملگر به یک متغیر میدان هر شیء از کلاس MultiDimStencil دارای تابعی به نام D است که برای این منظور در نظر گرفته شده است. برای مثال رابطه زیر مشتق متغیر میدان u را در راستای y به روش مرکزی (که عملگر آن در بالا تعریف شد) محاسبه و در متغیر میدان un ذخیره می‌نماید:

```
un() = central_y.D( u() );
```

در خطوط 24 تا 28 مشابه فوق، عملگرها به منظور حل مسأله داخل حلقه پیمایش زمانی (که از خط 22 آغاز شده)، استفاده شده‌اند. در خط 24 رابطه تفاضل محدود (2) به نقاط درونی اعمال شده است. خطوط 25 تا 28 برای اعمال شرایط مرزی می‌باشد. قرار دادن نام وصله‌ها درون پرانتز پس از نام متغیر سبب می‌گردد رابطه سمت راست تنها برای نقاط همان وصله محاسبه و در متغیر سمت چپ ذخیره شود.

در خط 29 انتقال داده‌ها برای متغیر تعیین شده انجام می‌گیرد. در حقیقت پس از تکمیل محاسبات برای تمام نقاط پیش از رفتن به گام زمانی بعد نیاز است نقاط حفره‌ها با مقادیر نقاط متناظرشان بِروز شوند. تابع فراخوانی شده در خط 29 پنهان از دید کاربر با استفاده از دستورات کتابخانه ام‌پی‌آی انتقال داده‌ها را انجام می‌دهد.

خطوط 30 و 31 نیز برای خروجی گرفتن از نتایج روی فایل مشخص‌شده

[1] Expression Templates
[2] Overloading
[3] Backward difference operator
[4] Central difference operator

می‌باشد.

3- آزمون‌ها

در این بخش مسائلی که با چارچوب چشمه حل شده‌اند، شرح داده می‌شود تا قابلیت‌های چارچوب چشمه و برخی دیگر از امکانات آن نمایان گردد.

3-1- جریان مافوق‌صوت روی یک برآمدگی

این آزمون شامل یک جریان مافوق‌صوت است که از روی یک برآمدگی دایره‌ای به ارتفاع 4% درون یک تونل عبور می‌کند. این آزمون توسط تعدادی از محققان با اعداد ماخ ورودی مختلف حل شده است، که می‌توان آنها را در مرجع [9] یافت. در این جا عدد ماخ ورودی برابر $M_{\text{in}} = 1.4$ است. معادلات حاکم، معادلات دینامیک گاز اویلر در مختصات منحنی‌الخط تعمیم‌یافته (ξ,η) می‌باشد (رابطه (5)).

$$\bar{U}_t + \bar{F}_\xi + \bar{G}_\eta = 0, \bar{U} = \frac{1}{J}U,$$
$$\bar{F} = \frac{1}{J}(\xi_x F + \xi_y G), \bar{G} = \frac{1}{J}(\eta_x F + \eta_y G)$$
$$U = \begin{pmatrix}\rho\\ \rho u\\ \rho v\\ E\end{pmatrix}, F = \begin{pmatrix}\rho u\\ \rho u^2 + p\\ \rho uv\\ (E+p)u\end{pmatrix}, G = \begin{pmatrix}\rho v\\ \rho vu\\ \rho v^2 + p\\ (E+p)v\end{pmatrix}$$
$$E = \rho\left(e + \frac{u^2+v^2}{2}\right), p = \rho e(\gamma - 1), \gamma = 1.4 \qquad (5)$$

که در آن u و v مؤلفه‌های بردار سرعت، ρ چگالی، p فشار، e انرژی درونی و γ نسبت گرماهای ویژه می‌باشد.

پس از تجزیه شارها به بردارهای شار مثبت و منفی، از روش پادبادسوی مرتبه اول برای حل معادلات استفاده می‌شود. شکل 2 (بالا) خطوط همتراز عدد ماخ را روی شبکه‌ای به اندازه 301×101 نشان می‌دهد. از آنجا که روش مرتبه اول است، از این شبکه ریز استفاده شده است. توزیع عدد ماخ روی دیواره‌های بالایی و پایینی تونل در شکل 2 (پایین) نمایش داده شده است.

