Звіт

Виконав Заяць Роман  
ПМІ-33

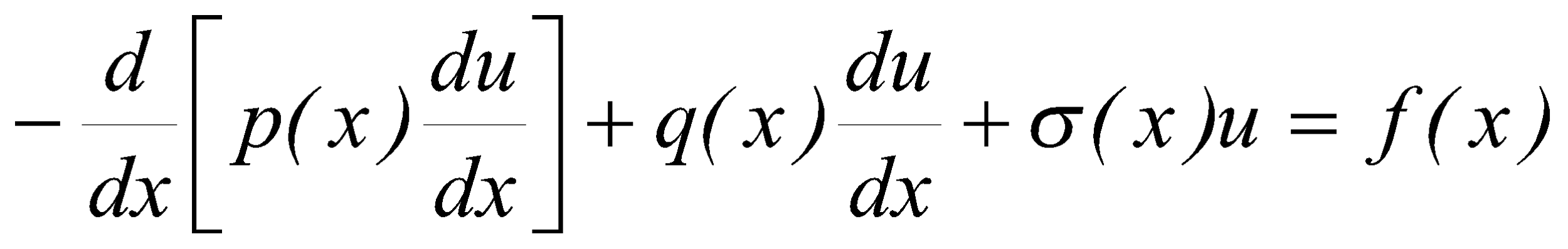
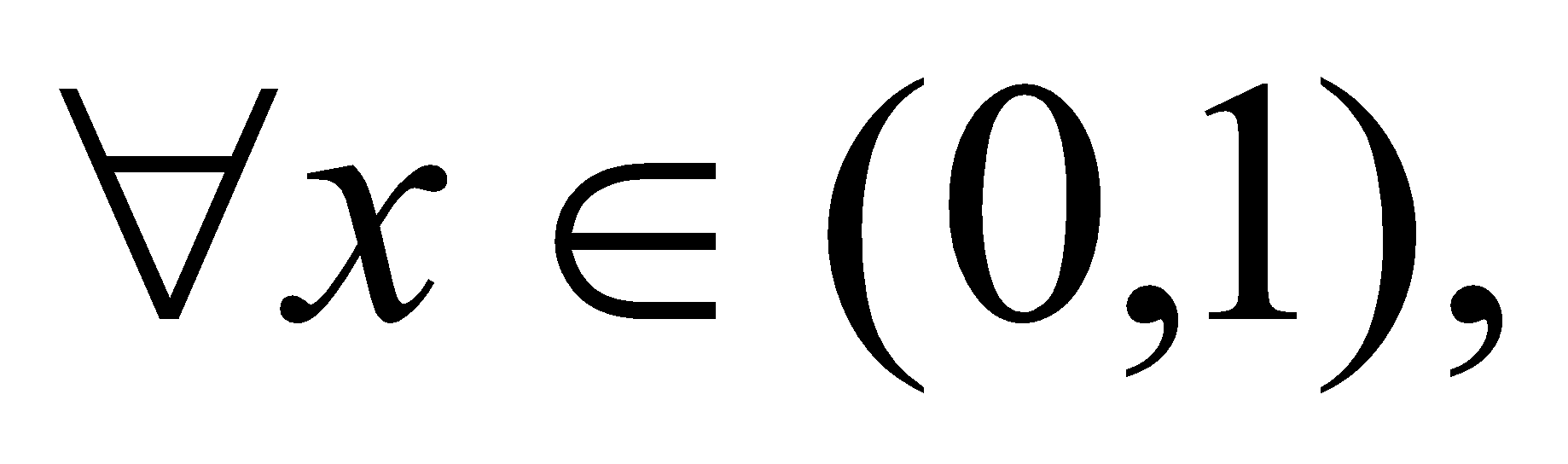
# *Вступ*

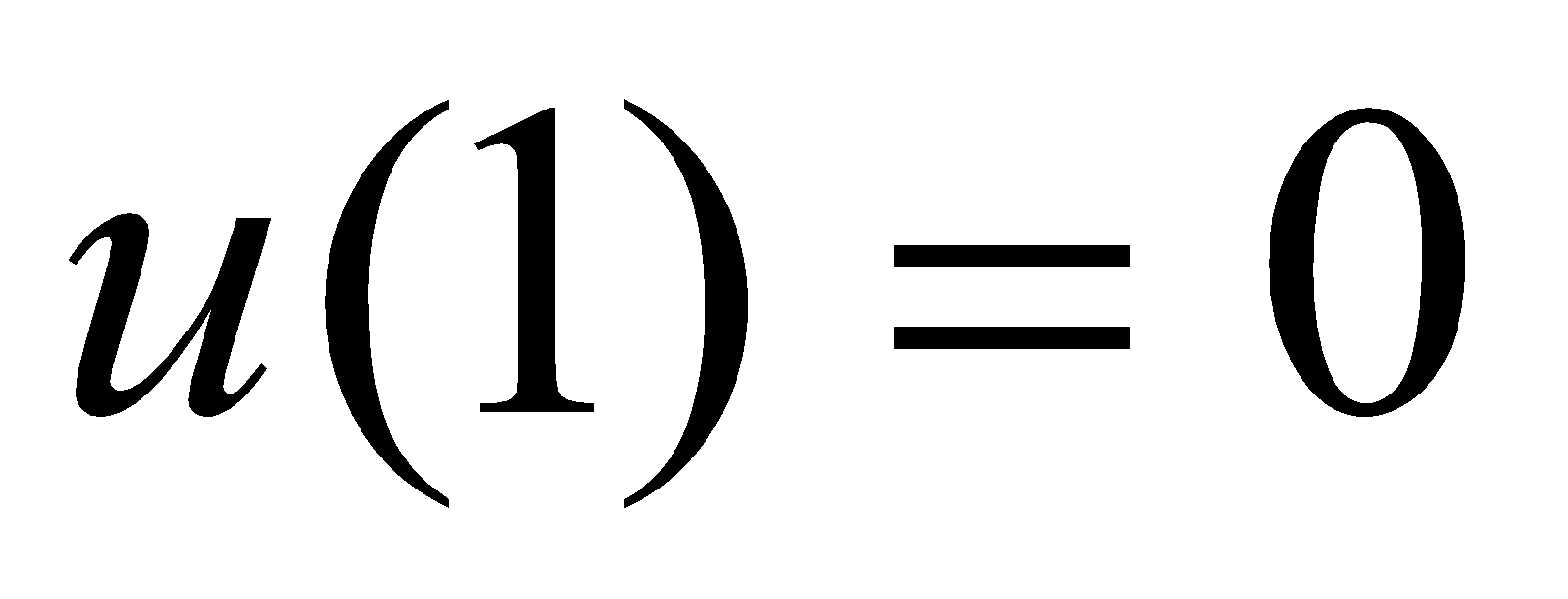
Чисельне розв’язування крайових задач реакції-дифузії h-адаптивним методом скінченних елементів: лінійні апроксимації.

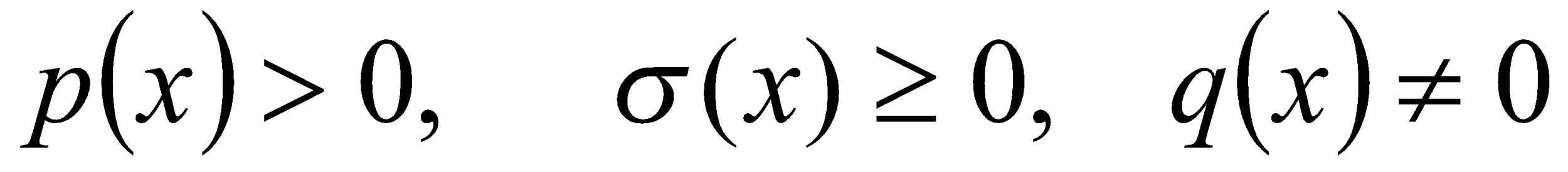
Основна ідея методу скінченних елементів полягає в тому, щоб будь-яку неперервну величину апроксимувати дискретною моделлю, яка будується на множині кусково-неперервних функцій, які визначені на скінченній кількості підобластей (елементів). Метод скінченних елементів є наближеним методом і вимагає оцінювання збіжності розв’язку до точного.

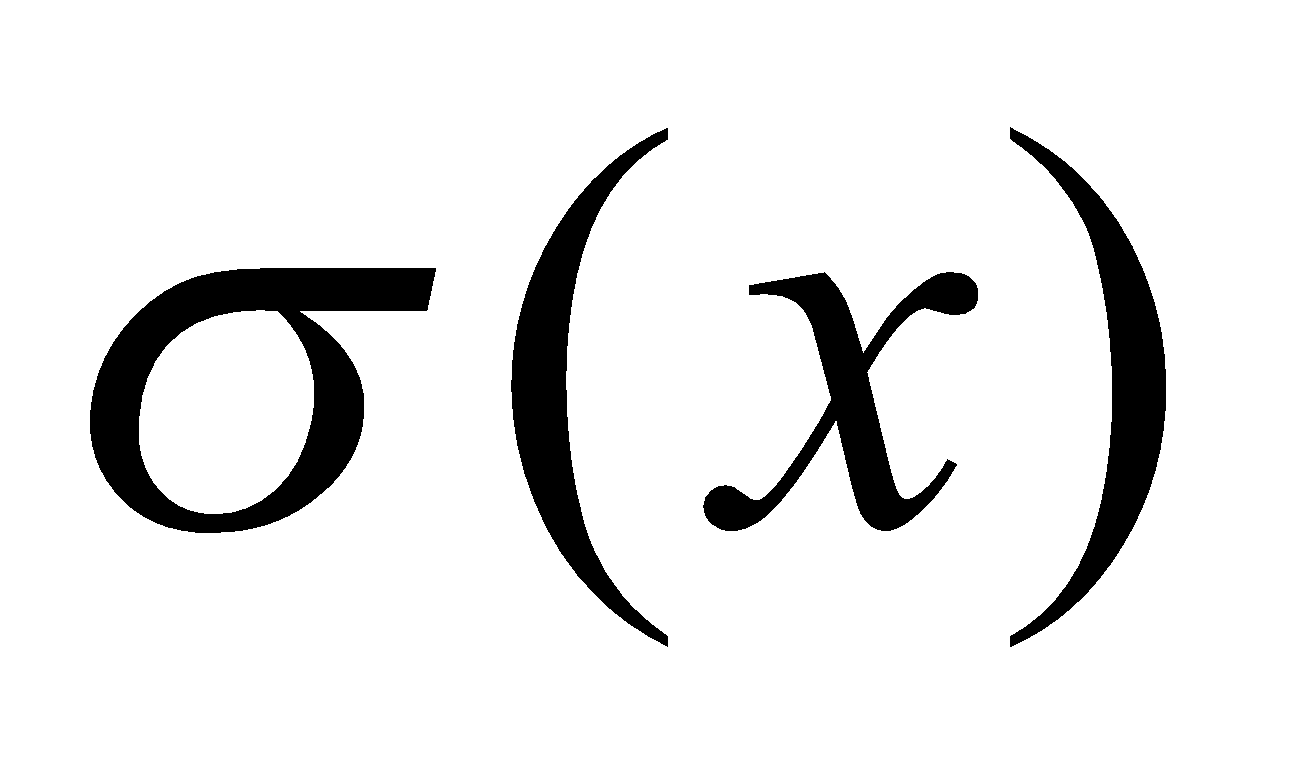
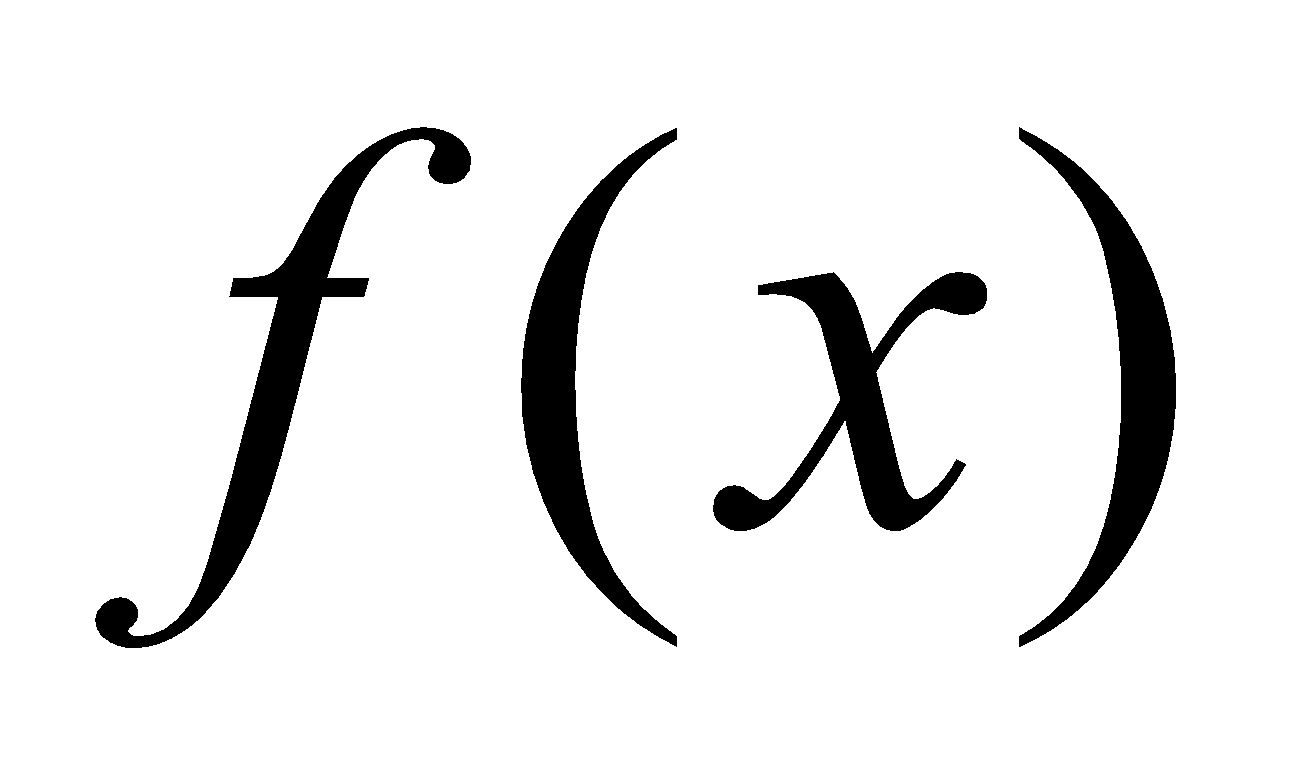
# *Формулювання крайової задачі*

Крайової задача:

  (1.1)

 (1.2)

