МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Московский физико-технический институт
(национальный исследовательский университет)

Разрыв распада в механике сплошной среды

Лабораторная работа №1 по курсу
Вычислительная математика
Вариант 2.2

Выполнил: студент группы Б04-856.
Крылов А. А.

г. Долгопрудный
2020 год

Содержание

1. Цели и задачи работы

Цель работы: найти скорость разрыва при распаде разрыва с обазованием ударной волны и волны разряжения.

Задачи:

- Свести исходную систему к нелинейному алгебраическому уравнению с одной переменной;
- Решить полученное уравнение методами вычислительной математики;
- Используя полученное решение найти скорость разрыва.

2. Распад разрыва собразованием ударной волны и волны разряжения

В начальный момент, до распада, все пространство можно разделить на два однородных полупространства, газодинамическим величинам в которых припишем индекс 0 для правого и индекс 3 для левого полупространств. Граница раздела между ними - плоскость.

В результате распада разрыва по правому полупространству начинает распространятся ударная волна с приписанным индексом 0, а по левому - волна разряжения с индексом 2. Из законов сохранения имеем:

$$\rho_1(U_1 - D_0) = \rho_0(U_0 - D_0), \tag{1}$$

$$P_1 + \rho_1(U_1 - D_0)^2 = P_0 + \rho_0(U_0 - D_0)^2, \tag{2}$$

$$(U_1 - D_0)\rho_1[\epsilon_1 + \frac{(U_1 - D_0)^2}{2}] + P_1 = (U_0 - D_0)\rho_0[\epsilon_0 + \frac{(U_0 - D_0)^2}{2}] + P_0. \tag{3}$$

Также на волне разряжения справедливы соотношения для адиабатического процесса:

$$U_3 + \frac{2C_3}{\gamma_3 - 1} = U_2 + \frac{2C_2}{\gamma_3 - 1}, \tag{4}$$

$$C_2 = C_3(\frac{P_2}{P_3})^{\frac{\gamma_3 - 1}{2\gamma_3}}. \tag{5}$$

Кроме того, на контактной границе остаются непрерывными давление и нормальные составляющие скорости:

$$P_1 = P_2, \tag{6}$$

$$U_1 = U_2, \tag{7}$$

Преобразование системы (1) - (7) с исключением D_0 приводит нелинейному уравнению порядка $2n$, где $n = \frac{2\gamma_3}{\gamma_3 - 1}$, имеющего следующий вид:

$$X^2Z^{2n} - \alpha_0\nu^2XZ^{n+2} + 2\alpha_0\nu(\mu+\nu)XZ^{n+1} - [2 + (\mu+\nu)^2\alpha_0]XZ^n - \nu^2Z^2 + 2\nu(\mu+\nu)Z - (\mu+\nu)^2 + 1 = 0. \tag{8}$$

Здесь $\alpha_0 = \frac{\gamma_0+1}{\gamma_0-1}, \mu = (U_3 - U_0)\sqrt{\frac{(\gamma_0-1)\rho_0}{2P_0}}, \nu = \frac{2}{\gamma_3-1}\sqrt{\frac{\gamma_3(\gamma_0-1)}{2}\frac{P_3}{P_0}\frac{\rho_0}{\rho_3}}, X = \frac{P_3}{P_0}, Z = (\frac{P_1}{P_3})^{\frac{1}{n}}$. Стоит отметить, что так как C_i – скорость звука $(i = 0, 1, 2, 3)$, то плотность среды можно расчитать как $\rho_i = \frac{\gamma_i P_i}{C_i^2}$.

Выразим $U_1 - D_0$ из (1) и получим

$$(U_0 - D_0)^2 = \frac{\rho_1(P_1 - P_0)}{\rho_0(\rho_1 - \rho_0)}. \tag{9}$$

ρ_1 можно получить из (1) и (3):

$$\rho_1 = \rho_0\frac{(\gamma_0 - 1) + (\gamma_0 + 1)\frac{P_1}{P_0}}{(\gamma_0 + 1) + (\gamma_0 - 1)\frac{P_1}{P_0}} \tag{10}$$

Только положительные решения уравнения (8) имеют физический смысл. P_1 найдем из этих решений, а rho_1 найдем из соотношения (10). Тогда D_0 легко найти из (10).