3-2- جریان تراکم‌ناپذیر لزج روی پله معکوس

این آزمون شامل یک جریان تراکم‌ناپذیر روی یک پله معکوس درون یک کانال است [10]. این آزمون را محققان زیادی مطالعه نموده‌اند که یک مرور جامع را می‌توان در مرجع [10] و مراجعی که در آن ذکر شده است، یافت. هدف از حل آزمون این است که نشان داده شود که چشمه صرفاً برای هندسه‌های مستطیلی کاربرد ندارد و با استفاده از وصله‌ها می‌توان هندسه‌های بیشتری را پوشش داد.

شکل 3 (بالا) هندسه و مرزهای پیرامون را نشان می‌دهد. معادلات حاکم، معادلات ناویر استوکس در شکل تابع جریان-تاوایی[1] است (رابطه (6)):

$$-(\psi_{xx} + \psi_{yy}) = \omega$$
$$\frac{1}{Re}(\omega_{xx} + \omega_{yy}) = u\omega_x + v\omega_y$$
$$u = \psi_y, v = -\psi_x \qquad (6)$$

که در آن u و v مؤلفه‌های بردار سرعت، ω تاوایی و ψ تابع جریان می‌باشد.

عدد رینولدز جریان بر اساس قطر هیدرولیک ورودی (H_2) و میانگین سرعت ورودی، برابر 100 می‌باشد. در ورود، جریان به طور کامل توسعه یافته است. شکل 3 (پایین) خطوط جریان و بردارهای سرعت را روی شبکه یکنواختی به اندازه 301×101 نشان می‌دهد که با چهار هسته پردازنده محاسباتی و با استفاده از روش مرکزی مرتبه دوم برای جملات جابجایی و لزج است.

[1] Stream function-Vorticity Formulation

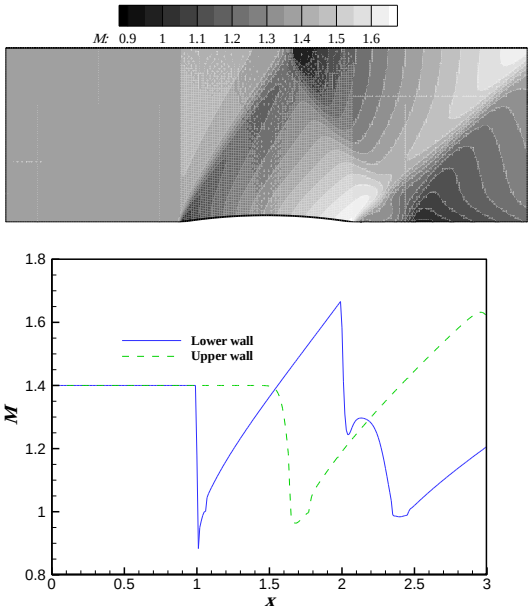

Fig 2 Supersonic flow over a bump; Mach number contour lines (top), Mach number distributions over wind tunnel walls (bottom)

شکل 2 جریان مافوق‌صوت روی برآمدگی؛ خطوط همتراز عدد ماخ (بالا)، توزیع عدد ماخ روی دیواره‌های تونل باد (پایین)

همچنین از روش تکرار گاوس-سایدل برای حل معادلات گسسته استفاده شده است. با مقایسه نتایج به دست آمده با آنچه که در [10] آورده شده، درستی آنها تأیید می‌گردد.

برای این آزمون شبکه و هندسه یک مستطیل می‌باشد. با این حال نقاط درون پله که جزئی از حل نمی‌باشند با تعریف وصله‌هایی از ناحیه حل عددی کنار گذاشته شده‌اند. همچنین سطوح بالا و سمت راست پله که نقاط آن درون مستطیل شبکه می‌باشند با وصله‌های دیگری برای اعمال شرایط مرزی دیوار تعریف شده‌اند. تمام شرایط مرزی شامل ورودی، خروجی، دیوارها و نقاط درونی در قطعه کُد زیر نمایش داده شده‌اند:

```
Patch interior("interior", multiGrid);
Patch fluid("fluid", multiGrid);
Patch down("down", multiGrid);
Patch middown("middown", multiGrid);
Patch up("up", multiGrid);
Patch inlet("inlet", multiGrid);
Patch midleft("midleft", multiGrid);
Patch outlet("outlet", multiGrid);

interior.MakeInteriorPatch();
inlet.MakeBoundaryPatch(0, X(1) >= 0.0);
outlet.MakeBoundaryPatch(1);
down.MakeBoundaryPatch(2, X(0) >= 0.0 && X(0) < L2);
up.MakeBoundaryPatch(3, X(0) < L2);
middown.MakePatch(X(0) <= 0.0 && abs(X(1)) < 1e-4);
midleft.MakePatch(X(1) <= 0.0 && abs(X(0)) < 1e-4);
fluid.MakePatch(interior, X(0) > 0.0 || X(1) > 0.0);
```