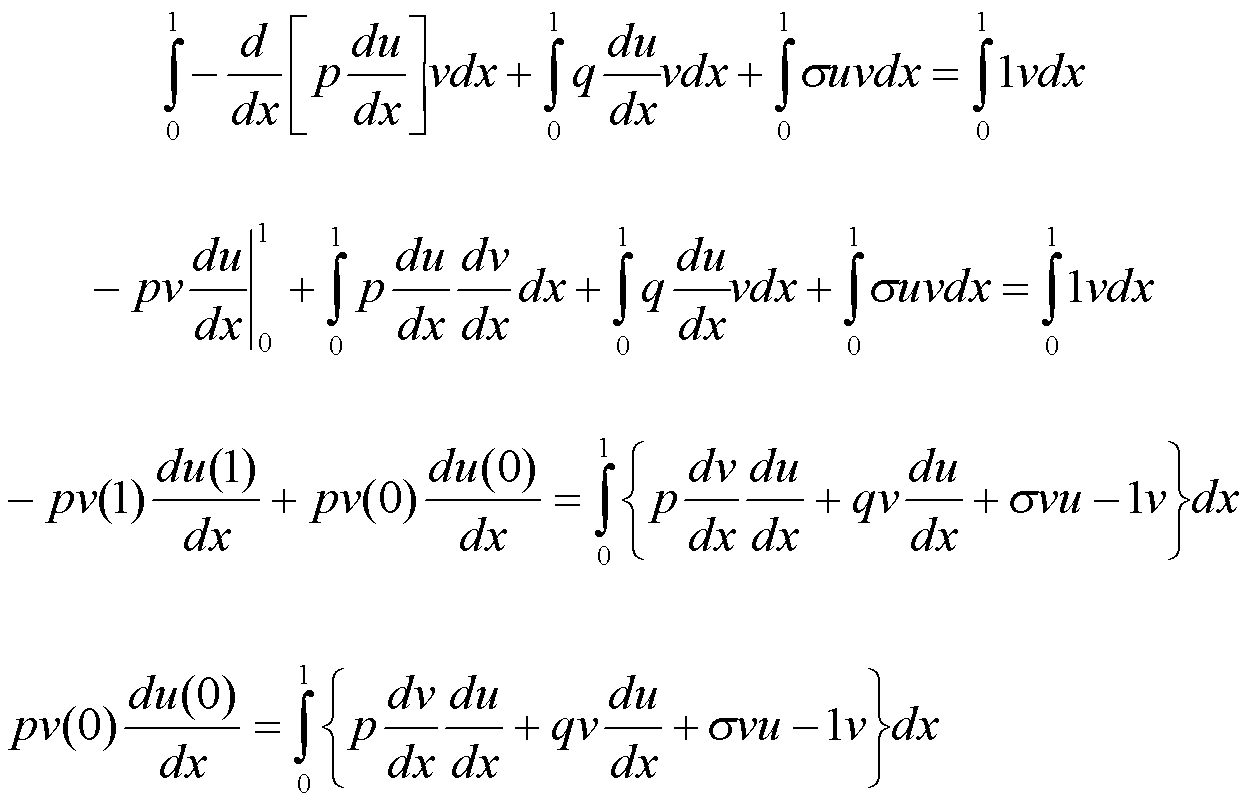
 (1.3)

Потрібно знайти функцію *u=u(x)*, яка на відрізку [0,1] задовольняє рівняння (1.1) і крайові умови (1.2) з врахуванням (1.3). В даному випадку ми розглядаємо, що *q(x)*, *p(x)* і є коефіцієнтами, а = 11. Зобразимо

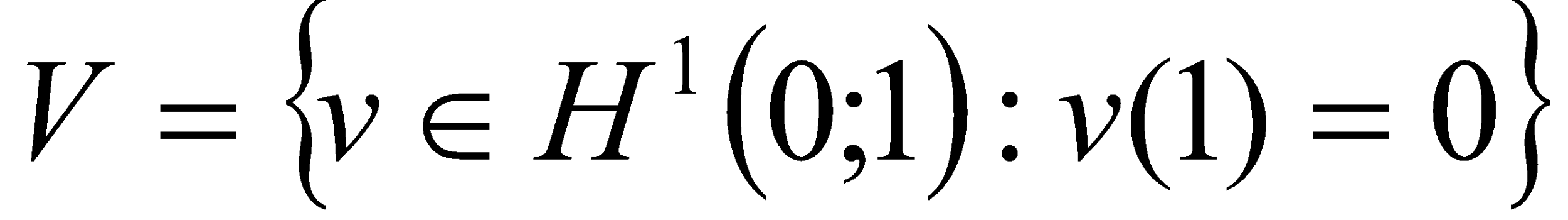
рівняння:  


# *2. Варіаційна постановка задачі*

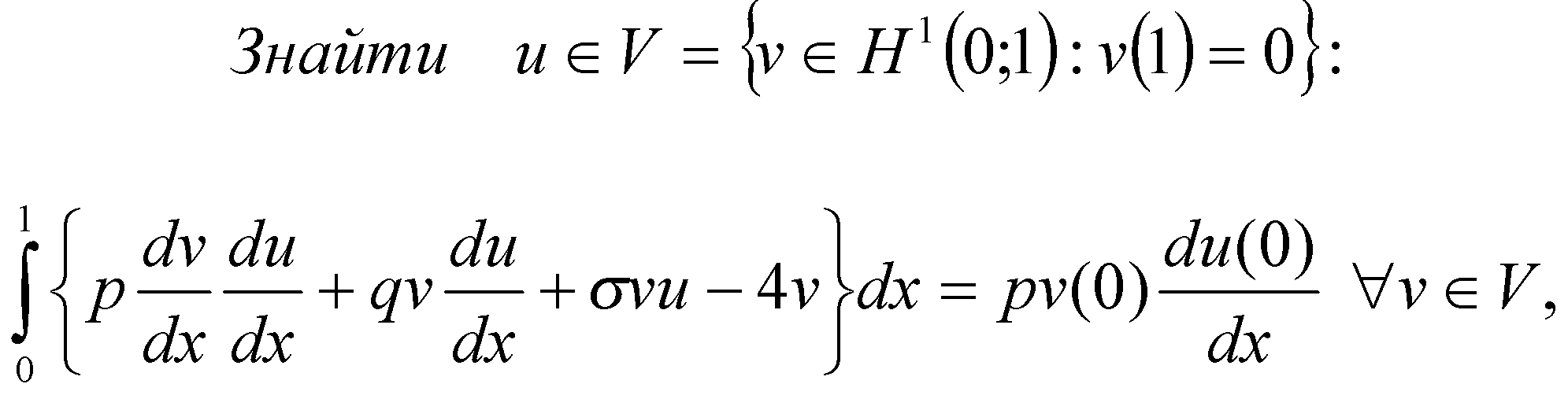
Щоб побудувати варіаційну постановку крайової задачі (1.1-1.2), введемо прості *V*. В ньому візьмемо деяку функцію *v=v*(*x*) і домножимо наше рівняння (1.1) на цю функцію, а потім проінтегруємо його частинами на проміжку [0,1]. У результаті отримаємо:

 (2.1)

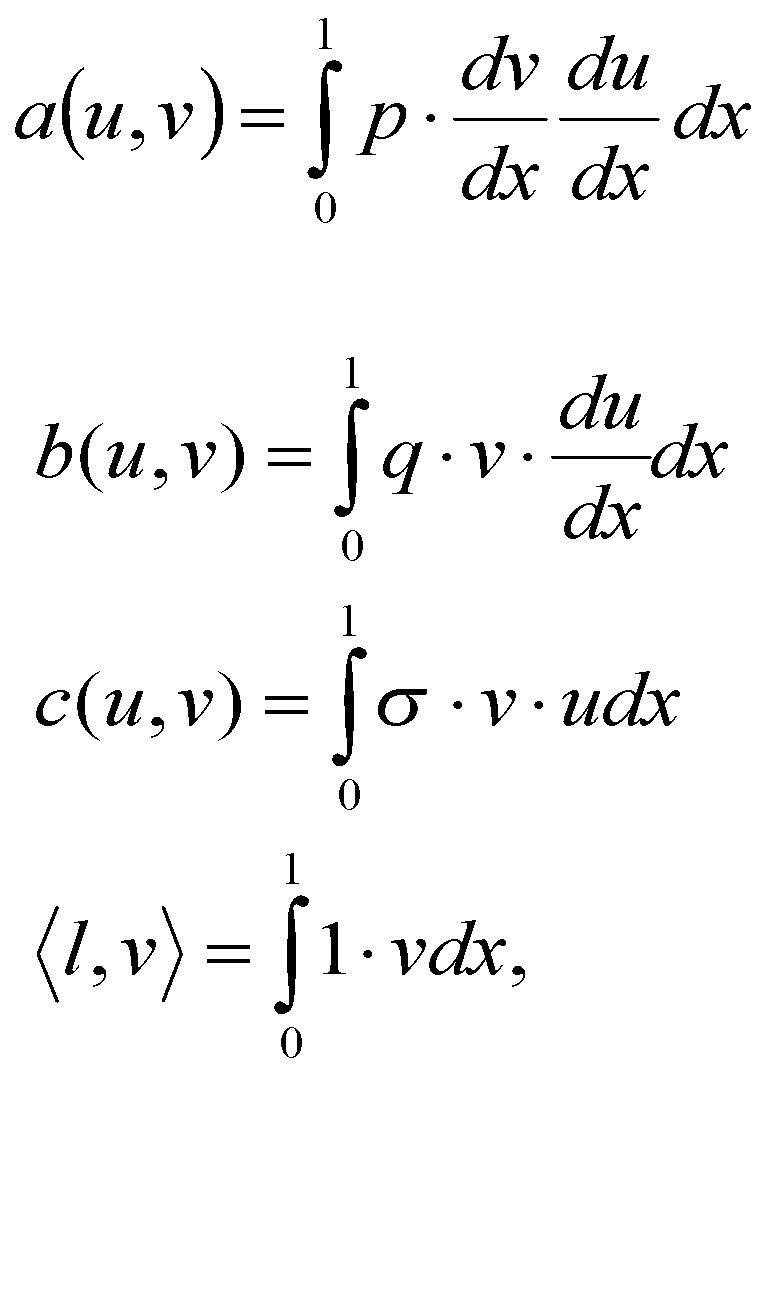
З огляду на одержану тотожність та крайові умови задачі (1.2) визначимо простір допустимих функцій

.

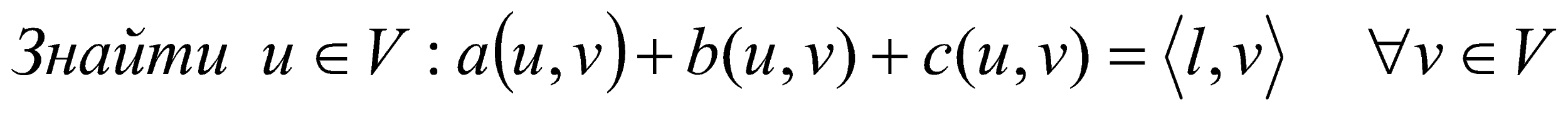
Тепер можна сформулювати відповідну (1.1)-(1.2) варіаційну задачу:

 (2.2)

