

Отчёт по лабораторной работе №3

Дисциплина: Научное программирование

Полиенко Анастасия Николаевна, НПМмд-02-23

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Простейшие операции	6
3.2	Операции с векторами	7
3.3	Матричные операции	9
3.4	Построение простейших графиков	11
3.5	Сравнение циклов и операций с векторами	18
4	Выводы	21

Список иллюстраций

3.1	Вектора и матрицы	7
3.2	Операции с векторами	8
3.3	Проекция	9
3.4	Операции с матрицами	10
3.5	Вектора x и y	11
3.6	График $y = \sin x$	12
3.7	Настройка графика	12
3.8	Улучшенный график $y = \sin x$	13
3.9	Точки x и y	14
3.10	Точки x и y на графике	15
3.11	Добавление функции регрессии	15
3.12	Результирующий график	16
3.13	Вектор x	17
3.14	Попытка построить график	17
3.15	График $y = x^2 \sin x$	18
3.16	loop_for	19
3.17	loop_vec	19
3.18	Сравнение двух кодов	20

1 Цель работы

Освоить основы работы с GNU Octave.

2 Задание

1. Изучить задание векторов и матриц
2. Изучить операции над векторами
3. Изучить операции над матрицами
4. Построить графики функций
5. Сравнить эффективность двух кодов

3 Выполнение лабораторной работы

3.1 Простейшие операции

Для ведения журналирования используется операция *diary on*. Консоль в Octave можно использовать как простой калькулятор для простейших вычислений.

Для задания векторов и матриц используются [и] (рис. 3.1).

```

>>diary on
>> 2*6+(7-4)^2
ans = 21
>> u=[1 -4 6]
u =

    1    -4     6

>> u=[1; -4; 6]
u =

     1
    -4
     6

>> A=[1 2 3; 2 4 0; 1 1 1]
A =

     1     2     3
     2     4     0
     1     1     1

>> |

```

Рис. 3.1: Вектора и матрицы

3.2 Операции с векторами

1. В Octave можно складывать вектора и умножать их на скаляр, вычислять скалярное и векторное произведение двух векторов и норму вектора (рис. 3.2).

```
>> u=[1; -4; 6]
u =

     1
    -4
     6

>> v=[2; 1; -1]
v =

     2
     1
    -1

>> 2*v+3*u
ans =

     7
    -10
    16

>> dot(u, v)
ans = -8
>> cross(u, v)
ans =

    -2
    13
     9

>> norm(u)
ans = 7.2801
>>
```

Рис. 3.2: Операции с векторами

1. Для вычисления проекции вектора используется формула $proj_{\vec{u}} = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|^2} \vec{v}$ (рис. 3.3).

```
>> u=[3 5]
u =
    3    5

>> v=[7 2]
v =
    7    2

>> proj = dot(u, v)/norm(v)^2 * v
error: parse error:

      syntax error

>>> proj = dot(u, v)/norm(v)^2 * v
      ^
>> proj = dot(u, v)/(norm(v)^2 * v
proj =
    4.0943    1.1698
```

Рис. 3.3: Проекция

3.3 Матричные операции

В Octave можно складывать и перемножать матрицы, умножать их на скаляр и транспонировать, вычислять определитель и обратную матрицу, находить собственные значения и ранг матрицы (рис. 3.4).

```

>> A=[1 2 -3; 2 4 0; 1 1 1]
A =

     1     2    -3
     2     4     0
     1     1     1

>> B=[1 2 3 4; 0 -2 -4 6; 1 -1 0 0]
B =

     1     2     3     4
     0    -2    -4     6
     1    -1     0     0

>> A*B
ans =

    -2     1    -5    16
     2    -4   -10    32
     2    -1    -1    10

>> B'*A
ans =

     2     3    -2
    -3    -5    -7
    -5   -10    -9
    16    32   -12

>> 2*A - 4*eye(3)
ans =

    -2     4    -6
     4     4     0
     2     2    -2

>> det(A)
ans = 6
>> inv(A)
ans =

    0.6667   -0.8333    2.0000
   -0.3333    0.6667   -1.0000
   -0.3333    0.1667     0

>> eig(A)
ans =

    4.5251 + 0i
    0.7374 + 0.8844i
    0.7374 - 0.8844i

>> rank(A)
ans = 3

