6. Нелинейные уравнения и системы в SCILAB

Если нелинейное уравнение достаточно сложное, то отыскание его корней процесс нетривиальный. Рассмотрим, какими средствами обладает `Scilab` для решения этой задачи.

6.1. Решение нелинейных уравнений

В общем случае аналитическое решение уравнения *f(x)=0* можно найти только для узкого класса функций. Чаще всего приходится решать это уравнение численными методами. *Численное решение уравнения* проводят в два этапа:

- *отделяют корни уравнения*, т.е. находят достаточно тесные промежутки, в которых содержится только один корень, эти промежутки называют *интервалами изоляции корня*, определить их можно, изобразив график функции или любым другим методом основанным на том, что непрерывная функции $f(x)$ имеет на интервале $[a,b]$ хотя бы один корень, если она поменяла знак $f(a)\cdot f(b)<0$, a и b называют *пределами интервала изоляции*;
- на втором этапе проводят *уточнение отделенных корней*, т.е. находят корни с заданной точностью.

6.2.1. Алгебраические уравнения

Любое уравнение $P(x)=0$, где $P(x)$ это многочлен, отличный от нулевого, называется *алгебраическим уравнением* (*полиномом* относительно переменных x. Всякое алгебраическое уравнение относительно x можно записать в виде

$$a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=0,$$

где $a_0\neq0$, $n\geq1$ и a_i – *коэффициенты* алгебраического уравнения n–й степени. Например, *линейное уравнение* это алгебраическое уравнение первой степени, *квадратное* – второй, *кубическое* – третьей и так далее.

Для определения алгебраического уравнения в `Scilab` существует функция

```
poly(a,"x", ["fl"]),
```

где `a` это число или матрица чисел, `x` – символьная переменная, `fl` – строковая переменная, определяющая способ задания полинома. Строковая переменная `fl` может принимать только два значения – "`roots`" или "`coeff`" (соответственно "`r`" или "`c`"). Если `fl= c`, то будет сформирован полином с коэффициентами, хранящимися в параметре `a`. Если же `fl= r`, то значения параметра a воспринимаются функцией как корни, для которых необходимо рассчитать коэффициенты соответствующего полинома. По умолчанию `fl= r`.

Рассмотрим несколько примеров формирования полиномов.

Листинг 6.1 отражает создание полинома `p`, имеющего в качестве корня тройку, и

полинома `f` с коэффициентом 3.

```
-->p=poly(3,'x','r');
-->f=poly(3,'x','c');
-->p
 p  =
  - 3 + x
-->f
 f  =
    3
```

Листинг 6.1.

На листинге 6.2 приведены примеры создания более сложных полиномов.

```
-->poly([1 0 2],'x')
 ans  =
           2
    2x - 3x + x
-->poly([1 0 2],'x','c')
 ans  =
          2
    1 + 2x
-->poly([-2 2],'x')
 ans  =
         2
  - 4 + x
```

Листинг 6.2

Листинг 6.3 содержит примеры операций с полиномами.

```
-->p1=poly([1 -2],'x','c')
 p1  =
    1 - 2x
-->p2=poly([3 -2],'x','c')
 p2  =
    3 - 2x
-->p1/p2
 ans  =
    1 - 2x
    ------
    3 - 2x
-->p1*p2
 ans  =
                2
    3 - 8x + 4x
```

Листинг 6.3

Решим несколько алгебраических уравнений.

ЗАДАЧА 6.1. Найти корни полинома $2x^4-8x^3+8x^2-1=0$.

Для решения этой задачи необходимо задать полином `p`. Сделаем это при помощи функции `poly`, предварительно определив вектор коэффициентов `V`. Обратите внимание, что в уравнении отсутствует переменная x в первой степени, это означает, что соответствующий коэффициент равен нулю. Отыскание корней полинома при помощи функции `roots(p)` приведено в листинге 6.4. Графическое решение, показанное на рис. 6.1 позволяет убедится,

что корни найдены верно.

```
-->V=[-1 0 8 -8 2];
-->p=poly(V,'x','c')
 p  =
                2    3    4
  - 1 + 8x - 8x + 2x
-->X=roots(p)
 X  =

!   0.4588039 !
! - 0.3065630 !
!   1.5411961 !
!   2.306563  !
```

