## Национальный исследовательский университет «МЭИ»

## (Московский Энергетический Институт)

**Кафедра математического и компьютерного моделирования**

## Численные методы

## Лабораторная работа №8:

## «ЧИСЛЕННОЕ РЕШЕНИЕ СТАЦИОНАРНЫХ ЗАДАЧ ТЕПЛОПРОВОДНОСТИ»

*Выполнил:* Солонин Е. В. А-14-19

*Преподаватель:* Амосова О. А.

Вариант 41

2021 Москва

**Постановка задачи:**

**Задача 8.1.** Найти с точностью  приближенное решение краевой задачи

 где 

### ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ

1. Построить тестовый пример, используя коэффициенты k(x) и q(x), указанные в индивидуальном варианте.
2. Cоставить разностную схему методом баланса. Выписать коэффициенты матрицы системы уравнений и коэффициенты правой части (интегралы, требуемые при построении разностной схемы, вычислить аналитически.)
3. Решение системы разностных уравнений найти прямым методом с помощью встроенной процедуры.
4. Построить на одном чертеже графики приближенного и точного решений для тестового примера и график погрешности. После проверки правильности работы программы перейти к решению основной задачи.
5. Для вычисления решения задачи с заданной точностью произвести расчет с начальным шагом  затем уменьшить шаг вдвое. Проверку достижения заданной точности производить в программе по правилу Рунге.
6. Построить графики найденного решения и графики погрешностей.
7. Используя программу, составленную в задаче, выполнить вычислительные эксперименты: найти решение задачи при следующих вариантах измененной правой части уравнения *F(x*) (*f(x)* - исходная функция правой части ):

A) ; B) 

C) F(x) – точечный источник тепла. Задать точечный источник можно следующим образом: F(x)=  где c – некоторая константа (мощность источника), - дельта функция , -точка из отрезка [a,b] в которую ставится источник. Рассмотреть различные расположения точки c на отрезке [a,b].

1. Построить на одном чертеже графики полученных приближенных решений.
2. Определить при каком варианте из A)-C) приближенное решение задачи достигает максимального и минимального значений.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| 8.1.41 | 1 | 4 |  |  |  | 8 | 6 |

**Непонятно, почему [a,b]=[1,4], когда k(x) определена на [0,3]. Беру [a,b]=[0,3]**

**Теоретический материал:**

Метод баланса: *i=1…N-1*

, .

Здесь , , 

**Тестовый пример:**

Соблюдение граничных условий:

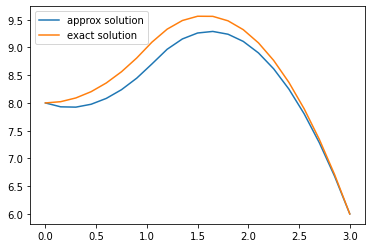
Согласование условий:

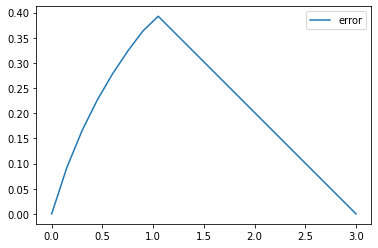
Возьмем

Возьмем уравнение

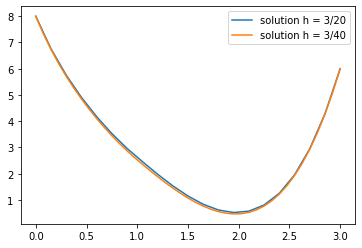
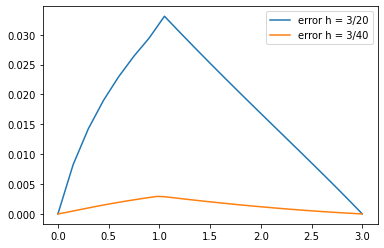
*Задача:* найти приближенное решение к функции u(x), являющейся решением задачи

*Будем работать с шагом h=(b-a)/20*

*****Графики точного и приближенного решения:*

*График погрешности:*

**Результаты:**

Исходная задача

Точность на шаге h=3/20 не достигнута, однако на шаге h=3/40 решение найдено с заданной точностью

