实用计算方法应急手册

陈国章 编著

天津科学技术出版社

实用计算方法应急手册

陈国章　编著

天津科学技术出版社

津新登字(90)003号

责任编辑:徐　彤

实用计算方法应急手册

陈国章　编著

*

天津科学技术出版社出版

天津市张自忠路189号　　邮编300020

天津市武清县振兴印刷厂印刷

新华书店天津发行所发行

*

开本787×1092毫米 1/16　印张11.25　字数267 000

1994年6月第1版

1994年6月第1次印刷

印数:1—3 500

ISBN 7-5308-1478-8

TP·45　定价8.50元

前　言

在人们从事教学、科研和编写计算机程序的过程中，几乎所有的程序设计人员都在为如何运用有效的经典算法，如何编写简洁而高效的算法程序而花费大量的时间和精力。如果能拥有一本《实用计算方法应急手册》，无疑将会节约大量的时间和精力。再有通过对本书的学习和使用又能进一步地提高您对常用算法的理解，开拓您的解题思路，缩短解题周期，减少重复劳动。

现在人类社会已经进入了计算机时代，作为计算机应用的一个重要方面——科学计算也正在迅速地发展。当今，能否熟练地运用计算机进行科学计算已经成为衡量科技人员实际水平的一项重要指标。在科学计算中所使用的计算方法是以计算机作为计算工具，以数学作为理论依据的，要提高运用计算机进行科学计算的能力，不应当将计算方法片面地理解为各种算法的简单罗列和堆积。同数学分析一样，它也是一门内容丰富，思想方法深刻而有着自身的理论体系的学科。

编写本书的指导思想是：兼顾算法的数学理论和实现程序，并偏重于程序。在本书中选编了有效的经典算法，以求解大量常见问题的算法为主。包括：方程求根，函数插值，数值积分，常微分方程的数值解法和线性方程组的解法几部分，共60个程序。在编写过程中，我们采用了目前市场上最为流行的Turbo C和Turbo Pascal语言，每个算法均包含六个内容：算法功能，算法简介，算法框图，算法中使用的变量说明，程序清单，算法的运行实例。

在本书的编写过程中，得到了南开大学管理系教授齐宣峰，河北农业大学基础课部教授张德培和陈文光的鼓励和帮助，编者对此表示深切的谢意。

由于时间仓促，加之编者的水平有限，因此，本书难免有不少的缺点和不当之处，恳切希望读者给予批评指正。

编　者

1993.5

目　录

第一篇　用 PASCAL 语言编写的实用算法

第二篇 用C语言编写的实用算法

第 一 篇

用 PASCAL 语言编写

的实用算法

第一章

第一章　方程求根

1.1　二　分　法

一、功能

求任意实函数方程 f(X)＝0 在自变量区间[a,b]上的全部实根。

二、算法简介

从 a 开始以一个基本步长 h 分隔，若某一步前后的函数值 y_0 与 y_1 异号或 y_1 等于零，则此区间中必有实根。把这个有根的区间记为$[a_1,b_1]$计算 $f((a_1+b_1)/2)$且与 y_0 比较，必能选出一个函数值异号的区间$[a_2,b_2]$再计算 $f((a_2+b_2)/2)$且与 y_0 比较，又能选出一个函数值异号的区间$[a_3,b_3]$重复上述二分区间的过程，直到区间的长度小于给定的精度要求，则认为求得一根。求出一根后继续分隔，重复上述步骤，直至求出[a,b]中全部实根为止。

三、框图

见图 1。

四、变量说明

a：实变量，区间左端点，输入参数。

b：实变量，区间右端点，输入参数。

h：实变量，逐步扫描分隔根的步长，输入参数。

ep：实变量，求得根的精度要求，输入参数。

m：整变量，m 不能小于 f(X)＝0 在[a,b]内根的个数，即 m＞＝n 的任意一个正整数，输入参数。

n：整变量，求得根的个数，返回参数。

rt：一维数组，rt[1]，rt[2]...rt[n]存放方程的根，返回参数。

五、过程清单

```
unit u _ erfen;
interface
  uses uf _ erfen;
```

```
type shuzu=array [1..20] of real;
procedure erfen(a,b,h,ep:real;var m,n:integer;var rt:shuzu);
implementation
procedure erfen(a,b,h,ep:real;var m,n:integer;var rt:shuzu);
  label
    1,2;
  var
    c,a1,b1,y,y0,y1,x:real;
  begin
    n:=0; c:=a;
    repeat
      y0:=f(c);
1:    c:=c+h;
      if  c<=b
        then
          begin
            y1:=f(c);
            if y1*y0>0
              then
                begin
                  y0:=y1;
                  goto 1
                end
              else
                begin

                  a1:=c-h; b1:=c; y0:=f(a1);
                  while b1-a1>ep do
                    begin
                      x:=(a1+b1)/2; y:=f(x);
                      if y*y0>0 then a1:=x;
                      if y*y0<0 then b1:=x
                      end
                      n:=n+1
                    end
                end
              else
                goto 2;
            rt[n]:=x
```

```
            until n>m;
        2:end;
    end.
```

