Практикум №4: системы уравнений, интегралы, производные

1 Системы уравнений

1. *Решить систему уравнений*

$$\begin{cases} -x_1 + x_2 + 2x_3 = 10; \\ 3x_1 - x_2 + x_3 = -20; \\ -x_1 + 3x_2 + 4x_3 = 40. \end{cases}$$

Представим ее в виде $\boldsymbol{Ax} = \boldsymbol{b}$. Инициализируем постоянные:

```
A=[-1 1 2; 3 -1 1; -1 3 4];
b=[10; -20; 40];
```

Нам необходимо проверить на вырожденность матрицу $\boldsymbol{A}$:

```
det(A)
ans =  10.000
```

Теперь решить данную систему можно несколькими способами.

1. Через обратную матрицу.

$$\boldsymbol{A}^{-1}\boldsymbol{Ax} = \boldsymbol{A}^{-1}\boldsymbol{b}, \quad \Rightarrow \quad \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}.$$

   В Octave это примет вид:

```
x = inv(A)*b
x =
1.00000
19.00000
-4.00000
```

   Проверим решение:

```
A*x
ans =
10.000
-20.000
40.000
```

2. Метод Гаусса. Приведем к верхней треугольной форме расширенную матрицу $(\boldsymbol{A} : \boldsymbol{b})$:

```
rref([A b])
ans =
1.00000   0.00000   0.00000    1.00000
0.00000   1.00000   0.00000   19.00000
0.00000   0.00000   1.00000   -4.00000
```

Слева мы получили единичную матрицу, что значительно упрощает вычисления. Однако, если бы матрица не имела нулей в правом верхнем углу, мы все равно могли бы найти корни системы (обратный ход метода Гаусса).

3. Автоматический метод Гаусса. В данном случае необходимо лишь воспользоваться уже известным вам оператором «левого» (или обратного) деления:

```
x = A\b
x =
1.0000
19.0000
-4.0000
```

Далее мы будем использовать именно этот способ решения линейных систем уравнений.

2. *Решить систему уравнений, заданную вырожденной матрицей*

$$\begin{cases} x_1 + 3x_2 + 7x_3 = 5; \\ -x_1 + 4x_2 + 4x_3 = 2; \\ x_1 + 10x_2 + 18x_3 = 12. \end{cases}$$

```
A = [ 1 3 7; -1 4 4; 1 10 18];
b = [5; 2; 12];
det(A)
ans = 0
```

Так как определитель матрицы коэффициентов равен нулю, невозможно найти обратную матрицу. Однако, можно воспользоваться способом решения через *псевдообратную матрицу*:

```
x = pinv(A)*b
x =
0.38498
-0.11033
0.70657
% check
A*x
ans =
5.0000
2.0000
12.0000
```

Однако, изменим вектор $\boldsymbol{b}$:

```
b = [3;6;0];
x = pinv(A)*b
```

```
x =
-1.08920
1.25117
-0.52347
% check
A*x
ans =
-1.0000
4.0000
2.0000
```

В этом случае решение не будет точным (точнее, оно вообще не является решением данной системы). Проверим, возможно ли найти общее решение данной системы уравнения, приведя к верхней треугольной форме расширенную матрицу $(\boldsymbol{A} : \boldsymbol{b})$:

```
rref([A b])
ans =
1.00000   0.00000   2.28571   0.00000
0.00000   1.00000   1.57143   0.00000
0.00000   0.00000   0.00000   1.00000
```

Последняя строка содержит ненулевой элемент лишь в столбце свободных членов, что однозначно свидетельствует об отсутствии решений данной системы уравнений.

2 Степенные уравнения

1. *Найти решение уравнения* $2x^2 - 4x + 5 = 0$.

Для этого необходимо инициализировать полином набором коэффициентов и найти корни командой `roots`.

```
p = [2 -4 5];
x = roots(p)
x =
1.0000 + 1.2247i
1.0000 - 1.2247i
```

Итак, корни нашего уравнения: $x = 1 \pm 1.2247i$. Точность вычислений Octave можно задать явно командой `format`. Для отображения результата в виде рациональных дробей можно указать следующее.

```
format rat
x =
1 + 4801/3920i
1 - 4801/3920i
```

Вернуться к прежнему виду результатов можно командой `format short`.

2. *Найти корни полинома* $p(x) = x^4 + 2x^3 - 3x^2 + 4x + 5$ *и получить его график на отрезке* $[-4, 2]$.

```
p = [1 2 -3 4 5];
x = roots(p)
x =
-3.18248 + 0.00000i
0.95560 + 1.11480i
0.95560 - 1.11480i
-0.72873 + 0.00000i

x=[-4:.05:2]; y=polyval(p,x);
plot(x,y)
```

Нарисуем ось X:

```
hold on
plot([-4 2], [0 0],'k')
```

Команда `hold on` позволяет «дорисовать» что-либо на уже имеющемся графике. Буква `'k'` в параметре означает рисование черным цветом. Отключить вывод на один и тот же график можно командой `hold off`.

