Одеський національний університет імені І.І.Мечникова
(повне найменування вищого навчального закладу)

Інститут математики, економіки і механіки
(повне найменування інституту/факультету)

Кафедра теоретичної механіки
(повна назва кафедри)

Дипломна робота

магістр
(освітньо-кваліфікаційний рівень)

на тему: **«Розв'язування однієї газодинамічної задачі методом великих частинок»**

«Решение одной газодинамической задачи методом крупных частиц»
«The solution of a gasdynamic problem by the method of big particles»

Виконав: студент денної форми навчання
напряму підготовки 8.040202 Механіка
Медушевський Володимир Геннадійович

Керівник проф. Асланов С.К. _____

Рецензент ст. викл. Царенко О.П. _____

Рекомендовано до захисту:

Протокол засідання кафедри

№ від 2017 р.

Завідувач кафедри д.ф.-м. наук, проф. Асланов С.К.
_____
(підпис)

Захищено на засіданні ДЕК №

протокол № від 2017 р.

Оцінка _____ / _____ / _____
(за національною шкалою, шкалою ECTS, бали)

Голова ДЕК д.ф.-м. наук, проф. Лещенко Д.Д.
_____
(підпис)

Одеса – 2017

ЗМІСТ

ВСТУП

Для розрахунку течій в складних задачах газової динаміки природно використовувати нестаціонарні схеми наскрізного розрахунку, де обчислення проводяться без попереднього виділення особливостей, поверхні розриву, тощо.

У ряді випадків, я вже мовилось вище, здається раціональнім введення в алгоритми елементів метода Харлоу *частинок у клітинках*, використовування якого межує з проведенням чисельного експерименту.

В останні роки був здійснений ряд *чисельних експериментів* по дослідженню складних газодинамічних течій. Вони були проведені за допомогою *модифікованого нестаціонарного метода крупних частинок*, поштовхом до розробки котрого послужили роботи Харлоу, Річа та Херта.

Основна ціль цих розробок – *побудова на базі процесу розщеплення загального чисельного алгоритму та дослідження з його допомогою широкого класу задач сучасної газової динаміки(від чисто дозвукових течій до надзвукових режимів, включаючи трансзвукові області, зони зриві і т.д.).*

Отримані схеми часто використовують порівняно невелике число розрахункових точок (зазвичай не більше 2-3 тисяч), що дозволяє проводити їх реалізацію на машинах середньої потужності та використовувати для серійних розрахунків у практиці прикладних НИИ та КБ.

Можливо, що окремі локальні властивості рішень будуть визначатись при цьому недостатньо точно, але, мабуть, лише тільки таким шляхом можна отримати загальні характеристики складних явищ та картину течій взагалі.

Як зазначалося у введені, метод Харлоу частинок у клітинках поєднує в собі у визначених рисах перевагу «лагранжевого» та «ейлерового» підходів.

Область рішення тут розбивається нерухомою фіксованою у просторі (ейлеровій) розрахунковою сіткою.

Проте загальне середовище трактується дискретною моделлю – розглядається сукупність частинок фіксованої маси («лагранжева» сітка частинок), які рухаються через «ейлерову» сітку клітинок.

Частинки служать для визначення параметрів самої рідини (маси, енергії, швидкості), в той час як «ейлерова» сітка використовується для визначення

параметрів поля (тиску, густини, температури).

Метод частинок у клітинках дозволяє досліджувати складні явлення у динаміці багато компонентних середовищ, частинки добре «стежать» за вільними поверхнями та лініями розділу середовищ, взаємодією розривів і т.п.

Проте дискретний метод частинок володіє рядом недоліків.

Головний з них, що лежить у самій природі метода, полягає у тому, що через дискретне представлення загального середовища кінцевим числом частинок у клітинці параметри течій також визначаються дискретним образом – а тільки частина перетне межу ейлерової клітинки, то маса, імпульс та енергія частинки віднімаються від відповідних величин попередньої клітинки та додаються до нових значень.

