DOI: 10.24143/2072-9502-2020-1-84-93
УДК 004:378.147+519.632.4

ИСПОЛЬЗОВАНИЕ СВОБОДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РЕШЕНИЯ ОДНОЙ ЗАДАЧИ МАТЕМАТИЧЕСКОЙ ФИЗИКИ

А. Е. Мартьянова

Астраханский государственный технический университет,
Астрахань, Российская Федерация

Рассматривается решение уравнения Лапласа с условиями Дирихле с помощью свободного программного обеспечения: систем компьютерной математики Maxima, Scilab, GNU Octave и языка программирования общего назначения Python. Алгоритм решения разностного уравнения Лапласа с условиями Дирихле, реализованный итеративным методом последовательной сверхрелаксации, позволяет получить решение в виде двумерного массива значений и 3D-графика. Полученное решение в виде двумерного массива сравнивается с тестовыми значениями. Установлено хорошее совпадение полученного массива с тестовыми значениями. Выбор пакета свободного программного обеспечения зависит как от вида поставленной задачи, так и от личных предпочтений.

Ключевые слова: свободное программное обеспечение, системы компьютерной математики, уравнение Лапласа, задача Дирихле, метод конечных разностей, язык программирования, граничные условия, метод последовательной сверхрелаксации.

Для цитирования: *Мартьянова А. Е.* Использование свободного программного обеспечения для решения одной задачи математической физики // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2020. № 1. С. 84–93. DOI: 10.24143/2072-9502-2020-1-84-93.

Введение

Современный образовательный процесс невозможен без применения математических пакетов. Поскольку коммерческое программное обеспечение (ПО) обычно имеет высокую стоимость, большой интерес представляет свободное ПО, распространяемое чаще всего по лицензии GNU GPL (англ. General Public License – универсальная общественная лицензия).

Для решения самых разных математических задач применяются различные системы компьютерной математики (СКМ). Также часто используется термин «системы компьютерной алгебры».

Системы компьютерной математики предназначены для выполнения преобразований и работы с математическими выражениями в численной и (иногда) аналитической (символьной) формах, включают в себя также язык программирования, позволяющий реализовывать алгоритмы, набор средств для проведения численных расчетов и построения графиков и пр.

Наиболее известными коммерческими СКМ являются Maple (Waterloo Maple Inc.), Mathematica (Wolfram Research), Mathcad (PTC). В научной и инженерной среде широко применяется MATLAB (The MathWorks), который представляет собой пакет прикладных программ для решения задач технических вычислений (прежде всего численных расчетов и работы с матрицами) и развитый высокоуровневый интерпретируемый язык программирования, используемый в этом пакете. Поддерживает пакетную обработку данных.

В образовательном процессе в качестве замены коммерческого ПО представляется возможным использовать следующее свободное или бесплатное ПО:

– СКМ Maxima (William Schelter), близкую по своим возможностям к коммерческим системам Maple и Mathematica; распространяется по лицензии GNU GPL;

– СКМ Scilab (ESI Group) – пакет прикладных математических программ, близкий по своим возможностям к коммерческому пакету MATLAB; распространяется по лицензии CeCILL (лицензия на свободное ПО, адаптированная к интернациональному законодательству и законодательству Франции, совместимая с GNU GPL v2);

– СКМ GNU Octave (John W. Eaton), близкую по своим возможностям к коммерческому пакету MATLAB; распространяется по лицензии GNU GPL.

Открытая кроссплатформенная система Maxima имеет несколько графических интерфейсов, чаще всего используется wxMaxima [1]. Maxima предназначена для работы с символьными

вычислениями, может работать и с численными методами. Язык Maxima интерпретируемый. Пакет имеет длительную историю развития (с 1968 г.), широко применяется в практике.

Scilab – открытая кроссплатформенная система, имеющая различные инструменты и пакеты расширений, а также схожий с MATLAB язык программирования [2].

GNU Octave – открытая кроссплатформенная система, близкая к MATLAB [3]. Язык GNU Octave написан с учетом совместимости с языком MATLAB.

Наряду с использованием СКМ для решения математических задач в образовательном процессе может быть использован и язык программирования, такой как Python (Guido van Rossum), – высокоуровневый язык общего назначения, который распространяется по лицензии PSFL, совместимой с GNU GPL [4]. Открытость, кроссплатформенность, лаконичность, активное развитие и использование языка Python позволяют рассматривать его как язык, подходящий для обучения программированию и решения различных вычислительных задач.

