Метод простых итераций

Одним из наиболее важных численных методов решения нелинейных уравнений является метод итераций. Сущность метода заключается в следующем.

Пусть функция f(x) монотонная на отрезке [a,b], причем выполнено условие: f(a)\*f(b)<0.

Заменим исходное нелинейное уравнение f(x)=0 эквивалентным ему уравнением вида: x=φ(x).

Пусть известно начальное приближение корня х=х0. Подставляя это значение в правую часть уравнения, получаем новое приближение: x1= φ(x0).

Затем аналогичным образом получим: x2 = φ(x1) .

Далее, подставляя каждый раз новое значение корня, получаем последовательность значений

xn+1 = φ(xn) , n=1,2, ... .

Итерационный процесс продолжается до тех пор, пока не станут близки результаты двух последовательных итераций:

.

Достаточным условием сходимости метода простых итераций является условие: | φl(x)| < 1,vвыполненное для любого x, принадлежащего некоторому отрезку [a,b], содержащему корень уравнения.

Рассмотрим геометрическую интерпретацию метода. Построим графики функций y=x и y= (x) . Корнем уравнения x=(x) является абсцисса точки пересечения кривой y= (x) с прямой y=x.

Из графиков видно, что при φl(x)>0 (а, б) и при φl(x)<0 (в, г) возможны как сходящиеся, так и расходящиеся итерационные процессы. Скорость сходимости зависит от абсолютной величины производной φl(x). Чем меньше |φl(x)| вблизи корня, тем быстрее сходится процесс

Итерационные процессы могут быть односторонними, если φl(x)>0 и двусторонними, если φl(x)<0.

**import** java.util.Scanner;

**public** **class** Number8 {

**public** **void** main(String [] args){

Scanner input = **new** Scanner(System.***in***);

System.***out***.println("x0=");

**double** x0 = input.nextDouble();

System.***out***.println("e=");

**double** e = input.nextDouble();

Scheht schet = **new** Scheht();

**double** x=schet.Schet(x0,e);

System.***out***.println(x);

}

}

**public** **class** Scheht {

**double** y;

**public** **double** Schet(**double** x, **double** e){

//введи в следующую строчку уравнение для решения

y=x;

**if**(Math.*abs*(y-x)>e){

Schet(y,e);

}

**return** y;

}

}