|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № \_\_5\_\_**

|  |  |
| --- | --- |
| **Тема:** Исследование математической модели на основе технологии вычислительного эксперимента.  **Студент \_\_\_\_**Оберган Т.М.**\_\_\_\_\_\_**  **Группа \_\_\_\_**ИУ7-65Б\_\_\_\_\_\_\_\_\_\_\_  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_\_**Градов В.М.\_\_\_ |  |

Москва.

2020 г.

**Оглавление**

[Исходные данные 3](#_Toc43230035)

[Листинг 4](#_Toc43230036)

[Результаты работы 9](#_Toc43230037)

**Цель работы**: получение навыков проведения исследований компьютерной математической модели, построенной на квазилинейном уравнении параболического типа.

# Исходные данные

1. Значения параметров (все размерности согласованы)

, Вт/см К,

, Дж/см3К.

Порядки величин (как в лаб. работе №4):

=0.0134, =1, =4.35 10-4, =1,

=2.049, =0.563 10-3, =0.528 105, =1.

.

Порядки величин (как в лаб. работе №4):

0.05 Вт/см2 К,

0.01 Вт/см2 К,

 10 см,

300К,

0.5 см,

2. Поток тепла  при 

, где - амплитуда импульса потока и время её достижения (Вт/см2 и с).

# Листинг

Листинг 1 – главный модуль

**import** numpy **as** np

**from** math **import** fabs

**from** model **import** \*

**from** visualization **import** draw\_graphs

# Прогонка

**def** get\_T\_new(T\_prev, t):

A\_list, B\_list, C\_list, F\_list = get\_coeffs(T\_prev)

K0, M0, P0 = left\_boundary\_condition(T\_prev, t)

KN, MN, PN = right\_boundary\_condition(T\_prev)

ksi = [-M0 / K0]

eta = [P0 / K0]

n = **len**(A\_list)

**for** i **in** **range**(n):

denominator = B\_list[i] - A\_list[i] \* ksi[i]

ksi.append(C\_list[i] / denominator)

eta.append((F\_list[i] + A\_list[i] \* eta[i]) / denominator)

T\_new = [0] \* (n + 2)

T\_new[-1] = (PN - MN \* eta[i]) / (KN + MN \* ksi[i])

**for** i **in** **range**(**len**(ksi) - 1, -1, -1):

T\_new[i] = ksi[i] \* T\_new[i+1] + eta[i]

**return** T\_new

**def** get\_result():

res = []

curr\_T = [T0 **for** i **in** np.arange(0, l, x\_step)]

res.append(curr\_T)

t = t\_step

t\_total = t

**while** True:

**if** t >= period:

t\_values.append(t)

t = 0

T\_prev = curr\_T

max\_diff = 1

**while** max\_diff > eps:

T\_new = get\_T\_new(T\_prev, t)

max\_diff = 0

**for** i **in** **range**(**len**(T\_prev)):

diff = fabs((T\_new[i] - T\_prev[i]) / T\_new[i])

**if** max\_diff < diff:

max\_diff = diff

T\_prev = T\_new

curr\_T = T\_new

res.append(T\_new)

update\_t\_values(t\_step)

t += t\_step

t\_total += t\_step

**if** t\_total > t\_last:

**break**

**return** res

**if** \_\_name\_\_ == '\_\_main\_\_':

# Вычисление результата

res = get\_result()

# Отображение графиков

draw\_graphs(res, l, t\_last, x\_step, t\_step)

Листинг 2 – модуль данных модели и вычислительных функций

**from** math **import** exp

# Параметры

l = 10

T0 = 300

Fmax = 50

tmax = 20

t\_last = 300

period = 160

t = 1

x\_step, t\_step = 0.1, 1

eps = 1***e***-2

a1 = 0.0134

b1 = 1

c1 = 4.35e-4

m1 = 1

a2 = 2.049

b2 = 0.563e-3

c2 = 0.528e5

m2 = 1

alpha\_0 = 0.05

alpha\_N = 0.01

R = 0.5

k\_0 = 0.4

k\_N = 0.1

b\_arg = (k\_N \* l) / (k\_N - 0.4)

a\_arg = k\_0 \* (-b\_arg)

d\_arg = (alpha\_N \* l) / (alpha\_N - alpha\_0)

c\_arg = alpha\_0 \* (-d\_arg)

t\_values = []

