Intermediate Programming

— Nonlinear equations (1) —

Waseda Univ.

Today's Topics

- Methods for solving nonlinear equations
  - Bisection Method
  - Newton's Method

Solution of equations

- How to obtain real solutions x for nonlinear equations?
  - If $f(x) = 0$ is less than or equal fourth order polynomial, there exists a general formula.
  - General directed formula for nonlinear equations does not exist.

$$\Downarrow$$

  - There are well-known methods which calculate solutions by using iterative method.
- In this lesson, we will write programs which calculate solutions by using the well-known iterative method:
  - Bisection Method
  - Newton's Method

Solution of equations

- How to obtain real solutions x for nonlinear equations?
  - If $f(x) = 0$ is less than or equal fourth order polynomial, there exists a general formula.
  - General directed formula for nonlinear equations does not exist.

$$\Downarrow$$

  - There are well-known methods which calculate solutions by using iterative method.
- In this lesson, we will write programs which calculate solutions by using the well-known iterative method:
  - Bisection Method
  - Newton's Method

Solution of equations

- How to obtain real solutions x for nonlinear equations?
  - If $f(x) = 0$ is less than or equal fourth order polynomial, there exists a general formula.
  - General directed formula for nonlinear equations does not exist.

$$\Downarrow$$

  - There are well-known methods which calculate solutions by using iterative method.
- In this lesson, we will write programs which calculate solutions by using the well-known iterative method:
  - Bisection Method
  - Newton's Method

Bisection Method

- For initial value, $f(a)$ and $f(b)$ are opposite sign ($+$ and $-$ or $-$ and $+$).
- Function$f$ must be a continuous function on the interval $[a, b]$.
- Bisection Method obtains a solution on the interval $[a, b]$. If there is more than one solution, depending on the initial values, some one will be obtained.

Algorithm of Bisection Method

Step 1. We will set initial values a and b which make $f(a)$ and $f(b)$ opposite sign.

Step 2. Calculating $c \equiv (a + b)/2$. If convergence condition $|a - b| < 2\varepsilon$, then go to Step3, where ε is a small positive number.

- If $f(c) = 0$, then go to Step3,
- If $f(c)$ and $f(a)$ are same sign, then substitute the value of c for a.
- If $f(c)$ and $f(b)$ are same sign, then substitute the value of c for b.

Repeat Step2.

Step 3. Let "c" be a solution.

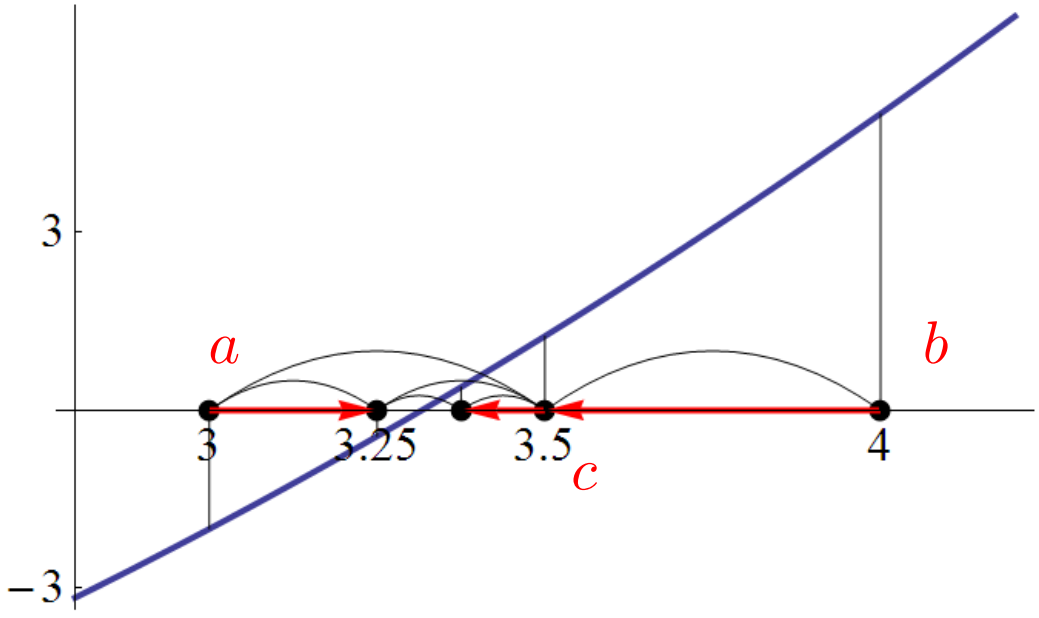
- The solution c obtained by the Bisection Method satisfies $|c - x^*| < \varepsilon$, where x^* is an exact solution.

Fundamental idea in Bisection Method

Intermediate-value theorem

For continuous function f, if a and b which make $f(a)$ and $f(b)$ opposite sign exist, at least one solution exist on the interval (a, b).

- It repeats using Intermediate-value theorem and narrowing the range of the solution with half and half.