六、例题

求方程 $x^3-3x-1=0$ 在区间[-8,8]内的全部实根。取 h=0.1,ep=10^{-5},m=3

```
program m - erfen;
  uses u - erfen;
  var
    a,b,h,ep:real;i,n,m:integer;rt:shuzu;
  begin
    writeln(' input data:');
    write(' a=');readln(a);
    write(' b=');readln(b);
    write(' h=');readln(h);
    write(' ep=');readln(ep);
    write(' m=');readln(m);
    erfen(a,b,h,ep,m,n,rt);
    writeln;
    writeln('result is:');
    for i:=1 to n do writeln(' x',i,'=',rt[i]:10:6)
  end.
unit uf - erfen;
interface
  function f(x:real):real;
implementation
  function f(x:real):real;
    begin
      f:=x*x*x-3*x-1
    end;
  end.
```

运行结果如下：

```
input data:
  a=-8
  b=8
  h=0.1
  ep=1e-5
  m=3
result is:
  x1=-1.532086
```

x2=－0.347296

x3= 1.879388

1.2 迭 代 法

一、功能

对较好的迭代初值，能求出收敛的实值函数方程 y=f(x)=0 的一个根。

二、算法简介

对于一般形式的方程 f(x)=0，我们先设法将它化为 x=g(x)的形式，如果给出根的某个近似值 x_k，将它代入 x=g(x)则有 $x_{(k+1)}=g(x_k)$这样，从给定的初始近似值 x_0 出发，按 $x_{(k+1)}=g(x_k)$形式不断迭代，得到一个数列 $x_0, x_1 \ldots x_k \ldots$ 如果这个数列有极限，则迭代格式 $x_{(k+1)}=g(x_k)$形式是收敛的，这时数列 x_k 当 k 趋于无穷时的极限就是方程 f(x)=0 的根。

三、框图

见图 2。

四、变量说明

x0：实变量，迭代初值，输入参数。

ep：实变量，求得根的精度要求，输入参数。

rt：一维数组，存放每一次迭代后的结果，返回参数。

n：整变量，记录迭代多少次后根满足精度，返回参数。

五、过程清单

```
unit u－die;
interface
  uses uf－die;
  type shuzu=array [0..50]of real;
  procedure die(var x0:real; ep:real; var rt:shuzu; var n:integer);
implementation
  procedure die(var x0:real; ep:real; var rt:shuzu; var n:integer);
    var
      x1:real;
      k:integer;
   begin
      n:0; rt[0]:=x0;
      for k:=1 to 50 do
        begin
          x1:=g(x0);
          rt[k]:=x1;
          n:=n+1;
          if abs(x1－x0)<ep
```

```
                then exit
                else
                  begin
                    x0:=x1
                  end;
            end;
        end;
end.
```

六、例题

求方程 $x=e^{-x}$ 在 $x=0.5$ 附近的一个根，要求精度 $ep=10^{-3}$。

```
program m-die(input,output);
  uses u-die;
  var
    x0,ep:real; rt:shuzu; i,n:integer;
  begin
    writeln('  input data ');
    write('  x0=');readln(x0);
    write('  ep=');readln(ep);
    die(x0,ep,rt,n);
    writeln;
    writeln('result is :');
    writeln('  ------------');
    writeln('  i     |    ','  x[i]');
    writeln('  ---|--------');
    for i:=0 to n do writeln(i:4,'      |   ',rt[i]:8:5);
    writeln('  ------------')
  end.
unit uf-die;
interface
  function g(x:real):real;
implementation
  function g(x:real):real;
    begin
      g:=exp(-x)
    end;
end.
```

运行结果如下：

```
input data
  x0=0.5
```

ep=1e-3

result is:

i	x[i]
0	0.50000
1	0.60653
2	0.54524
3	0.57970
4	0.56006
5	0.57117
6	0.56486
7	0.56844
8	0.56641
9	0.56756
10	0.56691

1.3 迭代加速法

一、功能

对较好的迭代初值，能求出收敛的实值函数方程 $f(x)=0$ 的一个实根。

二、算法简介

设用 $\bar{x}_{(k+1)}$ 表示近似值 x_k 经过一次迭代得到结果 $\bar{x}_{(k+1)}=g(x_k)$，则误差可表示为 $x^*-\bar{x}_{(k+1)}=g'(\xi)\cdot(x-x_k)$　其中 ξ 是 x^* 与 x_k 之间的某个点。假定 $g'(x)$ 在求根范围内改变不大，近似地取某个定值 q，由于收敛，自然要求 $|q|<1$，这样则有 $x^*-\bar{x}_{(k+1)}=q(x^*-x_k)$。就可得到 $x^*-\bar{x}_{(k+1)}=q/(1-q\cdot(\bar{x}_{(k+1)}-x_k)$，我们用 $q/1-q\cdot(\bar{x}_{(k+1)}-x_k)$ 作为计算结果 $\bar{x}_{(k+1)}$ 的一种补尝，可以期望所得到的 $x_{(k+1)}=\bar{x}_{(k+1)}+q/(1-q)\cdot(\bar{x}_{(k+1)}-x_k)=1/(1-q)\cdot\bar{x}_{(k+1)}-1/(1-q)\cdot x_k$ 是一个比 $\bar{x}_{(k+1)}$ 更好的结果。因此得到迭代加速公式