**Код программы** *(пояснения к коду красным цветом)***:**

**import** numpy  
**import** matplotlib.pyplot **as** plt  
  
*# задача Коши***def** dy(y, t):  
 u, v = y  
 du = t \* (u \*\* 2) \* numpy.sqrt(v)  
 dv = numpy.sqrt(u) + numpy.sqrt(v)  
 **return** [du, dv]  
  
*# формула метода Эйлера***def** Euler(y, t, h):  
 u, v = y  
 y0, y1 = dy(y, t)  
 euler1, euler2 = u + h \* y0, v + h \* y1  
 **return** [euler1, euler2]  
  
*# формула усовершенствованного метода Эйлера***def** Euler\_modified(y, t, h):  
 u, v = y  
 y0, y1 = dy(y, t)  
 y\_support0, y\_support1 = dy([u + h / 2 \* y0, v + h / 2 \* y1], t + h / 2)  
 eulerMod1, eulerMod2 = u + h \* y\_support0, v + h \* y\_support1  
 **return** [eulerMod1, eulerMod2]  
  
*# расчет числа точек N для шага h*get\_n = **lambda** h: int(numpy.ceil((t\_n - t\_0) / h))  
  
*# получить массив решений выбранным методом с шагом h для двух компонент u и v***def** get\_y\_data(h, Method):  
 n = get\_n(h)  
 t\_data = numpy.linspace(t\_0, t\_n, n + 1)  
 y\_data = [[1, 1]]  
 **for** i **in** range(1, len(t\_data)):  
 y\_data.append(Method(y\_data[i - 1], t\_data[i - 1], h))  
 y\_data = numpy.array(y\_data)  
 u, v = y\_data[:, 0], y\_data[:, 1]  
 **return** [u, v]  
  
*# максимальное значение погрешности по правилу Рунге***def** Runge(y\_data\_h, y\_data\_2h):  
 r\_data = [0]  
 **for** i **in** range(1, len(y\_data\_2h)):  
 r\_data.append(numpy.abs((y\_data\_h[2 \* i] - y\_data\_2h[i])))  
 **return** max(r\_data)  
  
*# функция, осуществляющая: поиск решения выбранным методом;  
# оценку погрешности по правилу Рунге***def** solve(Method, methodName):  
 u\_h, v\_h = get\_y\_data(h, Method)  
 u\_2h, v\_2h = get\_y\_data(2 \* h, Method)  
  
 max\_r\_u = Runge(u\_h, u\_2h)  
 max\_r\_v = Runge(v\_h, v\_2h)  
  
 print(**"\n"**, methodName, **":"**)  
 print(**"Max Runge error for u(t): "**, max\_r\_u)  
 print(**"Max Runge error for v(t): "**, max\_r\_v)  
 **return** [u\_h, v\_h]  
  
*# построение графиков для двух методов одной компоненты решения***def** show\_graph(data\_euler, data\_eulerMod, title):  
 n = get\_n(h)  
 t\_data = numpy.linspace(t\_0, t\_n, n + 1)  
 fig, ax = plt.subplots()  
 plt.plot(t\_data, data\_euler, label=**"Euler"**)  
 plt.plot(t\_data, data\_eulerMod, label=**"Euler modified"**)  
 ax.set\_title(title)  
 plt.legend()  
 plt.show()  
  
t\_0 = 0  
t\_n = 1  
h = 0.01  
  
u\_euler, v\_euler = solve(Euler, **"Euler"**)  
u\_eulerMod, v\_eulerMod = solve(Euler\_modified, **"Euler modified"**)  
  
show\_graph(u\_euler, u\_eulerMod, **"u(t)"**)  
show\_graph(v\_euler, v\_eulerMod, **"v(t)"**)

**Результаты выполнения:**

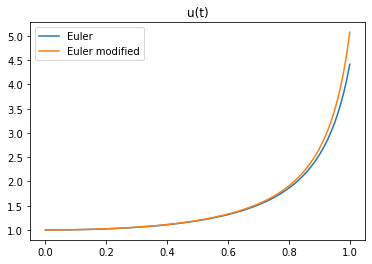
Euler :