به طور کلی برای تعریف وصله‌ها توابع مختلفی وجود دارد که در این آزمون استفاده گردیده‌اند. تابع MakeInteriorPatch برای انتخاب نقاط درونی (غیر مرزی و نیز غیر حفره) استفاده می‌شود. تابع MakeBoundaryPatch برای انتخاب نقاط مرزی استفاده می‌شود. اولین آرگومان این تابع یک عدد صحیح است که 0 و 1 به ترتیب متناظر مرز چپ و راست، 2 و 3 به ترتیب مرز پایین و بالا و ... می‌باشند. همچنین با استفاده از عبارات شرطی (به عنوان آرگومان دوم تابع) می‌توان نقاط مرزی را انتخاب کرد که شرط خاصی را ارضا می‌نمایند. تابع MakePatch نیز برای انتخاب نقاطی از یک وصله دیگر یا

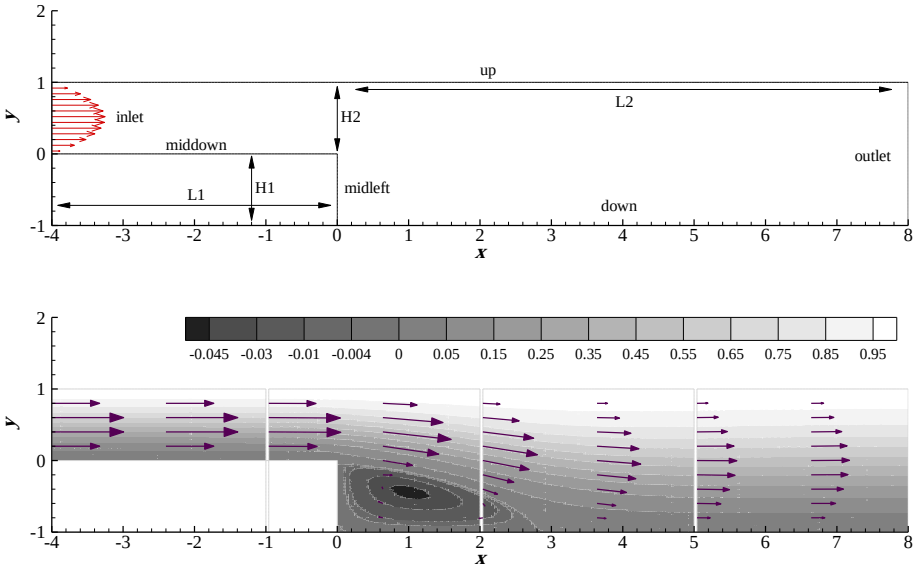

Fig 3 Incompressible flow over a backward facing step

شکل 3 جریان تراکم‌ناپذیر لزج روی پله معکوس

انتخاب نقاطی (درونی، مرزی یا حفره)که شرط خاصی را ارضا می‌نمایند، استفاده می‌شود. برای مثال وصله با نام fluid شامل نقاطی درون شبکه می‌باشد که شرط $(x > 0 \cup y > 0)$ را ارضا می‌نمایند. در حقیقت با این شرط نقاط گوشه پایین و سمت چپ مبدأ مختصات که بیانگر قسمت جامد پله می‌باشند (شکل 3)، عضو این وصله نمی‌باشند. همچنین وصله‌های با نام middown و midleft شامل نقاط دو سطح پله می‌باشند (شکل 3 بالا).