Такі скачки, вельми характерні для розрахунків по методу Харлоу, приводять до великих нефізичних флуктуацій величин що розраховуються (особливо густина), у рішеннях з'являються автоколивання, тощо.

Окрім того, самі чисельні схеми цього методу мають, взагалі кажучи, недостатню розрахункову стійкість (особливо в областях застою, при невеликому числі частинок у клітинці), тому доводиться вводити у схеми явні дисипативні члени зі штучною в'язкістю, використовувати неявні схемі першого шагу, розглядати частини різних форм, тощо.

Складно також отримання інформації до сильно розріджених областей, звідки майже уходять усі частинки та т.і. Значно же збільшити число частинок не дозволяють технічні можливості сучасних обчислювальних машин.

По суті, метод Харлоу, завдяки введенню додаткового параметру (числа частинок в даній клітинці), збільшує в програмному сенсі на одиницю розмірність задачі.

Для газодинамічних течій виявилось доцільним відійти від дискретної моделі частинок та виходити з концепції безперервності, розглядаючи замість частинок потік маси через межи «ейлерових» клітинок.

Густина газу тут буде вже знаходитись не шляхом ділення сумарної маси усіх частинок в клітинці на її об'єм, а з закону збереження маси, записаного у різницевій формі для даної клітинки (крупної частинки).

При цьому природно збереглися сильні сторони метода Харлоу – «ейлерово-лагранжевий» підхід та сам процес організації розрахунків.

Таким чином, замість сукупності частинок у клітинках тут розглядається маса усієї клітинки в цілому – *крупна частинка* – та на основі кінцево-різницевих уявлень законів збереження вивчаються нестаціонарні (та безперервні) потоки цих крупних частинок через «ейлерову» сітку.

По суті, при такому підході використовуються закони збереження, записані у вигляді рівнянь балансу для клітинки кінцевих розмірів (як це зазвичай робиться у процесі виводу газодинамічних рівнянь, але без подальшого граничного переходу від клітинки до точки). *Застосовуючи раціональні форми апроксимацій до різних видів течій, вдається різко скоротити вимоги до об'єму пам'яті та швидкодії.*

Метод, про який піде мова нижче, являється у деякому сенсі проміжним між методом дискретних частинок у клітинках та звичайними кінцево-різницевими підходами, так як організація обчислень тут проводиться аналогічно підходу Харлоу, але в той же час на усіх етапах зберігається структура кінцево-різницевих схем. У методах вказаного типу використовується розщеплення початкової системи рівнянь по фізичним процесам.

На кожному з етапів розрахунку тимчасового циклу розглядаються в залежності від характеру досліджуваного рішення різні види апроксимацій (явні або неявні схеми, орієнтовані схеми з урахуванням напрямку потоку, центральні різності, метод інтегральних співвідношень, тощо).

У даній роботі проводиться повний виклад та дослідження схем модифікованого метода крупних частинок стосовно до трансзвукових задач аеродинаміки, течіям зі вдувом струменя в основний потік, досліджуються відривні турбулентні зони та інше.

Такий підхід дав можливість вивчити складні картини обтікання плоских та асиметричних тіл різної форми у стисливому газі для широкого діапазону змінення швидкостей потоку. Розглядаються також внутрішні течії зі складною конфігурацією стрибків ущільнення, дифракційні задачі, тощо.

Підкреслимо, що використовування метода крупних частинок дозволило розглянути увесь цей клас складних течій для плоских та асиметричних задач з

єдиних позицій, за допомогою одного чисельного підходу.

Для оцінки точності та надійності отриманих результатів, постановки задачі, граничних умов проводились методичні обчислення на різних розрахункових сітках; мала місце перевірка виконання законів збереження; результати порівнювались з наявними розрахунками по іншим схема, а також з експериментальними та аналітичними даними.

Висновки

В рамках цієї магістерської роботи був створений **Windows**-додаток **BigParticles,** що реалізує чисельний експеримент, у рамках якого проводиться розрахунок газодинамічної задачі про розліт газового шару в середу з протитиском.

Додаток має зручний інтерфейс, за допомогою якого користувач має можливість змоделювати область обчислення задачі.

