Some Examples, Questions, and Answers: Lecture 2 – COMPSCI 369

Georgy Gimel'farb

Taylor Series

Example 2.1: Appropximate the function $f(x) = \log x$ (Fig. 1), i.e the natural logarithm of x, or the logarithm to the base $e = 2.718281828\ldots$, in the neighbourhood of point $x = 1.0$ using the Taylor polynomial of degree $n = 6$.

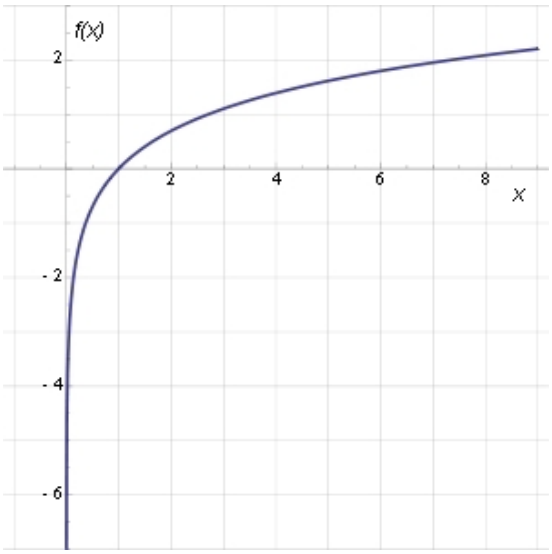

Figure 1: Function $\log x$ (`http://en.wikipedia.org/wiki/Natural_logarithm`).

Solution: First to sixth derivatives of the function $f(x) = \log x$ for $x > 0$:

$$\frac{df(x)}{dx} = \frac{1}{x}; \quad \frac{d^2f(x)}{dx^2} = -\frac{1}{x^2}; \quad \frac{d^3f(x)}{dx^3} = \frac{2!}{x^3};$$

$$\frac{d^4f(x)}{dx^4} = -\frac{3!}{x^4}; \quad \frac{d^5f(x)}{dx^5} = \frac{4!}{x^5}; \quad \frac{d^6f(x)}{dx^6} = -\frac{5!}{x^6}$$

Therefore, the values of the function $\log x$ and its derivatives at the point $x = 1$ are, respectively, $\log 1 = 0$ and

$$f^{(1)}(1) \equiv \left.\frac{df(x)}{dx}\right|_{x=1} = 1; \quad f^{(2)}(1) \equiv \left.\frac{d^2f(x)}{dx^2}\right|_{x=1} = -1;$$

$$f^{(3)}(1) \equiv \left.\frac{d^3f(x)}{dx^3}\right|_{x=1} = 2! = 2; \quad f^{(4)}(1) \equiv \left.\frac{d^4f(x)}{dx^4}\right|_{x=1} = -3! = -6;$$

$$f^{(5)}(1) \equiv \left.\frac{d^5f(x)}{dx^5}\right|_{x=1} = 4! = 24; \quad f^{(6)}(1) \equiv \left.\frac{d^6f(x)}{dx^6}\right|_{x=1} = -5! = -120$$

Substituting these values into the Taylor polynomial (Lecture `CS369-Roots-LinSys.pdf`, Slide 3):

$$f(x) = f(a) + (x-a)f^{(1)}(a) + \frac{1}{2!}(x-a)^2 f^{(2)}(a) + \frac{1}{3!}(x-a)^3 f^{(3)}(a)$$

$$+ \frac{1}{4!}(x-a)^4 f^{(4)}(a) + \frac{1}{5!}(x-a)^5 f^{(5)}(a) + \frac{1}{6!}(x-a)^6 f^{(6)}(a)$$

for $a = 1$ gives the following polynomial approximation of the logarithmic function in the vicinity of $x = 1$:

$$\widehat{\log}_6(x;1) = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \frac{(x-1)^5}{5} - \frac{(x-1)^6}{6}$$

Question 2.1 Approximate the same function $f(x) = \log x$ in the neighbourhood of point $x = a$ using the Taylor polynomial $\widehat{\log}_6(x;a)$ of degree $n = 6$.