پس از تعریف وصله‌ها می‌توان روابط تساوی را تنها به یک وصله محدود نمود. برای مثال می‌توان به نحوه اعمال شرط مرزی ورودی اشاره نمود که به صورت زیر (بِراحتی) اعمال می‌گردد:

```
u()(inlet) = 6.0*(X(1) / H2 - pow(X(1) / H2, 2));
```

که در حقیقت تساوی زیر را که بیانگر رابطه سهموی پروفیل توسعه یافته سرعت است، تنها برای نقاط وصله inlet اعمال می‌نماید (رابطه (7)):

$$u = 6\left(\frac{y}{H_2} - \left(\frac{y}{H_2}\right)^2\right) \tag{7}$$

تابع دیگری با نام MakeShadowPatch وجود دارد که در آزمون جریان روی یک ایرفویل توضیح داده می‌شود.

3-3- جریان مافوق‌صوت روی یک پله

این آزمون شامل یک جریان مافوق‌صوت غیردائم با عدد ماخ $M_{\text{in}} = 3.0$ گذرنده از روی یک پله درون یک تونل باد می‌باشد [11]. معادلات حاکم، معادلات دینامیک گاز اویلر می‌باشند (معادلات (5) در مختصات کارتزین). برای حل این مسأله که شامل یک شوک قوی می‌باشد، از روش ضرورتاً غیرنوسانی وزن‌دار[1] مرتبه پنجم [13,12] استفاده شده است و روش انتگرال‌گیری زمانی روش رونگه-کوتای مرتبه سوم [13,12] می‌باشد. در حقیقت هدف از حل این آزمون نشان دادن توانایی چشمه در استفاده از روش‌های غیرخطی می‌باشد.

برای اعمال روش فوق نیاز به تجزیه شار وجود دارد. تجزیه شار می‌تواند با بزرگترین مقدار ویژه صورت گیرد:

```
for (int eqn = 0; eqn < 4; eqn++)
{
    FP(eqn)(fluid) = weno::D(0, U(eqn), F(eqn), abs(u()) + c());
}
```

در قطعه کُد فوق، تمام مؤلفه‌های بردار F جداگانه با مقدار ویژه $|u| + c$ تجزیه و سپس روش ضرورتاً غیرنوسانی وزن‌دار اعمال می‌شود. آرگومان اول تابع weno::D راستای تجزیه شار و آرگومان‌های دوم و سوم به ترتیب متغیر پایستار و شار متناظر آن می‌باشند. آرگومان چهارم نیز مقدار ویژه لازم برای تجریه شار است.

روش مناسب‌تر این است که هر متغیر مشخصه با مقدار ویژه خود تجزیه شود که در این صورت نیاز به نگاشت متغیرها به فضای مشخصه‌ها می‌باشد که به اصطلاح تجزیه محلی مشخصه‌ها[2] گفته می‌شود. این روش به علت عملیات ضرب ماتریسی به لحاظ محاسباتی پُرهزینه‌تر است اما اتلاف عددی کمتری نسبت به روش قبلی دارد. این روش نیز پیاده‌سازی شده است و کاربر به صورت زیر آن را فراخوانی می‌کند:

```
FP(fluid) = wenochar<2>::D(0, U, F);
```

در قطعه کُد فوق، تمام بردار F و U به تابع داده می‌شود و درون تابع عملیات نگاشت و تجزیه بردارها و اعمال روش ضرورتاً غیرنوسانی وزن‌دار انجام می‌شود. آرگومان‌های تابع wenochar<2>::D مشابه تابع weno::D است با این تفاوت که کل بردار متغیرهای پایستار و کل بردار شار به تابع داده می‌شود ونیازی به مقدار ویژه وجود ندارد. آرگومان الگو (اینجا عدد 2) بُعد فضایی مسأله را مشخص می‌کند.

شکل 4 توزیع چگالی را در لحظه $t = 2.5$ نشان می‌دهد که در آن یک شوک به شکل کمان جلوی پله تشکیل شده است و با برخورد به دیواره‌های تونل منعکس شده است. همانند آزمون قبل نقاط پله با تعریف وصله‌هایی از ناحیه حل عددی جدا شده‌اند.