**def** update\_t\_values(h\_t):

**for** i **in** **range**(**len**(t\_values)):

t\_values[i] += h\_t

# Исходные данные модели

**def** k(T):

**return** a1 \* (b1 + c1 \* T \*\* m1)

**def** k\_x(x):

**return** a\_arg / (x - b\_arg)

**def** get\_c(T):

**return** a2 + b2 \* T \*\* m2 - (c2 / T \*\* 2)

**def** alpha(x):

**return** c\_arg / (x - d\_arg)

**def** p(x):

**return** (2 / R) \* alpha(x)

**def** f(x):

**return** (2 \* T0 / R) \* alpha(x)

# Поток тепла

**def** F0(t):

**return** Fmax / tmax \* t \* exp(1 - (t / tmax))

**def** get\_total\_F(t):

total\_F = F0(t)

**for** cur\_t **in** t\_values:

total\_F += F0(cur\_t)

**return** total\_F

# Простая аппроксимация

**def** approc\_plus\_half(func, n, step):

**return** (func(n) + func(n + step)) / 2

**def** approc\_minus\_half(func, n, step):

**return** (func(n) + func(n - step)) / 2

# Краевые условия

# При х = 0

**def** left\_boundary\_condition(T\_prev, t):

c\_0 = get\_c(T\_prev[0])

c\_half = (c\_0 + get\_c(T\_prev[1])) / 2

p\_0 = p(0)

p\_half = approc\_plus\_half(p, 0, x\_step)

X\_half = approc\_plus\_half(k\_x, 0, x\_step)

Ft = get\_total\_F(t)

K0 = x\_step \* (c\_half / 8 + c\_0 / 4 + t\_step / 8 \* p\_half +

(t\_step / 4 \* p\_0)) + X\_half \* t\_step / x\_step

M0 = x\_step / 8 \* c\_half - t\_step / x\_step \* X\_half + \

x\_step / 8 \* t\_step \* p\_half

P0 = x\_step \* (c\_half \* (T\_prev[0] + T\_prev[1]) / 8 +

c\_0 \* T\_prev[0] / 4 + t\_step \*

(approc\_plus\_half(f, 0, x\_step) + f(0)) / 4) + \

Ft \* t\_step

**return** K0, M0, P0

# При х = N

**def** right\_boundary\_condition(T\_prev):

T\_prev\_N = T\_prev[-1]

c\_minus = approc\_minus\_half(get\_c, T\_prev\_N, t)

k\_minus = approc\_minus\_half(k, T\_prev\_N, t)

cN = get\_c(T\_prev\_N)

KN = x\_step / 8 \* c\_minus + x\_step / 4 \* cN + t / x\_step \* k\_minus + t \* alpha\_N + \

t \* x\_step / 8 \* p(l - x\_step / 2) + t \* x\_step / 4 \* p(l)

MN = x\_step / 8 \* c\_minus - t / x\_step \* k\_minus + t \* x\_step / 8 \* p(l - x\_step / 2)

PN = x\_step / 8 \* c\_minus \* (T\_prev\_N + T\_prev[-2]) + \

x\_step / 4 \* cN \* T\_prev\_N + t \* alpha\_N \* T0 + t \* x\_step / 4 \* (f(l) + f(l - x\_step / 2))

**return** KN, MN, PN

**def** get\_coeffs(T\_prev):

A, B, C, F = [], [], [], []

**for** i **in** **range**(1, **len**(T\_prev) - 1):

cur\_x = i \* x\_step

a = approc\_plus\_half(k\_x, cur\_x, x\_step) \* t\_step / x\_step

c = approc\_minus\_half(k\_x, cur\_x, x\_step) \* t\_step / x\_step

A.append(a)

C.append(c)

B.append(a + c + get\_c(T\_prev[i]) \* x\_step + p(cur\_x) \* x\_step \* t\_step)

F.append(f(cur\_x) \* x\_step \* t\_step + get\_c(T\_prev[i]) \* T\_prev[i] \* x\_step)

**return** A, B, C, F

Листинг 3 – модуль построения графиков

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

**def** draw\_graphs(res, x\_max, t\_max, x\_step, t\_step):

# Графики

lenres = **len**(res)

t\_last = **len**(res[0])

res\_cutted = [i[0:t\_last:] **for** i **in** res]

# -- Трехмерный

x, y = np.mgrid[0:lenres:1, 0:t\_last:1]

z = np.array([np.array(i) **for** i **in** res\_cutted])

fig\_3d = plt.figure()

xyz = fig\_3d.add\_subplot(111, projection='3d')

xyz.plot\_surface(x, y, z, cmap='inferno')

fig\_3d.show()

