Расчетно-графическая работа №2

РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Введение

Решение нелинейных уравнений вида

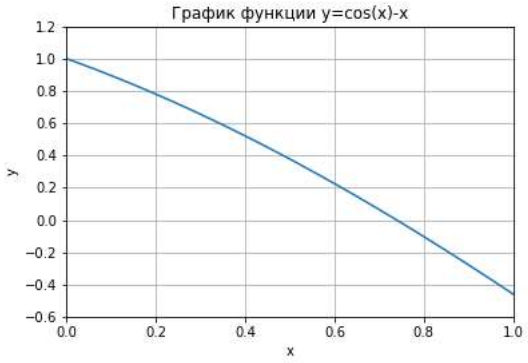
$$f(x)=0$$

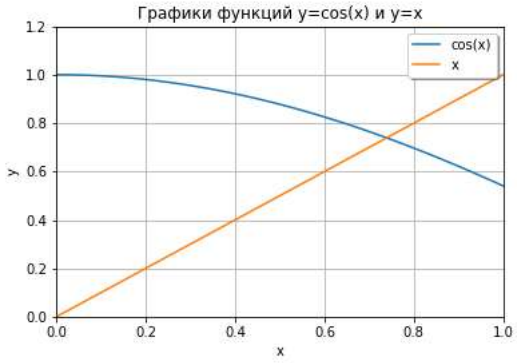
заключается в отыскании одного или нескольких корней, т.е. таких значений аргумента x, при которых функция $f(x)$ обращается в нуль. В общем случае аналитические формулы для корней функции $f(x)$ получить не удается, поэтому приходится использовать приближенные методы.

Решение нелинейных уравнений обычно состоит из двух этапов:

1) *Отделение*, или *локализация* корней, т.е. отыскание таких отрезков [a,b] (одного или нескольких), внутри которых имеется только один корень нелинейного уравнения.

Отделить корни в большинстве случаев можно графически. Для этого нужно построить график функции $y=f(x)$. Если $f(x)=f_1(x)-f_2(x)$, то построив графики функций $y=f_1(x)$ и $y=f_2(x)$, найдем приближенное значение корня как точку пересечения этих графиков. Так, для уравнения $\cos x - x = 0$ имеем:





Как видно из рисунков, корень рассматриваемого уравнения локализован на отрезке [a,b]=[0.6,0.8]. Графики построены приводимой ниже программой с использованием сторонних модулей *Numpy* и *Matplotlib*.

```
import numpy as np
import matplotlib.pyplot as plt

x=np.linspace(0,1,100)
y=np.cos(x)-x
plt.plot(x,y)
plt.xlabel("x")
plt.ylabel("y")
plt.title('График функции y=cos(x)-x')
plt.grid(True)
plt.axis([0,1,-0.6,1.2])
plt.savefig("f(x).png")
plt.show()
```

```
y1=np.cos(x)
y2=x
plt.plot(x,y1,x,y2)
plt.xlabel("x")
plt.ylabel("y")
plt.title('Графики функций y=cos(x) и y=x')
plt.legend(['cos(x)','x'], shadow=True)
plt.grid(True)
plt.axis([0,1,0,1.2])
plt.savefig("f1f2(x).png")
plt.show()
```

Локализовать корень уравнения можно также программно, вычисляя функцию $f(x)$ для значений x, изменяющихся с некоторым заданным шагом h. При этом отыскиваются два таких соседних значения x, для которых $f(x)$ имеет противоположные знаки.

2) *Уточнение* приближенного значения корня до заданной точности.

Для уточнения корня нелинейного уравнения до заданной точности можно воспользоваться несколькими численными методами.

Метод половинного деления (МПД) состоит в построении последовательности вложенных отрезков, на концах которых функция принимает значения разных знаков. Каждый последующий отрезок получают делением предыдущего пополам. Процесс построения последовательности отрезков позволяет найти корень уравнения $f(x)=0$ с любой заданной точностью.