Листинг 6.4

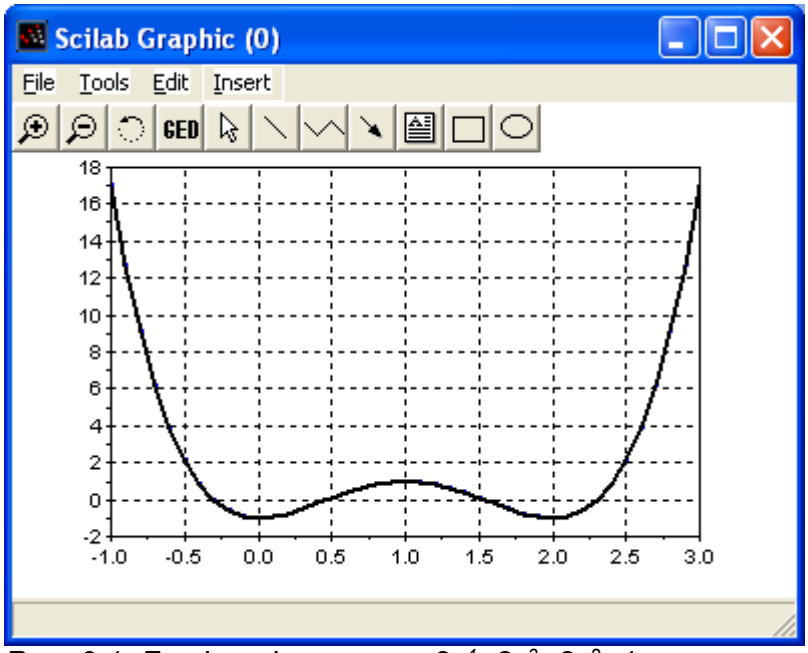

Рис. 6.1. График функции $y=2x^4-8x^3+8x^2-1$

ЗАДАЧА 6.2. Найти корни полинома $x^3+0.4x^2+0.6x-1=0$.

Решение этой задачи аналогично решению предыдущей. Разница заключается в способе вызова необходимых для этого функций. Не трудно заметить (листинг 6.5), что полином имеет один действительный и два комплексных корня, в отличии от полинома из задачи 6.1, в котором все корни действительные. Рис. 6.2 подтверждает это.

```
-->roots(poly([-1 0.6 0.4 1],'x','c'))
 ans  =
!   0.7153636               !
! - 0.5576818 + 1.0425361i !
! - 0.5576818 - 1.0425361i !
```

Листинг 6.5

ЗАДАЧА 6.3. Найти решение уравнения $y(x)=0$, если $y(x)=x^4-18x^2+6$.

Листинг 6.6 демонстрирует решение этой задачи. Обратите внимание на способ определения полинома.

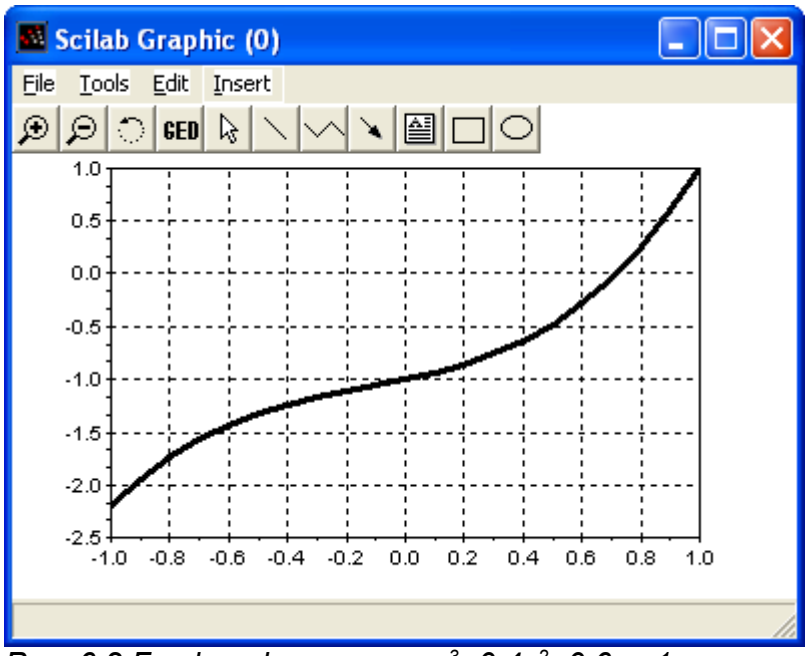

Рис. 6.2.График функции y= x^3+0.4x^2+0.6x –1

```
-->x=poly(0,'x');
-->y=x^4-18*x^2+.6;
-->roots(y)
 ans  =
!   0.1827438 !
! - 0.1827438 !
! - 4.2387032 !
!   4.2387032 !
```

Листинг 6.6

В Scilab существует функция fsolve(x0,f), которую так же можно применить для решения алгебраических уравнений. Подробно эта функция будет описана далее, так как ее можно использовать для решения нелинейных уравнений, отличных от алгебраических, и для решения систем линейных и нелинейных уравнений.