# -- Проекции

fig, (first\_graph, second\_graph) = plt.subplots(

nrows=1, ncols=2,

figsize=(8, 4))

# ------- Первая cm - K

x = **list**(np.arange(0, 10, x\_step))

x\_cutted = x[:t\_last:]

step1 = 10

**for** i **in** res\_cutted[::step1]:

first\_graph.plot(x\_cutted, i)

first\_graph.plot(x\_cutted, res\_cutted[-1])

first\_graph.set\_xlabel("x, cm")

first\_graph.set\_ylabel("T, K")

first\_graph.grid()

# ------- Вторая sec - K

step2 = 5

te = **list**(**range**(0, t\_max, t\_step))

**for** i **in** np.arange(0, x\_max / 3, 0.2 \* step2):

line = [j[**int**(i / x\_step)] **for** j **in** res]

second\_graph.plot(te, line[:-1])

second\_graph.set\_xlabel("t, sec")

second\_graph.set\_ylabel("T, K")

second\_graph.grid()

fig.show()

# Результаты работы

1. Провести исследование по выбору оптимальных шагов по времени и пространству h. Шаги должны быть максимально большими при сохранении устойчивости разностной схемы и заданной точности расчета.

Рассмотреть влияние на получаемые результаты амплитуды импульса и времени (определяют крутизну фронтов и длительность импульса).

Точность расчета можно оценить разными способами:

1. уменьшая шаги и наблюдая сходимость решений, как это делалось в лаб. работе №1;
2. проверяя, соблюдается ли при выбранных баланс мощности после выхода на стационарное распределение температуры (в установившемся режиме), реализующееся при , т.е. в этом режиме должно выполняться условие: подводимая мощность равна отводимой. Имеем

окончательно

Задать точность . Здесь - время выхода на стационарный режим, т.е. когда температура перестает меняться с заданной точностью (см. лаб. работу №4).

*Замечание.* Варьируя параметры задачи, следует иметь ввиду, что решения, в которых температура превышает значения примерно 2000К, физического смысла не имеют и практического интереса не представляют.

Fmax = 50

tmax = 20

t\_last = 300

period = 170

**Исследование шага по пространству:**

Примем t\_step = 1

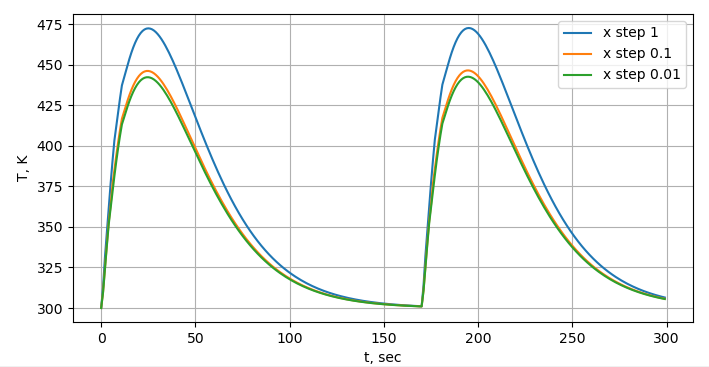


Рис. 1 – варьирование значения x\_step

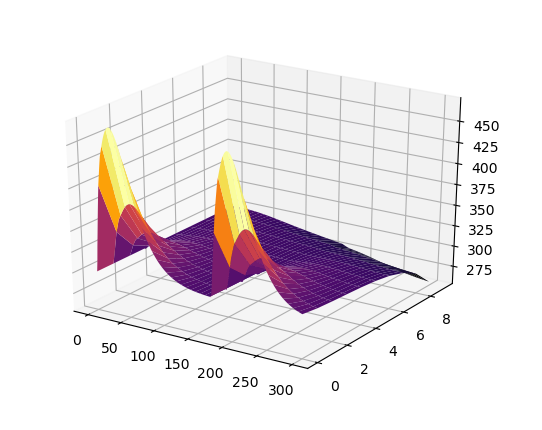


Рис. 2 – трехмерный график время – пространство – температура,

при x\_step = 1

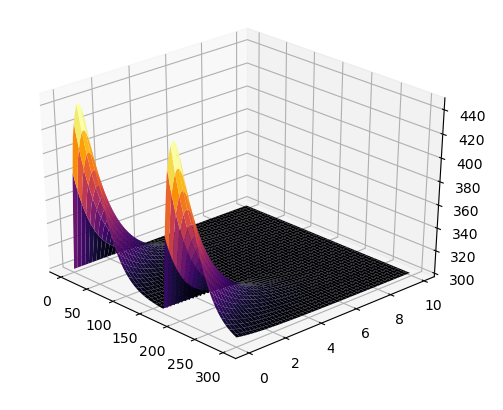


Рис. 3 – x\_step = 0.1

На рис.1 и рис.2 видно, что шаг 1 дает недостаточную точность расчетов. Особенно на рис. 2 заметно, как график «сползает вниз», чего быть не должно.