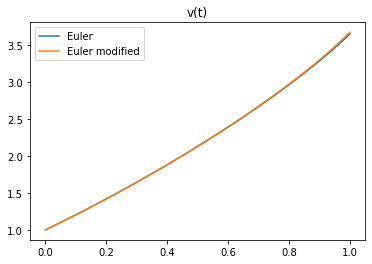
Max Runge error for u(t): 0.454982637144246

Max Runge error for v(t): 0.021519478844226114

Euler modified :

Max Runge error for u(t): 0.061646522345664145

****Max Runge error for v(t): 0.0012717212391528676

****

**Анализ результатов:**

Оба метода дают сносные результаты. По оценкам погрешности по Рунге, можно сделать вывод, что модифицированный метод справляется со своей задачей лучше, чем метод Эйлера.

**Постановка задачи:**

**Задача 7.2.** Дана задача Коши для двух систем обыкновенных дифференциальных уравнений с постоянными коэффициентами

 ,

 ,

где  и  – заданные матрицы,  - заданные векторы. Исследовать поведение решения систем уравнений

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Используя встроенную функцию пакета NUMPY для нахождения собственных чисел матриц *A* и *B*, найти коэффициенты жесткости обеих систем. Установить какая задача является жесткой.
2. Численно решить обе задачи на отрезке [0,1] c шагом *h*=0.01 явным методом Эйлера. Определить, для какой из задач явный метод неустойчив при данном шаге *h*. Построить графики компонент полученного решения.
3. Численно решить обе задачи на отрезке [0,1] c шагом *h*=0.01 по индивидуальному варианту из лабораторной работы 6. Определить, для какой из задач метод неустойчив при данном шаге *h*. *h*=0.01. Построить графики компонент полученного решения.
4. Для жесткой задачи экспериментально подобрать шаг *h*, при котором графики компонент решения, полученного по явному методу Эйлера, визуально совпадают с графиками компонент решения, полученного по неявному методу с шагом *h*=0.01. Сравнить найденное значение шага с теоретическим значением шага, при котором явный метод Эйлера для жестких задач должен быть устойчивым.
5. Сравнить полученные результаты.

**Теоретический материал:**

Явный метод Эйлера:

Неявный метод Эйлера:

Усовершенствованный метод Эйлера:

Число жесткости системы:

Условие абсолютной устойчивости метода Эйлера и усовершенствованного метода Эйлера:

**Код программы** *(пояснения к коду зеленым цветом)***:**

**import** numpy  
**from** numpy **import** linalg **as** LA  
**import** matplotlib.pyplot **as** plt  
  
*# рассуждения на тему собственных значений матрицы, устойчивости решений, жесткости системы***def** stiff\_coef(matrix, matrixName):  
 print(**"\nFor matrix "** + matrixName + **":"**)  
  
 l = LA.eigvals(matrix)  
 re\_l = numpy.real(l)  
 s = max(numpy.abs(re\_l)) / min(numpy.abs(re\_l))  
 h\_euler\_stiff = min(2 / numpy.abs(l))  
  
 print(**"Eigenvalues are: "**)  
 **for** i **in** range(len(l)):  
 print(**"l["**, i, **"] = "**, numpy.round(l[i]))  
  
 print(**"Stiff coefficient is: "**, **"%.2f"** % s)  
 **if** h <= h\_euler\_stiff:  
 print(**"Explicit Euler method is stable"**)  
 print(**"Modified Euler method is stable"**)  
 **else**:  
 print(**"Explicit Euler method is unstable"**)  
 print(**"Modified Euler method is unstable"**)  
  
 **if** s > 10:  
 *# для жесткой системы оцениваем шаг для неявного метода Эйлера, чтобы система уравнений была устойчивой* print(**"Matrix "** + matrixName + **" is stiff"**)  
 print(  
 **"\nh estimation for Explicit Euler method to make stiff linear system of equations stable is: "**,  
 **"%.5f"** % h\_euler\_stiff,  
 )  
  