Алгоритм

1) Проверим, правильно ли заданы исходные данные: если F(A)*F(B)>0 или A>B, то полагаем IER=1, печатаем соответствующее сообщение и проводим аварийную остановку.

2) Зададим текущие границы отрезка, содержащего корень A1=A, B1=B; зададим начальное число итераций N=0 и IER=0.

4) Вычислим середину отрезка [A1,B1]: C=0.5*(A1+B1); изменим счетчик итераций на единицу: N=N+1.

5) Если длина текущего отрезка [A1,B1] меньше заданной точности EPS, то возвращаемся в вызывающую программу.

6) Проверим, внутри какого из отрезков [A1,C] или [C,B1] лежит корень уравнения. Если корень лежит внутри [A1,C], то изменяем правую текущую границу отрезка, содержащего корень, в противном случае – левую: если F(A1)* F(C)<0, то B1=C, иначе A1=C; итерационный процесс повторяется с пункта 4.

В качестве параметров функции МПД можно выбрать следующие.

Входные:

F – имя функции $f(x)$;

A, B – соответственно левая и правая граница отрезка локализации [a,b];

EPS – точность вычисления корня.

Выходные:

C – корень уравнения (если он найден);

N – количество итераций, которое потребовалось выполнить для вычисления корня с заданной точностью;

IER – код ошибки: IER=0, если корень найден, IER=1, если A>B или на отрезке [A,B] нет корня, т.е. если F(A)*F(B)>0.

Метод Ньютона (метод касательных) является одним из наиболее эффективных методов нахождения корней нелинейных уравнений. Он состоит в построении итерационной последовательности

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i = 0,1,2,...$$

сходящейся к корню уравнения $f(x)=0$, если $f(x_0)f''(x_0) > 0$, $x_0 \in$[a,b]. Геометрическая интерпретация метода следующая: если через точку графика (x_i, $f(x_i)$) провести касательную, то точка пересечения этой касательной с осью абсцисс принимается за уточненное значение корня x_{i+1}. Метод Ньютона особенно эффективен, если известно хорошее начальное приближение x_0 к корню и в окрестности корня график функции имеет большую крутизну.

Алгоритм

1) Зададим: начальное значение количества итераций N=0; текущее значение корня X=X0.

2) Выполним итерационный цикл:

изменим содержимое счетчика итераций на единицу N=N+1;

если количество итераций сравнялось с NMAX, то положим IER=1 и осуществим возврат в вызывающую программу;

вычислим уточненное значение корня по формуле Y=X-F(X)/PF(X);

вычислим модуль разности между уточненным и текущим значениями корня E=ABS(Y-X);

проверим, следует ли продолжать уточнение корня: если E>=EPS, то положим X=Y и повторим итерационный цикл с пункта 2.

3) Если точность вычисления корня достигнута, то присвоим IER=0 и осуществим возврат в вызывающую программу.

В качестве параметров функции, реализующей метод Ньютона, можно рекомендовать следующие.

Входные:

F – имя функции f(x);

PF – имя функции f '(x);

X0 – начальное приближение для корня уравнения;

EPS – точность вычисления корня;

NMAX – максимальное количество итераций.

Выходные:

X – корень уравнения;

N – количество итераций, выполненных для вычисления корня с заданной точностью;

IER – код ошибки. Если корень найден, то IER=0; если корень не найден за NMAX итераций, то IER=1.

Метод секущих является модификацией метода Ньютона, в формуле которого при этом делается замена

$$f'(x_i) \approx [f(x_i + h) - f(x_i)]/h \quad (h \text{ – малая величина}),$$

и состоит в построении итерационной последовательности

$$x_{i+1} = x_i - \frac{hf(x_i)}{f(x_i + h) - f(x_i)}, \quad i = 0,1,2, \ldots$$

Условия сходимости и алгоритм метода, а также параметры функции, реализующей метод, аналогичны таковым метода Ньютона.