```

Рис. 3.4: Операции с матрицами

3.4 Построение простейших графиков

1. Построим график функции $\sin x$ на интервале $[0, 2\pi]$

Создаём вектор значений x и вектор значений y (рис. 3.5) и строим простейший график с помощью команды *plot* (рис. 3.6).

```
>> x = linspace(0, 2*pi, 50)
x =

Columns 1 through 11:
    0    0.1282    0.2565    0.3847    0.5129    0.6411    0.7694    0.8976    1.0258    1.1541    1.2823

Columns 12 through 22:
    1.4105    1.5387    1.6670    1.7952    1.9234    2.0517    2.1799    2.3081    2.4363    2.5646    2.6928

Columns 23 through 33:
    2.8210    2.9493    3.0775    3.2057    3.3339    3.4622    3.5904    3.7186    3.8468    3.9751    4.1033

Columns 34 through 44:
    4.2315    4.3598    4.4880    4.6162    4.7444    4.8727    5.0009    5.1291    5.2574    5.3856    5.5138

Columns 45 through 50:
    5.6420    5.7703    5.8985    6.0267    6.1550    6.2832

>> y = sin(x)
y =

Columns 1 through 11:
    0    0.1279    0.2537    0.3753    0.4907    0.5981    0.6957    0.7818    0.8551    0.9144    0.9587

Columns 12 through 22:
    0.9872    0.9995    0.9954    0.9749    0.9385    0.8866    0.8202    0.7403    0.6482    0.5455    0.4339

Columns 23 through 33:
    0.3151    0.1912    0.0641   -0.0641   -0.1912   -0.3151   -0.4339   -0.5455   -0.6482   -0.7403   -0.8202

Columns 34 through 44:
   -0.8866   -0.9385   -0.9749   -0.9954   -0.9995   -0.9872   -0.9587   -0.9144   -0.8551   -0.7818   -0.6957

Columns 45 through 50:
   -0.5981   -0.4907   -0.3753   -0.2537   -0.1279   -0.0000

>> plot(x,y)
```

Рис. 3.5: Вектора x и y

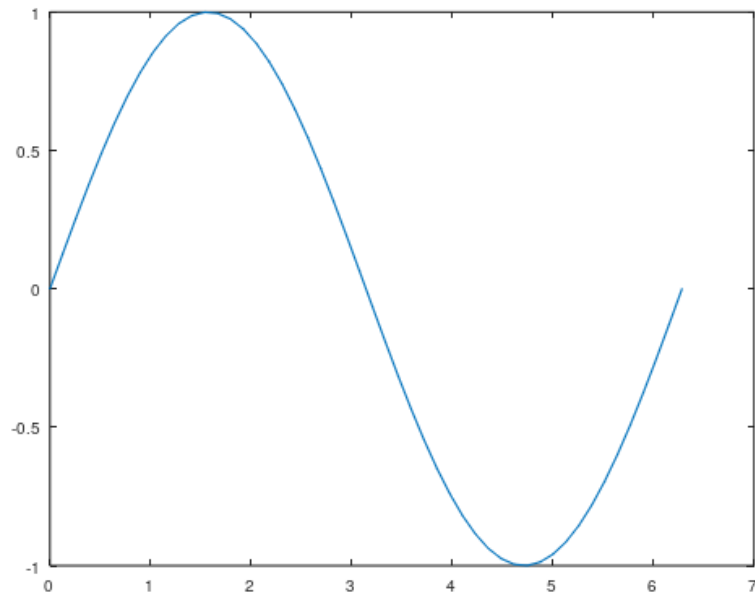


Рис. 3.6: График $y = \sin x$

1. Улучшаем график, изменяя цвет и толщину линии, подписывая оси и подгоняя их диапазон, добавляя сетку, легенду и название графика (рис. 3.7).

```
>> plot(x, y, 'r', 'linewidth', 3)
>> axis([0 2*pi -1 1])
>> grid on
>> xlabel('x')
>> ylabel('y')
>> title('Sine graph')
>> legend('y=sin(x)')
```

Рис. 3.7: Настройка графика

В результате получаем такой график (рис. 3.8).