3. *Найти решение уравнения* $y = x^3 + x^2 - 3x - 3$.

Зададим функцию:

```
f = inline("x^3+x^2-3*x-3");
```

Функция `fsolve` позволяет решать нелинейные уравнения, и ее можно применить в т.ч. к решению степенных уравнений. Необходимо задать начальное приближение для поиска. Задавая разные значения, получим разные корни:

```
fsolve (f, 1)
ans =  1.7321
fsolve (f, 0)
ans = -1
fsolve (f, -2)
ans = -1.7321
```

Можем проверить корни:

```
p=[1 1 -3 -3]
p =
1   1  -3  -3
roots(p)
ans =
1.7321
-1.7321
-1.0000
```

А теперь попробуем решить этим же методом систему уравнений:

$$\begin{cases} \mathrm{e}^{-\mathrm{e}^{-(x+y)}} = y(1+x^2), \\ x\cos y + y\sin x = 1/2. \end{cases}$$

Для начала конвертируем их к виду $F(x) = 0$:

$$\begin{cases} e^{-e^{-(x+y)}} - y(1+x^2) = 0, \\ x \cos y + y \sin x - 1/2 = 0. \end{cases}$$

Запишем функцию, позволяющую вычислить обе компоненты:

```
function F = F(x)
    F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1).^2);
    F(2) = x(1).*cos(x(2)) + x(2).*sin(x(1)) - 0.5;
endfunction
```

Теперь попробуем найти решение, начиная с $(0, 0)$:

```
fsolve(@F, [0 0])
ans =
0.35325   0.60608
```

3 Численное интегрирование, дифференциальные уравнения

1. *Найти интеграл* $\int\limits_0^3 x(\sin \frac{1}{x})\sqrt{|1-x|}\,dx$.

Для вычисления подынтегральной функции в каждом узле интегрирования, нам необходимо задать функцию `i1.m`:

```
function y = i1 (x)
    y = x .* sin (1./x) .* sqrt (abs (1 - x));
endfunction
```

Для интегрирования с оптимальным расчетом квадратур можно использовать функцию `quad`:

```
[q, ier, nfun, err] = quad (@i1, 0, 3)
ABNORMAL RETURN FROM DQAGP
q =  1.9819
ier =  1
nfun =  5061
err =  0.00000011522
```

`q` – результат интегрирования, `ier` – код ошибки интегрирования (при нормальной процедуре равен 0), `nfun` – количество узлов интегрирования, `err` – оценка ошибки интегрирования.

Здесь и во многих других функциях первым аргументом является либо строка с именем функции, либо ссылка на нее (как в данном случае), либо inline-функция.

Еще примеры интегрирования. Квадратурная формула Гаусса–Конрода:

```
f = inline ("x.^3");
quadgk (f, 0, 1)
ans =  0.25000
```

Квадратура Кленшоу–Куртиса (и бесконечный предел интегрирования):

```
f = @(x) x.^3 .* exp (-x);
quadcc (f, 0, Inf)
ans =  6.0000
```

Квадратура Симпсона:

```
f = inline ("x.^3");
quadv(f, 0, 1)
ans =  0.25000
```

Автоматический выбор квадратуры:

```
integral(f, 0, 1)
ans =  0.25000
```

2. *Найти интеграл* $\int\limits_0^5 (x^4 + 2x^2 - 1)dx$

Можно посчитать интеграл и другим способом, если задан полином: определим коэффициенты полинома, вычислим новый полином, являющийся интегралом нашего, а затем, вычитая первообразные, найдем искомый интеграл:

```
c = [1 0 2 0 -1];
i = polyint(c);
I = polyval(i, 5) - polyval(i, 0)
I =  703.33
```

Аналогичным образом мы можем вычислять производные:

```
d = polyder(c);
polyval(d, [1:5])
ans =
8    40   120   272   520
```

3. *Вычислить интеграл* $\int\limits_0^1 dx \int\limits_{-1}^1 \cos(\pi xy)\sqrt{x|y|}\, dy$

Для двухмерного интегрирования воспользуемся функцией `dblquad`

```
I = dblquad(@(x, y) cos (pi*x.*y) .* sqrt (x.*abs(y)), 0, 1, -1, 1)
I =  0.30892
% OR
I = quad2d(@(x, y) cos (pi*x.*y) .* sqrt (x.*abs(y)), 0, 1, -1, 1)
I =  0.30892
% OR
[I err] = integral2(@(x, y) cos (pi*x.*y) .* sqrt (x.*abs(y)), 0, 1, -1, 1)
I =  0.30892
err =  0.00000030870
```

Тройные интегралы — `triplequad` или `integral3`.

