

Конспект лекций к лабораторной работе по теме «Метод дихотомии»

Метод дихотомии (или метод половинного деления – иногда под ними подразумевают различные технологии, но обычно считают разными названиями одной и той же) используется для поиска корня уравнения $f(x)=0$, которое или не имеет аналитического решения (трансцендентное уравнение), или же вычисления при его поиске трудоемки, а итоговая их погрешность соизмерима с гораздо менее трудоемкой дихотомией. Перечислим условия применимости метода:

- известен отрезок $[L; R]$, где есть ровно один корень;
- в границах этого отрезка у функции нет разрывов;
- справа и слева от корня функция меняет знак см. рис.1:
Из рисунка видно, что можно принять $L=0, R=1$.

Этот пример трансцендентного уравнения позволяет легко описать все последующие шаги:

- 1) вычисляем середину отрезка $m=(L+R)/2$;
- 2) Проверяем знак произведения:
 - a. если $f(L)*f(m)>0$, то здесь корня нет и можно задать $L=m$
 - b. иначе корень есть, а значит зададим $R=m$

В любом случае отрезок поиска корня уменьшился вдвое.

- 3) Повторяем пункты 1-2 пока $R-L >$ заданной погрешности.

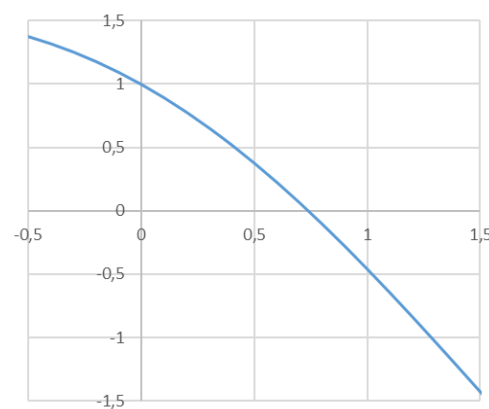


Рис.1. График $f(x)=\cos(x)-x$

Ниже представлен текст программы на языке PascalABC.net решения данной задачи:

```

1  ##
2  function f(x:real) :=cos(x)-x;
3  //L,R,m,e-координаты левой, правой границ и середины отрезка, e-погрешность
4  var (L,R,m,e) :=(0.0,1.0,0.0,0.0001); //числа указать вещественными!!!
5  var c:=0; //счетчик итераций
6  while R-L > e do begin
7      m:=(L+R)/2; c+=1;
8      if f(L)*f(m)>0 then L:=m else R:=m
9  end;
10 writeln('x=',L, ' ; c=',c);
    
```

Для данного случая выведется: $x=0.73907470703125; c=14$

Это трансцендентное уравнение имеет только численное решение. Количество итераций в методе известно заранее: $c = \log_2((R - L)/e) = \log_2(10^4) \cong 14$, но часто нужно использовать не абсолютную, а относительную погрешность и строка 5 программы примет вид: `while (R-L) / (abs(R)+abs(L)) > e do` В этом случае формула для количества итераций не работает (в данном примере получим $c=13$).

Рассмотрим случай, когда для уравнения существует аналитическое решение, где накопленная при вычислениях погрешность превышает погрешность дихотомии, например: $f(x)=x^4-2 \cdot x^3-3 \cdot x^2+4 \cdot x-0,9=0$

Для такого уравнения возможны от 0 до 4 корней и нужно найти отрезок $[L; R]$ для каждого из них. Построив для этого график функции в Excel (см. Рис 2) можно увидеть, что число корней ≥ 2 , а для определения их наличия и значений $[L; R]$ для отрезка $[0; 1]$ нужно рассмотреть его подробнее, изменив масштаб построения, причем как по X, так и по Y. Может оказаться также, что корни находятся ближе друг к другу, чем шаг по X, выбранный при вычислении таблицы для графика. Тогда для «подозрительного» участка строим отдельный график меньшим шагом - может оказаться, что здесь или нет корней (график ниже оси), или 1 корень (касается оси), или 2. В приведенном примере очевидные отрезки: $\{-2;-1\}, \{2;3\}$. Построив более детальный график (см. Рис.3.) увидим, что есть еще два отрезка: $\{0;0.5\}, \{0.5;1\}$, которые можно использовать для дихотомии. *Заметим, что если бы график касался X, то дихотомия была бы неприменима.*

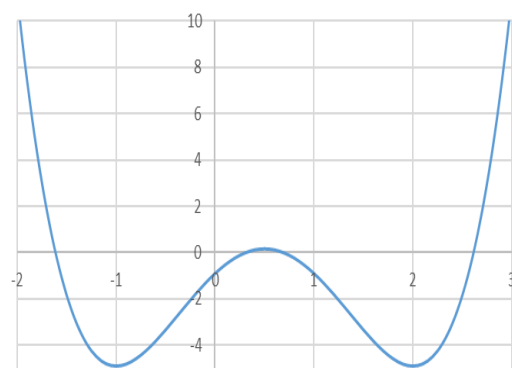


Рис. 2. График полинома 4-й степени

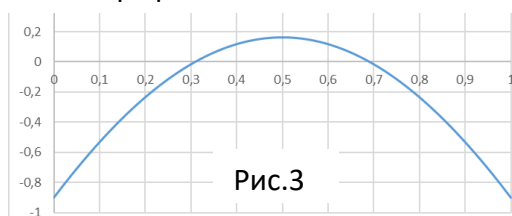


Рис.3

Конспект лекций к лабораторной работе по теме «Метод простых итераций»