ЗАДАЧА 6.4. Найти решение уравнения $y(x)=0$, если $y(x)=x^5-x^3+1$.

Решим эту задачу при помощи функции fsolve(x0,f), где x0 – начальное приближение, f – функция, описывающая левую часть уравнения $y(x)=0$. Листинг 6.7 содержит ход решения задачи. В первой строке происходит определение функции y(x) в виде исходного полинома. Во второй, вызывается команда fsolve(-2,y), для отыскания корней функции y. В качестве начального приближения задано число –2, так как не трудно определить (рис. 6.3), что полином имеет единственный действительный корень, в интервале от –2 до –1.

```
-->deff('[f]=y(x)','f=x^5-x^3+1')
-->X=fsolve(-2,y)
 X  =
  - 1.2365057
```

Листинг 6.7

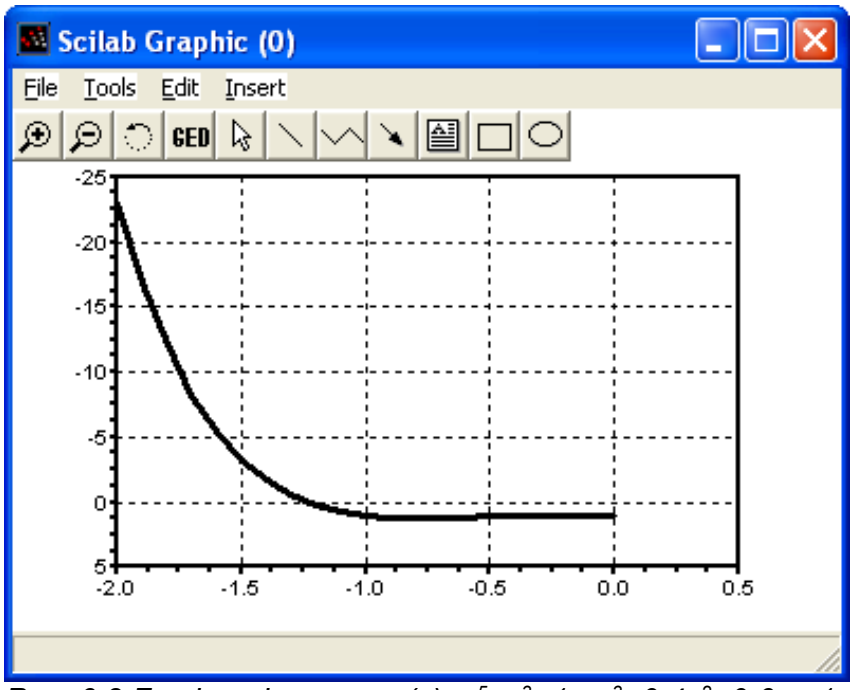

Рис. 6.3.График функции y(x)=x⁵–x³+1= x³+0.4x²+0.6x –1

Заметим, что заданное уравнение, кроме действительно корня, имеет и мнимые. Для отыскания всех корней полинома используйте функцию roots (листинг 6.8).

```
-->roots(poly([1 0 0 -1 0 1],'x','c'))
 ans  =
!   0.9590477 + 0.4283660i !
!   0.9590477 - 0.4283660i !
! - 0.3407949 + 0.7854231i !
! - 0.3407949 - 0.7854231i !
! - 1.2365057              !
```

Листинг 6.8

Далее будет рассмотрено применение функции `fsolve` для решения неалгебраических уравнений.

6.2.2. Трансцендентные уравнения

Уравнение, в котором неизвестное входит в аргумент трансцендентных функций, называется *трансцендентным уравнением*. К трансцендентным уравнениям принадлежат показательные, логарифмические, тригонометрические.

Рассмотрим применение функции `fsolve` для решения трансцендентных уравнений.

ЗАДАЧА 6.5. Найти решение уравнения: $\sqrt[3]{(x-1)^2}-\sqrt[3]{x^2}=0$.