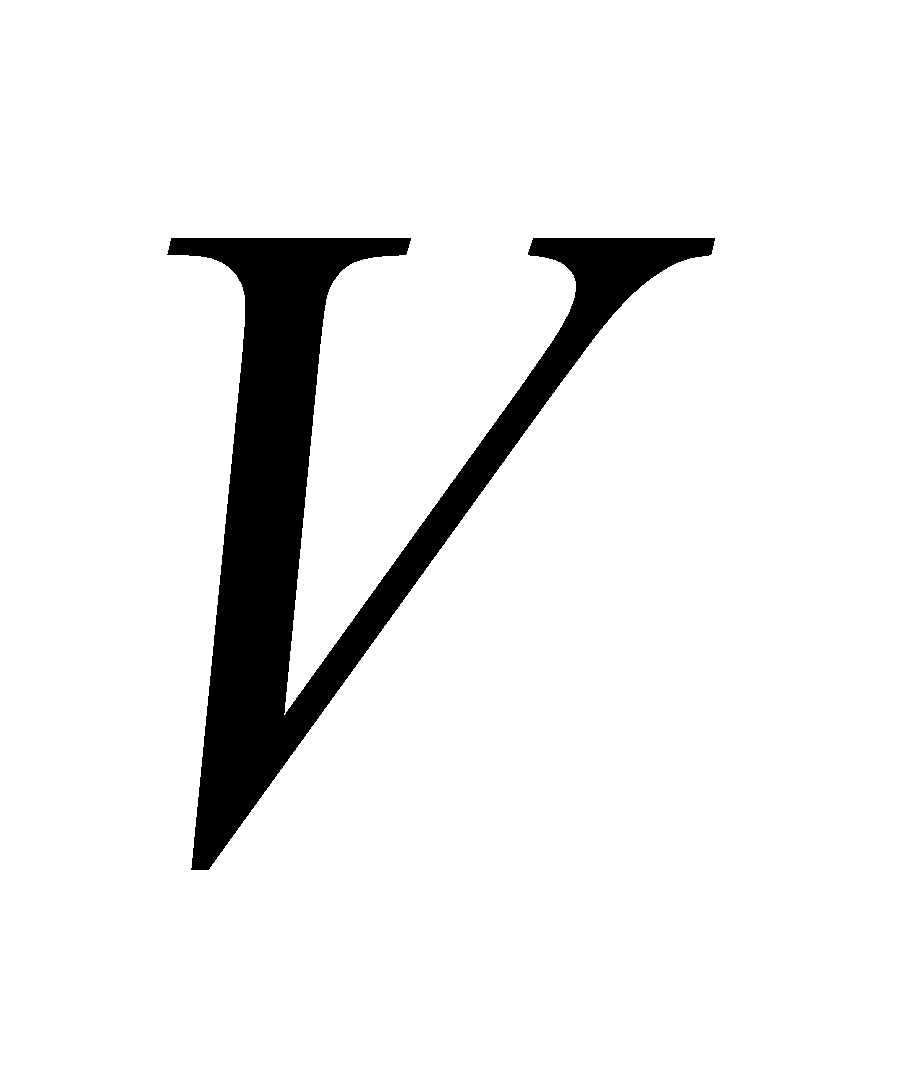
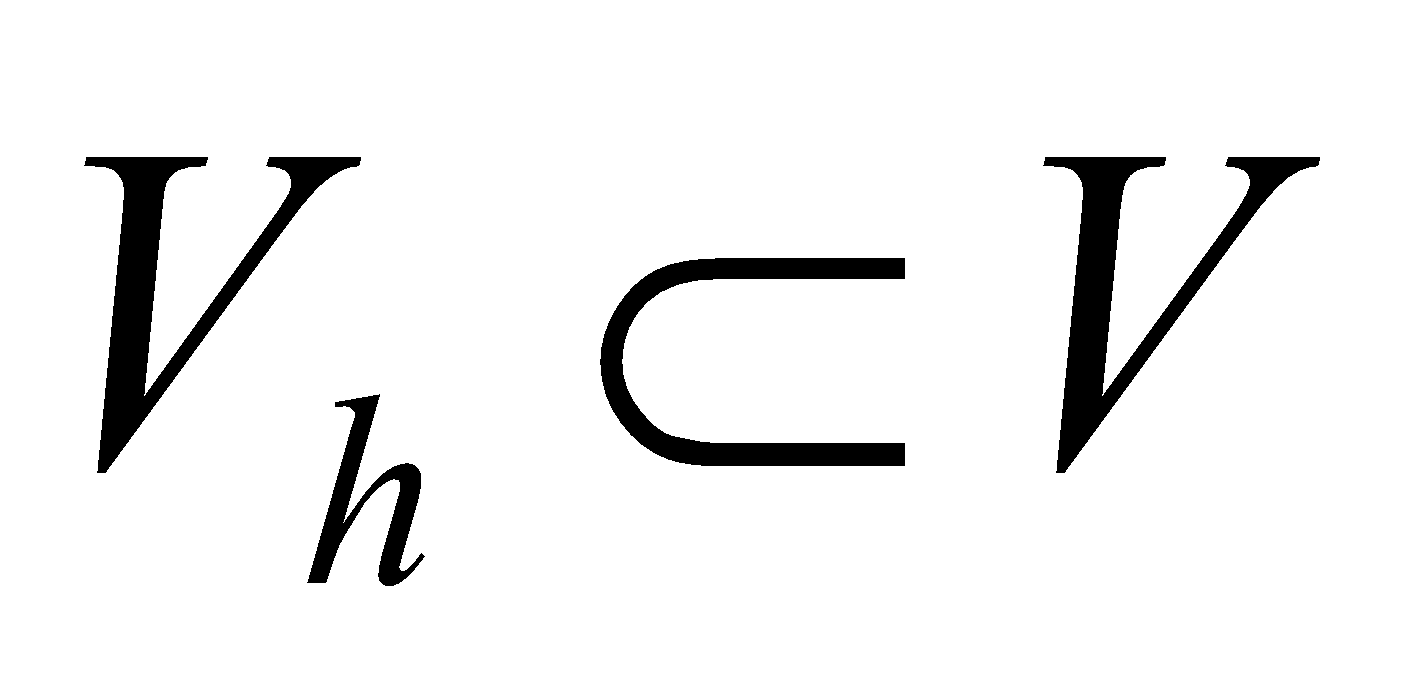
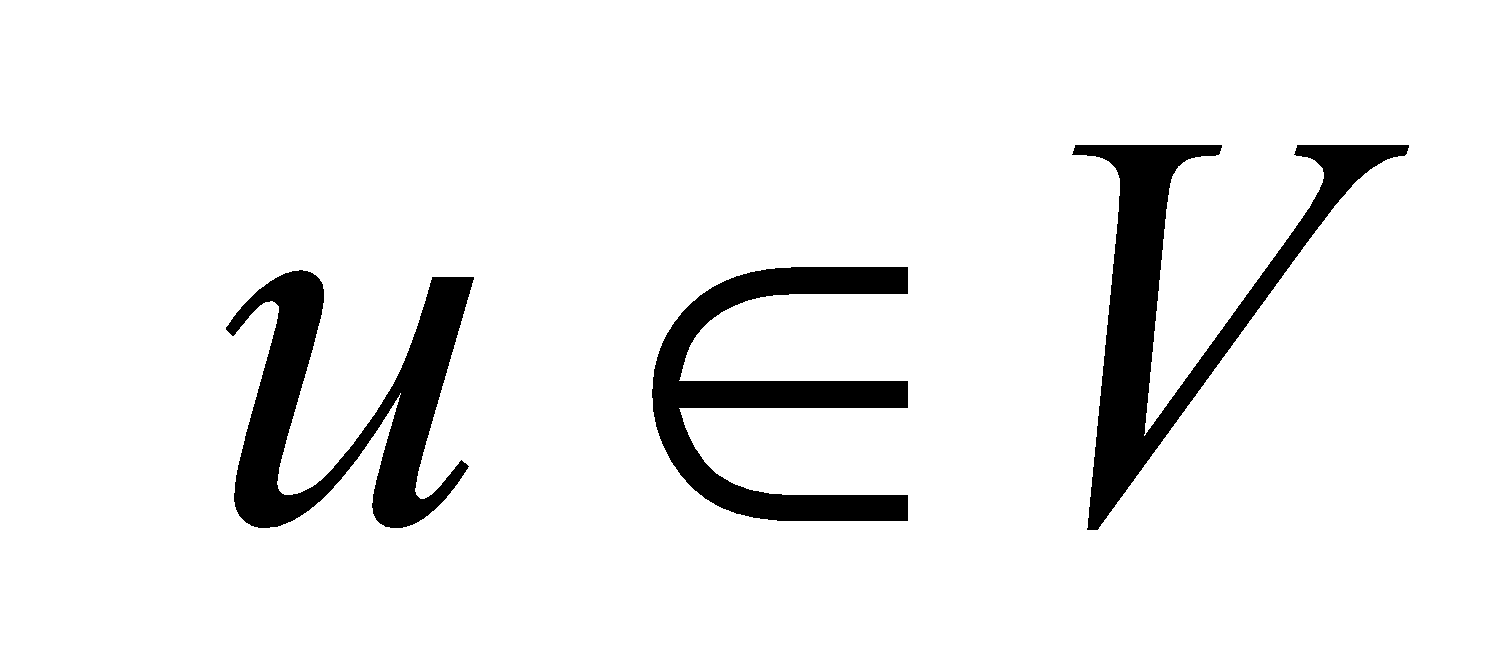
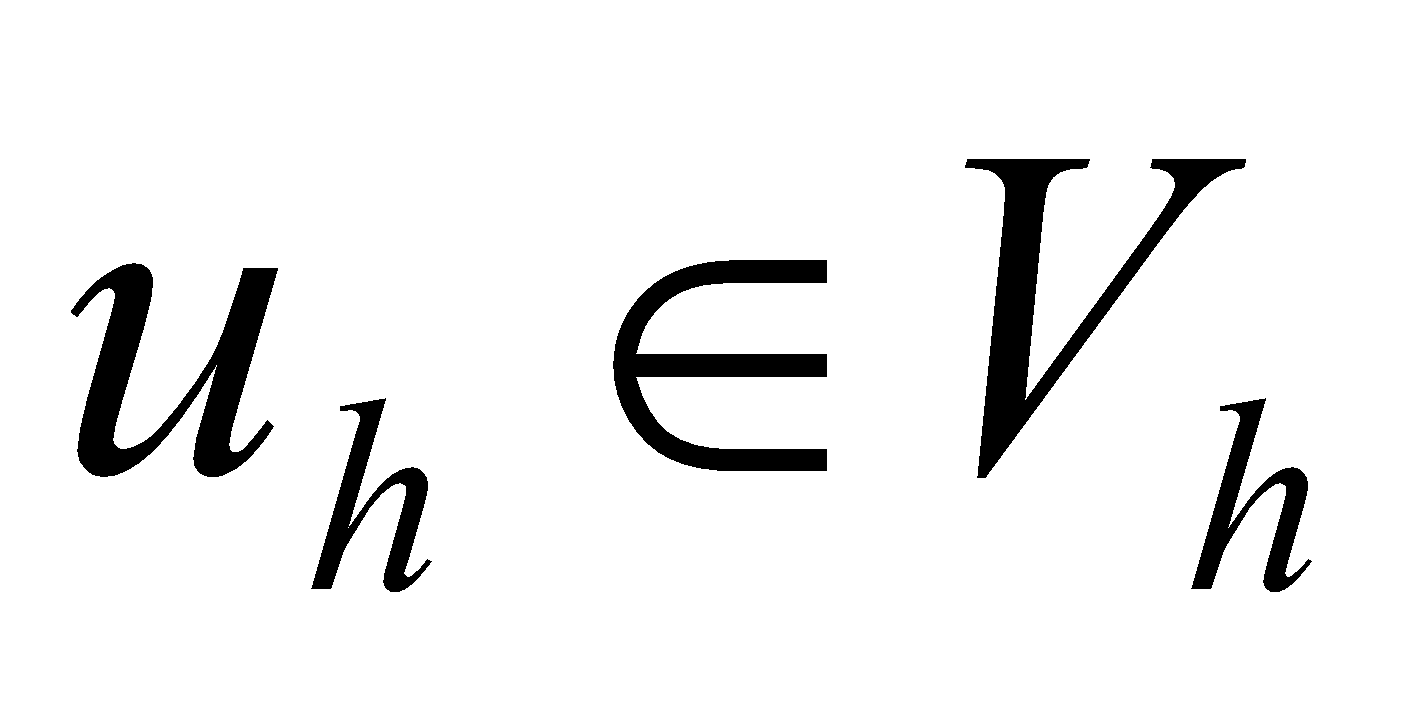
Введемо такі позначення:

 (2.3)

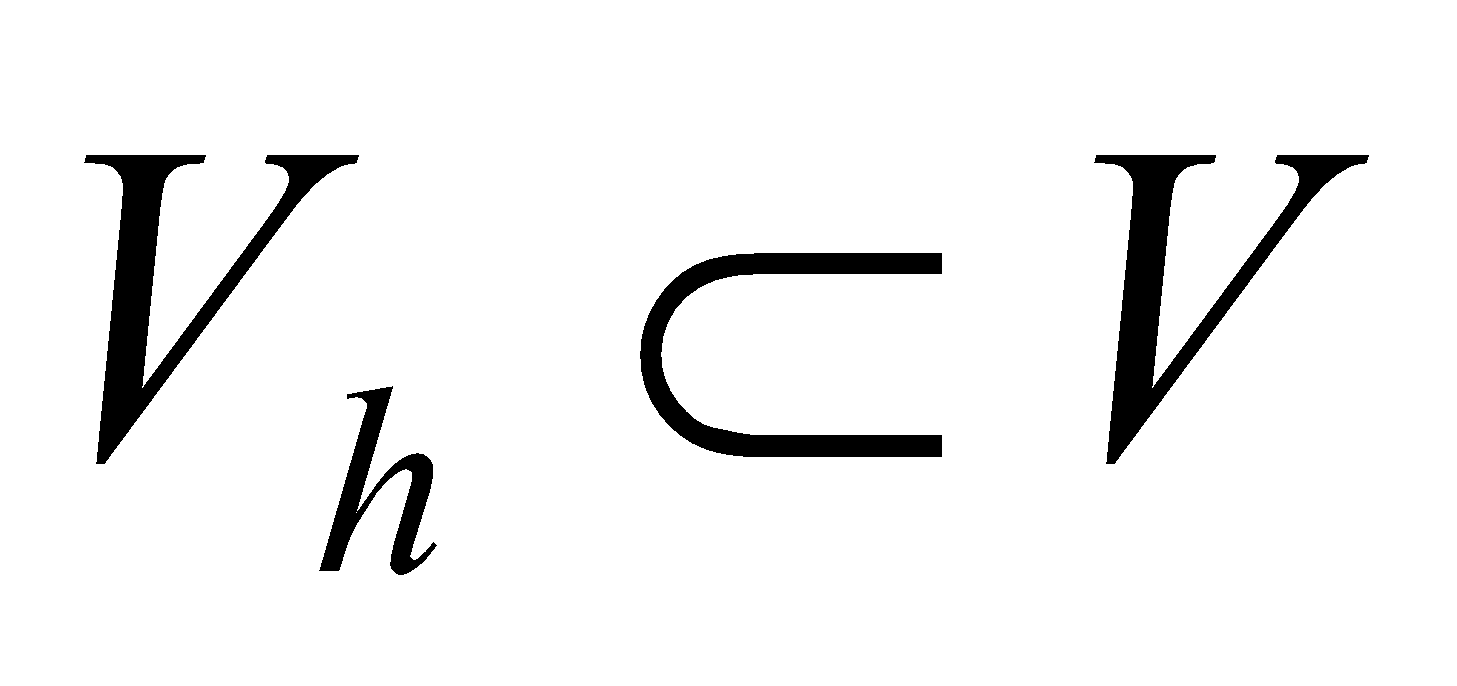
тоді дану задачу можна записати коротко у такому вигляді:

. (2.4)

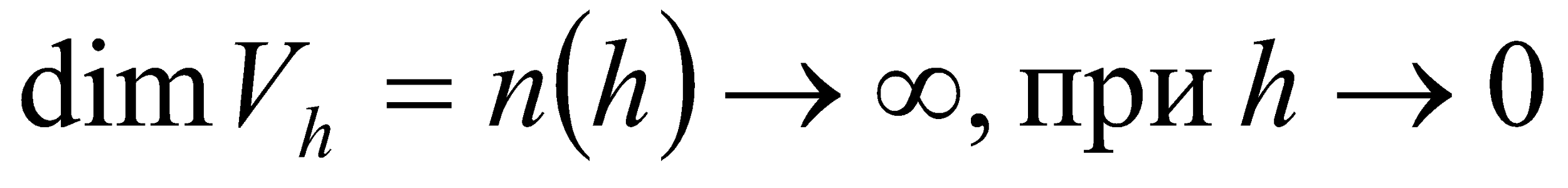
***3. Дискретизація Гальоркіна***

Основна ідея методу Гальоркіна полягає в перенесенні розв’язування варіаційної задачі з нескінченновимірного простору допустимих функцій  в належним чином вибраний скінченновимірний дискретний підпростір . Замість точного розв’язку  варіаційної задачі ми одержуємо наближення .