В настоящей статье рассматриваются особенности решения двумерной задачи Дирихле с использованием различного свободного ПО, а именно: СКМ Maxima, Scilab, GNU Octave и языка программирования общего назначения Python.

Формулировка задачи, алгоритм решения разностного уравнения Лапласа и таблица тестовых значений приняты по источнику [5].

Решение задачи Дирихле с помощью свободного ПО

Приближенное решение первой краевой задачи для уравнения Лапласа

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad (1)$$

осуществляется итеративным методом последовательной сверхрелаксации (ПСР) в виде распределения $u(x, y)$ в области, ограниченной прямоугольником $R = \{(x, y)\colon 0 \le x \le a,\ 0 \le y \le b$, где $b/a = m/n\}$. Прямоугольник R на плоскости разделен решеткой на $(n - 1)(m - 1)$ квадратов со стороной, равной h (т. е. $a = n \cdot h$ и $b = m \cdot h$).

Разностное уравнение Лапласа в методе ПСР для $2 \le i \le n-1$ и $2 \le j \le m-1$ представляется в виде итерационной формулы [5]

$$u_{i,j} = u_{i,j} + wr_{i,j} = u_{i,j} + w\left(\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{4}\right), \qquad (2)$$

где w – параметр релаксации – находится в области $1 \le w < 2$ [6]; $r_{i,j}$ – остаточный член в уравнении (2).

Граничные значения $u(x, y)$ известны в следующих точках решетки:

$$\begin{aligned} &u(x_1, y_j) = u_{1,j} \text{ для } 2 \le j \le m-1 \text{ (слева);} \\ &u(x_i, y_1) = u_{i,1} \text{ для } 2 \le i \le n-1 \text{ (внизу);} \\ &u(x_n, y_j) = u_{n,j} \text{ для } 2 \le j \le m-1 \text{ (справа);} \\ &u(x_i, y_m) = u_{i,m} \text{ для } 2 \le i \le n-1 \text{ (вверху).} \end{aligned} \qquad (3)$$

Начальные значения во всех внутренних точках решетки должны быть вычислены предварительно. Для этого можно использовать постоянную K, которая равна среднему арифметическому $2n + 2m - 4$ граничных значений, заданных в (3).

Для иллюстрации область $R = \{(x, y)\colon 0 \le x \le a,\ 0 \le y \le a\}$ принята в виде квадрата с граничными условиями (ГУ)

$$\begin{aligned} &u(x, 0) = 20 \text{ и } u(x, a) = 180 \text{ для } 0 < x < a; \\ &u(0, y) = 80 \text{ и } u(a, y) = 0 \text{ для } 0 < y < a \end{aligned} \qquad (4)$$

и разбита на 64 равных квадрата. Начальное значение во внутренних точках решетки $u_{i,j} = 70$ для каждого $i = 2, \ldots, 8$ и $j = 2, \ldots, 8$. Параметр ПСР для $n = 9$ и $m = 9$ вычислен [5, 6] как

$$w = 4 / \left(2 + \sqrt{4 - \left(\cos\left(\frac{\pi}{n-1} \right) + \cos\left(\frac{\pi}{m-1} \right) \right)^2} \right) = 1,44646.$$

Поскольку граничная функция в углах разрывна, граничные значения $u_{1,1} = \frac{u_{1,2} + u_{2,1}}{2} = 50$, $u_{9,1} = \frac{u_{8,1} + u_{9,2}}{2} = 10$, $u_{1,9} = \frac{u_{1,8} + u_{2,9}}{2} = 130$ и $u_{9,9} = \frac{u_{8,9} + u_{9,8}}{2} = 90$ не используются в вычислениях внутренних точек решетки.