4. *Решить дифференциальное уравнение* $\dot{x} = -\mathrm{e}^t x^2$ *при* $x(0) = 2$.

Запишем функцию, вычисляющую $\dot{x}$:

```
function xdot = ode1(x, t)
    xdot = -exp(-t)*x^2;
endfunction
```

Заданим аргумент $t \in [0, 5]$ как вектор в 50 экземпляров

```
    t = linspace(0,5,50);
    x = lsode(@ode1, 2, t);
    plot(t,x)
```

`lsode` решает простейшее уравнение $\frac{dy}{dx} = f(x, y)$ при начальных условиях $y(0)$ по заданному вектору x.

5. *Решить методом Рунге–Кутты дифференциальное уравнение ван дер Поля* $y'' + \mu(1 - y^2)y' + y = 0$, $\mu > 0$.

Для начала перепишем это уравнение с заменой $y_1 = y$, $y_2 = y_1'$: $y_2' = \mu(1 - y_1^2)y_2 - y_1$. Для простоты примем $\mu = 1$. Введем функцию, описывающую наше уравнение (ее необходимо ввести как новый m-файл и сохранить под именем `vdp1.m`):

```
% Initialisation of Van der Pol with mu=1
function dydt = vdp1(t,y)
    dydt = [y(2); (1-y(1)^2)*y(2)-y(1)];
endfunction
```

Теперь найдем решение уравнения и отобразим графики функции y и ее первой производной:

```
    [t, y] = ode45(@vdp1, [0 20], [2; 0]);
    plot(t, y(:,1), '-', t, y(:,2), '--')
```

Функция `ode45` в качестве первого параметра требует имя функции, в которой описано дифференциальное уравнение; второй параметр — интервал, в котором изменяется аргумент искомой функции; третий аргумент — начальные условия для функции и ее производной. Возвращаемое значение y содержит два столбца: в первом находится искомая функция, а во втором — ее первая производная.

Итак, для численного решения дифференциального уравнения в Octave необходимо сначала представить это уравнение в виде линейной системы

$$\begin{cases} y_1' = f_1(x, y_1, \ldots, y_n), \\ y_2' = f_2(x, y_1, \ldots, y_n), \\ \ldots \\ y_n' = f_n(x, y_1, \ldots, y_n). \end{cases}$$

Затем функции f_1, …, f_n следует определить как строки специальной функции, которая будет играть роль первого параметра функции, решающей данное уравнение.

4 Численное дифференцирование

1. *Для ряда данных вычислить производную и построить график функции и производной*

```
x = [0:0.01:10];
y = y=x.^2.*sin(x)+sin(x/11)-tan(x*222)/cos(x);
```

Простейший способ найти производную — воспользоваться методом разделенных разностей. Функция `diff` вычисляет разность $y(x_{i+1}) - y(x_i)$. Производную $y'(x)$ мы можем рассчитать в нулевом приближении либо как $\frac{y(x_{i+1})-y(x_i)}{x_{i+1}-x_i}$, либо как $\frac{y(x_i)-y(x_{i-1})}{x_i-x_{i-1}}$. Попробуем оба способа. Учитывая то, что мы имеем равномерно распределенный ряд, вычисления упрощаются.

```
dy1=[0 diff(y)]/0.01;
dy2=[diff(y) 0]/0.01;
plot(x,[y;dy1;dy2])
```

Благодаря гладкости функции и большому шагу, мы практически не видим разницы. Однако, если мы в 10 раз уменьшим шаг, сдвиг уже будет иметь значение.

Кстати, мы можем и простейшим образом (трапециями) вычислить интеграл:

```
iy = [cumsum(y)];
plot(x,[y;dy1;iy])
legend("F", "dF", "iF")
```

Если добавить ось X (`plot(x,[y;dy1;iy], x, zeros(size(x)))`), поведение интегральной кривой отлично отразится на оригинальной функции.

2. *Найти производную зашумленного ряда данных.*

(не удалять предыдущие данные!)