Выражение, стоящее в правой части уравнения можно представить в виде разности двух функций $f(x)$–$g(x)$=0, где $f(x)=\sqrt[3]{(x-1)^2}$, $g(x)=\sqrt[3]{x^2}$. Тогда решение задачи будет выглядеть, так как показано в листинге 6.9. В качестве приближенного корня был выбран ноль, т.к. на рис. 6.4 видно, абсцисса точки пересечения линий $f(x)$ и $g(x)$ лежит в интервале [0;1].

```
-->deff('[y]=f1(x)','y1=((x-1)^2)^(1/3),y2=(x^2)^(1/3),y=y1-y2')
-->fsolve(0,f1)
 ans  =
    0.5
```

Листинг 6.9

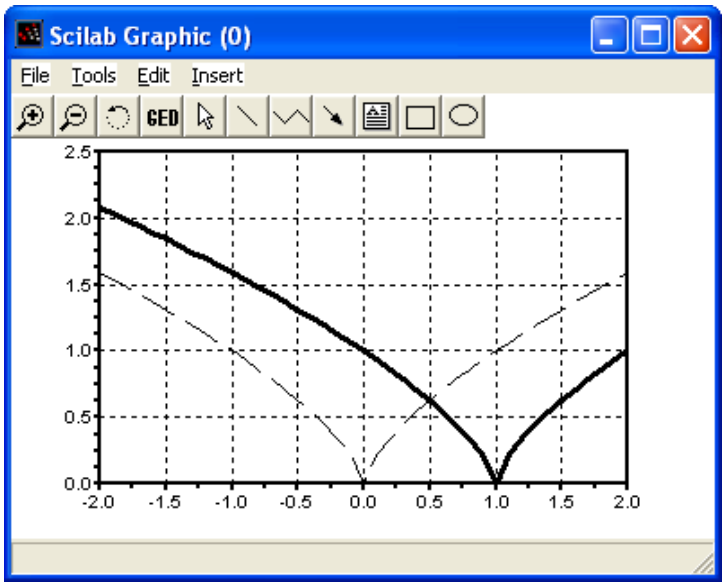

Рис. 6.4. График функций $f(x)=\sqrt[3]{(x-1)^2}$ *и* $g(x)=\sqrt[3]{x^2}$

Рис. 6.4.

ЗАДАЧА 6.6. Найти корни уравнения $f(x)=e^x/5-2(x-1)^2$.

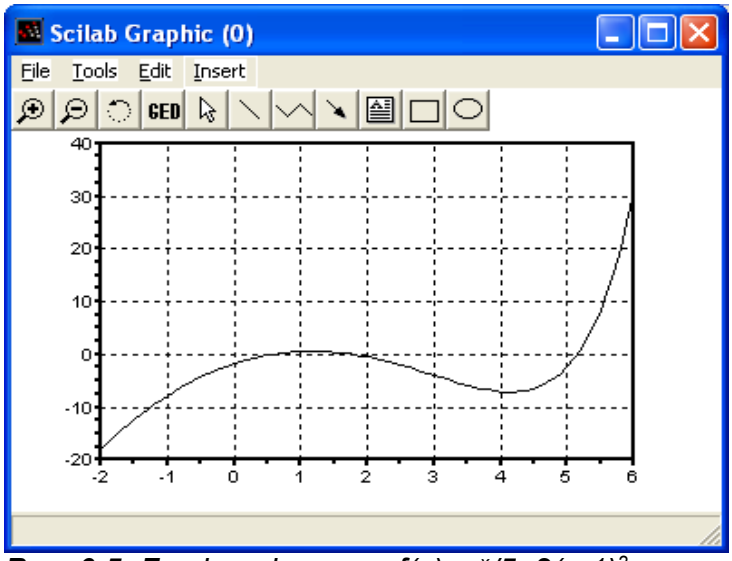

Рис. 6.5: График функции $f(x)=e^x/5-2(x-1)^2$

На рис. 6. 5 видно, что график функции $f(x)$ трижды пересекает ось абсцисс, то есть уравнение имеет три корня. Последовательно вызывая функцию `fsolve` с различными начальными приближениями, так как показано в листинге 6.10, получим все решения заданного уравнения.

```
-->deff('[y]=f(x)','y=exp(x)/5-2*(x-1)^2')
-->x(1)=fsolve(0,f);
-->x(2)=fsolve(2,f);
-->x(3)=fsolve(5,f);
```

```
-->x
 x  =
!   0.5778406 !
!   1.7638701 !
!   5.1476865 !
```

Листинг 6.10

Кроме того, начальные приближения можно задать в виде вектора и тогда функцию можно вызвать один раз (листинг 6.11).

```
-->fsolve([0;2;5],f)
 ans  =
!   0.5778406 !
!   1.7638701 !
!   5.1476865 !
```

Листинг 6.11

ЗАДАЧА 6.7. Вычислить корни уравнения $\sin(x)-0.4x=0$ в диапазоне $[-5\pi;5\pi]$.