Question 2.2 Approximate the exponential function $f(x) = \mathrm{e}^x$ in the neighbourhood of point $x = a$ using the Taylor polynomial $\widehat{\mathrm{e}}_n(x;a)$ of degree n.
Hint: See Lecture `CS369-Roots-LinSys.pdf`, Slide 3.

Newton's Root Finder

Example 2.2 The Newton's iterative search for the root x° of the cubic function $f(x) = 2x^3 - 15x^2 + 36x - 23$ (i.e. $f(x^\circ) = 0$) finds each next approximation x_{n+1} of the root by setting to zero the linear Taylor series expansion, i.e. by setting to zero the Taylor polynomial $\widehat{f}_1(x;x_n) = f(x_n) + (x - x_n)f^{(1)}(x_n)$ of degree 1.

In other words, the Newton's method assumes that $\widehat{f}_1(x_{n+1};x_n) = 0$, or – what is the same – $f(x_n)+(x_{n+1}-x_n)f^{(1)}(x_n) = 0$. This simple equation gives $x_{n+1} = x_n - \frac{f(x_n)}{f^{(1)}(x_n)}$. To implement this root finder, one needs the first derivative of the above cubic function: $f^{(1)}(x) = 6x^2 - 30x + 36$.

A simple C program implementing the Newton's search is given below:

```
/*********************
 *  Newton's method for a given cubic polynomial
 *********************/
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
double funct( double x );
double deriv( double x );
// Cubic polynomial
double funct( double x ) { return 2.*x*x*x-15.*x*x+ 36.*x-23.; }
 // Derivative
double deriv( double x ) {  return 6.*x*x-30.*x+36.; }

int main() {
       int    step = 0;        // iteration
       double drv,             // derivative value
              fnc,             // function value
              rc,              // current root value
              rn;              // new root value
       float  input;
       const double thresh   = 1.e-4;    // accuracy threshold
       const int    max_step = 100;      // max number of iterations

       printf("\nEnter initial root value: ");
       scanf("%f", &input );
       rn = (double)input;

       do {
             rc   = rn;
             fnc = funct( rc );
             drv = deriv( rc );
             if ( fabs( drv ) < thresh ) {
                 printf(
                    "\nClose or equal to zero derivative at step %d\n",

                 break;
             } else {
                 rn = rc - fnc / drv;       // Step towards the root
                 printf(
                 "\nStp%2i: old%7.5lf; fn%10.5lf  dr%10.5lf; new%7.5lf",
                      step++,     rc,      fnc,      drv,     rn     );
              }
       } while ( fabs( rn - rc ) >= thresh && step < max_step );
       printf("\n");
}
```

The above search converges to the goal value $x^\circ = 1$ in five steps for the input $x_0 = 0.0$ and in 14 steps for the input $x_0 = 2.5$:

```
Enter initial root value: 0.

Step  0: root 0.00000; fnc  -23.00000  drv   36.00000  new root 0.63889
Step  1: root 0.63889; fnc   -5.60112  drv   19.28241  new root 0.92937
Step  2: root 0.92937; fnc   -0.89320  drv   13.30132  new root 0.99652
Step  3: root 0.99652; fnc   -0.04189  drv   12.06274  new root 0.99999
Step  4: root 0.99999; fnc   -0.00011  drv   12.00016  new root 1.00000
```

```
Enter initial root value: 2.5

Step  0: root 2.50000; fnc    4.50000  drv   -1.50000  new root 5.50000
Step  1: root 5.50000; fnc   54.00000  drv   52.50000  new root 4.47143
Step  2: root 4.47143; fnc   16.86689  drv   21.81918  new root 3.69840
Step  3: root 3.69840; fnc    6.14458  drv    7.11695  new root 2.83503
Step  4: root 2.83503; fnc    4.07267  drv   -0.82655  new root 7.76235
Step  5: root 7.76235; fnc  288.05920  drv  164.65369  new root 6.01286
Step  6: root 6.01286; fnc   85.92943  drv   72.54113  new root 4.82830
Step  7: root 4.82830; fnc   26.25086  drv   31.02586  new root 3.98220
Step  8: root 3.98220; fnc    8.78927  drv   11.68155  new root 3.22980
Step  9: root 3.22980; fnc    4.18269  drv    1.69562  new root 0.76303
Step 10: root 0.76303; fnc   -3.37561  drv   16.60235  new root 0.96635
Step 11: root 0.96635; fnc   -0.41402  drv   12.61243  new root 0.99918
Step 12: root 0.99918; fnc   -0.00985  drv   12.01476  new root 1.00000
Step 13: root 1.00000; fnc   -0.00001  drv   12.00001  new root 1.00000
```