Метод простой итерации (МПИ) состоит в построении итерационной последовательности

$$x_{i+1} = \varphi(x_i), \quad i = 0,1,2, \ldots,$$

сходящейся к корню, если x_0 принадлежит отрезку локализации [a,b] и функция $\varphi(x)$ обладает на [a,b] свойством $|\varphi'(x)| < 1$. Функция $\varphi(x)$ может быть построена по универсальной формуле

$$\varphi(x) = x - \tau f(x),$$

где τ – малая по модулю величина, значение которой, обеспечивающее сходимость, либо подбирается вручную, либо может быть задано по формуле

$$\tau = 2\text{sign}(f'(x))/(m + M),$$

где $\text{sign}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases}$, $m = \min_{a \le x \le b} |f'(x)|$, $M = \max_{a \le x \le b} |f'(x)|$. В отдельных случаях $\varphi(x)$ может быть получена преобразованием исходного уравнения относительно переменной x. При этом обязательна проверка условия $|\varphi'(x)| < 1$. Алгоритм метода и параметры функции, реализующей данный метод, аналогичны таковым метода Ньютона.

Задание

Провести локализацию корней нелинейного уравнения f(x)=0 графическим способом с использованием модулей *Numpy* и *Matplotlib*. Используя заданные численные методы, уточнить корень нелинейного уравнения. Функцию f(x) и номера численных методов выбрать из таблицы по номеру варианта. Найденные значения корня сравнить с результатами, полученными с помощью модуля *Scipy.optimize*, а также в *Exel* и *Mathcad*.

Варианты заданий

№ варианта	Левая часть уравнения f(x)	№ методов	№ варианта	Левая часть уравнения f(x)	№ методов
1	x+sqrt(x)+x**(1/3)-2.5	1,2	2	cos(2/x)-2*sin(1/x)+1/x	1,3
3	cos(x)-exp(-x**2/2)+x-1	1,4	4	sin(x**2)+cos(x**2)-10*x	2,3
5	tg(x/2)-ctg(x/2)+x	2,3	6	x-sin(2x)-.25	3,4
7	sqrt(x)-cos(.387*x)	1,2	8	x**2+4*sin(2x) - 2	1,3

9	1.8*x**2-sin(10*x) - 1	1,4	10	3*x-cos(4x)-1	2,3
11	2*x-ln(x)-7	2,3	12	x**2*cos(2x)-1	3,4
13	sin(x+pi/3)-.5*x	1,2	14	cos(x+pi/8)-x**3	1,3
15	2*arctg(x)-3*x+2	1,4	16	4*x-cos(3x)-1	2,3
17	(x-3)*cos(2x)-1	2,3	18	sqrt(1-x)-tg(2x)	3,4
19	ctg(1.05*x)-x**2	1,2	20	$tg(1.5773*x^2)-2.3041*x$	1,3
21	$ln(7.6221*x)-8.591*x^3+1.5$	1,4	22	$9.33*sin(6.977*x^4)-7.25*x^2$	2,3
23	$e^x-6*x-3$	2,3	24	$x^2-cos^2(\pi x)$	3,4
25	$e^x-1+1.5*x^3$	1,2	26	$0.1e^x-sin^2x+0.5$	1,3
27	$(x+1)^{1/2}-1/x$	1,4	28	2*x+lg x+0.5	2,3

Численные методы по номеру: 1 – МПД, 2 – метод Ньютона, 3 – МПИ, 4 – метод секущих.

Указания к выполнению

1) В основной программе построить график функции y=f(x) для того, чтобы определить количество корней и отрезки, где они расположены. Уравнение может иметь один, несколько или бесчисленное множество корней. Необходимо решить с преподавателем, какой корень подлежат определению.

