Intermediate Programming

— Nonlinear equations (2) —

Waseda Univ.

Today's Topics

- Newton's Method using numerical differentiation.
- Function Pointer

Newton's method

Algorithm of Newton's Method

Step 1. We set an initial value $x^{(0)}$ in a domain. It is better to set a close value of solution, if you have the idea of a solution.

Step 2. We calculate sequences $x^{(1)}, x^{(2)}, x^{(3)}, \ldots$ using the following recurrence formula:

$$x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})} \qquad (i = 0, 1, 2, \cdots).$$

We assume $f'(x^{(i)}) \neq 0$.

Step 3. When satisfying some convergence condition, let $x^{(i+1)}$ be an approximate solution. For example: $\left|x^{(i+1)} - x^{(i)}\right| < \varepsilon$.

Program of Newton's Method

- For equations $f(x) = 0$, we use two functions:

  `double func(double x)` → Function $f(x)$

  `double func_d(double x)` → Derivative $f'(x)$

$$\Downarrow$$

- In case of the function is complicated, it is troublesome.
- There is a possibility of mistakes for calculating the derivative.

$$\Downarrow$$

- Using numerical differentiation, we can automatically compute an approximate derivative value from $f(x)$.

Numerical differentiation

For 1 variable function $f(x)$, the derivative is defined by

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

Instead of it, there is a method that using a small positive number, by computing

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

obtains an approximate derivative value. It is called numerical differentiation.

Numerical differentiation

- For calculating an approximate derivative value, we shall defined step size h as a very small value (e.g. $h = 1e - 4$). We can rewrite the program to solve nonlinear equations using newton's method like the following:

```
int main(void){
   ...


   i=0;
   while (i<max){
      f  = func(x);
      df = func_d(x);
      x_n= x- f/df;
      ...
   }
   ...
   return 0;
}
```

⇒

```
int main(void){
   ...
   h=1e-4;

   i=0;
   while (i<max){
      f  = func(x);
      df = (func(x+h)-func(x))/h;
      x_n= x- f/df;
      ...
   }
   ...
   return 0;
}
```

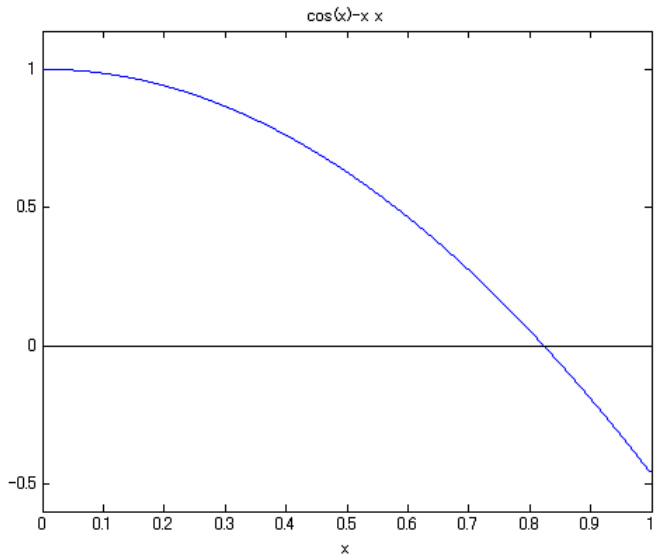
Example

Write a program to solve equations $\cos(x) = x^2$ using newton's method. Use numerical differentiation in computing $f'(x) = -\sin(x) - 2x$.

- Let the initial value be $x^{(0)} = 1.0$. Output of the program is as follows:

```
initial value: 1
x(001) = ...
x(002) = ...
...
answer = ...
```

- The figure on the right is the graph of $f(x) = \cos x - x^2$.



Answer

```
#include<stdio.h>
#include<math.h>

double func(double x){
  return cos(x)-x*x;
}
double Newton(double init, double eps, int imax){
  int i;
  double x_old=init, f, df, x_new,h=1e-04;
  for(i=0;i<imax;i++){
    x_new=x_old-func(x_old)/((func(x_old+h)-func(x_old))/h);
    printf("x(%03d)=%+.6f\n",i+1,x_new);
    if(fabs(x_new-x_old)<eps) return x_new;
    x_old = x_new;
  }
  return 0.0/0.0;/*Nan*/

}
int main(void){

  double error, x, x_n;
  int i, max;
  printf("initial value:");
  scanf("%lf",&x);
  x = Newton(x,1e-6,20);
  if(!isnan(x))/*Determining whether a value is NaN or not*/
    printf("answer =%+.6f\n",x);
  else
    printf("not converged\n");
  return 0;
}
```

Program of Newton's method

The written program is

```
double func(double x){
    return cos(x) - x*x;
}

double  newton(double x0, double e, int max){
   ...
   i=0;
   while(i<max){
      f  = func(x);
      df = (func(x+h)-func(x))/h;
      ...
   }
```

- We use func() that computes equations straightforwardly in newton() that compute a solution using Newton's Method.

- In this case, when the function that compute another equations is given by a different name (for example, func2()), it is necessary to modify the contents of the newton( ) function.

Program of Newton's method

The written program is

```
double newton(double x0, double e, int max){
   ...
   i=0;
   while(i<max){
      f  = func2(x);
      df = (func2(x+h)-func2(x))/h;
      ...
   }

double func2(double x){
    return x - cos(x);
}
```

Using function pointer, we can pass the function name to express an equation to newton() as the argument. For example: newton (func2,1.0,1e-6,100);

Function pointer

- The address of a function is given on the memory like a variable and an array.
- An array name is replaced with a pointer to the first element of array in program.
- Similarly, function name is replaced with a pointer to the first address of function in program. For example, in case of the function.

```
double func2(double x){
    ...
}
```

is defined, function name func2 shows a pointer to the first element of function.

Function pointer

- Programming example using function pointer:

```
#include <stdio.h>

double add(double x, double y){
  return x+y;
}

double sub(double x, double y){
  return x-y;
}

int main(void){
  double (*func_p)(double,double);/*func_p that takes two double type of argument*/
                                  /*Declaring func_p as function pointer*/
  func_p = add;                   /*Substituting the pointer to add for func_p*/
  printf("%f\n",func_p(3.0,2.0));

  func_p = sub;                   /*Substituting the pointer to sub for func_p*/
  printf("%f\n",func_p(3.0,2.0));

  return 0;
}
```

Summry

- Newton's method using numerical differentiation.
- Function Pointer