На рис. 1 представлены фрагмент кода приближенного решения уравнения (1) с ГУ (4) в окне СКМ Maxima и график поверхности, построенный там же по результатам этого решения.

```
--> for j:1 thru m do (
    u[1,j]:80.0, u[n,j]:0.0
    );

--> u[1,1]:(u[1,2]+u[2,1])/2;

--> u[1,m]:(u[1,m-1]+u[2,m])/2;

--> u[n,1]:(u[n-1,1]+u[n,2])/2;

--> u[n,m]:(u[n-1,m]+u[n,m-1])/2;

--> /* Вычисление параметра метода последовательной релаксации */
    w:4/(2+sqrt(4-(cos(%pi/(n-1))+cos(%pi/(m-1)))^2));

--> /* Организация итеративного метода последовательной релаксации */
    err:1; cnt:0;

--> while (err>tol and cnt<=max1) do (
    err:0,
    for j:2 thru m-1 do (
    for i:2 thru n-1 do (
    (relx:float(w*((u[i,j+1]+u[i,j-1]+u[i+1,j]+u[i-1,j])-4*u[i,j])/4),r
    u[i,j]:u[i,j]+relx,
    if err<=abs(relx) then
    err:abs(relx)
```

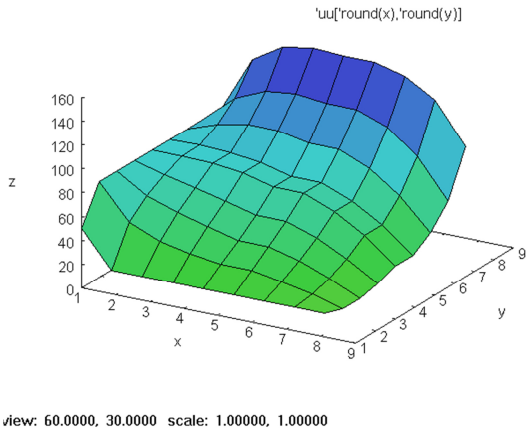

Рис. 1. Решение задачи в пакете Maxima

Ниже приводится весь программный код приближенного решения по методу ПСР уравнения (1) с ГУ (4) в СКМ Maxima:

```
- - > kill(all); fpprintprec:6;
- - > tol:0.001; max1:1000;
- - > n:9; m:9; k:70.0;
- - > /* Инициализация массива u */
        for j:1 thru m do (
        for i:1 thru n do (
        arraymake(u, [i,j]), u[i,j]:k
        ));
- - > /* Задание граничных значений */
        for i:1 thru n do (
        u[i,1]:20.0, u[i,m]:180.0
        );
- - > for j:1 thru m do (
        u[1,j]:80.0, u[n,j]:0.0
        );
- - > u[1,1]:(u[1,2]+u[2,1])/2;
- - > u[1,m]:(u[1,m-1]+u[2,m])/2;
```

```
- - > u[n,1]:(u[n-1,1]+u[n,2])/2;
- - > u[n,m]:(u[n-1,m]+u[n,m-1])/2;
- - > /* Вычисление параметра метода последовательной релаксации */
        w:4/(2+sqrt(4-(cos(%pi/(n-1))+cos(%pi/(m-1)))^2));
- - > /* Организация итеративного метода последовательной релаксации */
- - > err:1; cnt:0;
        while (err>tol and cnt<=max1) do (
        err:0,
        for j:2 thru m-1 do (
        for i:2 thru n-1 do (
        (relx:float(w*((u[i,j+1]+u[i,j-1]+u[i+1,j]+u[i-1,j])-4*u[i,j])/4),numer),
        u[i,j]:u[i,j]+relx,
        if err<=abs(relx) then
        err:abs(relx)
        )),
        cnt:cnt+1
        );
- - > /* Формирование двумерного массива */
        uu:genmatrix(lambda([i,j], u[i,j]), n, m);
- - > /* Преобразование массива в функцию двух переменных */
        f(x,y):=float('uu[round(x), round(y)]);
- - > /* Вывод 3D-графика */
plot3d(f(x,y), [x,1,n], [y,1,m], [grid,n,m]);
```

Здесь, в программе, и далее, в следующих программах, приняты обозначения: m, n – число точек сетки на сторонах прямоугольной области R с ГУ (4); *tol* – допустимое отклонение; *max*1 – максимальное число итераций; k – постоянная, равная среднему арифметическому $2n + 2m - 4$ граничных значений, заданных в (3); w – параметр релаксации в итерационной формуле (2); *relx* – остаточный член в формуле (2).