4-3- جریان گذرصوتی روی ایرفویل ناکا 0012

در این آزمون جریان تراکم‌پذیر غیرلزج روی ایرفویل ناکا 0012[3] شبیه‌سازی شده است. عدد ماخ جریان آزاد برابر $M_\infty = 0.85$ و زاویه حمله آن $\alpha = 1^\circ$

[1] WENO

[2] Local characteristic decomposition

[3] NACA0012

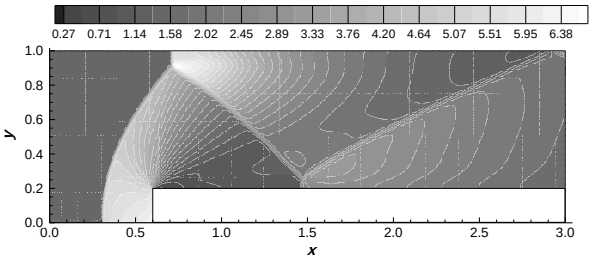

Fig 4 Supersonic flow over a step: Density contour lines at $t = 2.5$

شکل 4 جریان مافوق‌صوت روی یک پله؛ خطوط همتراز چگالی در لحظه $t = 2.5$

درجه می‌باشد. بالا و پایین ایرفویل یک شوک تشکیل می‌شود. در اینجا روش حل عددی روش تفاضل محدود فشرده مرتبه چهارم [14] برای گسسته‌سازی مکانی و روش رونگه-کوتای مرتبه سوم برای انتگرال‌گیری زمانی می‌باشد. برای پایداری و کنترل کردن نوسانات حول شوک‌ها از فیلتر معرفی‌شده در مرجع [15] استفاده شده است. شکل 5 نتایج حل عددی را برای ضریب فشار با نتایج آزمایش [16] مقایسه می‌نماید. همچنین مشاهده می‌شود که فیلتر نوسانات گیبز[1] حول شوک که ناشی از روش غیراتلافی استفاده شده می‌باشد، را به خوبی حذف کرده است.

شایان ذکر است در این آزمون از یک شبکه سی[2] استفاده شده است. هنگام اجرای موازی این آزمون برخی نقاط در دو انتهای شبکه در قسمت دنباله[3] که به صورت هندسی منطبق بر یکدیگر هستند، روی پردازنده‌های مختلف قرار دارند. در چشمه قابلیت تبادل اطلاعات بین این نقاط قرار داده شده است. در حقیقت ابتدا نقاط منطبق باید شناسایی و سپس حین حل، تبادل اطلاعات بین نقاط مورد نظر صورت گیرد. تابع زیر با استفاده از متغیر مکان نقاط منطبق (با فاصله کمتر از 10^{-6}) را می‌یابد و اینکه نقاط منطبق متعلق به کدام پردازنده (rank) هستند را نیز معین می‌نماید:

```
wake.MakeShadowPatch(rank, face[2], X, 1e-6);
```

در حقیقت تابع فوق فاصله نقاطی که روی وصله face[2] و در حافظه پردازنده به شماره rank قرار دارند، با سایر نقاط روی این وصله که در حافظه سایر پردازنده‌ها قرار دارند محاسبه و در صورتی که فاصله آنها کمتر از 10^{-6} باشد به عوان نقاط منطبق در نظر می‌گیرد که لیست این نقاط در حقیقت وصله wake را می‌سازد. این تابع تنها یک بار و پیش از آغاز حلقه زمانی باید فراخوانی شود. سپس هر زمان که نیاز باشد، همانند قطعه کُد زیر، تبادل داده‌ها با استفاده از تابع Update انجام می‌شود:

```
wake.Update(rank, rho(), rho(), 1, 1);
wake.Update(rank, V(0), V(0), 1, 1);
wake.Update(rank, V(1), V(1), 1, 1);
wake.Update(rank, p(), p(), 1, 1);
```

آرگومان اول که شماره پردازنده و آرگومان دوم و سوم به ترتیب متغیر مبدأ و مقصد برای تبادل داده‌ها می‌باشند. آرگومان چهارم و پنجم مشخص کننده این است که در راستای شماره 1 (یعنی y) و به فاصله 1 از وصله wake تبادل داده‌ها صورت گیرد. با اینکه برنامه‌نویسی دو تابع فوق دشوار می‌باشد اما کاربر بدون درگیر بودن با پیچیدگی‌های برنامه‌نویس پردازش موازی (با کتابخانه ام‌پی‌آی) شرط مرزی دنباله را برای استفاده پیاده‌سازی می‌نماید.