迭代：$\bar{x}_{(k+1)}=g(x_k)$

加速：$x_{(k+1)}=\bar{x}_{(k+1)}+q/(1-q)\cdot(\bar{x}_{(k+1)}-x_k)$

三、框图

见图 3。

四、变量说明

x0：实变量，迭代初值，输入参数。

ep：实变量，求得根的精度要求，输入参数。

rt:一维数组,rt[0],rt[1]…rt[n]存放每一次迭代结果,返回参数。

n:整变量,记录迭代多少次后根满足精度,返回参数。

五、过程清单

```
unit u - jdie;
interface
  uses uf - jdie;
  type shuzu=array [0..20]of real;
  procedure jdie(var x0:real; ep,q:real; var rt:shuzu; var n:interger);
implementation
  procedure jdie(var x0:real; ep,q:real; var rt:shuzu; var n:integer);
    var
      x,x1:real; k:integer;
    begin
      n:=0;rt[0]:=x0;
      for k:=1 to 20 do
        begin
          x:=g(x0);
          x1:=x+q*(x-x0)/(1-q);
          rt[k]:=x1;
          n:=n+1;
          if abs(x1-x0)<ep
            then exit
            else x0:=x1
        end;
    end;
end.
```

六、例题

求方程 $x=e^{-x}$ 在 $x=0.5$ 附近的一个根。取 $q=-0.6$ $ep=10^{-5}$

```
program m - jdie(input,output);
  uses u - jdie;
  var
    x0,ep,q:real; n,i:integer; rt:shuzu;
  begin
    writeln(' input data ');
    write(' x0=');readln(x0);
    write(' ep=');readln(ep);
    write(' q=');readln(q);
    jdie(x0,ep,q,rt,n);
    writeln;
```

```
    writeln('result is :');
    writeln('  ------------');
    writeln('  k     |   ',' x(k)');
    writeln('  ----|--------');
    for i:=0 to n do writeln(i:4,'   |      ',rt[i]:6:5);
    writeln('  ------------')
  end.
unit uf-jdie;
interface
  function g(x:real):real;
implementation
  function g(x:real):real;
    begin
      g:=exp(-x)
    end;
end.
```

运行结果如下：

```
input data
x0=0.5
ep=1e-5
q=-0.6
result is:
```

k	x(k)
0	0.50000
1	0.56658
2	0.56713
3	0.56714
4	0.56714

1.4 牛 顿 法

一、功能

对给定的初值，能求出非线性实值函数方程 $f(x)=0$ 的一个实根。

二、算法简介

设已知方程 $f(x)=0$ 的一个近似根 x_0，则函数 $f(x)$ 在点 x_0 附近可用一阶泰勒多项式 $p_1(x)$

$=f(x_0)+f'(x_0)\cdot(x-x_0)$来近似。因此方程$f(x)=0$在点$x_0$附近可近似表示为$f(x_0)+f'(x_0)\cdot(x-x_0)=0$设$f'(x_0)$不等于零，解得$x=x_0-f(x_0)/f'(x_0)$根据初值$x_0$得到新的近似根$x_1$，然后用$x_1$替代$x_0$，又计算得到更新的近似根，如此下去，直到满足精度，就可得到方程$f(x)=0$的根。迭代公式为$x_{(k+1)}=x_k-f(x_k)/f'(x_k)$

三、框图

见图4。

四、变量说明

x_0：实变量，迭代初值，输入参数。

ep：实变量，求得根的精度要求，输入参数。

n：整变量，最大迭代次数，输入参数。

m：整变量，记录迭代多少次后根满足精度，返回参数。

rt：一维数组，rt[0]，rt[1]…rt[m]…存放每一次迭代结果，返回参数。

五、过程清单

```
unit u_newtn;
interface
  uses uf_newtn;
  type shuzu=array [0..20]of real;
  procedure newtn(var x0:real; ep:real; n:integer;
                  var m:integer;  var rt:shuzu);
implementation
  procedure newtn(var x0:real; ep:real; n:integer;
                  var m:integer;  var rt:shuzu);
    label 100;
    var
      k:integer; x1,y:real;
    begin
      m:=0;
      rt[m]:=x0;
      for k:=1 to n-1 do
        begin
          y:=g(x0);
          if y=0 then
                begin
                  writeln('&&&&&');
                  goto 100
                end
              else
                begin
                  x1:=x0-f(x0)/y;
```