3. Метод и результаты

Для вычисления D_0 при известных $\gamma_0, \rho_0, P_0, \gamma_3, C_3, P_3, U_0$ необходимо найти положительные корни (8), и, зная их, необходимо найти P_1, а также ρ_1 из (10).

Нахождение корней уравнения (8) делится на две подзадачи: локализации корней и уточнения значения корня до нужной степени точности. Под задачей локализации подразумевается поиск таких областей определения уравнения, в которых существует только одно решение.

Все корни алгебраического уравнения вида

$$f(z) = \sum_{i=0}^{n} a_i z^{n-i} = 0 \tag{11}$$

находятся в кольце $\frac{|a_n|}{|a_n|+B} < |z| < 1 + \frac{A}{|a_0|}$, где $A = \max(|a_1|, ..., |a_n|), B = \max(|a_0|, ..., |a_{n-1}|)$ вследствие основной теоремы алгебры. В нашем случае нас интересуют только $z > 0$, поэтому в нашем случае $|z|$ можно заменить на z.

Как только найдено кольцо существования корней, для локализации нужно выполнить следующий алгоритм.

Всё кольцо делится на N частей, где N - достаточно большое число. Для дальнейшего рассмотрения берутся только те из N подобластей, на которых $f(z)$ меняет знак. Стоит отметить несовершенство этого метода локализации корней, ведь можно упустить из виду такие корни, в которых локальный экстремум $f(z)$ равен 0.

После того как решена задача локализации корней, уточнение до нужной степени точности может быть выполнено, например, методом половинного деления.

Пусть известно, что на отрезке $[a, b]$ находится только один корень уравнения (11) с непрерывной функцией $f(z)$ и $f(a)f(b) < 0$.

Вычислим $f(c), c = \frac{a+b}{2}$, и оставим для рассмотрения отрезок $[a, c]$, если $f(a)f(c) < 0$, либо $[c, b]$, если $f(b)f(c) < 0$ Обозначим левый конец отрезка a_i, а правый b_i,где i - номер итерации. Процесс будет повторятся, пока $\frac{|b-a|}{2^n} = |b_n - a_n| < \epsilon$, где ϵ - заданная точность. Этот метод весьма прост в понимании и реализации, однако такой метод имеет лишь первый порядок сходимости.

Реализация метода поиска D_0 на языке програмироанния C представлена в Приложении 5.1. В ходе программы сначала считаются коэффиценты уравнения (8), затем происходит локализация в кольце всех положительных корней, далее идет локализация каждого отдельного положительного корня, потом идет процесс уточнения каждого корня методом половинного деления, и в конце с использованием найденных Z находится соотвествующий корню D_0.

Так, при исполнении программы с исходными данными $\gamma_0 = 1.666667, \rho_0 = 0.000169, U_0 = 0.000000, P_0 = 1013000.000000, \gamma_3 = 1.400000, C_3 = 36537.000000, U_3 = 12290.000000, P_3 = 1676800.000000$ с заданной точностью $\epsilon = 0.00001$ были найдены $D_0^1 = -44716.725481, D_0^2 = -63379.476913, D_0^3 = -162012.621300$, соответствующие корням $Z_0^1 = 0.335204, Z_0^2 = 0.765000, Z_0^3 = 1.090827$. Возможно, были упущенны те корни, в которых локальный экстремум уравнения (8) равен 0, но метод отделения корней, укзанный выше не позволяет обнаружить такие корни. Полный вывод программы приведен в Приложении 5.2.

Из вывода программы видно, что число смен знаков в получившемся уравнении равно 5, а число корней равно 3, что согласуется с теоремой Декарта.