Example

Solve $x = 1.2e^{-x}$ by using Bisection Method.

- Let $f(x) = x - 1.2e^{-x}$, create the function $f(x)$ and its data type of the return value and argument are double data type.

  double func(double x);

- Create the function which returns the solution by using the Bisection Method. Its arguments are the initial value $a,\ b$ and ε.

  double Bisection(double a, double b, double eps);

- Let the initial values be $a = 0,\ b = 3$ and ε be $\varepsilon = 10^{-6}$

Example

- Use format specifiers `"%+.6f"` for output.
- Output of the program is as follows:

```
a = +0.000000, b = +3.000000, c = +1.500000
a = +0.000000, b = +1.500000, c = +0.750000
a = +0.000000, b = +0.750000, c = +0.375000
a = +0.375000, b = +0.750000, c = +0.562500
a = +0.562500, b = +0.750000, c = +0.656250
a = +0.562500, b = +0.656250, c = +0.609375
a = +0.609375, b = +0.656250, c = +0.632812
a = +0.632812, b = +0.656250, c = +0.644531
a = +0.632812, b = +0.644531, c = +0.638672
a = +0.632812, b = +0.638672, c = +0.635742
a = +0.632812, b = +0.635742, c = +0.634277
a = +0.634277, b = +0.635742, c = +0.635010
a = +0.635010, b = +0.635742, c = +0.635376
a = +0.635376, b = +0.635742, c = +0.635559
a = +0.635559, b = +0.635742, c = +0.635651
a = +0.635559, b = +0.635651, c = +0.635605
a = +0.635559, b = +0.635605, c = +0.635582
a = +0.635559, b = +0.635582, c = +0.635571
a = +0.635559, b = +0.635571, c = +0.635565
a = +0.635559, b = +0.635565, c = +0.635562
a = +0.635562, b = +0.635565, c = +0.635563
a = +0.635563, b = +0.635565, c = +0.635564
answer = +0.635564
```

Example

- Using the infinite loop of repeating Step2.
  We may use either while loop or for loop like the following:

  ```
  while (1) { ... }
  ```

  ```
  for (; ;) { ... }
  ```

- Write convergence condition $|a - b| < 2\varepsilon$ like the following:

  ```
  fabs(a - b) < 2 * eps
  ```

- Use `break;` for going to Step3 in the algorithm.
- We can define $f(x) = x - 1.2e^{-x}$ as the following function:

  ```
  double func(double x){
      double y;
      y=x-1.2*exp(-x);
      return y;
  }
  ```

- We can write 10^{-6} as 1e-6 (scientific notation).

Example(clue)

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

double Function1(double x) {
  Calculating formula
}

double Bisection(double a,
         double b, double eps) {
  double fa = Function1(a);
  double fb = Function1(b);
  double c, fc;

  while (1) {
  Repeat calculation by using infinite loop
  }

  return c;
}

int main(void) {
  double x;

  Bisection Method

  return 0;
}
```

- Equation:

```
double Function1(double x) {
  See clue the previous page
}
```

- Bisection Method:

```
double Bisection(double a, double b, double eps) {
  double fa, fb, fc, c;
  while (1) {  /* Infinite loop */
    c = ;/*c is middle point of a and b*/
    printf("");/*Output of progress */
    if (Convergence test) {
      break;
    }
    fa = Function1(a);
    fb = Function1(b);
    fc = Function1(c);
    if (fc == 0) {
      break;
    }
    else if ((fc > 0 && fa > 0) || (fc < 0 && fa < 0)) {
      a = c;
    }
    else if ((fc > 0 && fb > 0) || (fc < 0 && fb < 0)) {
      b = c;
    }
  }
  return c;
}
```

- main function