На рис. 2 представлены фрагмент кода решения уравнения (1) с ГУ (4) в окне текстового редактора SciNotes и график поверхности, построенный после запуска в консоли СКМ Scilab функции [*u*] = laplas(*n*, *m*, *tol*, *max*1).

```
end
u(1,1)=(u(1,2)+u(2,1))/2;
u(1,m)=(u(1,m-1)+u(2,m))/2;
u(n,1)=(u(n-1,1)+u(n,2))/2;
u(n,m)=(u(n-1,m)+u(n,m-1))/2;
// Вычисление параметра метода последовательной релаксации
w=4/(2+sqrt(4-(cos(%pi/(n-1))+cos(%pi/(m-1)))^2));
// Организация итеративного метода последовательной релаксации
err=1;
cnt=0;
while((err>tol)&(cnt<=max1))
    err=0;
    for j=2:m-1
        for i=2:n-1
            relx=w*(u(i,j+1)+u(i,j-1)+u(i+1,j)+u(i-1,j)-4*u(i
,j))/4;
            u(i,j)=u(i,j)+relx;
            if (err<=abs(relx))
                err=abs(relx);
            end
        end
    end
    cnt=cnt+1;
end
endfunction
```

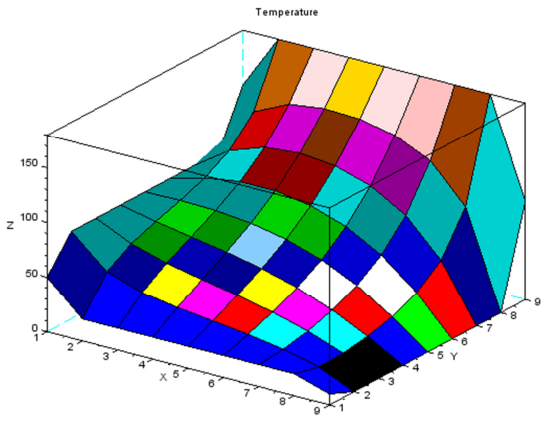

Рис. 2. Решение задачи в пакете Scilab

Для вывода массива *u* и построения графика поверхности в консоли СКМ Scilab последовательно были запущены следующие команды (комментирование может быть опущено):

```
- - >  // Вызов функции
- - >  [u]=laplas(9,9,0.001,1000);
- - >  // Вывод массива u
- - >  full(u)
- - >  // Построение графика функции
- - >  surf(u);
title('Temperature');
```

Ниже приводится код приближенного решения уравнения (1) с ГУ (4) по методу ПСР, написанный в СКМ Scilab:

```
function [u]=laplas(n, m, tol, max1)
// Инициализация массива u k=70.0;
u=k*ones(n,m);
// Задание граничных значений
for i=1:n
   u(i,1)=80.0;
   u(i,m)=0.0;
end
for j=1:m
   u(1,j)=20.0;
   u(n,j)=180.0;
end
u(1,1)=(u(1,2)+u(2,1))/2;
u(1,m)=(u(1,m-1)+u(2,m))/2;
u(n,1)=(u(n-1,1)+u(n,2))/2;
u(n,m)=(u(n-1,m)+u(n,m-1))/2;
// Вычисление параметра метода последовательной релаксации
w=4/(2+sqrt(4-(cos(%pi/(n-1))+cos(%pi/(m-1)))^2));
// Организация итеративного метода последовательной релаксации
err=1;
cnt=0;
while((err>tol)&(cnt<=max1))
   err=0;
   for j=2:m-1
      for i=2:n-1
         relx=w*(u(i,j+1)+u(i,j-1)+u(i+1,j)+u(i-1,j)-4*u(i,j))/4;
         u(i,j)=u(i,j)+relx;
         if (err<=abs(relx))
            err=abs(relx);
         end
      end
   end
   cnt=cnt+1;
end
       endfunction
```

Программа сохранена на диске в виде файла сценария, который потом был запущен на исполнение.