Notice that convergence of the Newton's search is not necessarily monotone depending on closeness of linear approximation at each successive guess x_n, i.e. $\widehat{f_1}(x; x_n) = f(x_n) + (x - x_n) f^{(1)}(x_n)$, starting from the initial guess x_0, to the function $f(x)$ between x_n and the true root x°. Moreover, in some cases the process may even diverge from the goal root.

Question 2.3: What result of the above search will be obtained for the input guess $x_0 = 2.0$ and why? Can you predict the search result for the input $x_0 = 3.0$?

Question 2.4: Find results of the above search for the input guesses $x_0 = -1$, -0.5, 0.5, 1.5, 2.1, 3.1, 3.9, 4.0, 5.0. Does the number of steps to convergence relate only to the closeness between the initial guess and the true root $x^\circ = 1.0$ or it depends also on the closeness between the function and its linear approximation?

Question 2.5: Implement the Newton's search and find the root of the function $f(x) = \log x - \frac{1}{x}$; $x > 0$, starting from the initial guesses $x_0 = 1.0$ and $x_0 = 2.0$.

Hint: the first derivative is positive, $f^{(1)}(x) = \frac{1}{x} + \frac{1}{x^2} > 0$, for all $x > 0$.

Systems of Linear Equations: Gaussian Elimination

Example 2.3: Transforming the system

$$\left\{\begin{array}{rcrcrcr} 6x & + & 2y & - & 3z & = & 1 \\ 12x & - & y & + & 4z & = & 22 \\ 18x & + & 5y & - & 5z & = & 13 \end{array}\right. , \text{ or } \overbrace{\begin{bmatrix} 6 & 2 & -3 \\ 12 & -1 & 4 \\ 18 & 5 & -5 \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}^{\mathbf{x}} = \overbrace{\begin{bmatrix} 1 \\ 22 \\ 13 \end{bmatrix}}^{\mathbf{b}}$$

into upper triangular form by sequential elimination of lower parts of the columns.

Eliminating the first column: Deducting the scaled first row from each lower row with the scale factors, being the ratio between the corresponding first coefficients (in order to eliminate the first column elements):

$$\begin{array}{l} \text{Row } 1 : 6x + 2y - 3z = 1 \\ \text{Row } 2 : 12x - y + 4z - \frac{12}{6}(6x + 2y - 3z) = 22 - \frac{12}{6} \cdot 1 \quad \Rightarrow \quad -5y + 10z = 20 \\ \text{Row } 3 : 18x + 5y - 5z - \frac{18}{6}(6x + 2y - 3z) = 13 - \frac{18}{6} \cdot 1 \quad \Rightarrow \quad -y + 4z = 10 \end{array}$$

– eliminates all the elements of the first column below the main diagonal:

$$\left\{\begin{array}{rcrcrcr} 6x & + & 2y & - & 3z & = & 1 \\ & - & 5y & + & 10z & = & 20 \\ & - & y & + & 4z & = & 10 \end{array}\right. , \text{ or } \begin{bmatrix} 6 & 2 & -3 \\ 0 & -5 & 10 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 20 \\ 10 \end{bmatrix}$$

In the matrix form such elimination is represented by left-multiplying the initial matrix $\mathbf{A}$ and vector $\mathbf{b}$ with the eliminating matrix $\mathbf{M}_1$, which has the unit main diagonal and the negated scaling factors along the first column, $\mathbf{M}_1\mathbf{A}\mathbf{x} = \mathbf{M}_1\mathbf{b}$:

$$\underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -\frac{12}{6} = -2 & 1 & 0 \\ -\frac{1}{6} = -3 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_1} \underbrace{\begin{bmatrix} 6 & 2 & -3 \\ 12 & -1 & 4 \\ 18 & 5 & -5 \end{bmatrix}}_{\mathbf{A}}}_{\mathbf{M}_1\mathbf{A} = \begin{bmatrix} 6 & 2 & -3 \\ 0 & -5 & 10 \\ 0 & -1 & 4 \end{bmatrix}} \mathbf{x} = \underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -\frac{12}{6} = -2 & 1 & 0 \\ -\frac{1}{6} = -3 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_1} \underbrace{\begin{bmatrix} 1 \\ 22 \\ 13 \end{bmatrix}}_{\mathbf{b}}}_{\mathbf{M}_1\mathbf{b} = \begin{bmatrix} 1 \\ 20 \\ 10 \end{bmatrix}}$$