  
*# формула явного метода Эйлера*Euler = **lambda** y, matrix, h: y + h \* matrix \* y  
  
*# формула неявного метода Эйлера***def** Euler\_implicit(y, matrix, h=0.01):  
 E = numpy.eye(3)  
 invMatrix = LA.inv(E - h \* matrix)  
 **return** invMatrix \* y

*# задача Коши*

dy = lambda y, matrix: numpy.array(matrix \* y)

*# формула усовершенствованного метода Эйлера*

def Euler\_modified(y, matrix, h):

    y\_predict = y + h / 2 \* dy(y, matrix)

    y\_correct = y + h \* dy(y\_predict, matrix)

    return y\_correct  
  
*# расчет числа точек N для шага h*get\_n = **lambda** h: int(numpy.ceil((t\_n - t\_0) / h))  
  
*# получить массив решений выбранным методом с шагом h для трех компонент u, v, w***def** get\_y\_data(h, Method, matrix, y0):  
 n = get\_n(h)  
 y\_data = [y0]  
 **for** i **in** range(1, n):  
 y\_data.append(Method(y\_data[i - 1], matrix, h))  
 y\_data = numpy.array(y\_data)  
 u, v, w = y\_data[:, 0], y\_data[:, 1], y\_data[:, 2]  
 **return** [u, v, w]  
  
  
*# построение графика для одной компоненты решения***def** show\_graph(data, title, comp\_title):  
 n = get\_n(h)  
 data = data[:, 0]  
 t\_data = numpy.linspace(t\_0, t\_n, n)  
 fig, ax = plt.subplots()  
 plt.plot(t\_data, data, label=title)  
 ax.set\_title(comp\_title)  
 plt.legend()  
 plt.show()  
  
  
*# построение графиков для двух методов для одной компоненты решения***def** show\_graph\_2(data1, data2, title1, title2, comp\_title, h\_try):  
 n1 = get\_n(h\_try)  
 n2 = get\_n(h)  
  
 t\_data1 = numpy.linspace(t\_0, t\_n, n1)  
 t\_data2 = numpy.linspace(t\_0, t\_n, n2)  
  
 fig, ax = plt.subplots()  
 plt.plot(t\_data1, data1[:, 0], label=title1)  
 plt.plot(t\_data2, data2[:, 0], label=title2)  
 ax.set\_title(comp\_title)  
 plt.legend()  
 plt.show()  
  
  
*# функция, осуществляющая:  
# поиск решения явным методом Эйлера;  
# поиск решения модифицированным методом Эйлера***def** solve(matrix, y0, matrixName):  
 print(**"\nFor matrix "** + matrixName + **": "**)  
 u, v, w = get\_y\_data(h, Euler, matrix, y0)  
 show\_graph(u, **"Euler"**, **"u(t)"**)  
 show\_graph(v, **"Euler"**, **"v(t)"**)  
 show\_graph(w, **"Euler"**, **"w(t)"**)  
 u, v, w = get\_y\_data(h, Euler\_modified, matrix, y0)  
 show\_graph(u, **"Euler modified"**, **"u(t)"**)  
 show\_graph(v, **"Euler modified"**, **"v(t)"**)  
 show\_graph(w, **"Euler modified"**, **"w(t)"**)  
  
  
*# функция, осуществляющая:  
# сравнение результатов явного и неявного метода Эйлера***def** solve\_for\_stiff(matrix, y0):  
 print(**"\n\nFor stiff matrix: "**)  
  
 h\_try = 10 \*\* (-6)  
 print(**"h to try to match graphics is: "**, h\_try)  
 u, v, w = get\_y\_data(h\_try, Euler, matrix, y0)  
 u\_imp, v\_imp, w\_imp = get\_y\_data(h, Euler\_implicit, matrix, y0)  
 show\_graph\_2(u, u\_imp, **"Euler"**, **"Euler implicit"**, **"u(t)"**, h\_try)  
 show\_graph\_2(v, v\_imp, **"Euler"**, **"Euler implicit"**, **"v(t)"**, h\_try)  
 show\_graph\_2(w, w\_imp, **"Euler"**, **"Euler implicit"**, **"w(t)"**, h\_try)  
  