همچنین شایان ذکر است که چارچوب چشمه برای حل دستگاه‌های سه‌قطری حاصل از اعمال روش فشرده مرتبه چهارم آنها را بین پردازنده‌ها توزیع می‌نماید. در قطعه برنامه زیر نحوه فراخوانی تابع برای محاسبه مشتق به روش فشرده مرتبه چهارم نشان داده شده است.

```
CompactDerivative(0, dx, distributor, FB[0](eqn), FBv[0](eqn), physical, face[0], 1, face[1], 1);
CompactDerivative(1, dx, distributor, FB[1](eqn), FBv[1](eqn), physical, face[2], 1, face[3], 1);
```

خط اول مشتق در راستای x و خط دوم مشتق در راستای y را محاسبه می‌نماید. آرگومان اول مشخص کننده راستای مشتق‌گیری و آرگومان دوم اندازه شبکه می‌باشد. آرگومان سوم توزیع کننده می‌باشد که جهت توزیع دستگاه معادلات سه‌قطری بین پردازنده‌ها می‌باشد. آرگومان چهارم متغیر ورودی و آرگومان پنجم جهت ذخیره‌سازی مشتق محاسبه شده می‌باشد. آرگومان ششم وصله مربوط به نقاط درونی و آرگومان‌های هشتم و دهم وصله‌های مربوط به نقاط مرزی دو انتهای دستگاه معادلات می‌باشند. آرگومان‌های هفتم و نهم نیز مشخص‌کننده رابطه مرزی استفاده شده در دو مرز می‌باشد. برای مثال عدد 1 بیانگر استفاده از روش تفاضل محدود فشرده یک‌سویه در مرز می‌باشد.

3-5- جریان زیرصوت لزج روی روی ایرفویل ناکا 0012

در این آزمون جریان غیردائم تراکم‌پذیر لزج روی ایرفویل ناکا 0012 شبیه‌سازی می‌گردد. معادلات حاکم، معادلات ناویر-استوکس در مختصات منحنی‌الخط تعمیم‌یافته می‌باشد. عدد ماخ جریان آزاد $M_\infty = 0.2$، عدد رینولدز آن $Re = 10^5$ و زاویه حمله $\alpha = 18^\circ$ درجه می‌باشد. روش حل عددی همانند آزمون قبل می‌باشد. جریان در حالت‌های دوبُعدی و هم سه‌بُعدی اجرا شده است. در حالت سه‌بُعدی راستای z به صورت متناوب در نظر گرفته شده است. شکل 6 خطوط همتراز تاوایی را در شش زمان متفاوت نشان می‌دهد. مشاهده می‌شود که گردابه‌هایی در انتهای ایرفویل تشکیل می‌گردد و سپس جدا می‌شود. در شکل 7 نتایج حل سه‌بعدی برای صفحات هم‌مقدار تاوایی نشان داده شده است.

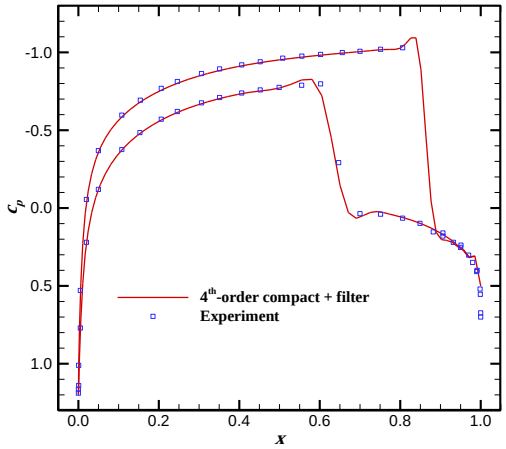

Fig 5 Transonic flow over an airfoil; Pressure distribution for a Mach number of 0.85 and an angle of attack of 1 degree.

شکل 5 جریان گذرصوتی روی ایرفویل؛ توزیع ضریب فشار در عدد ماخ 0.85 و زاویه حمله 1 درجه

4- آزمودن کارایی کُد

در این بخش مقایسه‌ای بین زمان اجرای چارچوب نرم‌افزاری چشمه و یک کُد غیر شیءگرا صورت گرفته است. برای مقایسه روش ضرورتاً غیرنوسانی مرتبه پنجم که یک روش غیرخطی با هزینه محاسباتی بالا می‌باشد، انتخاب می‌شود. معادلات حاکم معادلات دینامیک گاز اویلر می‌باشند. جدول 1