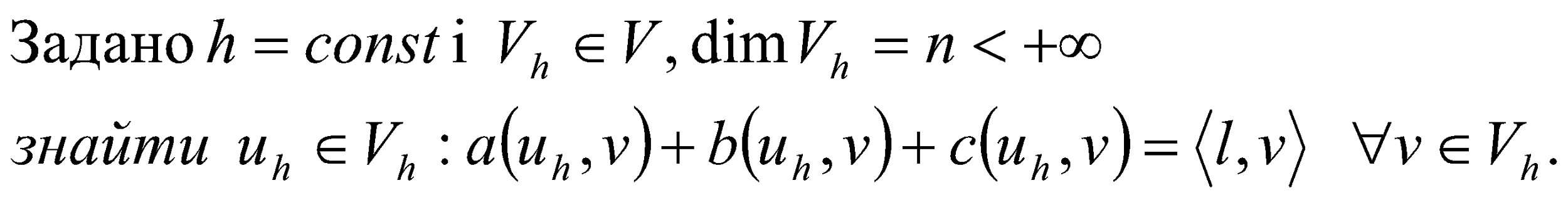
У заданому нами просторі допустимих функцій *V* виберемо підпростір

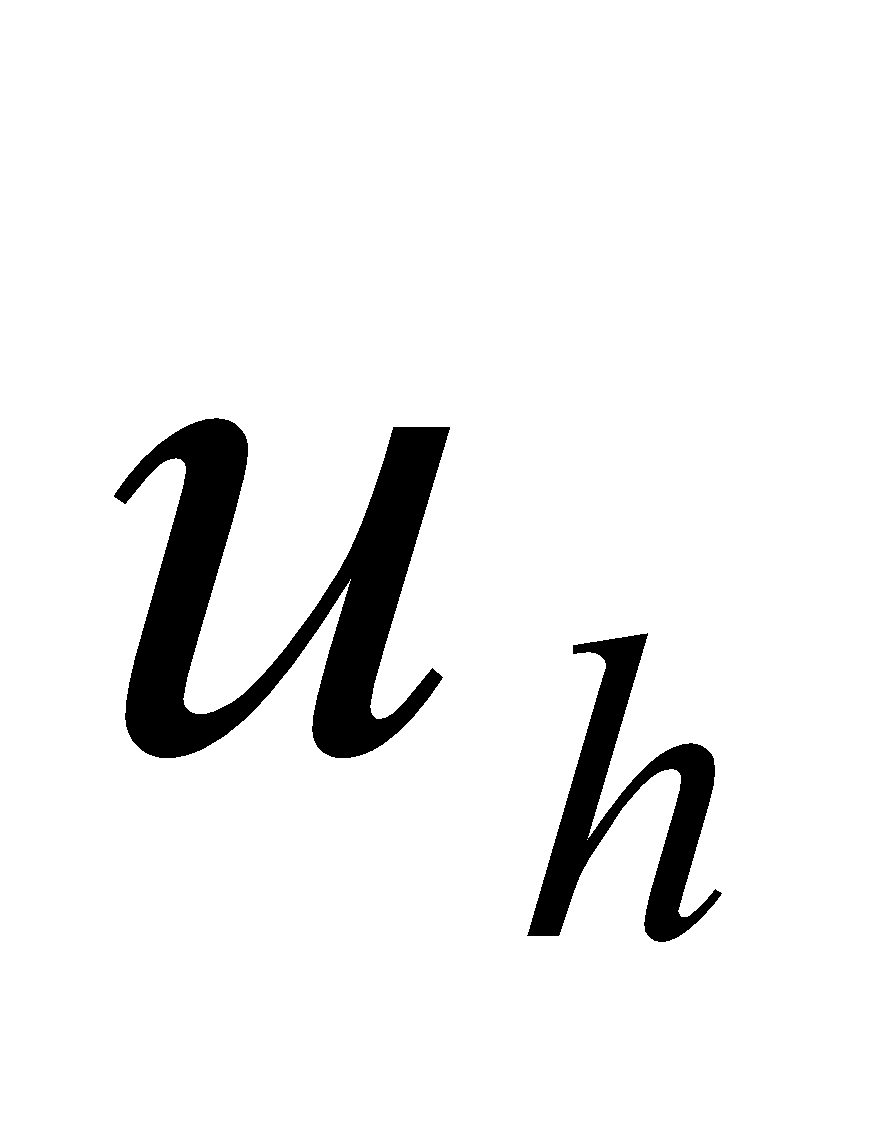
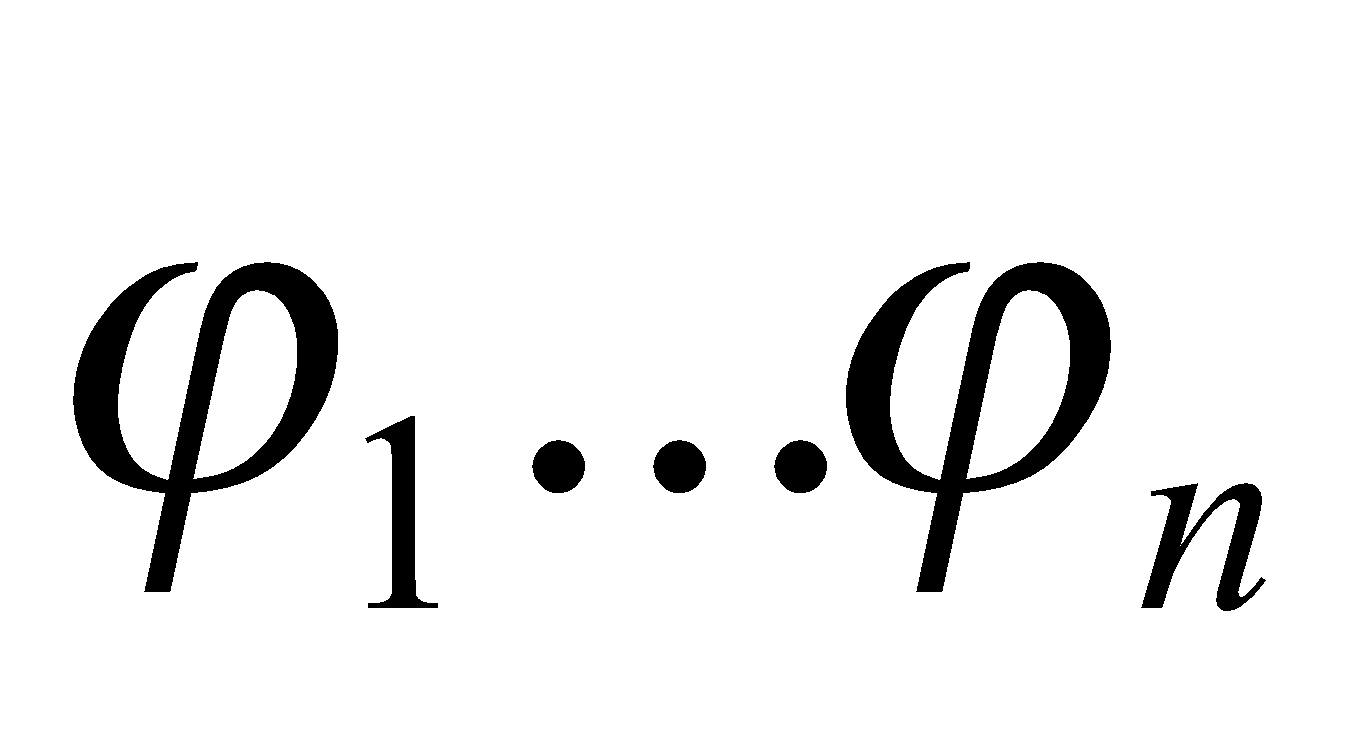
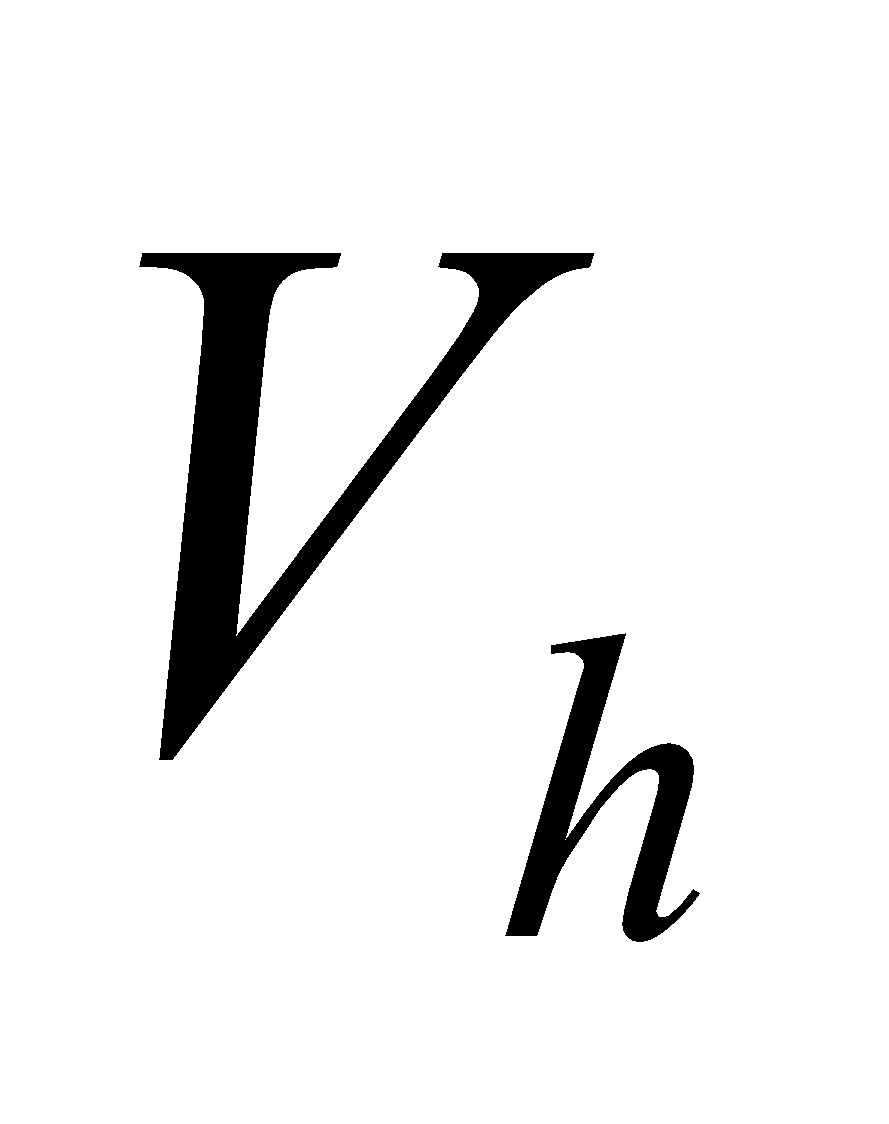


такий, що виконується умова

,

тоді варіаційна задача переформульовується в послідовність задач

 (3.1)

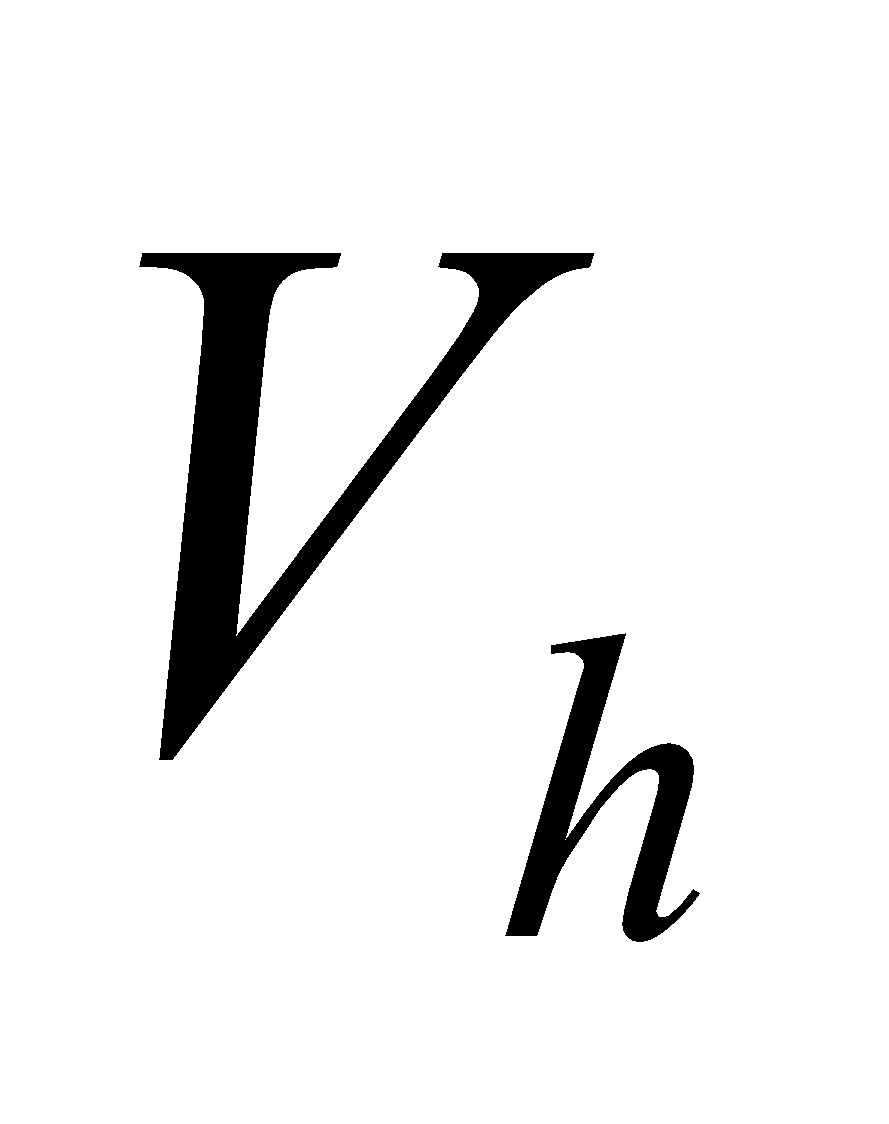
Щоб обчислити  із (3.1) потрібно вибрати базис  в просторі ; тоді :

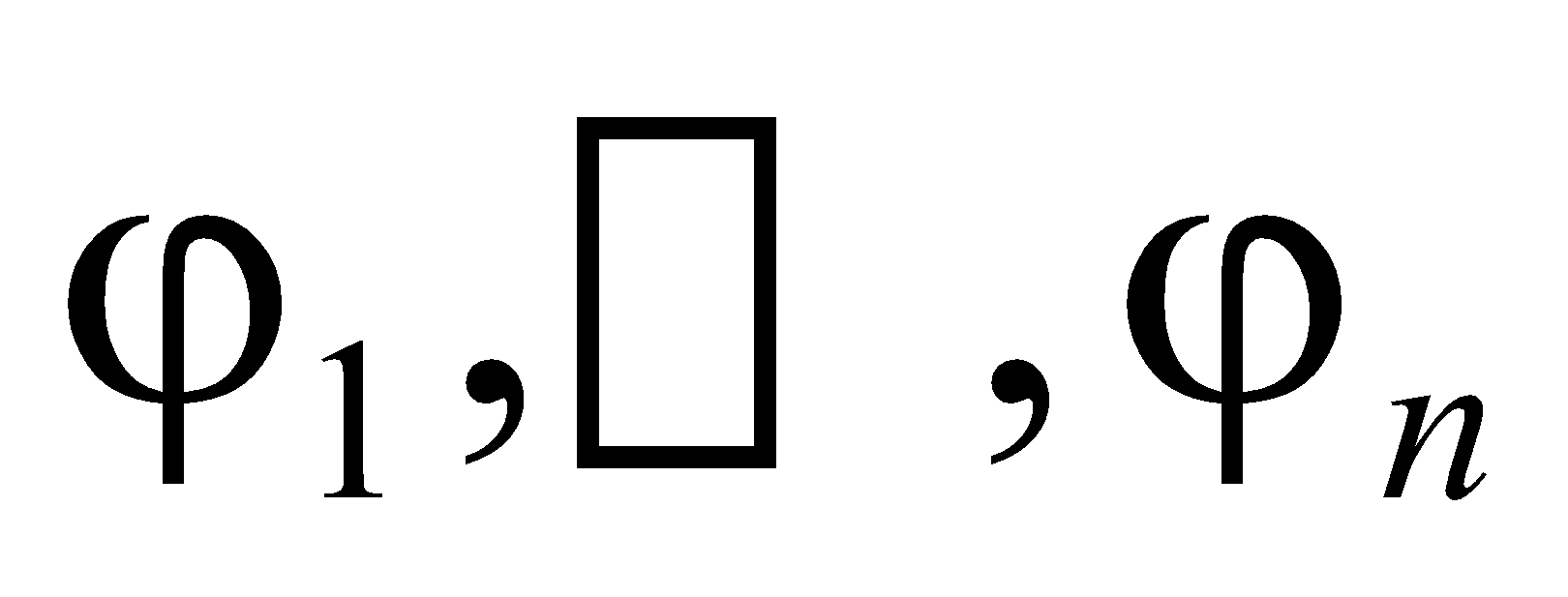
Перепишемо задачу (2.3) :