Для проведення обчислень при виконанні чисельного експерименту був реалізований «метод крупних частинок».

Усі розрахунки проводяться в окремому потоці даних у фоновому режимі та не вимагають участі користувача.

Хід чисельного експерименту повністю контролюється користувачем, який має можливість за бажанням припиняти та відновляти хід розрахунку.

При цьому результати експерименту можуть бути збереженні в зручному для використання форматі в будь-який момент часу.

Проте, у ході дослідження способів збереження інформації було виявлено, що серіалізовані результати експерименту (стан розрахункової матриці на кожній ітерації) займають достатньо великий об’єм пам’яті на носії при великій області розрахунку.

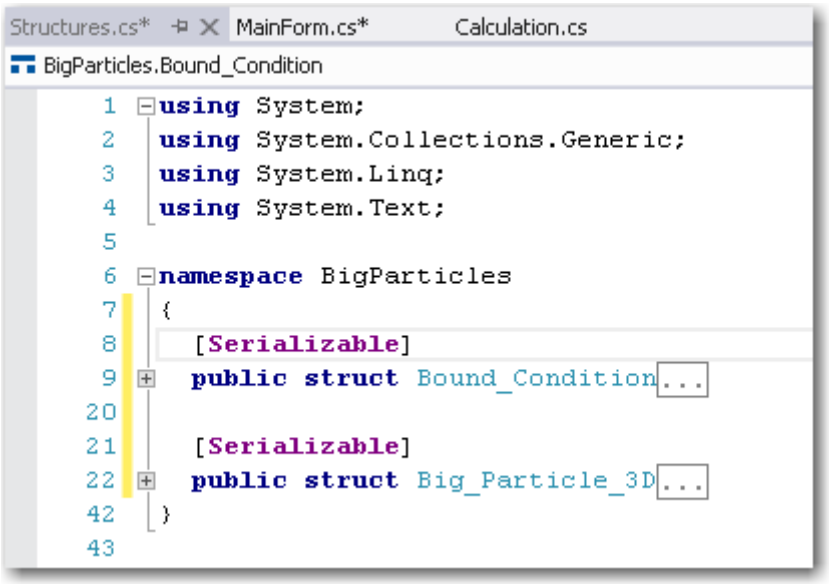
Тому такий спосіб збереження інформації, незважаючи на те, що він має багато зручностей для подальшого використання, не є раціональним для ПК.