```
int main(void) {
  double x;
  x = Bisection();/*What's argument?*/
  printf("");/*Output the answer*/
  return 0;
}
```

Newton's Method

- It's an iterative method for solving nonlinear equation $f(x) = 0$.
- In many cases, the Newton's Method often converges on a solution faster than Bisection Method, but does not converge all the time.
- We need to compute $f(x)$ and first - derivative $f'(x)$.

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: $\left| x^{(i+1)} - x^{(i)} \right| < \varepsilon$.

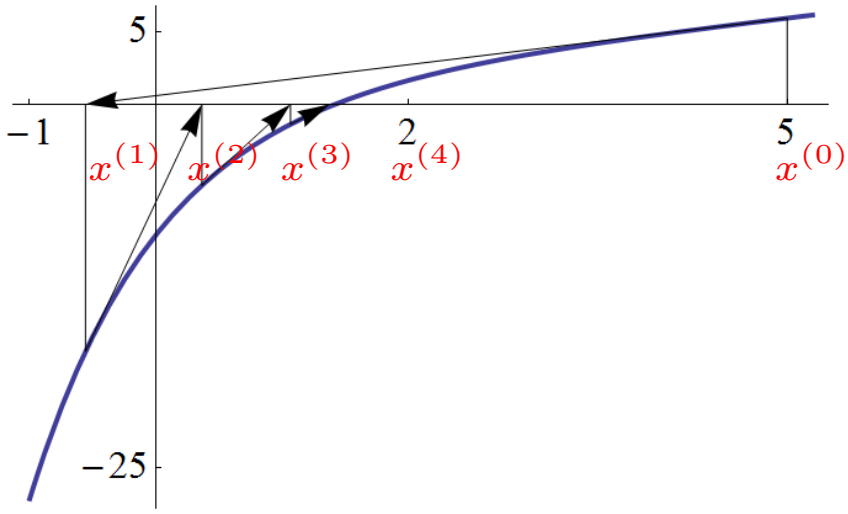
Geometric interpretation

- A tangent line of $f(x)$ in $x = x^{(i)}$ is

$$y = f'(x^{(i)})(x - x^{(i)}) + f(x^{(i)}).$$

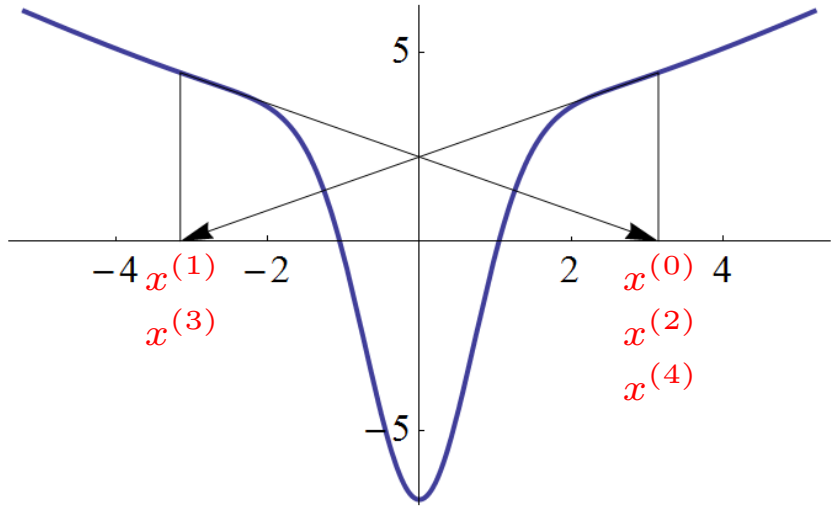
- Repeating to calculate x-coordinate point at intersection of the tangent line and x-axis($y = 0$):

$$x = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}$$



limit on the number of iterations

- In Newton's Method, even if there is a solution, it may not converge to a solution, depending on the initial value. Therefore, we set the upper limit of the number of iterations, in case of reaching limitation, then it ends the iteration.



limit on the number of iterations

Let I_{MAX} be the max iterative number.
In case of reaching limitation without satisfying convergence condition, it is considered that an approximate solution was not obtained. For example:
$I_{\mathrm{MAX}} = 20$

Creating a program

A program which solves the following equation:

Solve the equation

$$x = 1.2e^{-x}$$

using Newton's Method.

For

$$x = 1.2e^{-x} \Longleftrightarrow x - 1.2e^{-x} = 0$$

, we hold $f(x) = x - 1.2e^{-x}$,

$$f'(x) = 1 + 1.2e^{-x}.$$

We solve the equation using iteration:

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

Hints

- Computation of $f(x), f'(x)$: $f(x), f'(x)$ are

$$f(x) = x - 1.2e^{-x}, \qquad f'(x) = 1 + 1.2e^{-x}$$

  For computing the above values, we will define $f(x), f'(x)$ as functions.

We can define $f(x) = x - 1.2e^{-x}$ as the following function:

```
double func(double x){
    double y;
    y=x-1.2*exp(-x);
    return y;
}
```

Similarly, we can define $f'(x)$ as something like functioin_name`func_d( )`

Hints

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

- In Newton's Method, we set some initial value $x^{(0)}$, according to the above formula, compute the next point in sequence with increasing the value of i.
- When satisfying stopping criterion, it ends computing. Let a obtained value be an approximate solution.
- Setting of the initial value: Declare a variable x that represents the current point and substitute some value for x.
- Iteration of Newton's Method: Use while, for loop.

Hint

Stopping criterion of iterations

- For stopping iteration, we use the following stopping criterion:

$$|x^{(i+1)} - x^{(i)}| < \varepsilon$$

- We need to set a very small value as error (e.g. `error=1e-6`: 10^{-6})

Setting the maximum number of iterations

- In Newton's Method, it may not converge to a solution by choosing of an initial point. Therefor, we set the maximum number of iterations, if it does not converge to a solution, it is forcibly finished.
- For example, we substitute a variable max for the maximum number of iterations (e.g. 100), we use it as stopping criterion. When iteration ends, in case of the value of i is equal to max, it does not converge to a solution. Otherwise, it converges to the solution.

Summary

- Methods for solving nonlinear equations
  - Bisection Method
  - Newton's Method