  
A = numpy.matrix(  
 [  
 [-116.967, 38.887, -110.397],  
 [-52.101, -102.573, 275.23],  
 [104.81, -277.406, -100.46],  
 ]  
)  
  
Y0 = numpy.array([[4.4, 3.2, 5.2]]).transpose()  
  
B = numpy.matrix(  
 [[-47.173, 40.843, 27.459], [34.392, -81.93, 26.961], [35.24, 14.016, -86.897]]  
)  
  
Z0 = numpy.array([[4, 3.6, 4.8]]).transpose()  
  
t\_0 = 0  
t\_n = 1  
h = 0.01  
  
stiff\_coef(A, **"A"**)  
stiff\_coef(B, **"B"**)  
solve(A, Y0, **"A"**)  
solve(B, Z0, **"B"**)  
solve\_for\_stiff(B, Z0)

**Результаты выполнения:**

**For matrix A:**

Eigenvalues are:

l[0] = (-120+0j)

l[1] = (-100+300j)

l[2] = (-100-300j)

Stiff coefficient is: **1.20**

Explicit Euler method is **unstable**

Modified Euler method is **unstable**

**For matrix B:**

Eigenvalues are:

l[0] = (-6+0j)

l[1] = (-105+8j)

l[2] = (-105-8j)

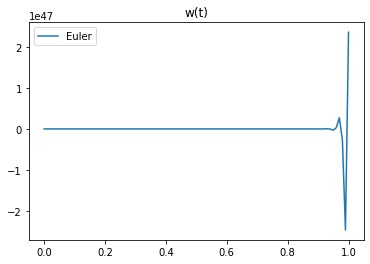
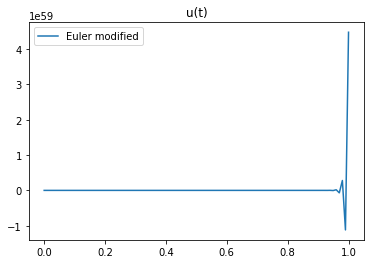
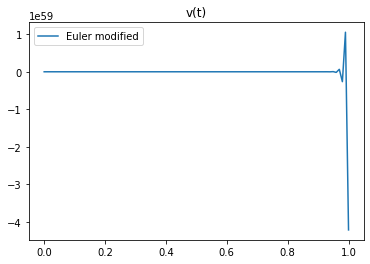
Stiff coefficient is: **18.06**

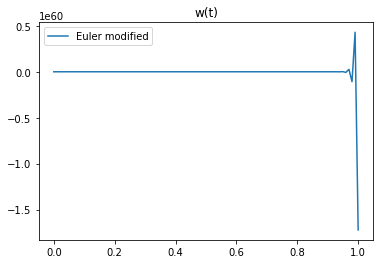
Explicit Euler method is **stable**

Modified Euler method is **stable**

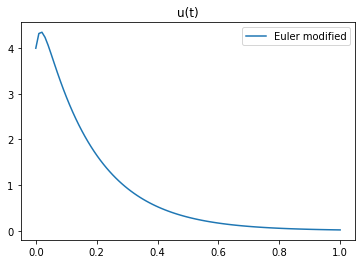
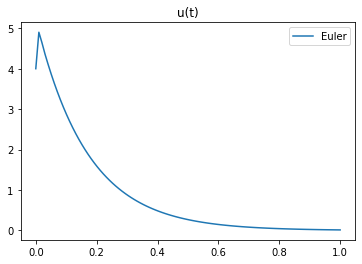
Matrix B is **stiff**