Eliminating the second column: The like deduction of the scaled second row of the system $\mathbf{M}_1\mathbf{A}\mathbf{u} = \mathbf{M}_1\mathbf{b}$ from the lower one finalises the transformation in this particular example:

$$\begin{array}{l} \text{Row } 1 : 6x + 2y - 3z = 1 \\ \text{Row } 2 : -5y + 10z = 20 \\ \text{Row } 3 : -y + 4z - \frac{-1}{-5}(-5y + 10z) = 10 - \frac{-1}{-5} \cdot 20 \quad \Rightarrow \quad 2z = 6 \end{array}$$

by eliminating all the elements of the second column below the main diagonal:

$$\left\{\begin{array}{rcrcrcr} 6x & + & 2y & - & 3z & = & 1 \\ & - & 5y & + & 10z & = & 20 \\ & & & & 2z & = & 6 \end{array}\right. , \text{ or } \begin{bmatrix} 6 & 2 & -3 \\ 0 & -5 & 10 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 20 \\ 6 \end{bmatrix}$$

In the matrix form such elimination is represented by left-multiplying the matrix $\mathbf{M}_1\mathbf{A}$ and vector $\mathbf{M}_1\mathbf{b}$ with the eliminating matrix $\mathbf{M}_2$, which has the unit main diagonal and the negated scaling factors below the main diagonal along the second column, $\mathbf{M}_2\mathbf{M}_1\mathbf{A} = \mathbf{M}_2\mathbf{M}_1\mathbf{b}$:

$$\underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{-1}{-5} & 1 \end{bmatrix}}_{\mathbf{M}_2} \underbrace{\begin{bmatrix} 6 & 2 & -3 \\ 0 & -5 & 10 \\ 0 & -1 & 4 \end{bmatrix}}_{\mathbf{M}_1\mathbf{A}}}_{\overbrace{\mathbf{M}_2\mathbf{M}_1\mathbf{A}}^{\mathbf{C}} = \begin{bmatrix} 6 & 2 & -3 \\ 0 & -5 & 10 \\ 0 & 0 & 2 \end{bmatrix}} \mathbf{x} = \underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{-1}{-5} & 1 \end{bmatrix}}_{\mathbf{M}_2} \underbrace{\begin{bmatrix} 1 \\ 20 \\ 10 \end{bmatrix}}_{\mathbf{M}_1\mathbf{b}}}_{\overbrace{\mathbf{M}_2\mathbf{M}_1\mathbf{b}}^{\mathbf{c}} = \begin{bmatrix} 1 \\ 20 \\ 6 \end{bmatrix}}$$

Solving the system: The resulting upper triangular system $\mathbf{C}\mathbf{x} = \mathbf{c}$ is solved by back-substitution:

$$\left.\begin{array}{rcl} 2z = 6z & \Rightarrow & z = \frac{6}{2} = 3 \\ -5y + 10z = 20 & \Rightarrow & y = \frac{1}{-5}(20 - 10 \cdot 3) = 2 \\ 6x + 2y - 3z = 1 & \Rightarrow & x = \frac{1}{6}(1 - 2 \cdot 2 + 3 \cdot 3) = 1 \end{array}\right\}, \text{that is, } \mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Question 2.6: How does Gaussian elimination relate to LU decomposition that represents a square matrix $\mathbf{A}$ as the product of a lower triangular matrix, $\mathbf{L}$, and an upper triangular matrix, $\mathbf{U}$, i.e. $\mathbf{A} = \mathbf{L}\mathbf{U}$?

Hint: See `http://en.wikipedia.org/wiki/LU_decomposition`.