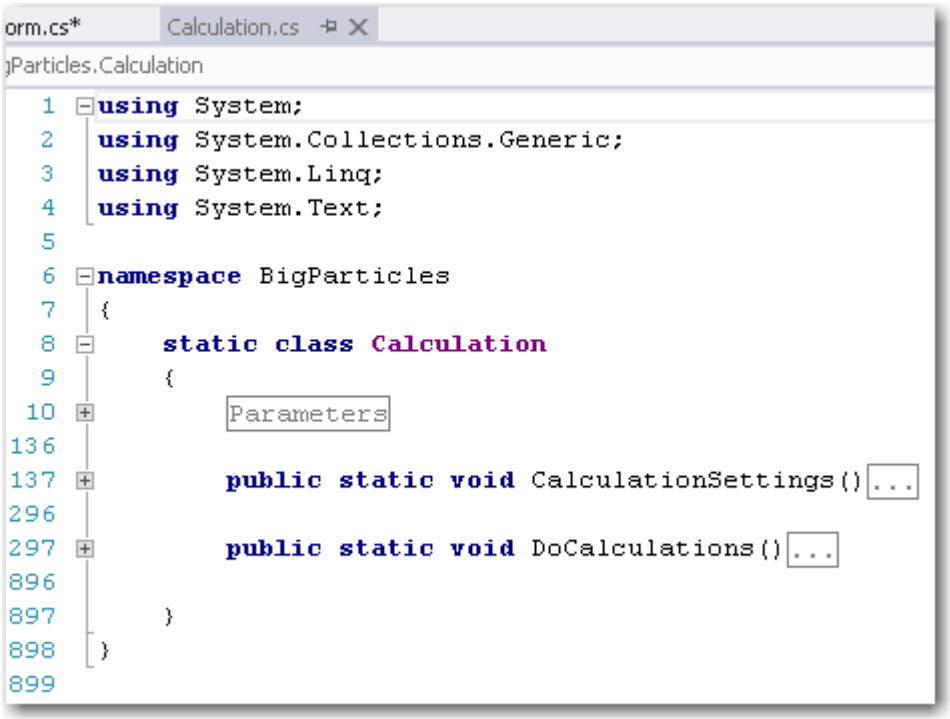
Перелік посилань

1. Эндрю Троелсен. «***Язык программирования C# 2005 и платформа .NET 2.0***», 3-е издание.: Пер. С англ.. – М.: ООО “И.Д. Вильямс”, 2007. – 1168 с.
2. Ватсон Карли «***Язык программирования C#***». Издательский дом «Питер», 2004
3. Черный Г.Г. ***Газовая динамика***. – М.: Наука, 1988. – 424 с.
4. Белоцерковский О.М., Давыдов Ю.М. ***Метод крупных частиц в газовой динамике.*** – М.: Наука. Главная редакция физико-математической литературы, 1982. – 392 с.

Додатки

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace BigParticles
{

  public partial class MainForm : Form
  {
    int iterrations = 10;
    int currentIteration = 1;
    string directoryPath;
    bool isCalculationSettingsReady = false;

    NumericUpDown ValueChanged

    public MainForm()...

    private void btn_Start_Click(object sender, EventArgs e)...

    private void btn_Cancel_Click(object sender, EventArgs e)...

    // Перемикач кнопок
    private void ButtonSwitcherStartStop()...

    private void backgroundWorker1_DoWork(object sender, DoWorkEventArg

    private void
      backgroundWorker1_ProgressChanged(object sender, ProgressChangedE

    private void
      backgroundWorker1_RunWorkerCompleted(object sender, RunWorkerComp

    private void Serialize()...

    private void button1_Click(object sender, EventArgs e)...

    private void button2_Click(object sender, EventArgs e)...
  }
}
```


Structures.cs*
MainForm.cs*
Calculation.cs
BigParticles.Bound_Condition
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace BigParticles
7 {
8 [Serializable]
9 public struct Bound_Condition...
20
21 [Serializable]
22 public struct Big_Particle_3D...
42 }
43


orm.cs*
Calculation.cs
gParticles.Calculation
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace BigParticles
7 {
8 static class Calculation
9 {
10 Parameters
136
137 public static void CalculationSettings()...
296
297 public static void DoCalculations()...
896
897 }
898 }
899

Програмний код тестового Windows-додатку

```
namespace Practice
{
    // структура представляє з себе клітинку розрахункової матриці
    public struct IntegralCell
    {

        public double ap, tr, er, up;
        public bool isCalculated;
        public IntegralCell(double ap, double tr, double er, double up, bool
isCalculated)
        {
            this.ap = ap;
            this.tr = tr;
            this.er = er;
            this.up = up;
            this.isCalculated = isCalculated;
        }
    }
    public partial class Form1 : Form
    {
        public delegate double f_x(double x);

        public static double a, b, A, B, eps, ap, tr, er;
        public static double n, m;
        f_x fx; public const double pi = Math.PI;
        public IntegralCell[,,] matrix;
        public int x, y, z;
        int i_ = 0, j_ = 0, k_ = 0;
        int time = 0;
        bool matrixCreated = false;


        public Form1()
        {
            InitializeComponent();
            UpD_eps.SelectedIndex = 0;
        }


        void Clear()
        {
            tBx_ap.Text = ""; tBx_tr.Text = "";
            tBx_er.Text = ""; tBx_up.Text = "";
        }



        private void UpD_eps_SelectedItemChanged(object sender, EventArgs e)
        {
            Clear();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            time++;
            labelTime.Text = time.ToString();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            InterfaceState(2);
            matrixCreated = false;
            btn_Calc.Text = "Почати";
            btn_Calc.Enabled = true;
        }