[1] Gibbs oscillations
[2] C-Grid
[3] Wake

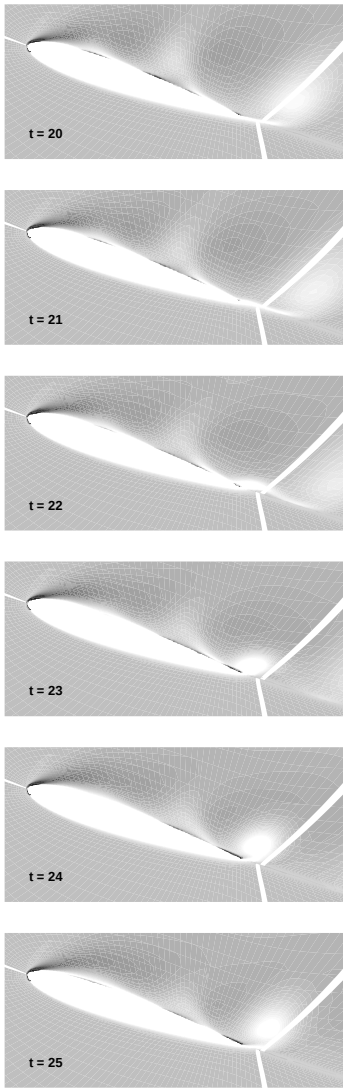

Fig 6 Subsonic flow over an airfoil; vorticity contour lines in two-dimensions, flow specifications: Mach number 0.2, Reynolds number 10^5, angle of attack 18 degree

شکل 6 جریان زیرصوت روی ایرفویل؛ مقدار تاوایی در حالت دوبُعدی، مشخصات جریان: عدد ماخ 0.2، عدد رینولدز 10^5 و زاویه حمله 18 درجه

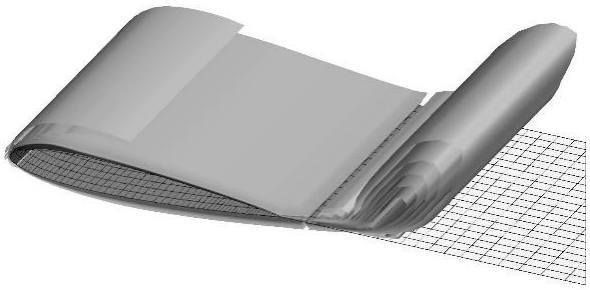

Fig 7 Subsonic flow over an airfoil; vorticity iso-surfaces in three-dimensions, flow specifications: Mach number 0.2, Reynolds number 10^5, angle of attack 18 degree.

شکل 7 جریان زیرصوت روی ایرفویل؛ سطوح هم‌مقدار تاوایی در حالت سه‌بُعدی، مشخصات جریان: عدد ماخ 0.2، عدد رینولدز 10^5 و زاویه حمله 18 درجه.

جدول 1 مقایسه زمان اجرای روش ضرورتاً غیرنوسانی مرتبه پنجم بین چارچوب نرم‌افزاری چشمه و یک کُد غیر شیءگرا

Table 1 Runtime comparison of the fifth-order weighted essentially non-oscillatory scheme between Cheshmeh framework and a non-object-oriented code

شبکه	چارچوب چشمه (ثانیه)	کُد غیر شیءگرا (ثانیه)
26×26	9.6	10.3
51×51	30.2	36.0
101×101	133.6	129.9
201×201	463.0	474.8

مقایسه‌ای بین زمان‌های اجرا برای 1000 گام زمانی بین چارچوب نرم‌افزاری چشمه و کُد توسعه یافته در مرجع [17] می‌باشد. کُد اخیر یک کُد غیرشیءگرا می‌باشد که با دستورات پایه زبان سی توسعه یافته و در آن تمام ملاحظات جهت افزایش کارایی محاسباتی صورت گرفته است. اجراها روی یک پردازنده اینتل[1] با ویژوال استودیو 2015 با پیکربندی ریلیس 64 بیتی[2] انجام شده است. مشاهده می‌شود کارایی هر دو کُد برای شبکه‌های مختلف تقریباً یکسان است.