В качестве шага по x будет принято значение 0.1. **x\_step = 0.1**

**Исследование шага по времени**

Примем x\_step = 0.1

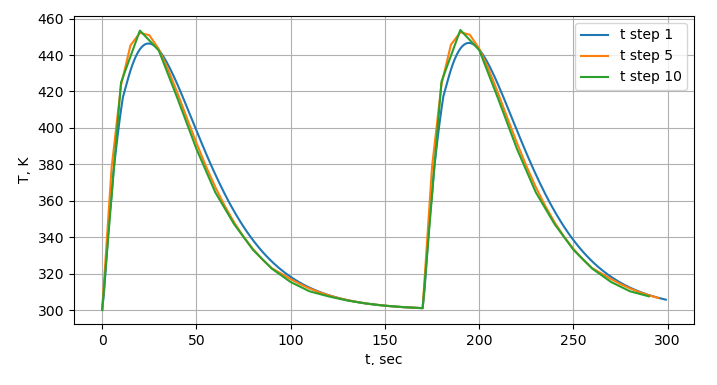


Рис. 4 – варьирование значения t\_step

В качестве значения t\_step будет принято 1. **t\_step = 1**

2. График зависимости температуры при 3-4 значениях параметров и/или теплоемкости.

*Справка*. С ростом теплоемкости темп нарастания температуры снижается.

Fmax = 50; tmax = 20; t\_last = 300; period = 300

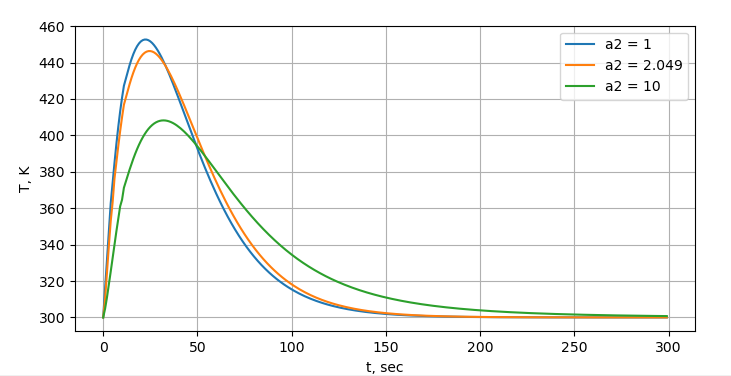


Рис. 4 – варьирование значения a2

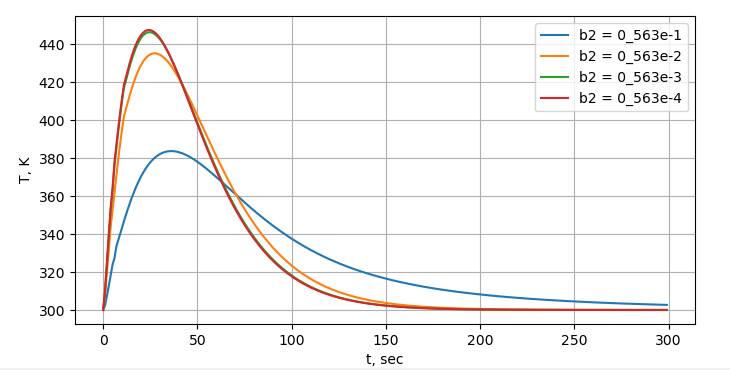


Рис. 5 – варьирование значения b2

3. График зависимости температуры в частотном режиме теплового нагружения. Импульсы следуют один за другим с заданной частотой (частота определяется количеством импульсов в 1 секунду).

Показать, что при большом количестве импульсов температурное поле начинает в точности воспроизводиться от импульса к импульсу.

Продемонстрировать, как по мере роста частоты импульсов размах колебаний температуры уменьшается (вплоть до нуля), т.е. реализуется квазистационарный режим, при котором в торец поступает постоянный поток . Здесь - длительность импульса, определяемая как момент времени, когда . Если взять прямоугольные импульсы длительностью , т.е. , то .