(3.2)

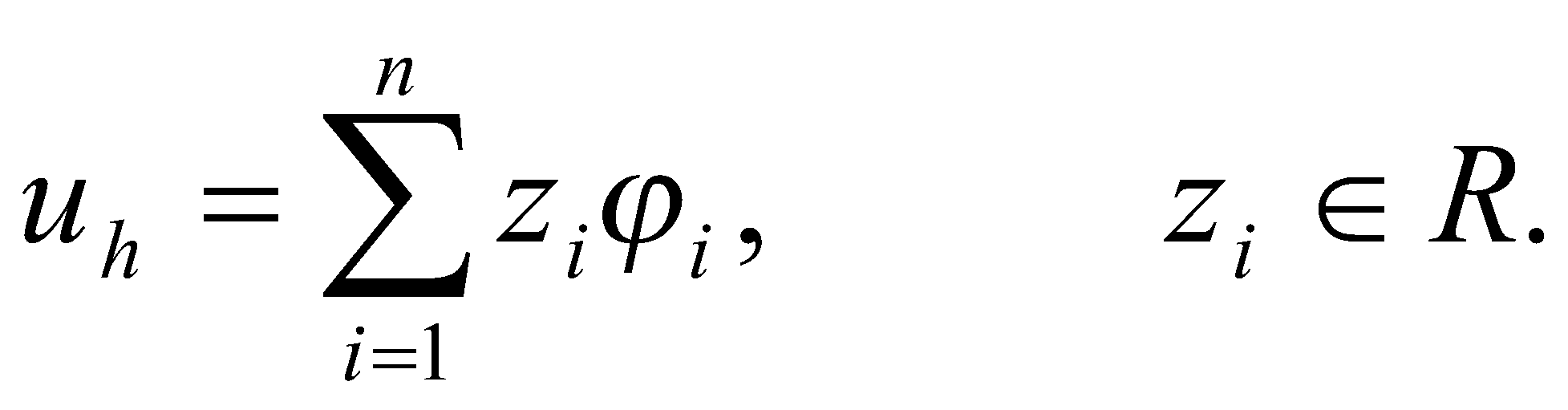
***4. Метод скінченних елементів***

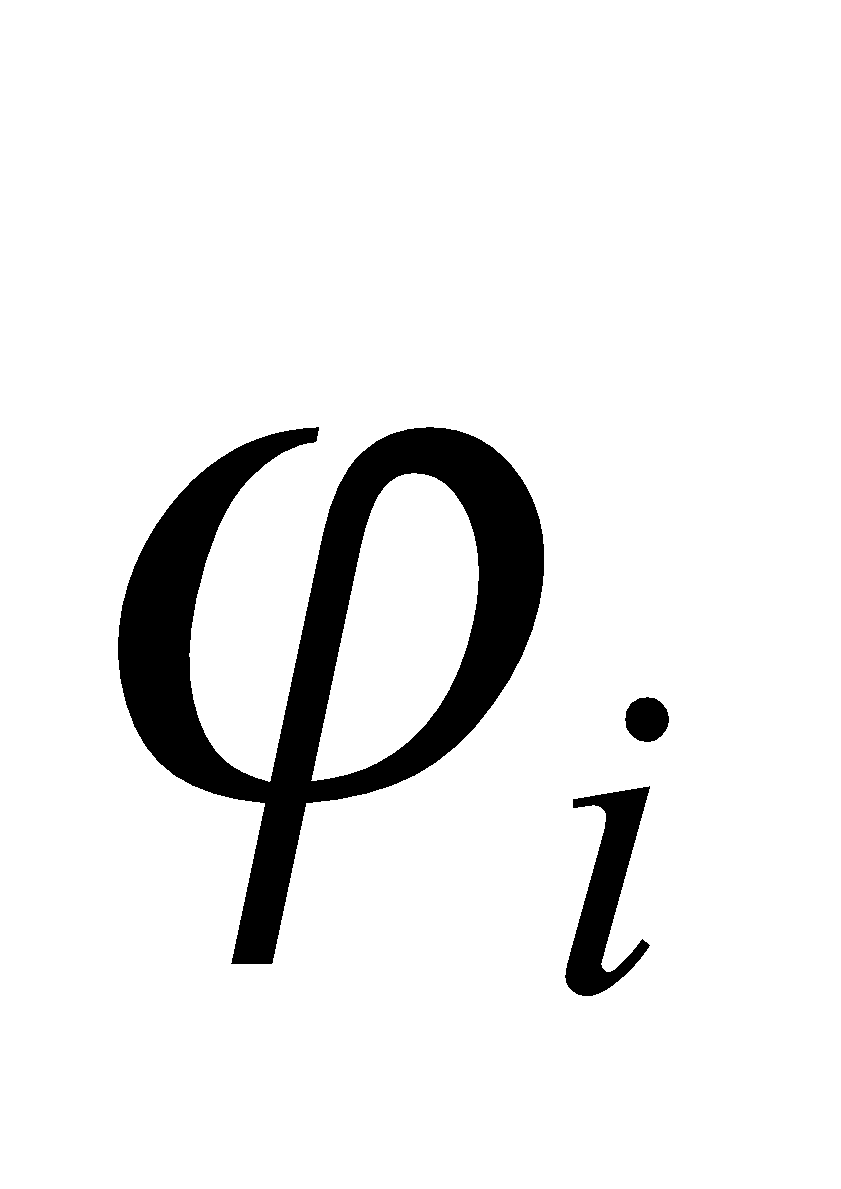
**4.1 Кусково-лінійні апроксимації: функції Куранта**

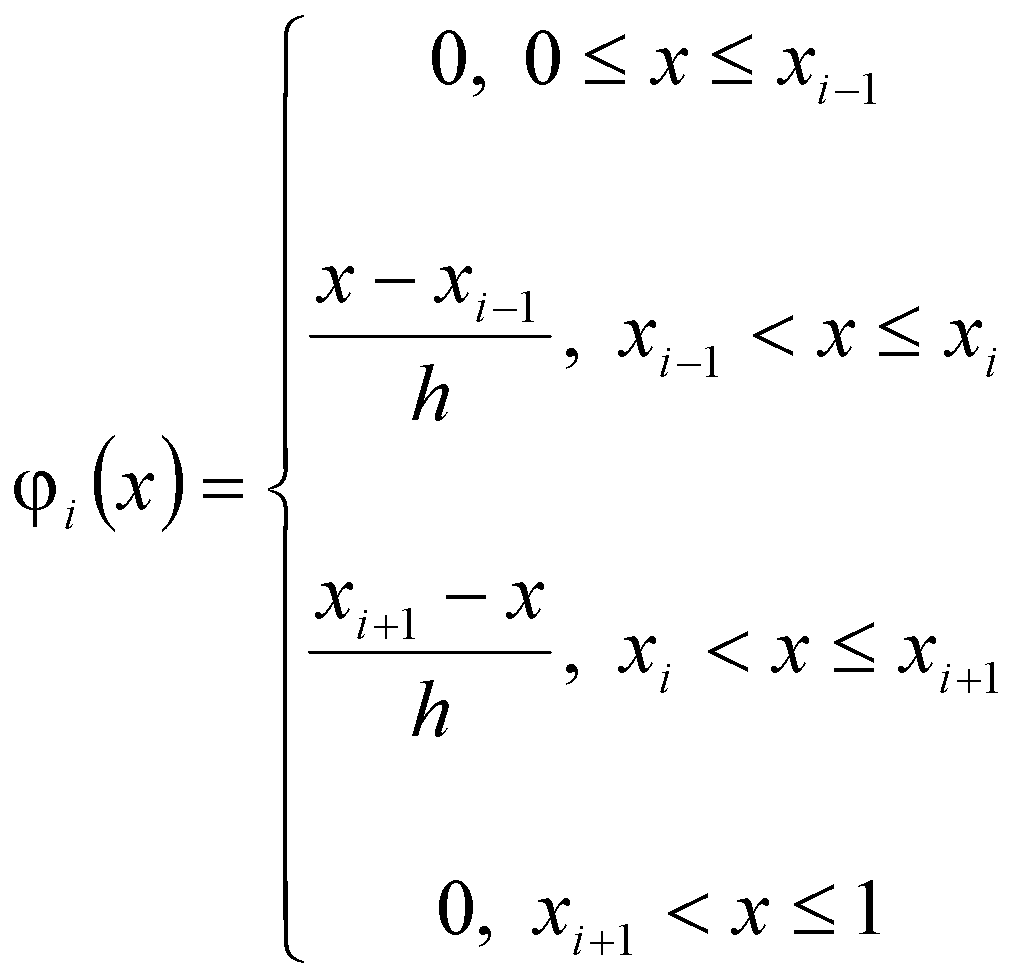
Нехай базисом простору  будуть функції

,

тоді будь–який елемент з цього простору ми можемо розписати за базисними функціями

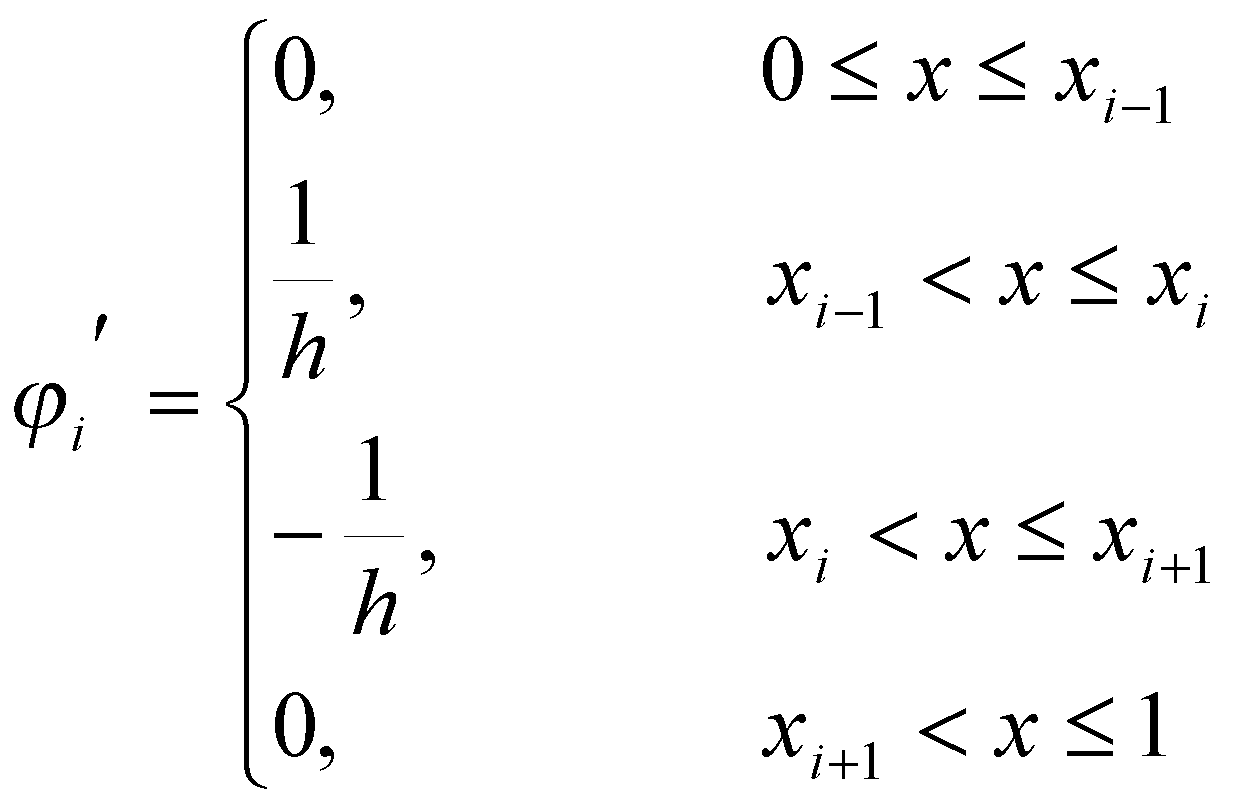
 (4.1)

Нехай  за припущенням є функціями Куранта для рівномірного розбиття. Ці функції мають такий вигляд:

, (4.2)

де *h=1/n*, *xi=ih*, *i=1,…,n*.