Question 2.7: Solve by hand the system $\underbrace{\begin{bmatrix} 1 & 4 \\ 2 & 6 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 3 \\ 4 \end{bmatrix}}_{\mathbf{b}}$ of two linear equations using Gaussian elimination and perform LU decomposition of its matrix $\mathbf{A}$.

Answers

To Question 2.1: At an arbitrary point $x = a$ the function value is $\log a$ and the values of the derivatives are $f^{(k)} = (-1)^{k-1}\frac{(k-1)!}{a^k}$. Therefore,

$$\log x = \log a + \frac{x-a}{a} - \frac{(x-a)^2}{2a^2} + \frac{(x-a)^3}{3a^3} - \frac{(x-a)^4}{4a^4} + \frac{(x-a)^5}{5a^5} - \frac{(x-a)^6}{6a^6}$$

To Question 2.2: At an arbitrary point $x = a$ both the function and all the derivatives have the same value e^a. Therefore, the Taylor polynomial of degree n is as follows:

$$\widehat{\mathrm{e}}_n(x;a) = \mathrm{e}^a\left(1 + \frac{x-a}{1!} + \frac{(x-a)^2}{2!} + \frac{(x-a)^3}{3!} + \ldots + \frac{(x-a)^n}{n!}\right)$$

To Question 2.3: The search fails for the initial guess $x_0 = 2.0$ because at this point the first derivative is equal to zero: $f^{(1)}(2.0) = 0.0$:

```
Enter initial root value: 2

Close or equal to zero derivative at step 0
```

In this example the first derivative is a quadratic polynomial, $6x^2 - 30x + 36$, with two own roots, $x = 2.0$ and $x = 3.0$, which correspond to the local maximum and the local minimum of the cubic polynomial $f(x)$ under consideration. Therefore, the like search failure is expected for the input $x_0 = 3.0$.