На рис. 3 представлены фрагмент текста m-файла GNU Octave и график поверхности, построенный после запуска *m*-файла.

```
end;
u(1,1)=(u(1,2)+u(2,1))/2;
u(1,m)=(u(1,m-1)+u(2,m))/2;
u(n,1)=(u(n-1,1)+u(n,2))/2;
u(n,m)=(u(n-1,m)+u(n,m-1))/2;
# Вычисление параметра метода последовательной релаксации
w=4/(2+sqrt(4-(cos(pi/(n-1))+cos(pi/(m-1)))^2));
# Организация итеративного метода последовательной релаксации
err=1;
cnt=0;
while((err>tol)&(cnt<=max1))
    err=0;
    for j=2:m-1
        for i=2:n-1
            relx=w*(u(i,j+1)+u(i,j-1)+u(i+1,j)+u(i-1,j)-4*u(i,j
            u(i,j)=u(i,j)+relx;
            if (err<=abs(relx))
                err=abs(relx);
            end
        end
```

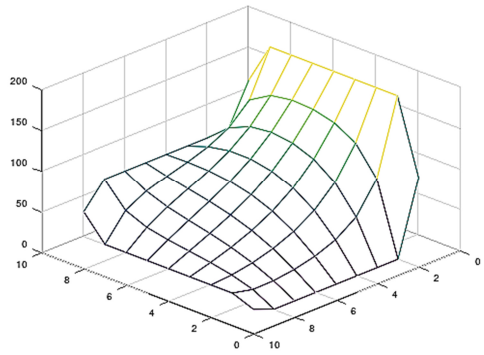

Рис. 3. Решение задачи в пакете GNU Octave

Текст программы близок к таковому же для СКМ Scilab, т. к. изначально язык этих пакетов разрабатывался с ориентацией на MATLAB.

На рис. 4 представлены фрагмент программного кода для решения уравнения (1) с ГУ (4) по методу ПСР и реализации варианта построения графика поверхности по результатам вычисления в среде IDLE Python.

```
for i in range(n):
    u[i][0] = 80.000
    u[i][m-1] = 0.000

for j in range(m):
    u[0][j] = 20.000
    u[n-1][j] = 180.000

u[0][0] = (u[0][1] + u[1][0])/2
u[0][m-1] = (u[0][m-2] + u[1][m-1])/2
u[n-1][0] = (u[n-2][0] + u[n-1][1])/2
u[n-1][m-1] = (u[n-2][m-1] + u[n-1][m-2])/2

# Вычисление параметра метода последовательной релаксации
w =(4/(2 + math.sqrt(4 - pow((math.cos(math.pi/(n - 1)) + math.cos(math.pi

# Организация итеративного метода последовательной релаксации
err = 1
cnt = 0
while((err>tol) and (cnt<=max1)):
    err = 0
    for j in range(1, m - 1):
        for i in range(1, n - 1):
            relx = w * (u[i][j+1] + u[i][j-1] + u[i+1][j] + u[i-1][j] - 4
            u[i][j] = u[i][j] + relx
            if (err <= abs(relx)):
                err = abs(relx)
    cnt = cnt + 1
```

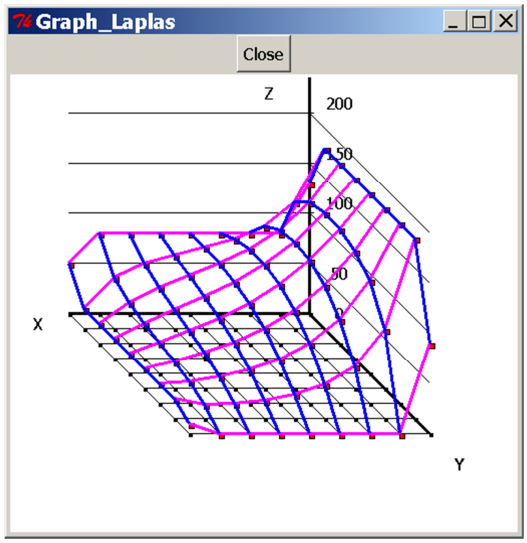

Рис. 4. Решение задачи в среде IDLE Python

Для отрисовки графика привлекается библиотека tkinter в среде IDLE Python. Библиотека tkinter предназначается для создания кроссплатформенных программ с графическим интерфейсом. Библиотека tkinter поставляется вместе с Python, и сама среда IDLE Python была создана с помощью этой библиотеки.

Следует отметить, что при построении графиков гораздо больше возможностей предоставляет библиотека matplotlib, которую необходимо устанавливать дополнительно, что не всегда возможно.

