

Теория

Для решения трансцендентных уравнений наибольшую популярность имеет метод дихотомии. Рассмотрим основные условия его применимости:

- 1) Известен числовой отрезок $[L, R]$ содержащий ровно 1 корень (график функции $y=f(x)$, соответствующей уравнению $f(x)=0$, имеет точку на оси абсцисс $X_r \in [L, R]$);
- 2) На всём отрезке $[L, R]$ функция определена;
- 3) В точке X_r функция $y=f(x)$ меняет знак.

Если корень есть, а функция не меняет знак, то это точка ее экстремума, где график КАСАЕТСЯ оси абсцисс, но не пересекает ее. В этом случае алгоритм поиска корня не работает.

Для решения проблемы «корня в точке экстремума» переходят от функции $y=f(x)$ к ее производной $y = \frac{df(x)}{dx}$, поскольку известно, что в точке экстремума производная функции равна нулю и имеет противоположные знаки слева и справа по оси абсцисс от точки экстремума(!), что и надо.

В случае, когда расположение каждого из корней уравнения неизвестно применяют метод простых итераций (используется также и для решения систем уравнений). Рассмотрим его детально.

Пусть есть уравнение вида: $f_1(x)=f_2(x)$. Для построения итерационной схемы перейдем к следующему виду этого же уравнения: $x = f_1(x)-f_2(x)+x$ и обозначим аргумент в левой части x_{new} , а в правой части x_{old} : $x_{new} = f_1(x_{old})-f_2(x_{old})+x_{old}$. В цикле последовательно вычисляется новое значение переменной x через старое (предыдущее) до тех пор, пока относительная разность старого и нового значений не станет меньше заданной погрешности:

Пока заданная точность не достигнута (цикл):

- 1) Задаем новое значение аргумента правой части равным результату предыдущего шага:

$$x_{old} = x_{new}$$

или же как среднее арифметическое вида (так называемая «неявная схема»):

$$x_{old} = (x_{new} + x_{old})/2 \text{ (возможны и другие варианты).}$$

Считается, что второй вариант имеет лучшую сходимость (чаще удастся найти корень), чем первый. Обычно это так и есть, но при этом количество шагов приближения к корню существенно возрастает.

- 2) Вычисляем $x_{new} = f_1(x_{old})-f_2(x_{old})+x_{old}$ (обычно правую часть оформляют как функцию)

Конец цикла.

В зависимости от задаваемого начального значения аргумента метод может сойтись к разным корням уравнения или же не сойтись вовсе. Редко когда удастся найти все корни уравнения – разве что корень всего один! В любом случае для сходимости необходимо, чтобы функция справа в п.2 алгоритма изменялась не быстрее, чем линейная функция вида $y=x$.

Ниже на примере полинома 4-й степени показаны некоторые технологические приемы для решения трансцендентных уравнений с плохой сходимостью при поиске корней методом простых итераций.

Использование метода простых итераций для решения уравнений с плохой сходимостью

Рассмотрим уравнение $x^4+a \cdot x^3+b \cdot x^2+c \cdot x+d=0$. Представим его как функцию: $y=f(x)=x^4+a \cdot x^3+b \cdot x^2+c \cdot x+d$.

- 1) $x_{new} = f(x_{old})+x_{old}$. Этот метод имеет плохую сходимость, поскольку $f(x)$ растет со скоростью полинома 4-й степени. Найти решение возможно, если корень вблизи нуля и коэффициенты a, b, c, d порядка единицы. Здесь можно пробовать обычную или же неявную схему (см. п.1 алгоритма).

- 2) Можно выразить аргумент, перенеся в левую часть уравнения любой член полинома, но реально это может быть только компонент нечетной степени, чтобы не потерять отрицательные корни:

$$-a \cdot x^3 = x^4 + b \cdot x^2 + c \cdot x + d \Rightarrow x = -\sqrt[3]{\frac{x^4 + b \cdot x^2 + c \cdot x + d}{a}} \Rightarrow x_{new} = -\sqrt[3]{\frac{x_{old}^4 + b \cdot x_{old}^2 + c \cdot x_{old} + d}{a}}$$

В этом случае выражение справа изменяется гораздо медленнее, чем полином 4-й степени, (степень полинома равна 4/3), а потому сходимость метода итераций намного выше.

- 3) Для еще большего повышения устойчивости алгоритма сделаем степень полинома нечетной домножим уравнение на x : $x \cdot (x^4+a \cdot x^3+b \cdot x^2+c \cdot x+d)=0 \Rightarrow x^5+x \cdot (a \cdot x^3+b \cdot x^2+c \cdot x+d)=0$ тогда:

$$x = -\sqrt[5]{x \cdot (a \cdot x^3 + b \cdot x^2 + c \cdot x + d)} \Rightarrow x_{new} = -\sqrt[5]{x \cdot (a \cdot x_{old}^3 + b \cdot x_{old}^2 + c \cdot x_{old} + d)}$$

(следует иметь ввиду, что здесь добавляется лишнее решение $x=0$, которое нужно исключить).