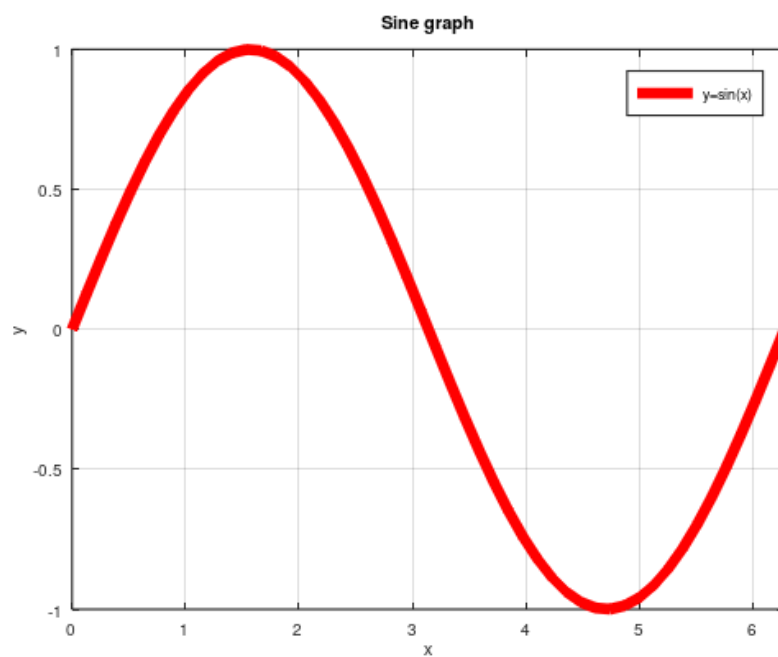


Рис. 3.8: Улучшенный график $y = \sin x$

1. Построим несколько графиков на одной картинке.

Создаём точки x и y (рис. 3.9).

```
>> clear
>> clf
>> x = [1 2 3 4]
x =

    1    2    3    4

>> y = [1 2 5 4]
y =

    1    2    5    4
```

Рис. 3.9: Точки x и y

Выведем их на график (рис. 3.10).

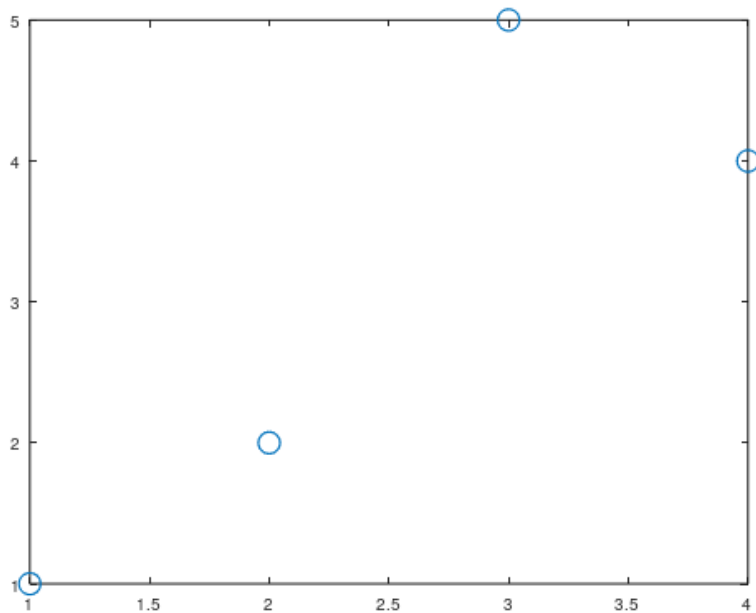


Рис. 3.10: Точки x и y на графике

Добавим функцию регрессии и добавим её на существующий график с помощью функции *hold on* (рис. 3.11).

```
>> plot(x,y, 'o')
>> hold on
>> plot(x, 1.2*x)
>> grid on
>> axis([0 5 0 6])
>> legend('data points', 'regressionline')
```

Рис. 3.11: Добавление функции регрессии

В результате получаем такой график (рис. 3.12).

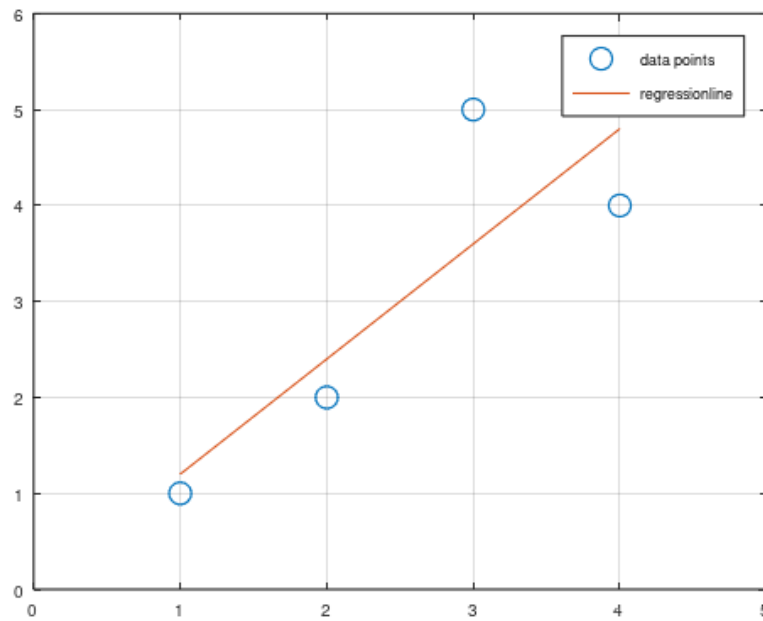


Рис. 3.12: Результирующий график

1. При построении графиков важно использовать поэлементное, а не матричное умножение.