Решение задачи приведено в листинге 6.12.

```
-->deff('[y]=fff(x)','y=-0.4+sin(x)')
-->V=[-5*%pi:%pi:5*%pi];
-->X=fsolve(V,fff);
-->X
 X  =
!-16.11948 -12.154854 -9.8362948 -5.8716685 -3.5531095 0.4115168
2.7300758 6.6947022 9.0132611 12.977887 15.296446!
```

Листинг 6.12

6.2.3. Системы уравнений

Если заданы *m* уравнений с *n* неизвестными и требуется найти последовательность из *n* чисел, которые одновременно удовлетворяют каждому из *m* уравнений, то говорят о *системе уравнений*. Для решения систем уравнений в Scilab так же применяют функцию fsolve(x0,f).

ЗАДАЧА 6.8. Решить систему уравнений: $\{x^2+y^2=1; x^3-y=0\}$.

Графическое решение системы (рис. 6.6) показывает, что она имеет две пары корней. Окружность и гипербола пересекаются в точках [0.8;0.6] и [-0.8;-0.6]. Эти значения приблизительны. Для того чтобы уточнить их, применим функцию fsolve, предварительно определив систему с помощью файл–функции (листинг 6.13).

```
function [y]=fun(x)
y(1)=x(1)^2+x(2)^2-1;
y(2)=x(1)^3-x(2);
endfunction
-->exec('C:\Program Files\scilab-4.0-rc1\bin\fun.sce');
  disp('exec done');
 exec done
-->fsolve([0.5 0.5],fun)
 ans  =
    0.8260314    0.5636242
```

```
-->fsolve([-0.5 -0.5],fun)
 ans  =
  - 0.8260314  - 0.5636242
```

Листинг 6.13

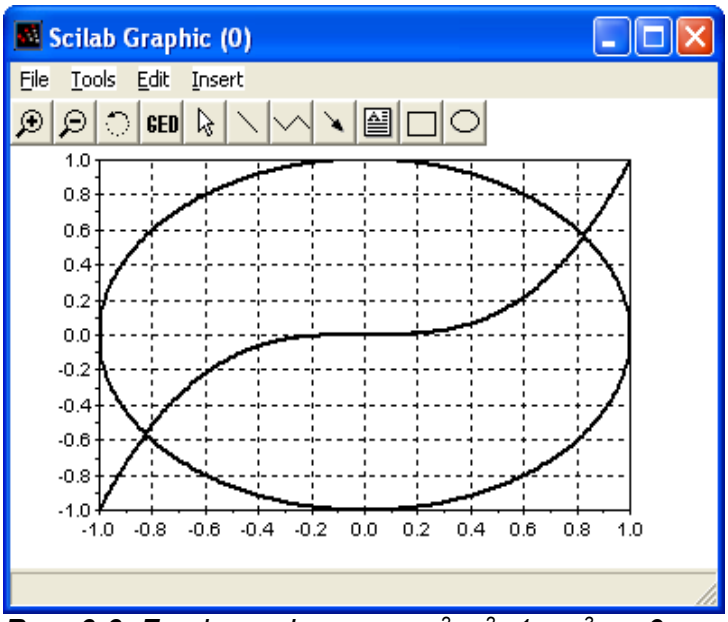

Рис. 6.6. Графики функции $x^2+y^2=1$ и $x^3-y=0$.

ЗАДАЧА 6.9. В данной задаче исследуется система из трех нелинейных уравнений с тремя неизвестными (листинг 6.13).

```
function [y]=fun(x)
y(1)=x(1)^2+x(2)^2+x(3)^2-1
y(2)=2*x(1)^2+x(2)^2-4*x(3)
y(3)=3*x(1)^2-4*x(2)+x(3)^2
endfunction
-->exec('D:\scilab 3\fun');disp('exec done');//вызов функции
 exec done
-->fsolve([0.5 0.5 0.5],fun)//решение системы
ans  =
!   0.7851969    0.4966114    0.3699228 !
```

Листинг 6.14