        private void UpD_1_ValueChanged(object sender, EventArgs e)
        {
```

```
            Clear();
        }

        private void UpD_2_ValueChanged(object sender, EventArgs e)
        {
            Clear();
        }

        private void btn_Calc_Click(object sender, EventArgs e)
        {

            InterfaceState(0);
            ButtonsSwitcher();
            btn_Calc.Text = "Продовжити";
            btnNewParams.Enabled = false;
            timer1.Start();

            x = (int)nMax.Value - (int)nMin.Value;
            y = (int)mMax.Value - (int)mMin.Value;
            z = (int)aMax.Value - (int)aMin.Value;

            // Створюємо матрицю у разі якщо вона ще не створена
            if (!matrixCreated)
            {
                matrix = new IntegralCell[x + 1, y + 1, z + 1];
                matrixCreated = true;
            }
            xValue.Maximum = x; yValue.Maximum = y; zValue.Maximum = z;

            // Кількість кроків (клітинок у матриці) розрахунку
            int steps = (x + 1) * (y + 1) * (z + 1);
            progressBar1.Value = 0;
            progressBar1.Maximum = steps + 1;



            backgroundWorker1.RunWorkerAsync();

        }
private void btn_Stop_Click(object sender, EventArgs e)
        {
            ButtonsSwitcher();
            backgroundWorker1.CancelAsync();
        }

        // Власне сам метод розрахунку однієї клітинки
        private void DoCalc(double n_, double m_, double a_, out IntegralCell cell)
        {
            eps = 0.01 * Math.Pow(10.0, -(int)UpD_eps.SelectedIndex);
            A = 0.0; B = 1.0;

            n = n_;
            m = m_;
            a = a_;
            if (n >= m) tr = m * pi / 2.0;
            else tr = n * pi / 2.0;
            fx = new f_x(Dwight_858_711);

            ap = Improper_1(A, ref B, eps, fx);

            er = Math.Abs(tr - ap);

            cell = new IntegralCell(ap, tr, er, B, true);
        }

        private void btn_CellValue_Click(object sender, EventArgs e)
        {
            IntegralCell cell = matrix[(int)xValue.Value, (int)yValue.Value,
(int)zValue.Value];
            tBx_ap.Text = string.Format("{0:F15}", cell.ap).PadLeft(23);
            tBx_tr.Text = string.Format("{0:F15}", cell.tr).PadLeft(23);
```

```
            tBx_er.Text = string.Format("{0:F15}", cell.er).PadLeft(23);
            tBx_up.Text = string.Format("{0:F1}", cell.up);
        }

        // Підінтегральна функція
        public static double Dwight_858_711(double x)
        {
            if (x > 0) return (Math.Sin(m * x)) * (Math.Sin(n * x) / (x * x + a * a));
            else return m * n;
        }

        // Основний цикл розрахунку
        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            for (int i = (int)nMin.Value; i <= (int)nMax.Value; i++)
            {
                i_++;
                for (int j = (int)mMin.Value; j <= (int)mMax.Value; j++)
                {
                    j_++;
                    for (int k = (int)aMin.Value; k <= (int)aMax.Value; k++)
                    {
                        k_++;
                        if (backgroundWorker1.CancellationPending)
                        {
                            e.Cancel = true;
                            break;
                        }
                        if (matrix[i_ - 1, j_ - 1, k_ - 1].isCalculated == false)
                        {
                            // Виконуємо розрахунок у клітинці з індексами поточного
циклу
                            DoCalc(i, j, k, out matrix[i_ - 1, j_ - 1, k_ - 1]);

                            backgroundWorker1.ReportProgress(1);
                        }
                        else
                        {
                            backgroundWorker1.ReportProgress(0);
                        }
                    }
                    k_ = 0;
                }
                j_ = 0;
            }
            i_ = 0;
        }


 // Метод, що викликається при зміні прогресу виконання розрахунку
        private void backgroundWorker1_ProgressChanged(object sender,
ProgressChangedEventArgs e)
        {
            if (i_ != 0)
                labelCell.Text = "Розрахунок клітинки: " + (i_ - 1) + ";" + (j_ - 1) +
";" + (k_ - 1);
            progressBar1.Value += 2;
            progressBar1.Value--;