Создаём вектор значений x (рис. 3.13).


```

>> clf
>> clear
>> x = linspace(-10, 10, 100)
x =

Columns 1 through 10:
-10.0000 -9.7980 -9.5960 -9.3939 -9.1919 -8.9899 -8.7879 -8.5859 -8.3838 -8.1818
Columns 11 through 20:
-7.9798 -7.7778 -7.5758 -7.3737 -7.1717 -6.9697 -6.7677 -6.5657 -6.3636 -6.1616
Columns 21 through 30:
-5.9596 -5.7576 -5.5556 -5.3535 -5.1515 -4.9495 -4.7475 -4.5455 -4.3434 -4.1414
Columns 31 through 40:
-3.9394 -3.7374 -3.5354 -3.3333 -3.1313 -2.9293 -2.7273 -2.5253 -2.3232 -2.1212
Columns 41 through 50:
-1.9192 -1.7172 -1.5152 -1.3131 -1.1111 -0.9091 -0.7071 -0.5051 -0.3030 -0.1010
Columns 51 through 60:
0.1010 0.3030 0.5051 0.7071 0.9091 1.1111 1.3131 1.5152 1.7172 1.9192
Columns 61 through 70:
2.1212 2.3232 2.5253 2.7273 2.9293 3.1313 3.3333 3.5354 3.7374 3.9394
Columns 71 through 80:
4.1414 4.3434 4.5455 4.7475 4.9495 5.1515 5.3535 5.5556 5.7576 5.9596
Columns 81 through 90:
6.1616 6.3636 6.5657 6.7677 6.9697 7.1717 7.3737 7.5758 7.7778 7.9798
Columns 91 through 100:
8.1818 8.3838 8.5859 8.7879 8.9899 9.1919 9.3939 9.5960 9.7980 10.0000

```

Рис. 3.13: Вектор x

При попытке построить график возникает ошибка (рис. 3.14).

```

>> plot(x, x^2*sin(x))
error: for x*y, only square matrix arguments are permitted and one argument must be scalar. Use .^ for
elementwise power.

```

Рис. 3.14: Попытка построить график

Исправляем матричное умножение на поэлементное, получаем график (рис. 3.15).

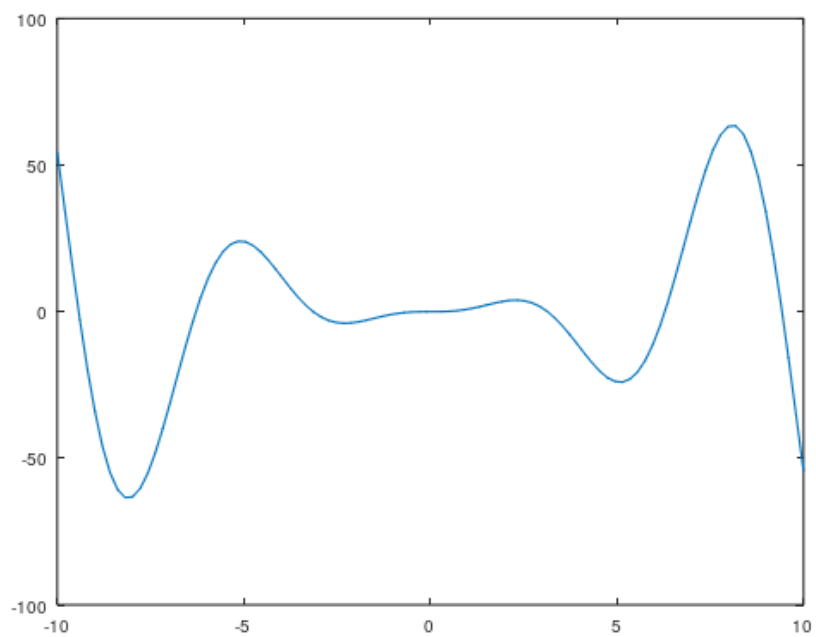
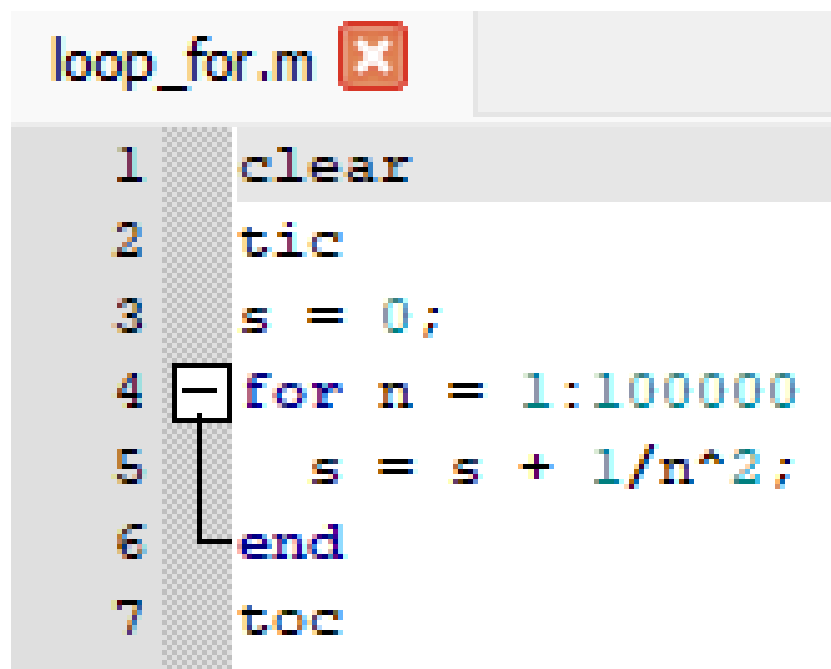