5- نتیجه‌گیری

در این مقاله یک چارچوب نرم‌افزار متن‌باز بومی با نام چشمه برای حل عددی معادلات سیال روی شبکه‌های باسازمان به روش تفاضل محدود ارائه گردید. این چارچوب با استفاده از زبان برنامه‌نویسی سی‌پلاس‌پلاس و قابلیت‌های جدید آن توسعه یافت. آزمون‌های مختلفی با معادلات و شرایط مرزی مختلف جهت آزمودن قابلیت‌های چارچوب نرم‌افزاری اجرا گردید. بسیاری از قابلیت‌هایی که برای روش‌های تفاضل محدود نیاز است، در چارچوب پیش‌بینی شده است. از جمله نشان داده شد که چارچوب چشمه قابلیت اجرا با روش‌های خطی و غیرخطی و نیز روش‌های فشرده سه‌قطری را دارد. همچنین با تعریف وصله‌هایی امکان اعمال شرایط مرزی مختلف وجود دارد. قابلیت‌های چارچوب چشمه به محققان این امکان را می‌دهد که شبیه‌سازی‌های تفاضل محدود را برای مسأله مورد نظرشان به سادگی انجام دهند.

6- تقدیر و تشکر

از حمایت مالی دانشگاه تهران و بنیاد ملی نخبگان از این تحقیق در قالب طرح پژوهشی شماره 28745/1/02 قدردانی می‌گردد.

7- مراجع

[1] The OpenFOAM foundation, Accessed on 1__September 2017; http://www.openfoam.org.
[2] H. Jasak, A. Jemcov, U. Kingdom, Openfoam: A C++ library for complex physics simulations, *International Workshop on Coupled Methods in Numerical Dynamics, IUC*, pp. 1-20, 2007.
[3] SU2: The Open-Source CFD code, Accessed on 1 September 2017; https://su2code.github.io.
[4] F. Palacios, M. Colonno, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, T. Lukaczyk, T. Taylor, J. Alonso, Stanford university unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design, *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Grapevine, Texas, January 7-10, 2013.
[5] The KIVA family of Computational Fluid Dynamics (CFD) software, Accessed on 1 September 2017; https://www.lanl.gov/projects/feynman-center/deploying-innovation/intellectual-property/software-tools/kiva/index.php.

[1] Intel Core 2 Quad Processor Q6600
[2] Release x64 configuration

[6] D. Abrahams, A. Gurtovoy, *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond*, 1st Edition, Addison-Wesley, 2005.
[7] T. Veldhuizen, *Expression Templates*, *C++ Report*, Vol. 7, No. 5, pp. 26-31, 1995.
[8] D. Gregor, J. Jarvi, Variadic templates for C++0x, *Journal of Object Technology*, Vol. 7, No. 2, pp. 31-1, 2008.
[9] M. H. Djavareshkian, M. H. Abdollahi Jahdi, Shock-capturing method using characteristic-based dissipation filters in pressure-based algorithm, *Acta Mechanica*, Vol. 209, No. 1, pp. 99-113, 2009.
[10] E. Erturk, Numerical solutions of 2-d steady incompressible flow over a backward-facing step, part i: High reynolds number solutions, *Computers & Fluids*, Vol. 37, No. 6, pp. 633-655, 2008.
[11] P. R. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, *Journal of Computational Physics*, Vol. 54, No. 1, pp. 115-173, review Article, 1984.
[12] X. D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, *Journal of Computational Physics*, Vol. 115, No. 1, pp. 200-212, 1994.
[13] G. S. Jiang, C. W. Shu, Efficient implementation of weighted ENO schemes, *Journal of Computational Physics*, Vol. 126, No. 1, pp. 202-228, 1996.
[14] S. K. Lele, Compact finite difference schemes with spectral-like resolution, *Journal of Computational Physics*, Vol. 103, No. 1, pp. 16-42, 1992.
[15] H. Mahmoodi Darian, V. Esfahanian, K. Hejranfar, A shock-detecting sensor for filtering of high-order compact finite difference schemes, *Journal of Computational Physics*, Vol. 230, No. 1, pp. 494-514, 2011.
[16] H. Yoshihara, P. Sacher, *Test Cases for Inviscid Flow Field Methods: Report of Fluid Dynamics Panel Working Group 07*, AGARD advisory report 211, AGARD, 1985.
[17] H. Mahmoodi Darian, V. Esfahanian, Assessment of WENO schemes for multi-dimensional Euler equations using GPU, *International Journal for Numerical Methods in Fluids*, Vol. 76, No. 1, pp. 961-981, 2014.