Похідні від функцій Куранта будуть рівні наступним виразам

, (4.3)

Графічно функції Куранта для рівномірного розбиття можна зобразити так:

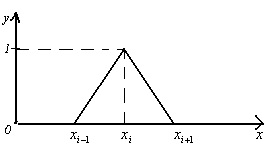
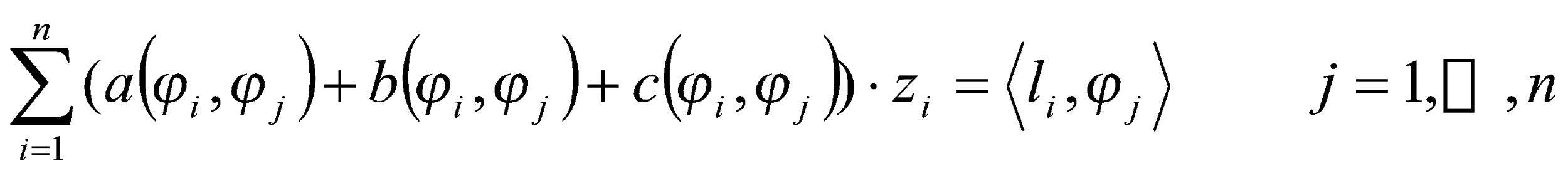


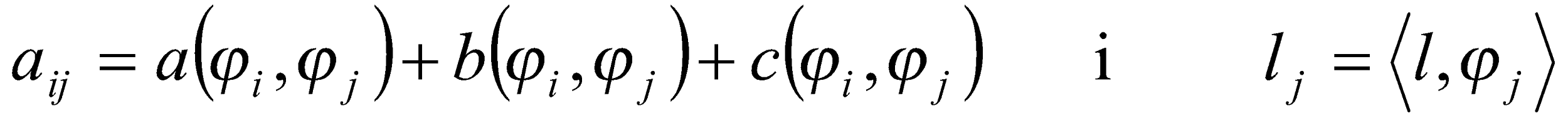
Рис.4.1. Графік функції Куранта для рівномірного розбиття

У результаті отримаємо співвідношення

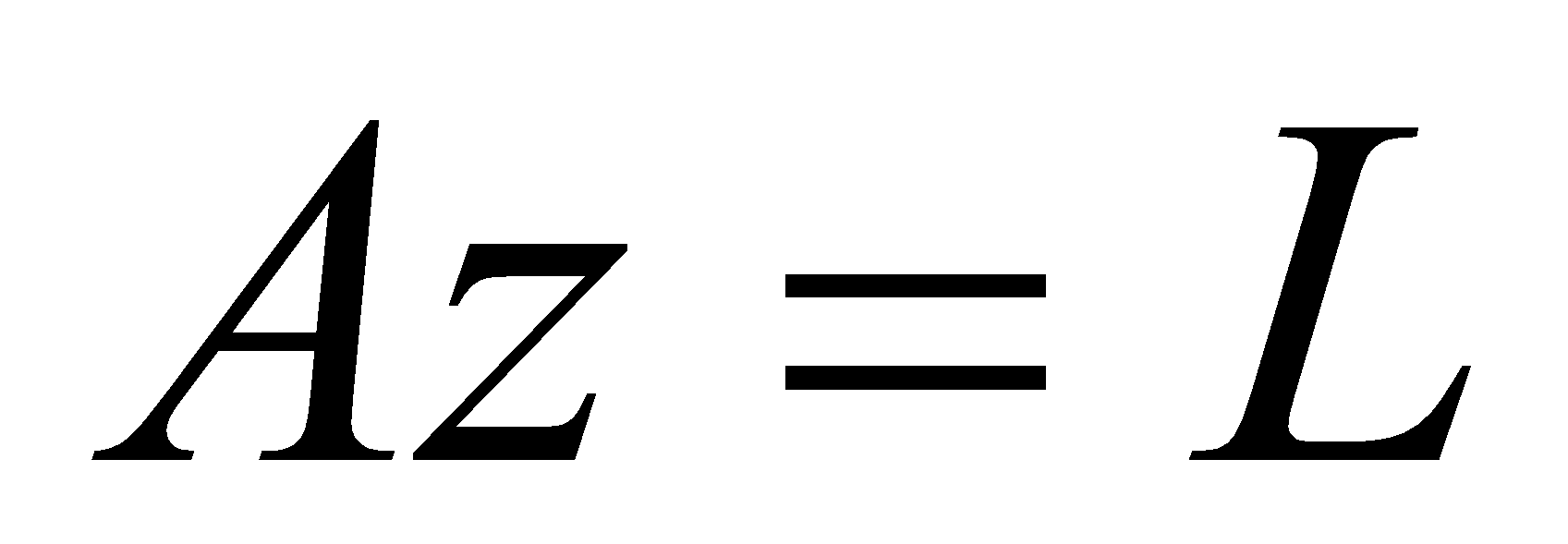
.

**4.1.1. Побудова системи лінійних алгебраїчних рівнянь**

Якщо позначити

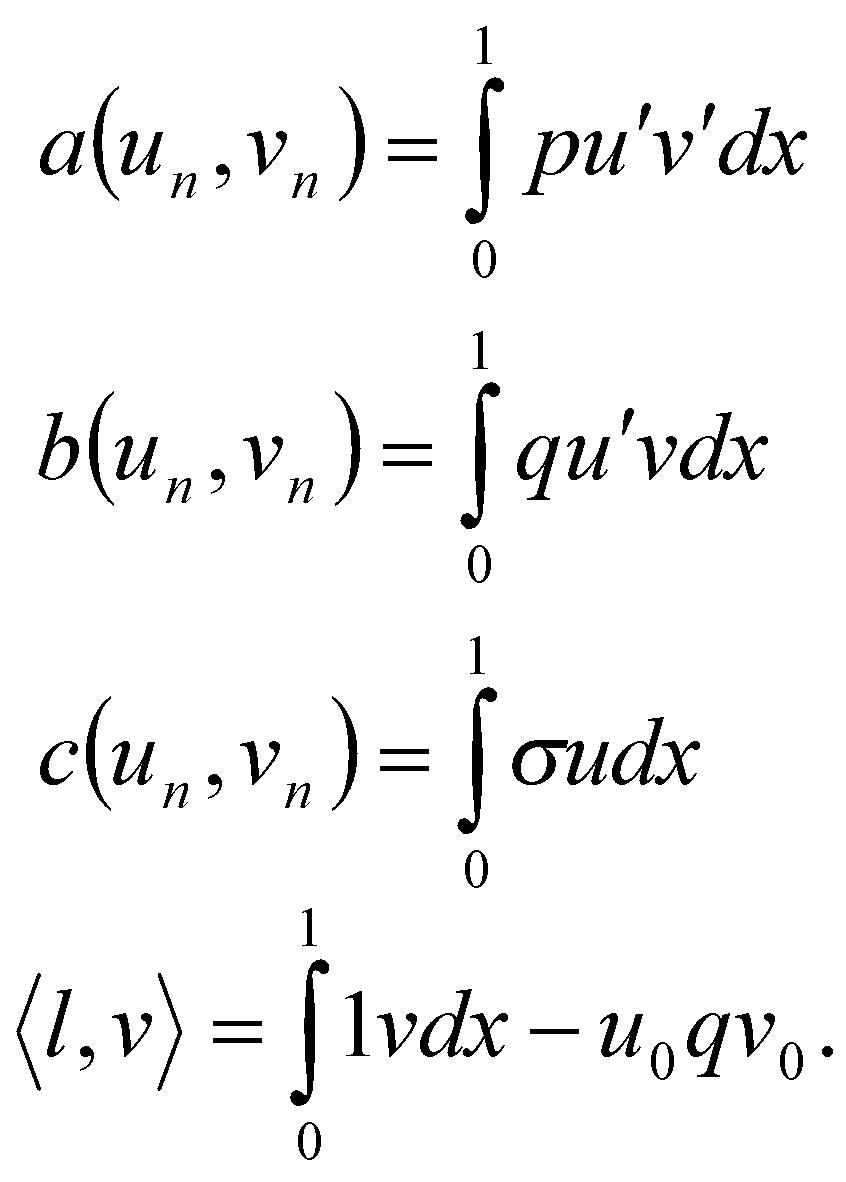
,

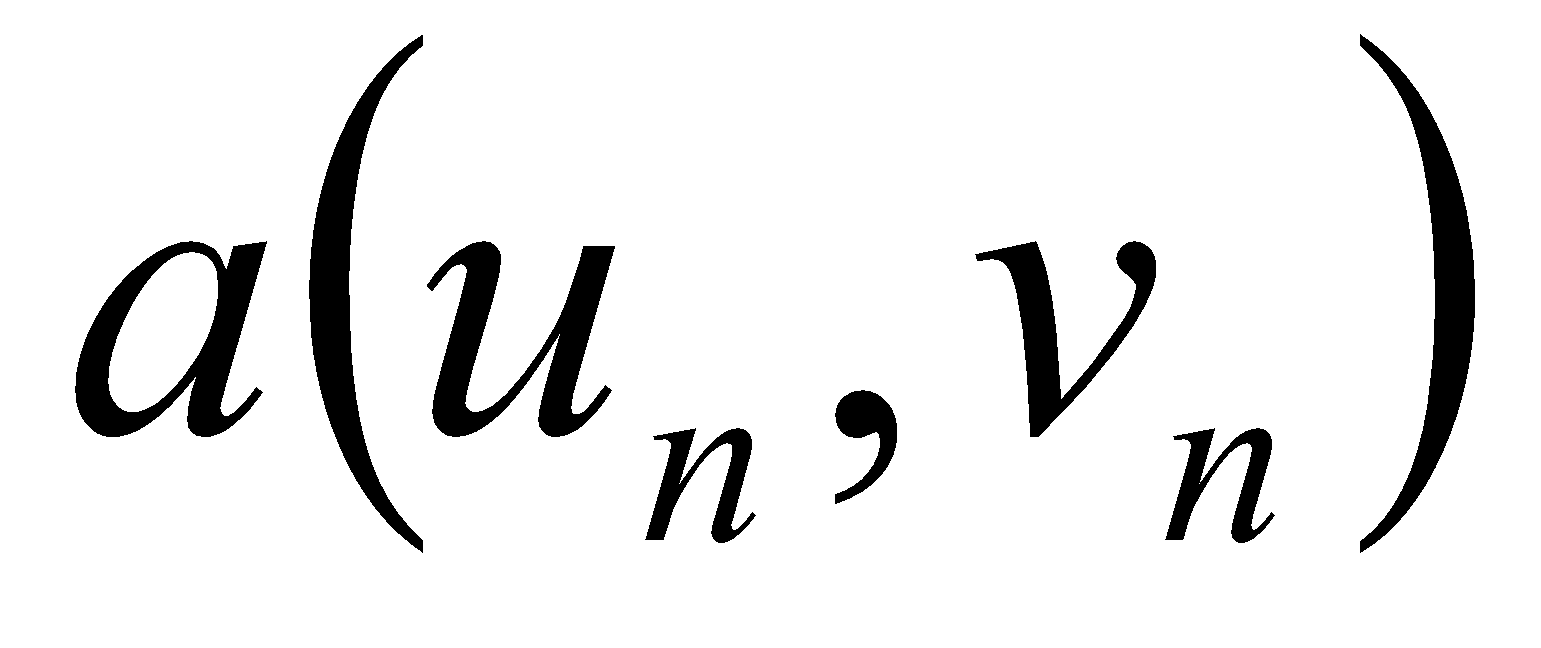
то ми отримаємо систему лінійних алгебраїчних рівнянь з *n* невідомими

 (4.4)

Таким чином ми, використавши метод Гальоркіна, звели варіаційну задачу (3.2) до системи лінійних алгебраїчних рівнянь (4.4).

Коефіцієнти системи лінійних алгебраїчних рівнянь можна записати так

 (4.5)

Обчислюємо  та формуємо матрицю А:

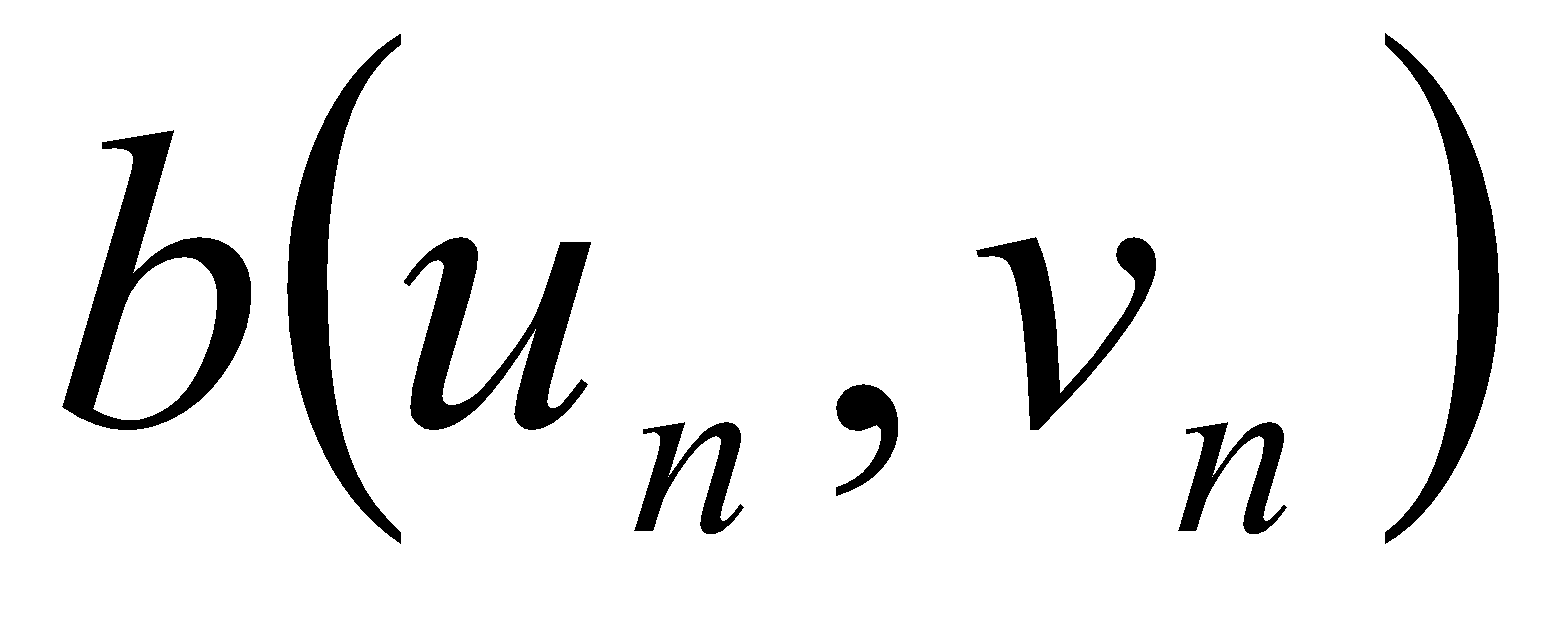
Для

Для

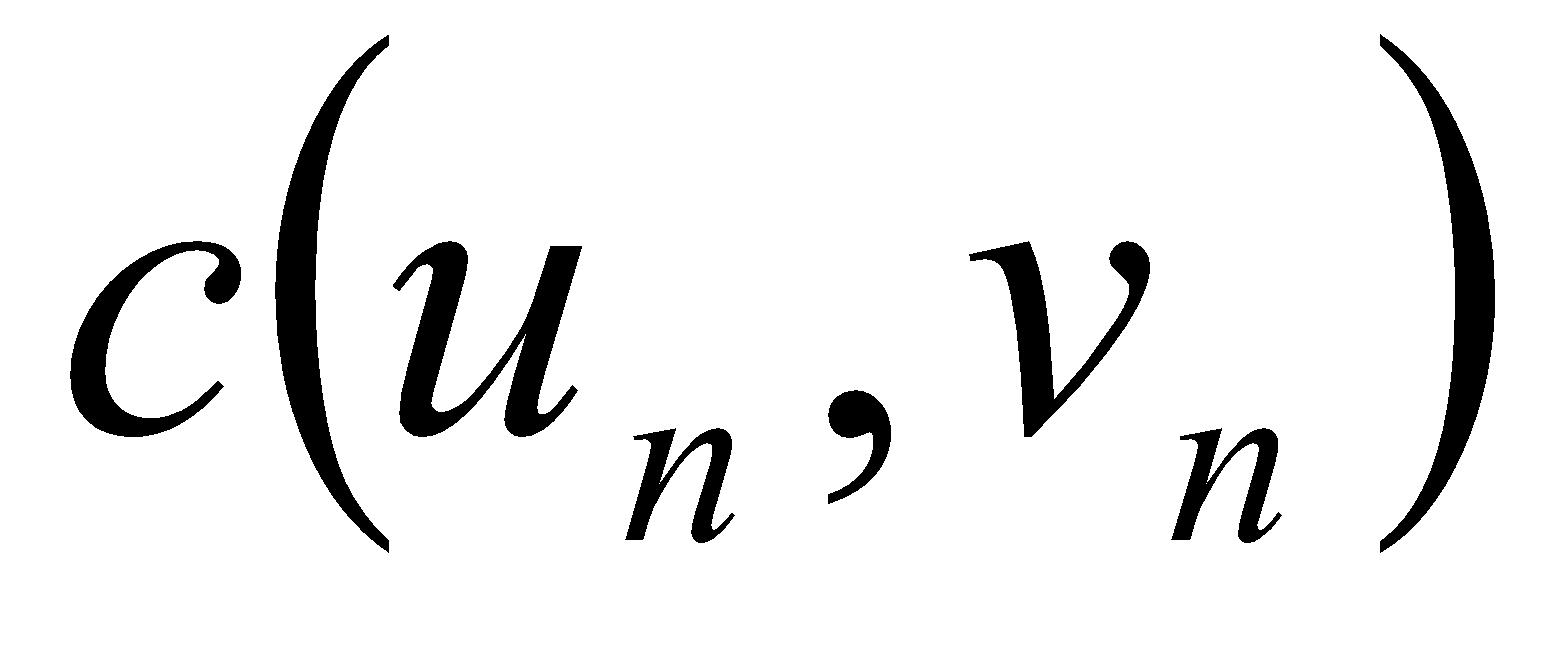
Для

Для

Напишемо загальний вигляд матриці А(тридіагональна):