Код на языке Python для приближенного решения уравнения (1) с ГУ (4) по методу ПСР и построения соответствующего графика поверхности приведен ниже:

```
import tkinter
import math

n = 9
m = 9
tol = 0.001
```

```
max1 = 1000
k = 70.0
# Инициализация массива u
u = [[k for j in range(m)] for i in range(n)]
# Задание граничных значений
for i in range(n):
   u[i][0] = 80.000
   u[i][m-1] = 0.000
for j in range(m):
   u[0][j] = 20.000
   u[n-1][j] = 180.000
u[0][0] = (u[0][1] + u[1][0])/2
u[0][m-1] = (u[0][m-2] + u[1][m-1])/2
u[n-1][0] = (u[n-2][0] + u[n-1][1])/2
u[n-1][m-1] = (u[n-2][m-1] + u[n-1][m-2])/2
# Вычисление параметра метода последовательной релаксации
w =(4/(2 + math.sqrt(4 - pow((math.cos(math.pi/(n - 1)) + math.cos(math.pi/(m - 1))),2))))
# Организация итеративного метода последовательной релаксации
err = 1
cnt = 0
while((err>tol) and (cnt<=max1)):
   err = 0
   for j in range(1, m - 1):
      for i in range(1, n - 1):
         relx = w * (u[i][j+1] + u[i][j-1] + u[i+1][j] + u[i-1][j] - 4 * u[i][j])/4
         u[i][j] = u[i][j] + relx
         if (err <= abs(relx)):
            err = abs(relx)
   cnt = cnt + 1
# Вывод двумерного массива u
for j in range(m):
   for i in range(n):
      print("%f" %u[i][j], end = "  ")
   print()
print()

# Создание экземпляра интерфейсного объекта Tk, окно
tk=tkinter.Tk()
tk.title("Graph_Laplas")
# Заполнение окна интерфейсными объектами, кнопка и холст
button=tkinter.Button(tk)
button["text"]="Close"
button["command"]=tk.quit
button.pack()
canvas=tkinter.Canvas(tk)
canvas["height"]=450
canvas["width"]=500
canvas["background"]="white"
canvas["borderwidth"]=3
canvas.pack()
# Рисование координатных осей X, Y, Z
canvas.create_text(30, 250, text="X")
canvas.create_text(450, 390, text="Y")
canvas.create_text(260, 20, text="Z")
x0=300
x1=60
dx=(x0-x1)/(n-1)
```

```
ax=240
y0=240
y1=480
dy=dx*0.5
ay=240*0.5
az=ax
x_axe=[]
xx=(x0, y0)
x_axe.append(xx)
xx=(x0-ax, y0)
x_axe.append(xx)
canvas.create_line(x_axe, fill="black", width=3)
z_axe=[]
zz=(x0, y0)
z_axe.append(zz)
zz=(x0, y0-az)
z_axe.append(zz)
canvas.create_line(z_axe, fill="black", width=3)
y_axe=[]
yy=(x0, y0)
y_axe.append(yy)
yy=(x0+ay, y0+ay)
y_axe.append(yy)
canvas.create_line(y_axe, fill="black", width=3)
canvas.create_text(330, 240, text="0")
canvas.create_text(330, 200, text="50")
canvas.create_line(x0-5, 190, x0+5, 190, fill="black", smooth=0)
canvas.create_line(x1, 190, x0, 190, fill="black", smooth=0)
canvas.create_line(x0, 190, x0+ay, 190+ay, fill="black", smooth=0)
canvas.create_text(330, 130, text="100")
canvas.create_line(x0-5, 140, x0+5, 140, fill="black", smooth=0)
canvas.create_line(x1, 140, x0, 140, fill="black", smooth=0)
canvas.create_line(x0, 140, x0+ay, 140+ay, fill="black", smooth=0)
canvas.create_text(330, 80, text="150")
canvas.create_line(x0-5, 90, x0+5, 90, fill="black", smooth=0)
canvas.create_line(x1, 90, x0, 90, fill="black", smooth=0)
canvas.create_line(x0, 90, x0+ay, 90+ay, fill="black", smooth=0)
canvas.create_text(330, 30, text="200")
canvas.create_line(x0-5, 40, x0+5, 40, fill="black", smooth=0)
canvas.create_line(x1, 40, x0, 40, fill="black", smooth=0)
canvas.create_line(x0, 40, x0+ay, 40+ay, fill="black", smooth=0)
# Рисование 3D-графика
for j in range(m):
    for i in range(n):
        z_=(u[i][j])
        # Рисование точек разностной сетки на X0Y-плоскости
        canvas.create_rectangle(x1+i*dx+j*dy, y0+j*dy, x1+3+i*dx+j*dy, y0+3+j*dy, fill="black")
        # Рисование точек разностной сетки поверхности u(x,y)
        canvas.create_rectangle(x1+i*dx+j*dy, az-z_+j*dy, x1+5+i*dx+j*dy, az-z_+5+j*dy, fill="red")
        # Рисование линий разностной сетки на X0Y-плоскости
        canvas.create_line(x1+i*dx+0*dy, y0+0*dy, x1+i*dx+(m-1)*dy, y0+(m-1)*dy, \
                           fill="black", smooth=0)
        canvas.create_line(x1+0*dx+j*dy, y0+j*dy, x1+(n-1)*dx+j*dy, y0+j*dy, fill="black", smooth=0)
# Рисование точек линий разностной сетки на поверхности u(x,y)
for j in range(m):
    for i in range(n-1):
        z_=(u[i][j])
```

```
    z__=(u[i+1][j])
    canvas.create_line(x1+i*dx+j*dy, az-z_+j*dy, x1+(i+1)*dx+j*dy, az-z__+j*dy, \
                fill="magenta", width=3, smooth=0)
for j in range(m-1):
  for i in range(n):
    z_=(u[i][j])
    z___=(u[i][j+1])
    canvas.create_line(x1+i*dx+j*dy, az-z_+j*dy, x1+i*dx+(j+1)*dy, az-z___+(j+1)*dy, \
                fill="blue", width=3, smooth=0)

    tk.mainloop()
```