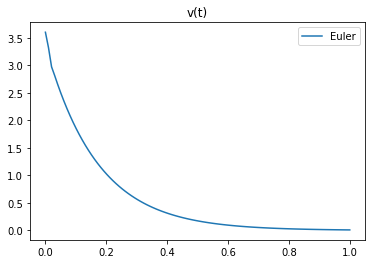
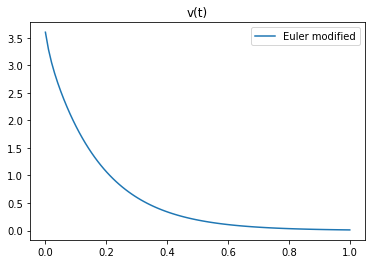
h estimation for Explicit Euler method to make stiff linear system of equations stable is: 0.01897

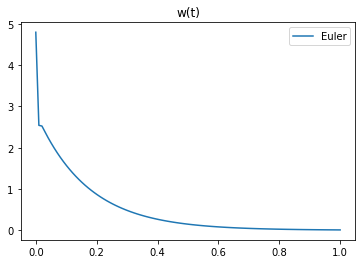
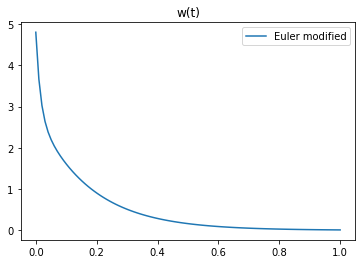
**For matrix A:**

****

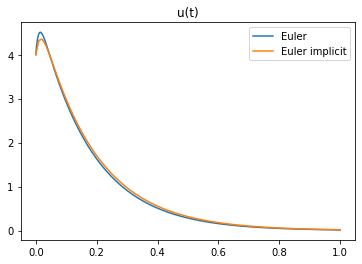
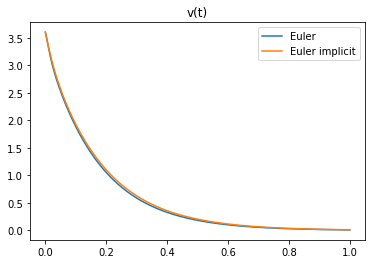
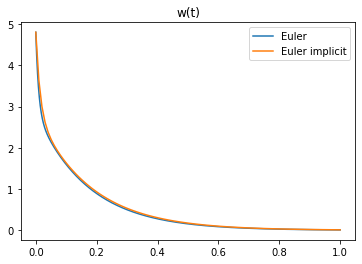
**For matrix B:**

****

****

****

**For stiff matrix:**

****h to try to match graphics is: 1e-06

**Анализ результатов:**

Все собственные значения для обеих задач имеют неположительную вещественную часть, значит можем определить число жесткости систем. Кроме того, сразу же можно сделать вывод, что неявный метод Эйлера будет устойчивым для любого шага.

Для задачи (А) коэффициент жесткости s = 1.2 - никуда не годится, задача не жесткая. Для задачи (В) s = 18.06. Вторая задача является жесткой т.к. s >> 1.

Явный метод Эйлера и модифицированый метод Эйлера имеют одну формулу для оценки шага h для соблюдения условия абсолютной устойчивости:

Для задачи (А) условие не выполняется для данного шага h=0.01.

Из графиков также понятно, что метод ведет себя неустойчивым образом что для метода Эйлера, что для модифицированного метода Эйлера.

Для задачи (B) условие устойчивости выполняется для данного шага h=0.01. Действительно,

Из графиков следует, что у метода Эйлера для всех трех компонент решения на отрезке [0; 0.05] переходный участок, а на отрезке [0.05, 1] - участок жесткости.

Для модифицированого метода Эйлера ситуация лучше - переходный участок заметен только для компоненты u(t) решения, а для v(t) и w(t) - участком жесткости можно назвать весь отрезок [0; 1]

Что касается попытки подобрать h для явного метода Эйлера - графики явного и неявного методов практически совпадают для шага h = 10^-6. Добиться полного наложения не получилось, т.к. при меньших h вычисления занимают очень много времени. Сравнивая найденное значение шага с теоретическим значением шага, при котором явный метод Эйлера для жестких задач должен быть устойчивым -разочаровываемся. Согласно теоретической оценке шаг 0.019 гарантировал устойчивость.