4. Вывод

Получены возможные значения $D_0^1 = -44716.725481, D_0^2 = -63379.476913, D_0^3 = -162012.621300$ для разрыва при распаде разрыва с обазованием ударной волны и волны разряжения путем сведения системы уравнений, описывающих явление к нелинейному уравнению (8). Применены методы вычислительной математики для решения поставленной задачи. Указаны возможные недостатки примененных методов.

5. Приложения

5.1. Код программы

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define ftype long double

ftype getMax(ftype* array, int size) {
if (size < 1) {
return -1;
}
else {
ftype max = array[0];
for (int i = 1; i < size; i++) {
if (array[i] > max) {
max = array[i];
}
}
return max;
}
}

ftype sign(ftype a) {
if (a > 0) {
return 1;
}
else if (a == 0) {
return 0;
}
else {
return -1;
}
}
ftype a0, a1, a2, a3, a4, a5, a6, n;
ftype evaluate(ftype X) {
ftype result = a0 * powl(X, 2.0 * n) + a1 * powl(X, n + 2.0) + a2 * powl(X, n + 1.0) +
a3 * powl(X, n) + a4 * powl(X, 2) + a5 * X + a6;
return result;
}

typedef struct {
ftype left;
ftype right;
} interval;

void printInterval(interval a) {
printf("\t%Lf<=Z<=%Lf\n", a.left, a.right);
}
```

```
int main()
{
printf("Computational Mathematics Lab #1\nVariant II.2\n\n");

//Initial values
ftype gamma0 = 5.0 / 3.0;
ftype rho0 = 1.694 * powl(10.0, -4.0);
ftype U0 = 0.0;
ftype P0 = 1.013 * powl(10.0, 6.0);

ftype gamma3 = 7.0 / 5.0;
ftype C3 = 3.6537 * powl(10.0, 4.0);
ftype U3 = 1.229 * powl(10, 4.0);
ftype P3 = 1.6768 * powl(10.0, 6.0);

printf("Initial values:\n\tgamma_0=%Lf\n\trho_0=%Lf\n\tU_0=%Lf\n\tP_0=%Lf\n\n\t
gamma_3=%Lf\n\tC_3=%Lf\n\tU_3=%Lf\n\tP_3=%Lf\n", gamma0, rho0, U0, P0,
 gamma3, C3, U3, P3);
//Data to get coefficients
n = 2.0 * gamma3 / (gamma3 - 1.0);

ftype alpha0 = (gamma0 - 1.0) / (gamma0 + 1.0);
ftype rho3 = (gamma3 * P3) / (powl(C3, 2.0));
ftype X = P3/P0;
ftype mu = (U3-U0) * sqrtl(rho0 * (gamma0 - 1.0)/(2.0 * P0));
ftype nu = (2.0 / (gamma3 - 1.0)) * sqrtl((gamma3 * (gamma0 - 1.0) / 2.0) *
(P3 / P0) * (rho0 / rho3));

//Get coefficients
a0 = powl(X, 2.0);
a1 = -1.0 * alpha0 * powl(nu, 2.0);
a2 = 2.0 * alpha0 * nu * (mu + nu) * X;
a3 = -1.0 * (2.0 + powl(mu + nu, 2.0)) * X;
a4 = -1.0 * powl(nu, 2.0);
a5 = 2.0 * nu * (mu + nu);
a6 = -1.0 * powl((mu + nu), 2.0) + 1;

printf("\nFirst step results(coefficients):\n\ta0=%Lf\n\ta1=%Lf\n\ta2=%Lf\n\t
a3=%Lf\n\ta4=%Lf\n\ta5=%Lf\n\ta6=%Lf\n", a0, a1, a2, a3, a4, a5, a6);