Исследование этой части будет проводиться при значениях: Fmax = 50, tmax = 10

С разными значениями period.

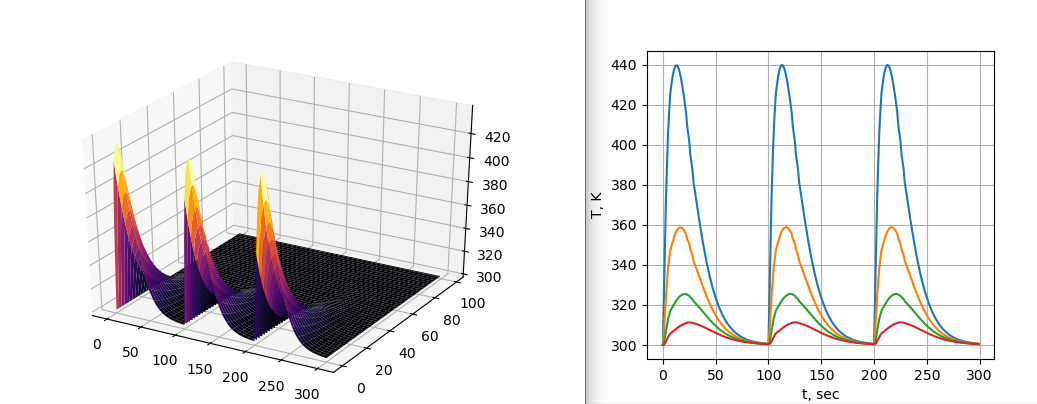


Рис. 6 – Fmax = 50, tmax = 10, period = 100

На этом графике видно, что температурное поле в точности воспроизводится от импульса к импульсу.

При уменьшении периодичности подачи импульсов можно увидеть, как происходит наложение:

Замечание по частотному режиму.  
Вариант перекрытия импульсов сложнее для расчета и малоинтересен для практики. При этом заранее понятно, что размах колебаний температуры будет уменьшаться с ростом частоты.   Когда же перекрытия импульсов нет, эффект уменьшения размаха обусловливается тепловой инерцией материала, здесь физика и числа другие.

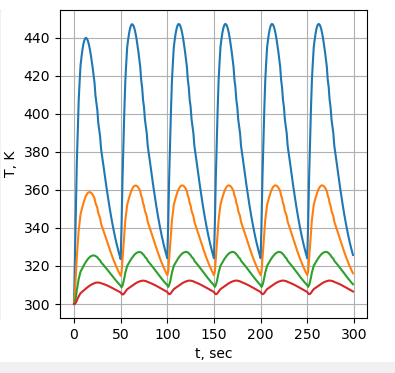


Рис. 7 – Fmax = 50, tmax = 10, period = 50

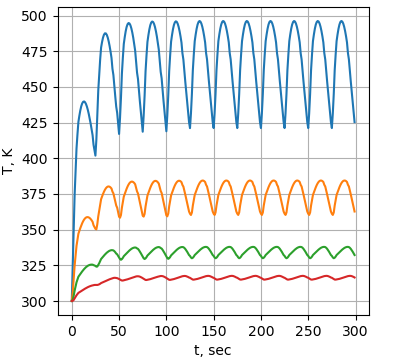
****

Рис. 8 – Fmax = 50, tmax = 10, period = 25

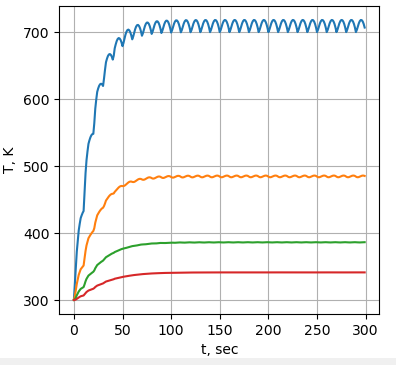


Рис. 9 – Fmax = 50, tmax = 10, period = 10

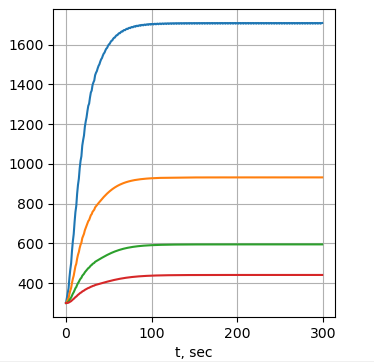


Рис. 10 – Fmax = 50, tmax = 10, period = 3