В отличие от метода дихотомии, где известен отрезок поиска корня, в данном случае будем полагать, что ни количество, ни область существования корней нам не известны. Строго говоря, пружный метод также является итерационным, а потому выбранное название достаточно условно, а потому примем, что для данного метода важно лишь начальное значение аргумента (*стартовая точка*) и выбранный *алгоритм приближения к корню*. Обычно, если корень единственный, то всегда можно построить нужный итерационный процесс, но при наличии нескольких корней обычно удается найти лишь некоторые из них. Рассмотрим данный метод подробнее.

Пусть есть уравнение $f(x)=0$ и мы пытаемся привести его к виду $x_{i+1}=\varphi(x_i)$, где скорость изменения функции справа не быстрее линейной. Можно предположить, что в этом случае x_{i+1} находитесь ближе к корню уравнения, чем x_i . Как пример рассмотрим функцию $f(x)=\cos(x)-x$ (график – см.Рис.1.). Нас интересует $f(x)=0$, тогда можно записать итерационное уравнение: $x_{i+1}=\cos(x_i)$. Ниже представлен фрагмент компьютерной программы для метода простых итераций:

```
1  ##
2  function f(x:real):=cos(x);
3  var (xnew,xold,e,c):=(0.0,0.0,0.0001,0);
4  repeat
5      xold:=xnew; c+=1; xnew:=f(xold);
6  until (abs(xnew-xold)/(abs(xnew)+abs(xold))<e) or (c>1000);
7  if c=1001 then println('Процесс не сходится, корень не найден')
8  else writeln(' x=',xnew,'; c=',c);
```

Окно вывода

```
x=0.739130176529671; c=23
```

Корень найден, но по сравнению с методом дихотомии понадобилось больше шагов алгоритма. В то же время здесь не нужно знать границы поиска корня, и если взять в качестве стартовой точки $x=1000$, то нужная точность будет достигнута даже за меньшее число шагов: $c=22$.

Рассмотрим решение уравнения : $f(x)=x^4-2\cdot x^3-3\cdot x^2+4\cdot x-0,9=0$. По аналогии с прежним решением здесь следует записать итерационную формулу, хотя сделать это уже сложнее. Покажем это.

Добавим к обеим частям уравнения переменную x . Тогда получим: $x_{i+1}=x_i^4-2\cdot x_i^3-3\cdot x_i^2+5\cdot x_i-0,9$. Как уже было сказано, итерационный процесс будет сходящимся в том случае, если скорость изменения функции не выше линейной, а здесь полином четвертой степени! В этом случае можно найти корень на отрезке $[-1;1]$ с соответствующей стартовой точкой, но в нашем случае там нет корня.

Выразим x^4 через оставшуюся часть уравнения: $x_{i+1}=(2\cdot x_i^3+3\cdot x_i^2-4\cdot x_i+0,9)^{1/4}$. Здесь скорость роста правой части соответствует степени $\frac{3}{4}$, – медленнее линейной. При составлении программы здесь следует учитывать, что в процессе решения возможно отрицательное подкоренное выражение, а потому его нужно брать как абсолютное значение. Соответствующая функция имеет вид:

```
2  function f(x:real):real;
3  begin
4      var t:=2*x**3+3*x**2-4*x+0.9*x;
5      result:=t**0.2; if t<0 then result:=-result
6  end;
```

В этом случае найден один верный корень: 2.61 (см. Рис.2) при широком диапазоне значений стартовой точки. Другие положительные корни не найдены, а отрицательные в принципе не могли быть найдены, поскольку для корня четной степени в итерационном процессе это невозможно.

Домножив уравнение на x и применим предыдущую схему! Тогда: $x_{i+1}=(2\cdot x_i^4+3\cdot x_i^3-4\cdot x_i^2+0,9\cdot x_i)^{1/5}$. В этом случае можно найти и отрицательные корни, но, к сожалению, ни один язык программирования не позволяет выполнять извлечение корня из отрицательного числа даже в случае нечетной степени этого корня! Извлечем корень из абсолютного значения, добавив нужный знак:

```
2  function f(x:real):real;
3  begin
4      var t:=2*x**4+3*x**3-4*x**2+0.9*x;
5      result:=t**0.2; if t<0 then result:=-result
6  end;
```

Подобный подход дает возможность найти отрицательные корни в области сходимости итерационного процесса, хотя в нашем случае отрицательный корень найти и не удалось. В целом можно утверждать:

- метод простых итераций позволяет найти по крайней мере один корень уравнения при верном выборе итерационной схемы;
- для полиномов четных степеней удобно перейти к нечетной, домножив обе его части на аргумент, а затем используя алгоритм, включая извлечение корня, показанные выше.