//Roots localization in ring
ftype arrayOneToN[6] = { a1, a2, a3, a4, a5, a6 };
ftype arrayZeroToNMinOne[6] = { a0, a1, a2, a3, a4, a5 };

ftype A = getMax(arrayOneToN, 6);
ftype B = getMax(arrayZeroToNMinOne, 6);

ftype leftEdge = fabsl(a6) / (fabsl(a6) + B);
ftype rightEdge = 1 + (A / fabsl(a0));
```

```
printf("\nSecond step result(localization):\nRing:\n\t
%Lf<=|Z|<=%Lf\n", leftEdge, rightEdge);
//Root separating
ftype N = 1024;
ftype step = (rightEdge - leftEdge) / N;
interval intervals[6];
int intervalsAmount = 0;
int lastSign = sign(evaluate(leftEdge));

N = N * 2;
step = (rightEdge - leftEdge) / N;
for (ftype X = leftEdge; X <= rightEdge; X += step) {
if (sign(evaluate(X)) != lastSign) {
lastSign = sign(evaluate(X));
intervalsAmount++;
intervals[intervalsAmount - 1].left = X - step;
intervals[intervalsAmount - 1].right = X;
}
}

printf("Intervals:\n");
for (int i = 0; i < intervalsAmount; i++) {
printInterval(intervals[i]);
}
//Getting roots with required precision
printf("Third step result(clarification by dichotomy method):\nInput epsilon: ");
ftype epsilon = 0.0;
ftype* roots = (ftype*) malloc(intervalsAmount * sizeof(ftype));
scanf_s("%Lf", &epsilon);
for (int i = 0; i < intervalsAmount; i++) {
ftype l = intervals[i].left;
ftype r = intervals[i].right;
while ((r - l) > epsilon) {
ftype middle = (l + r) / 2.0;

ftype leftValue = evaluate(l);
ftype middleValue = evaluate(middle);

ftype testVal = leftValue * middleValue;

if (testVal < 0) {
r = middle;
}
else {
l = middle;
}
}
ftype root = (l + r) / 2.0;
roots[i] = root;
```

```
printf("Root number %d Z = %Lf\n", i, root);
}
printf("\nFourth step result(D0):\n");
for (int i = 0; i < intervalsAmount; i++) {
if (roots[i] > 0) {
printf("Root Z[%d]=%Lf\n", i, roots[i]);
printf("--------------------------\n");

ftype P1 = powl(roots[i], n) * P3;

ftype rho1 = rho0 * ((gamma0 -1.0) + (gamma0 + 1.0) * (P1 / P0)) /
((gamma0 + 1.0) + (gamma0 - 1.0) * (P1 / P0));

ftype D0 = U0 - sqrtl((rho1 * (P1 - P0)) / (rho0 * (rho1 - rho0)));

printf("\tD0=%Lf\n", D0);
}
}
return 0;
}
```

5.2. Вывод программы

```
Computational Mathematics Lab #1
Variant II.2

Initial values:
        gamma_0=1.666667
        rho_0=0.000169
        U_0=0.000000
        P_0=1013000.000000

        gamma_3=1.400000
        C_3=36537.000000
        U_3=12290.000000
        P_3=1676800.000000

First step results(coefficients):
        a0=2.739956
        a1=-0.465081
        a2=1.643260
        a3=-6.818181
        a4=-1.860324
        a5=3.970951
        a6=-1.119047

Second step result(localization):
Ring:
        0.219852<=|Z|<=2.449275
Intervals:
```

```
        0.334154<=Z<=0.335242
        0.764145<=Z<=0.765233
        1.090721<=Z<=1.091809
Third step result(clarification by dichotomy method):
Input epsilon: 0.00001
Root number 0 Z = 0.335204
Root number 1 Z = 0.765000
Root number 2 Z = 1.090827

Fourth step result(D0):
Root Z[0]=0.335204
---------------------------
        D0=-44716.725481
Root Z[1]=0.765000
---------------------------
        D0=-63379.476913
Root Z[2]=1.090827
---------------------------
        D0=-162012.621300
```