2) Написать и отладить необходимые для расчетов программы:

- функции для вычисления f(x), f '(x);
- функции для локализации корней численными методами;

Предусмотреть печать не только окончательных, но и промежуточных результатов (отрезков [A,B], числа итераций, кодов ошибок и т.д.).

3) Провести расчеты с помощью Питон-программ пункта 2.

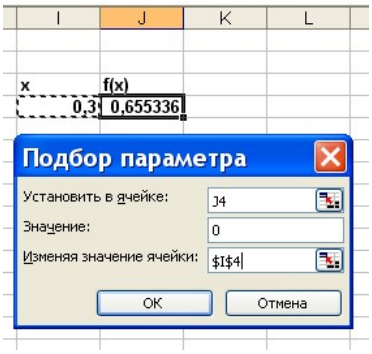
4) Провести расчеты с помощью модуля *Scipy.optimize*. В приводимом ниже примере для уточнения корня используется четыре метода. После имени метода на экран печатаются найденный корень, невязка уравнения и количество выполненных итераций.

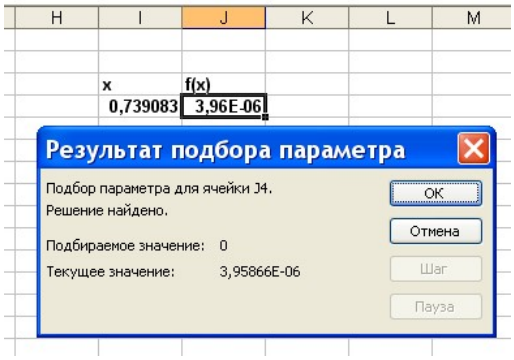
```
import numpy as np
from scipy import optimize

def f(x):
   return np.cos(x)-x
def df(x):
   return -np.sin(x)-1

a=0.6; b=0.8
sol = optimize.root_scalar(f, bracket=[a, b], method='brentq')
print('brentq:',sol.root, '%9.3e' % (f(sol.root)), sol.iterations)
sol = optimize.root_scalar(f, bracket=[a, b], method='bisect')
print('bisect:',sol.root, '%9.3e' % (f(sol.root)), sol.iterations)
sol = optimize.root_scalar(f, bracket=[a, b], method='ridder')
print('ridder:',sol.root, '%9.3e' % (f(sol.root)), sol.iterations)
sol = optimize.root_scalar(f, x0=0.2, fprime=df, method='newton')
print('newton:',sol.root, '%9.3e' % (f(sol.root)), sol.iterations)
```

```
brentq: 0.7390851332151579 4.552e-15 6
bisect: 0.7390851332151214 6.573e-14 37
ridder: 0.7390851332141972 1.612e-12 4
newton: 0.7390851332151607 0.000e+00 5
```

5) Провести расчет корня в *Excel*. Для этого задать в пустой ячейке числовое значение начального приближения к корню x_0 (для рассмотренного выше примера ввести «0,3») в соседней ячейке создать формулу расчета функции f(x), причем аргументом формулы задать

ссылку на ячейку с x_0. Если ячейка аргумента «I4», то для примера в ячейке «J4» написать «=cos(I4)-I4». Затем вызвать команду «Сервис->Подбор параметра» и заполнить поля: в поле «Установить в ячейке» задать ссылку на ячейку с f(x), в поле «Значение» задать «0», в поле «Изменяя значение ячейки» – ссылку на ячейку аргумента. Нажать «ОК». Появится окно «Результат подбора параметра», а в ячейке аргумента – приближенное значение корня.





6) Провести расчет корня в *Mathcad*. Для рассматриваемого примера:

$f(x) := \cos(x) - x$

$x := 1$

$z := \text{root}(f(x), x)$ $z = 0.7390851334$

7) Полученные результаты оформить в виде таблицы. Сравнить результаты Питон-программы со значениями корня, полученными с помощью *Scipy.optimize* (четырьмя методами), *Excel* и *Mathcad*, и сделать выводы.