Рис. 3.15: График $y = x^2 \sin x$

3.5 Сравнение циклов и операций с векторами

Подсчитаем сумму $\sum_{n=1}^{100000} \frac{1}{n^2}$.

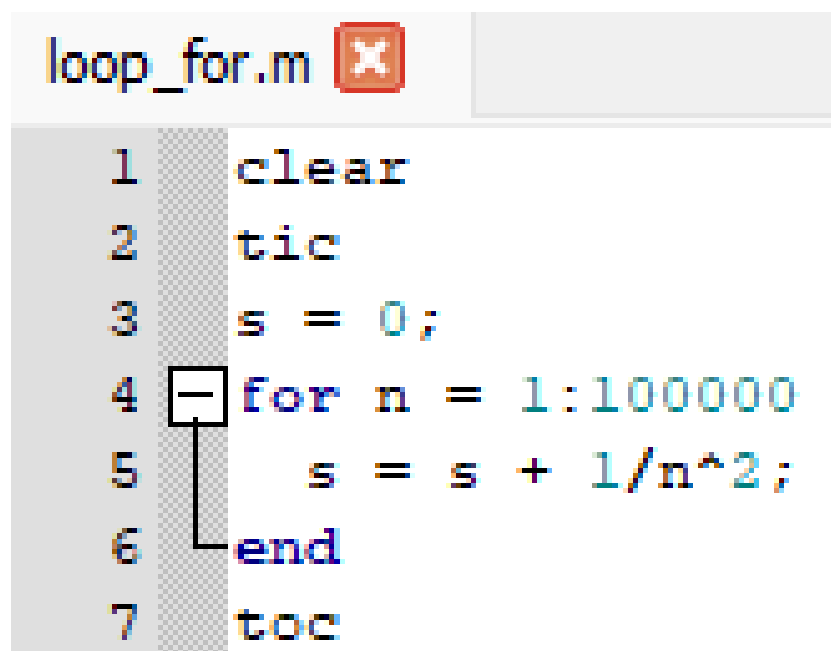
Это можно сделать с помощью цикла *for* (рис. 3.16).



```
1 clear
2 tic
3 s = 0;
4 for n = 1:1000000
5     s = s + 1/n^2;
6 end
7 toc
```

Рис. 3.16: loop_for

Или операции *sum* для вектором (рис. 3.17).



```
1 clear
2 tic
3 s = 0;
4 for n = 1:1000000
5     s = s + 1/n^2;
6 end
7 toc
```

Рис. 3.17: loop_vec

Операции с векторами намного эффективнее циклов (рис. 3.18).

```
>> loop_for -  
Elapsed time is 0.292815 seconds.  
>> loop_vec  
Elapsed time is 0.00413609 seconds.
```

Рис. 3.18: Сравнение двух кодов

4 Выводы

Изучила основы языка Octave и научилась работе с векторами и графиками.