        }
        // Метод викликається по завершенню роботи потоку
        private void backgroundWorker1_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            btnNewParams.Enabled = true;
            InterfaceState(1);
            timer1.Stop();
            time = 0;
            if (btn_Stop.Enabled) ButtonsSwitcher();
            if (!e.Cancelled)
            {
```

```
            btn_Calc.Text = "Почати";
            btn_Calc.Enabled = false;
        }
    }


    private void ButtonsSwitcher()
    {
        btn_Calc.Enabled = !btn_Calc.Enabled;
        btn_Stop.Enabled = !btn_Stop.Enabled;
    }

    // Метод перемикання стану інтерфейсу
    private void InterfaceState(int state)
    {
        switch (state)
        {
            case 0:
                gb1.Enabled = false;
                gb2.Enabled = false;
                btnNewParams.Enabled = false;
                break;
            case 1:
                gb1.Enabled = true;
                gb2.Enabled = false;
                btnNewParams.Enabled = true;
                break;
            case 2:
                gb1.Enabled = true;
                gb2.Enabled = true;
                btnNewParams.Enabled = true;
                break;
            case 3:
                gb1.Enabled = false;
                gb2.Enabled = true;
                btnNewParams.Enabled = true;
                break;
            default:
                break;
        }
    }

public static double
     NC_10_1(double a, double b, double eps, f_x f)
        {
            int n, i;
            double s0, s1, hx, x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10;
            double f0, f1, f2, f3, f4, f5, f6, f7, f8, f9, f10;

            n = 1; // початкова кількість інтервалів інтегрування
            s1 = 1.0E20; // віртуальне значення інтегральної суми
            do
            {
                s0 = s1; x10 = a; s1 = 0; hx = (b - a) / n / 10;
                for (i = 1; i <= n; i++)
                {
                    x0 = x10; f0 = f(x0);
                    x1 = x0 + hx; f1 = f(x1);
                    x2 = x1 + hx; f2 = f(x2);
                    x3 = x2 + hx; f3 = f(x3);
                    x4 = x3 + hx; f4 = f(x4);
                    x5 = x4 + hx; f5 = f(x5);
                    x6 = x5 + hx; f6 = f(x6);
                    x7 = x6 + hx; f7 = f(x7);
                    x8 = x7 + hx; f8 = f(x8);
                    x9 = x8 + hx; f9 = f(x9);
                    x10 = x9 + hx; f10 = f(x10);

                    s1 += f0 + f10 + (f1 + f9) * 6.616045310263273E0 -
                      (f2 + f8) * 3.020165556731188E0 + (f3 + f7) *
                                          1.695400510362856E1 -
```

```
                    (f4 + f6) * 1.621646853799714E1 + f5 * 26.59911620090870E0;
                }
                s1 = s1 * hx * 2.683414836192614E-1; n = n * 2;
            } while (Math.Abs(s0 - s1) > eps);
            return s1;
        }


        // Невласний одноразовий інтеграл від < a > до нескінченності
        // реалізован з використанням квадратурної формули по 11 вузлам
        public static double
          Improper_1(double a, ref double b, double eps, f_x f)
        {
            double ai, bi, err, imp1, imp;
            err = eps / 2;          // err - похибка обчислення для частичного інтегралу
            ai = a;                 //  ai -  нижнія межа для першого частичного
інтегралу
            bi = ai + 7.0;          //  bi - верхня межа для першого частичного
інтегралу
                                    // imp1 - перший частичний інтеграл ( основний )
            imp1 = NC_10_1(ai, bi, eps, f);
            imp = imp1;
            do
            {
                ai = bi; bi = bi + 1.0;
                imp1 = NC_10_1(ai, bi, eps, f);
                imp += imp1;
            } while (Math.Abs(imp1) >= err);
            b = bi;               // досягнуте значення верхньої межі інтегрування
            return imp;
        }
    }
}
```