Обчислюємо  та формуємо матрицю B:

Напишемо загальний вигляд матриці B(діагональна):

Обчислюємо  та формуємо матрицю C:

Напишемо загальний вигляд матриці C(діагональна):

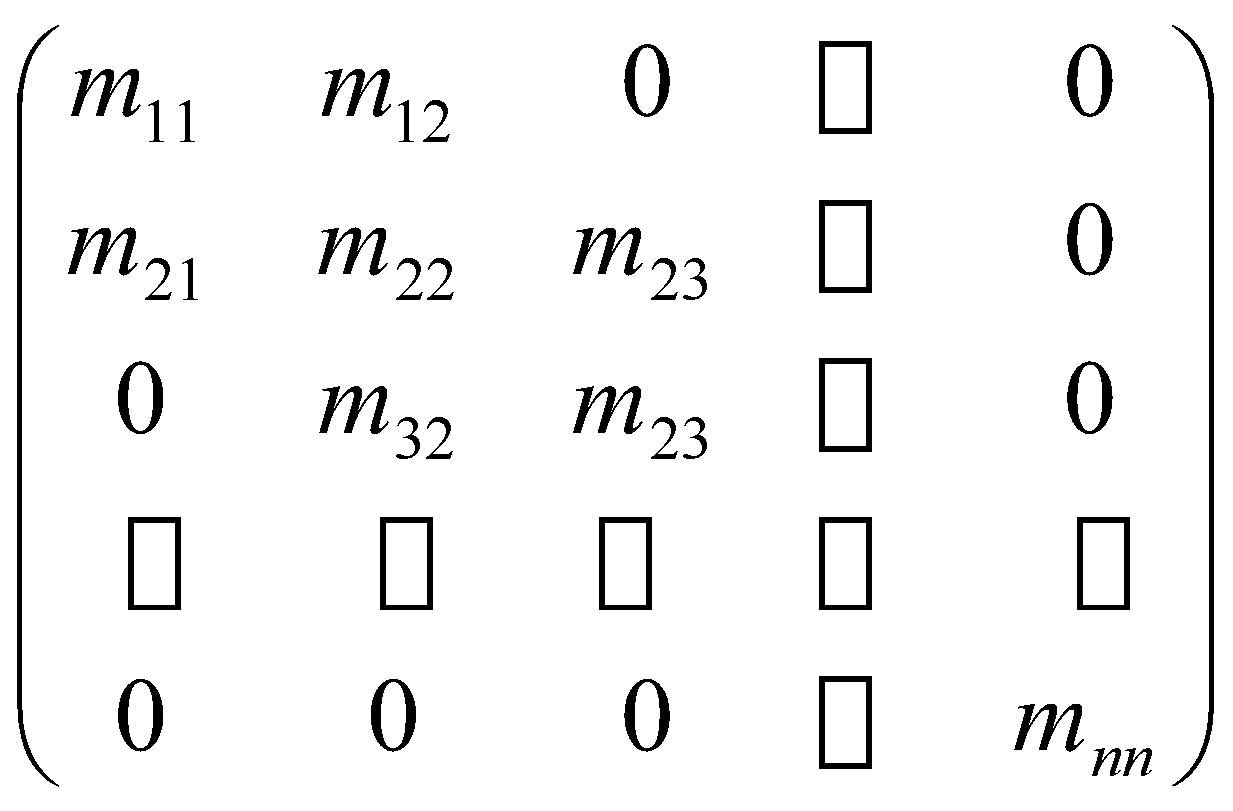
Напишемо загальний вигляд матриці (тридіагональна):

Обчислюємо та формуємо матрицю L:

Напишемо загальний вигляд матриці L(діагональна):

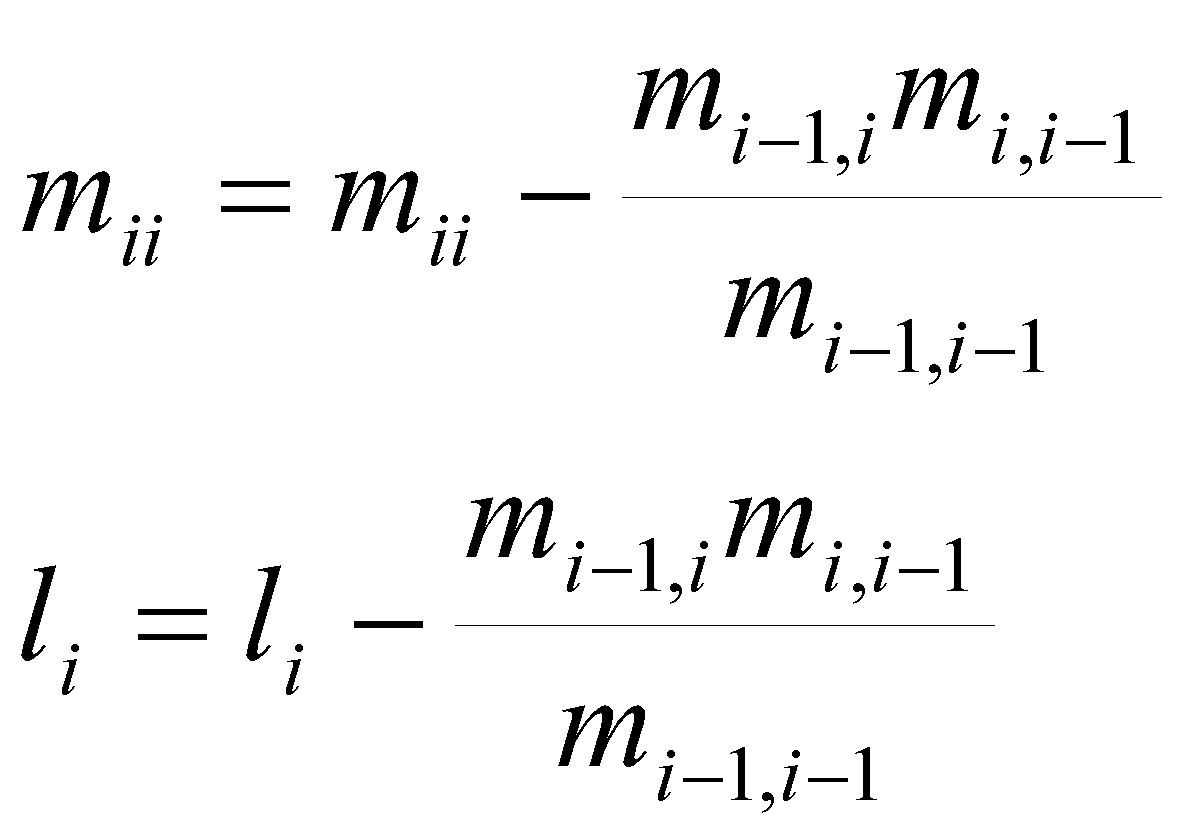
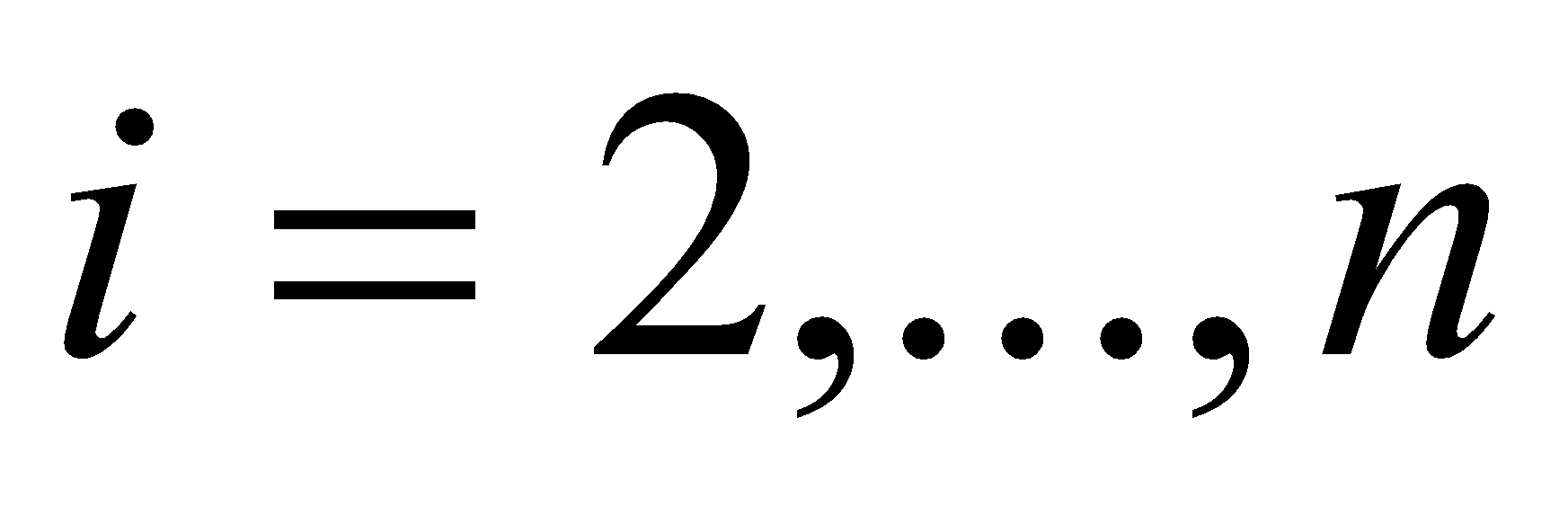
**4.1.2. Розв’язування системи рівянь МСЕ: метод прогонки**

Матриця, яку ми отримали є тридіагональною. Для розв’язування системи лінійних алгебраїчних рівнянь з такою матрицею зручно використовувати метод прогонки. Зберігання всієї матриці є нераціональним щодо використання пам’яті комп’ютера, оскільки нас цікавлять лише ненульові елементи. Тому будемо зберігати лише три діагоналі утвореної матриці. Наприклад, нехай дана матриця

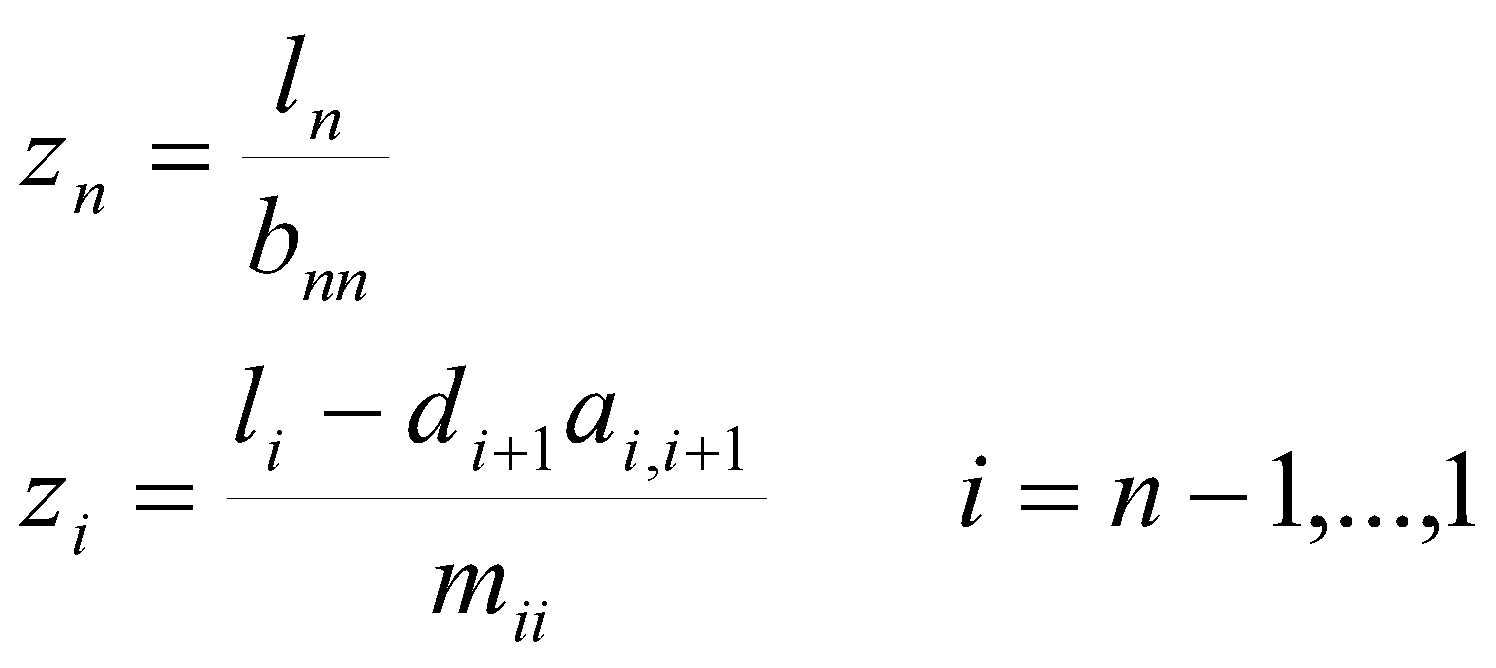
,

тоді в нашій схемі вона буде виглядати так

Сформулюємо ідею методу прогонки: піддіагональні елементи потрібно зробити нулями на прямому ході, а розв’язок знайти на зворотньому. Ось основні формули реалізації прямого ходу методу прогонки

і зворотнього



Код програми

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using ZedGraph;

using Accord.Math;

namespace SimpleSignal\_Zaiats

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

static double m(double x)

{

return Math.Sin(x);

}

static double b(double x)

{

return Math.Pow(x, 2) + 1;

}

static double si(double x)

{

return 5;

}

static double f()