Таблица тестовых значений функции $u(x, y)$ в узловых точках сетки разностной схемы приводится в источнике [5] с 6-ю значащими цифрами. Сравнение расчетных результатов, полученных здесь в этой работе, с тестовыми значениями по источнику [5] показало хорошее совпадение: 2-3 знака после запятой, поскольку допустимое отклонение *tol* в тестовых расчетах тоже было задано как 0,001. Результаты вычислений в IDLE Python показали, что после 19 итераций остаточный член $r_{i,j}$ в уравнении (2) становится меньше *tol* ($r_{i,j}$ = 0,00060984 < 0,001).

Как видно, код, реализованный в рассматриваемых здесь пакетах свободного ПО, близок в части решения уравнения (1) с ГУ (4) и требует примерно равных затрат по времени на его реализацию. Что касается построения графика поверхности, то в то время, как в среде IDLE Python необходимо написание кода, формирующего график по результатам решения уравнения с помощью той или иной библиотеки (см. здесь также [7]), в СКМ Maxima, Scilab, GNU Octave реализованы команды, позволяющие довольно просто и в приемлемом виде сформировать 3D-графики с подключением свободной программы gnuplot.

С другой стороны, при выборе СКМ в научной и инженерной среде чаще используется СКМ Scilab, поскольку этот пакет имеет больше возможностей, чем пакет GNU Octave, который, в свою очередь, представляет скорее среду программирования. С точки зрения удобства работы в них, пакеты Scilab и GNU Octave примерно одинаковы и уступают коммерческому ПО. Система компьютерной математики Maxima изначально была ориентирована на работу с математическими выражениями в аналитической (символьной) форме, и в этом ее дополнительное преимущество. Кроме того, в ней также имеется возможность реализации вычислительных методов. Пакет хорошо документирован на русском языке [8]. Графические возможности интерфейса wxMaxima делают пакет достаточно удобным и дружественным для использования в образовательном процессе.

Заключение

Таким образом, в настоящей работе рассмотрено решение задачи Дирихле при помощи свободного ПО, а именно: СКМ Maxima, Scilab, GNU Octave и языка программирования общего назначения Python. Полученные результаты расчета сравнивались с тестовыми результатами. Установлено хорошее соответствие расчетных значений тестовым. Показано, что для образовательного процесса в учебных и научных целях возможно и целесообразно использование свободного ПО в качестве альтернативы коммерческому ПО, причем выбор пакета ПО зависит как от вида поставленной задачи, так и от личных предпочтений.