О функции `polyder` мы уже упоминали. Она отлично подходит для тех наборов данных, которые можно аппроксимировать полиномом. Давайте повторим предыдущие вычисления `y`, но добавим шум в 10дБ:

```
yn = awgn(y, 10, "measured");
plot(x,[y;yn], x, zeros(size(x)))
```

Естественно, функции `diff` и `cumsum` в данном случае будут давать ужасный результат:

```
plot(x,[yn;[0 diff(yn)]/0.01])
```

Попробуем аппроксимировать нашу кривую полиномом десятой степени и сравнить на графике (а потом сравним с оригиналом):

```
p=polyfit(x,yn, 10);
plot(x,[yn; polyval(p,x)])
plot(x,[y; polyval(p,x)])
```

Естественно, в самом начале (в районе нуля) шумы настолько велики, что аппроксимация получается, мягко говоря, не очень. Но это все равно лучше, чем начальный зашумленный ряд.

Теперь вычисляем производную и сравним с предыдущей.

```
dp = polyder(p);
dyp=polyval(dp, x);
plot(x,[dy1;dyp])
```

И еще:

```
plot(x,[y;yn;dy1;dyp])
```

Можно попробовать разные степени полинома для аппроксимации этой функции, сравнив результаты.

Еще одним вариантом вычисления производной является функция `gradient`. Здесь можно «автоматически» учесть шаг:

```
plot(x,[y;dy1;gradient(y,0.01)])
```

А в случае неравномерно распределенных данных, мы можем задать вектор `x`.

```
x = [0 0.1 0.5 1 1.1 1.5 7 7.1 7.2 7.5 10 10.5 12 15 20 20.1 25 45 47 56 100];
y = x.^2.*sin(x)+sin(x/11)-tan(x*222)/cos(x);
dyy = gradient(y, x);
x1 = [0:0.1:100];
y1 = x1.^2.*sin(x1)+sin(x1/11)-tan(x1*222)/cos(x1);
plot(x, dyy, x1, [0 diff(y1)])
% и сравним с ходом оригинальной функции
plot(x, [y; dyy], x1, [y1; [0 diff(y1)]])
% who is who
legend("bad", "dbad", "ori", "dori")
```

3. *Вычислите вторую производную предыдущей функции*

Для этого можно воспользоваться функцией `del2` (дискретный Лапласиан):

```
plot(x,[y;del2(y)*1e4])
```

Не забываем, что т.к. мы вычисляем вторую производную, то интервал необходимо возвести в квадрат!

Естественно, N-ю производную мы можем вычислить и многократным вызовом функции `diff`, если данные распределены равномерно.

5 Задания для самостоятельного выполнения

1. Решите систему уравнений
$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1; \\ 2x_1 - x_2 + 4x_3 = 2; \\ x_1 - 3x_2 + x_3 = 3. \end{cases}$$

2. Решите систему уравнений
$$\begin{cases} x_1 + x_2/2 + x_3/3 = 1; \\ x_1/2 + x_2/3 + x_3/4 = 0; \\ x_1/3 + x_2/4 + x_3/5 = 0. \end{cases}$$

   Обратите внимание, что определитель матрицы коэффициентов `det(A) = 4.6296e-04`. Такие системы называются **плохо обусловленными**. Их решения сильно осциллируют при малейших изменениях коэффициентов матрицы.

3. Решите уравнение $x^7 - 2x^5 + 3x^3 - 4x = 0$.
4. Решите систему уравнений

$$\begin{cases} \mathrm{e}^{x+y} = \sin x; \\ \cos x = \ln y - 1. \end{cases}$$

5. Вычислите $\int\limits_0^1 \ln(x+1)\sin x\,dx$.
6. Вычислите $\int\limits_{-1}^{2} dx \int\limits_{-\pi}^{0} dy \int\limits_0^1 \frac{ln(xyz)}{\cos(xy)}dz$.
7. Найдите решение уравнения ван дер Поля при $\mu = 5$.
8. Постройте график решения задачи Коши методом Рунге–Кутты на интервале $[0,1]$ для уравнения $y' = x^3 \sin y + 1$ при $y(0) = 0$.
9. Найдите решение системы уравнений:

$$\begin{cases} 2(x-4)^2 + 7(y-8)^2 = z^2; \\ 5(x-1)^2 + 1 + 2z^2 = 4(y+3)^2; \\ x^2 + y^2 + z^2 = 0. \end{cases}$$

10. Вычислите производную и интеграл для ряда данных $y = y(t)$.

```
t = [0 1 3 5 9 10 11 15 20 21 23 25 50 52 57 59 60 73 94 96 99 100];
y = [41.6 -0.4 7.6 -25.8 5.3 23.1 636.7 -46.7 -3.7 -29.1 96.6 3.3 -9.4 56.7
     17.5 -17.1 17.4 4.3 -0.3 12.3 85.9 44.2];
```

11. Вычислить производную и интеграл для функции $\left(\frac{\sin x}{x}\right)^2$ с отношением сигнал-шум 10дБ на промежутке $[-10, 10]$.