{

return 11;

}

static double[] u(double h,int n)

{

double[] u = new double[n+1];

double[,] arr = new double[n+1, n+1];

double x = 0;

// заповнюємо матрицю, білінійна форма

for (int i = 0; i < n + 1; i++)

{

for (int j = 0; j < n + 1; j++)

{

arr[i, j] = 0;

if (i == j + 1)

{

arr[i, j] = -m(x) / h - b(x) / 2 + si(x) \* h / 6; // 4 рядок

}

if (j == i + 1)

{

arr[i, j] = -m(x) / h + b(x) / 2 + si(x) \* h / 6; // 3 рядок

}

if (i == j && i != 0 && i != n)

{

arr[i, j] = 2 \* (m(x) / h + si(x) \* h / 3); // 2 рядок

}

if ((i == 0 && j == 0) || (i == n && j == n))

{

arr[i, j] = m(x) / h + b(x) / 2 + si(x) \* h / 3; // 1 рядок

}

}

x += h;

}

arr[0, 0] \*= 1000000;

var invers = arr.Inverse(); //обернена матриця

// рахуємо функціонал

double[] l = new double[n+1]; //функціонал

for(int i=1;i<n;i++)

{

l[i] = f()\*h/2+f()\*h/2;

}

l[0] = f() \* h / 2;

l[n] = f() \* h / 2 + 0.5;

//// А \* U = L => U = inverse(A) \* L;

for (int i = 0; i < n+1; i++)

{

for (int j = 0; j < n+1; j++)

{

u[i] += l[j] \* invers[i, j];

}

}

return u;

}

static double[] fi(double x, double []xi,double h,double[] u)

{

//обчислює фі - функцію куранта

double[] phi = new double[xi.Length];

for (int i = 0; i < xi.Length; i++)

{

if (i == 0)

{

if (x > xi[i] && x <= xi[i + 1])

{

phi[i] = (xi[i] - x) / (h);

}

else

{

phi[i] = 0;

}

}

else if (i == xi.Length - 1)

{

if (x > xi[i - 1] && x <= xi[i])

{

phi[i] = (x - xi[i - 1]) / (h);

}

else

{

phi[i] = 0;

}

}

else

{

if (x > xi[i - 1] && x <= xi[i])

{

phi[i] = (x - xi[i - 1]) / (h);

}

else if (x > xi[i] && x <= xi[i + 1])

{

phi[i] = (xi[i + 1] - x) / (h);

}

else

{

phi[i] = 0;

}

}

}

for (int i=0;i<phi.Length;i++)

{

phi[i]= phi[i] \* u[i];

}

return phi;

}

static double fi2(double x, double[] xi, double h, double[] u)

{

// обчислює похідні функції куранта

// повертає суму базисних функцій

double[] phi = new double[xi.Length];

for (int i = 0; i < xi.Length; i++)

{

if (i == 0)

{

if (x > xi[i] && x <= xi[i + 1])

{

phi[i]= -1.0 / (xi[i + 1] - xi[i]);

}

else

{

phi[i] = 0;

}

}

else if (i == xi.Length-1)

{

if (x > xi[i - 1] && x <= xi[i])

{

phi[i] = 1.0 / (xi[i] - xi[i - 1]);

}

else

{

phi[i] = 0;

}

}

else

{

if (x > xi[i - 1] && x <= xi[i])

{

phi[i] = 1.0 / (xi[i] - xi[i - 1]);

}

else if (x > xi[i] && x <= xi[i + 1])

{

phi[i] = -1.0 / (xi[i + 1] - xi[i]);

}

else

{

phi[i] = 0;

}

}

}

double sum = 0;

for (int i = 0; i < xi.Length; i++)

{

sum += u[i] \* phi[i];

}

return sum;

}

private void DrawGraph(double[] x, double a, double b,double h,int n)

{

richTextBox1.Clear();

GraphPane pane = zedGraph.GraphPane;

pane.CurveList.Clear();

pane.XAxis.Scale.Min = a;

pane.XAxis.Scale.Max = b;

var uu = u(h, n);

PointPairList list = new PointPairList();

for(double xm=0;xm<=1;xm+=0.01)

{

var uh = fi(xm,x,h,uu);

double sum = 0;

for(int i=0;i<uh.Length;i++)

{

//сума то uh

sum += uh[i];

}

list.Add(xm,sum);

}

PointPairList list1 = new PointPairList();

for (double xm = 0.01; xm <= 1; xm += 0.01)

{

// uh ми вже знайшли в функції fi2

var uh = fi2(xm, x, h, uu);

list1.Add(xm, uh);

}

pane.YAxis.Scale.Min = 0;

pane.YAxis.Scale.Max = 10;

pane.AxisChange();

LineItem myCurve = pane.AddCurve("u", list, Color.Blue, SymbolType.None);

LineItem myCurve1 = pane.AddCurve("u'", list1, Color.Red, SymbolType.None);

zedGraph.AxisChange();

zedGraph.Invalidate();

}

private double L2(int n,double h)

{

double[] x = new double[n + 1];

x[0] = 0;

for(int i=1;i<n+1;i++)

{

x[i] = x[i - 1] + h;

}

double sum = 0;

var uu = u(h, n);

for(int i=1;i<n+1;i++)

{

sum += Math.Sqrt(Math.Abs((uu[i]) \* uu[i] \* ((x[i] \* x[i] \* x[i] / 3) - 2 \* (x[i] \* x[i] \* x[i]) - (x[i - 1] \* x[i - 1] \* x[i - 1]) / 3 + (x[i - 1] \* x[i - 1] \* x[i]) + (x[i - 1] \* x[i] \* x[i]))));

}

h = h / 2;

n = n \* 2;

double[] x1 = new double[n + 1];

x1[0] = 0;

for (int i = 1; i < n + 1; i++)

{

x1[i] = x1[i - 1] + h;

}

double sum1 = 0;

var uu1 = u(h, n);

for (int i = 1; i < n + 1; i++)

{

sum1 += Math.Sqrt(Math.Abs((uu1[i]) \* uu1[i] \* ((x1[i] \* x1[i] \* x1[i] / 3) - 2 \* (x1[i] \* x1[i] \* x1[i]) - (x1[i - 1] \* x1[i - 1] \* x1[i - 1]) / 3 + (x1[i - 1] \* x1[i - 1] \* x1[i]) + (x1[i - 1] \* x1[i] \* x1[i]))));

}

h = h / 2;

n = n \* 2;

double[] x2 = new double[n + 1];

x2[0] = 0;

for (int i = 1; i < n + 1; i++)

{

x2[i] = x2[i - 1] + h;

}

double sum2 = 0;

var uu2 = u(h, n);

for (int i = 1; i < n + 1; i++)

{

sum2 += Math.Sqrt(Math.Abs((uu2[i]) \* uu2[i] \* ((x2[i] \* x2[i] \* x2[i] / 3) - 2 \* (x2[i] \* x2[i] \* x2[i]) - (x2[i - 1] \* x2[i - 1] \* x2[i - 1]) / 3 + (x2[i - 1] \* x2[i - 1] \* x2[i]) + (x2[i - 1] \* x2[i] \* x2[i]))));

}

return Math.Abs(Math.Log(Math.Abs((sum - sum1) / (sum1 - sum2)), 2));

}

private double H1(int n, double h)

{

double[] x = new double[n + 1];

x[0] = 0;

for (int i = 1; i < n + 1; i++)

{

x[i] = x[i - 1] + h;

}

double sum = 0;

var uu = u(h, n);

for (int i = 1; i < n + 1; i++)

{

sum += Math.Sqrt(Math.Abs((uu[i]) \* uu[i] \* ((x[i] \* x[i] \* x[i] / 3) - 2 \* (x[i] \* x[i] \* x[i]) - (x[i - 1] \* x[i - 1] \* x[i - 1]) / 3 + (x[i - 1] \* x[i - 1] \* x[i]) + (x[i - 1] \* x[i] \* x[i]))+uu[i]/h));

}

h = h / 2;

n = n \* 2;

double[] x1 = new double[n + 1];

x1[0] = 0;

for (int i = 1; i < n + 1; i++)

{

x1[i] = x1[i - 1] + h;

}

double sum1 = 0;

var uu1 = u(h, n);

for (int i = 1; i < n + 1; i++)

{

sum1 += Math.Sqrt(Math.Abs((uu1[i]) \* uu1[i] \* ((x1[i] \* x1[i] \* x1[i] / 3) - 2 \* (x1[i] \* x1[i] \* x1[i]) - (x1[i - 1] \* x1[i - 1] \* x1[i - 1]) / 3 + (x1[i - 1] \* x1[i - 1] \* x1[i]) + (x1[i - 1] \* x1[i] \* x1[i]))+uu1[i] / h));

}

h = h / 2;

n = n \* 2;

double[] x2 = new double[n + 1];

x2[0] = 0;

for (int i = 1; i < n + 1; i++)

{

x2[i] = x2[i - 1] + h;

}

double sum2 = 0;

var uu2 = u(h, n);

for (int i = 1; i < n + 1; i++)

{

sum2 += Math.Sqrt(Math.Abs((uu2[i]) \* uu2[i] \* ((x2[i] \* x2[i] \* x2[i] / 3) - 2 \* (x2[i] \* x2[i] \* x2[i]) - (x2[i - 1] \* x2[i - 1] \* x2[i - 1]) / 3 + (x2[i - 1] \* x2[i - 1] \* x2[i]) + (x2[i - 1] \* x2[i] \* x2[i])) + uu2[i] / h));

}

return Math.Abs(Math.Log(Math.Abs((sum - sum1) / (sum1 - sum2)), 2));

}

private void button1\_Click(object sender, EventArgs e)

{

int a = 0;

int b = 1;

int n = Convert.ToInt32(textBox14.Text);

double h = ((double)b - (double)a) / (double)n;

double[] x = new double[n + 1];

double temp = a;

for (int i = 0; i < n + 1; i++)

{

// визначаємо точки хі

x[i] = temp;

temp += h;

}

DrawGraph(x, a, b, h, n);

var g = u(h, n);

for (int i=0;i<n+1;i++)

{

richTextBox1.Text += Convert.ToString(g[i])+"\n";

}

// порядки збіжності в Н1 і L2

label1.Text = L2(n, h).ToString();

label2.Text = H1(n, h).ToString();

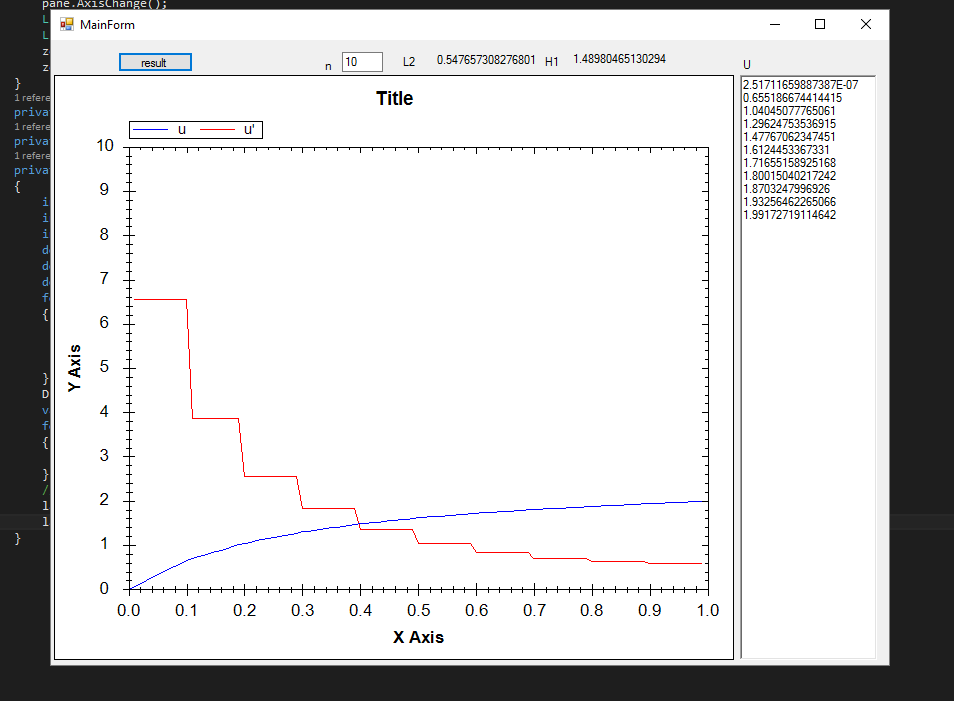
}

}

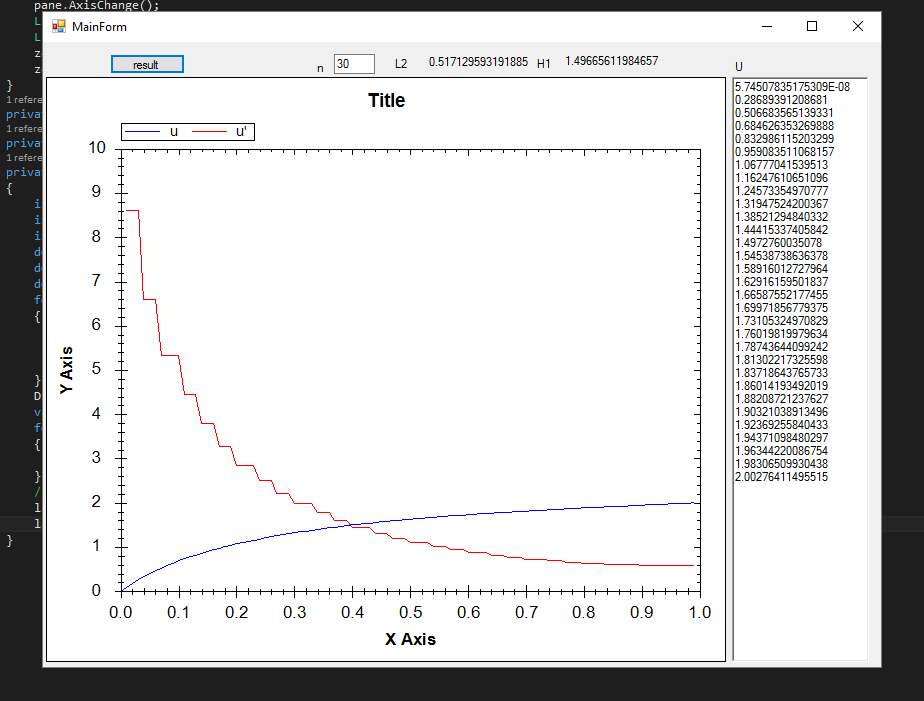
}

**Результат виконання програми:**

**При** n=10 (кількість розбиттів)



**При n=30**



**Червона лінія – функція u(x)**

**Синя лінія – похідна u(x)**

**Висновок:** Завдяки даному практичному завданню, я навчився знаходити розвязок варіаційної задачі методом скінченних елементів і оцінювати порядки збіжності цих розвязків в задежності від кроку розбиття.