СПИСОК ЛИТЕРАТУРЫ

1. *Maxima* Documentation. URL: http://maxima.sourceforge.net/documentation.html (дата обращения: 01.10.2019).
2. *Scilab*. URL: https://www.scilab.org/tutorials (дата обращения: 01.10.2019).
3. *GNU* Octave (version 5.1.0). URL: https://octave.org/doc/interpreter/ (дата обращения: 01.10.2019).
4. *Python*^{TM}. URL: https://www.python.org/doc/ (дата обращения: 01.10.2019).
5. *Мэтьюз Дж. Г., Финк К. Д.* Численные методы. Использование MATLAB. М.: Вильямс, 2001. 720 с.
6. *Вержбицкий В. М.* Основы численных методов. М.: Высш. шк., 2002. 840 с.
7. *Хахаев И. А.* Практикум по алгоритмизации и программированию на Python. М.: Альт Линукс, 2010. 126 с.
8. *Чичкарев Е. А.* Компьютерная математика с Maxima: рук. для школьников и студентов. М.: ALT Linux, 2012. 384 с.

Статья поступила в редакцию 21.11.2019

ИНФОРМАЦИЯ ОБ АВТОРЕ

Мартьянова Александра Евгеньевна – Россия, 414056, Астрахань; Астраханский государственный технический университет; канд. техн. наук, доцент; доцент кафедры физики; mrtnva@rambler.ru.

USING FREE SOFTWEAR FOR SOLVING ONE PROBLEM OF MATHEMATICAL PHYSICS

A. E. Martianova

Astrakhan State Technical University,
Astrakhan, Russian Federation

Abstract. The article presents the solution of Laplace equation with Dirichlet conditions using free software: computer mathematics systems Maxima, Scilab, GNU Octave and general-purpose programming language Python. The algorithm for solving Laplace difference equation with Dirichlet conditions is realized by using the iterative method of successive over-relaxation and helps to obtain the solution in the form of a two-dimensional array of values and 3D-graphs. The resulting solution in the form of a two-dimensional array is compared with the test values. The resulting array was found to match the test values. The choice of a free software depends on the type of task and on personal preferences.

Key words: free software, computer mathematics systems, Laplace equation, Dirichlet problem, finite-difference method, programming language, boundary conditions, method of successive over-relaxation.

For citation: Martianova A. E. Using free software for solving one problem of mathematical physics. *Vestnik of Astrakhan State Technical University. Series: Management, Computer Science and Informatics*. 2020;1:84-93. (In Russ.) DOI: 10.24143/2072-9502-2020-1-84-93.

REFERENCES

1. *Maxima Documentation*. Available at: http://maxima.sourceforge.net/documentation.html (accessed: 01.10.2019).
2. *Scilab.* Available at: https://www.scilab.org/tutorials (accessed: 01.10.2019).
3. *GNU Octave (version 5.1.0).* Available at: https://octave.org/doc/interpreter/ (accessed: 01.10.2019).
4. *Python*^{TM}. Available at: https://www.python.org/doc/ (accessed: 01.10.2019).
5. Mathews J. H., Fink K. D. *Numerical Methods. Using MATLAB*. NJ, Prentice Hall, 1999. 655 p. (Rus. ed.: Met'iuz Dzh. G., Fink K. D. Chislennye metody. Ispol'zovanie MATLAB. M.: Vil'iams, 2001. 720 s.).
6. Verzhbitskii V. M. *Osnovy chislennykh metodov* [Principles of numerical methods]. Moscow, Vysshaia shkola Publ., 2002. 840 p.
7. Khakhaev I. A. *Praktikum po algoritmizatsii i programmirovaniiu na Python* [Python algorithmization and programming workshop]. Moscow, Al't Linuks Publ., 2010. 126 p.
8. Chichkarev E. A. *Komp'iuternaia matematika s Maxima: rukovodstvo dlia shkol'nikov i studentov* [Computer mathematics with Maxima: tutorial for pupils and students]. Moscow, ALT Linux Publ., 2012. 384 p.

The article submitted to the editors 21.11.2019

INFORMATION ABOUT THE AUTHOR

Martianova Aleksandra Evgenevna – Russia, 414056, Astrakhan; Astrakhan State Technical University; Candidate of Technical Sciences, Assistant Professor; Assistant Professor of the Department of Physics; mrtnva@rambler.ru.