Покаже все описанные методы ниже в виде фрагментов программ.

Реализации

Метод дихотомии (случай пересечения оси x в корне и касания оси x)

```
function dix(L,R,e:real; f:function(x:real):real):(real,integer);
begin //L,R - границы, e - заданная погрешность, f - заданная функция
  result:=(-1.0 ,0); //функция возвращает кортеж: корень и количество итераций
  if f(L)*f(R)<0 then //проверка наличия корня
    while (R-L)/(abs(R)+abs(L))>e do begin //условие по относительной погрешности
      result:=(L+R)/2,result.item2+1; //вычисление середины и увеличение счетчика
      if f(result.item1)*f(L)>0 then //дихотомия
        L:=result.item1
      else
        R:=result.item1
    end;
  end;
end;
begin
  var t:=dix(0,0.7,0.000001,x->cos(x)-x); //функция передается  $\lambda$  - выражением
  //здесь рассматривается случай пересечения оси x графиком функции)
  if t.item2=0 then t:=dix(0,1,0.000001,x->-sin(x)-1);
  //здесь рассматривается случай касания графиком функции оси x - решаем через производную
  println(t); //выводится верное значение кортежа t
end.
```

Вариант на Java (то же самое уравнение)...

```
import static java.lang.Math.*;
public class Main {
  static int c=0; //счетчик
  public interface ff{double f(double x);} //интерфейс ff с абстрактной функцией f
  public static double dix(double L, double R, double e, ff t){
    c=0; double m;
    while ((R-L)/(abs(R)+abs(L))>e) {
      m=(R+L)/2; c+=1;
      if (t.f(L)*t.f(m)>0) L=m; else R=m;
    } return L;
  }
  //краткая справка по производным:  $(x^n)'=n \cdot x^{n-1}$ ,  $(\cos(x))'=-\sin(x)$ ,  $(\sin(x))'=\cos(x)$ 
  public static void main(String[] args) { //уравнение  $x=\cos(x) \rightarrow f(x)=\cos(x)-x$ 
    var t= dix(0, 1, 0.00001,x-> cos(x)-x); //функция для интерфейса задается как  $\lambda$ 
    if (c==0)t=dix(0, 1, 0.00001,x-> -sin(x)-1); //f(x)  $\rightarrow$  f'(x)=-sin(x)-1
    System.out.println(t+" "+c);
  }
}
```

Метод простых итераций (на примере полинома 4-й степени)

```
function pwr(a,b:real):=sign(a)*power(abs(a),b); //для нечетных корней (для четных power)
function iter(xo,e:real; f:function(x:real):real):(real,integer); //корень и счетчик
begin //xo-стартовое значение, e-заданная погрешность, f(x)-заданная функция
  result:=(xo ,0); //стартовая точка xold и счетчик
  var xnew:=xo+0.113; //начальное значение xnew (небольшое, но значимое отличие от xold)
  while (abs((abs(xo)-abs(xnew)))/(abs(xo)+abs(xnew))>e) and (result.item2<100000) do
  begin
    xo:=(xnew+xo)/2; //неявная схема
    //xo:=xnew; //явная схема
    xnew:=f(xo);
    result:=(xnew,result.item2+1);
  end;
end;
procedure roots(xx:real; f:function(x:real):real; k:integer);
begin //xx-расчетная точка, f - заданная функция и k - ее тип (вид зависимости)
  var t:=(0.0,0);
  t:=iter(xx,0.000000001,f); t:=(round(t.item1,1),t.item2); //корень с округлением
  if (t.item2<>100000) and not (t.item1 in r) and (t.item1<>real.NaN) and not
    real.IsInfinity(t.item1) then println(t,round(xx,3),k);
end; //попробуйте разобраться в этом наборе условий (опреатор If) самостоятельно (ДЗ)
begin //рассмотрены все 3 подхода, описанные в теории для уравнений с плохой сходимостью
  var a,b,c,d:real; //a,b,c,d - задаются любым способом
  roots(xx,x-> pwr(-x*(a*x*x*x+b*x*x+c*x+d),1.0/5),5); //...через корень 5-й степени (п.3)
  roots(xx,x-> pwr((x*x*x*x+b*x*x+c*x+d)/(-a),1.0/3),3); //...через корень 3-й степени (п.2)
  roots(xx,x-> x*x*x*x+a*x*x*x+b*x*x+c*x+d,x,1); //через линейную (п.1)
end. //вариант на Java предлагается написать самим пользуясь примером в дихотомии
```