To Question 2.4: The search results below illustrate that the closer the linear approximations at the initial point x_0 and all the subsequent points x_n; $n = 1, 2, \ldots$ to the function $f(x)$ itself within neighbourhood of the true root x°; $f(x^\circ) = 0$, the faster the convergence. If the linear approximations deviate much from the function itself, then the convergence may become non-monotonous or the process may even divergence from the goal root.

```
Enter initial root value: -1.

Step  0: root-1.00000; fnc  -76.00000  drv   72.00000  new root 0.05556
Step  1: root 0.05556; fnc  -21.04595  drv   34.35185  new root 0.66821
Step  2: root 0.66821; fnc   -5.04522  drv   18.63264  new root 0.93899
Step  3: root 0.93899; fnc   -0.76611  drv   13.12057  new root 0.99738
Step  4: root 0.99738; fnc   -0.03153  drv   12.04725  new root 0.99999
Step  5: root 0.99999; fnc   -0.00006  drv   12.00009  new root 1.00000

Enter initial root value: -0.5

Step  0: root-0.50000; fnc  -45.00000  drv   52.50000  new root 0.35714
Step  1: root 0.35714; fnc  -11.96501  drv   26.05102  new root 0.81643
Step  2: root 0.81643; fnc   -2.51842  drv   15.50636  new root 0.97885
Step  3: root 0.97885; fnc   -0.25788  drv   12.38344  new root 0.99967
Step  4: root 0.99967; fnc   -0.00394  drv   12.00591  new root 1.00000
Step  5: root 1.00000; fnc   -0.00000  drv   12.00000  new root 1.00000
```

```
Enter initial root value: 0.5

Step  0: root 0.50000; fnc   -8.50000  drv   22.50000  new root 0.87778
Step  1: root 0.87778; fnc   -1.60476  drv   14.28963  new root 0.99008
Step  2: root 0.99008; fnc   -0.11992  drv   12.17914  new root 0.99993
Step  3: root 0.99993; fnc   -0.00088  drv   12.00131  new root 1.00000
```

```
Enter initial root value: 1.5

Step  0: root 1.50000; fnc    4.00000  drv    4.50000  new root 0.61111
Step  1: root 0.61111; fnc   -6.14540  drv   19.90741  new root 0.91981
Step  2: root 0.91981; fnc   -1.02118  drv   13.48199  new root 0.99555
Step  3: root 0.99555; fnc   -0.05353  drv   12.08014  new root 0.99999
Step  4: root 0.99999; fnc   -0.00018  drv   12.00027  new root 1.00000
```

```
Enter initial root value: 2.1

Step  0: root 2.10000; fnc    4.97200  drv   -0.54000  new root11.30742
Step  1: root11.30742; fnc 1357.68119  drv  463.92338  new root 8.38089
Step  2: root 8.38089; fnc  402.45927  drv  206.00955  new root 6.42730
Step  3: root 6.42730; fnc  119.75590  drv   91.04210  new root 5.11191
Step  4: root 5.11191; fnc   36.21939  drv   39.43242  new root 4.19339
Step  5: root 4.19339; fnc   11.67176  drv   15.70544  new root 3.45022
Step  6: root 3.45022; fnc    4.79063  drv    3.91756  new root 2.22736
Step  7: root 2.22736; fnc    4.86842  drv   -1.05402  new root 6.84629
Step  8: root 6.84629; fnc  162.18502  drv  111.84122  new root 5.39615
Step  9: root 5.39615; fnc   48.73975  drv   48.82610  new root 4.39792
Step 10: root 4.39792; fnc   15.32608  drv   20.11256  new root 3.63590
Step 11: root 3.63590; fnc    5.72740  drv    6.24165  new root 2.71829
Step 12: root 2.71829; fnc    4.19336  drv   -1.21409  new root 6.17221
Step 13: root 6.17221; fnc   98.03232  drv   79.41087  new root 4.93772
Step 14: root 4.93772; fnc   29.81554  drv   34.15480  new root 4.06476
Step 15: root 4.06476; fnc    9.81547  drv   13.19093  new root 3.32066
Step 16: root 3.32066; fnc    4.37440  drv    2.54087  new root 1.59904
Step 17: root 1.59904; fnc    4.38877  drv    3.37037  new root 0.29688
Step 18: root 0.29688; fnc  -13.58214  drv   27.62251  new root 0.78858
Step 19: root 0.78858; fnc   -2.95819  drv   16.07370  new root 0.97262
Step 20: root 0.97262; fnc   -0.33533  drv   12.49731  new root 0.99945
Step 21: root 0.99945; fnc   -0.00656  drv   12.00984  new root 1.00000
Step 22: root 1.00000; fnc   -0.00000  drv   12.00000  new root 1.00000
```

```
Enter initial root value: 3.1
```

```
Step  0: root 3.10000; fnc    4.03200  drv    0.66000  new root -3.00910
Step  1: root -3.00910; fnc -321.64024  drv  180.60092  new root -1.22815
Step  2: root -1.22815; fnc  -93.54386  drv   81.89473  new root -0.08591
Step  3: root -0.08591; fnc  -26.20463  drv   38.62150  new root 0.59259
Step  4: root 0.59259; fnc   -6.51799  drv   20.32925  new root 0.91321
Step  5: root 0.91321; fnc   -1.11055  drv   13.60737  new root 0.99483
Step  6: root 0.99483; fnc   -0.06233  drv   12.09329  new root 0.99998
Step  7: root 0.99998; fnc   -0.00024  drv   12.00036  new root 1.00000
```

```
Enter initial root value: 3.9

Step  0: root 3.90000; fnc    7.88800  drv   10.26000  new root 3.13119
Step  1: root 3.13119; fnc    4.05615  drv    0.89040  new root -1.42424
Step  2: root -1.42424; fnc -110.47741  drv   90.89788  new root -0.20884
Step  3: root -0.20884; fnc  -31.19054  drv   42.52679  new root 0.52460
Step  4: root 0.52460; fnc   -7.95382  drv   21.91333  new root 0.88756
Step  5: root 0.88756; fnc   -1.46586  drv   14.09971  new root 0.99153
Step  6: root 0.99153; fnc   -0.10232  drv   12.15294  new root 0.99995
Step  7: root 0.99995; fnc   -0.00064  drv   12.00096  new root 1.00000
```

```
Enter initial root value: 4

Step  0: root 4.00000; fnc    9.00000  drv   12.00000  new root 3.25000
Step  1: root 3.25000; fnc    4.21875  drv    1.87500  new root 1.00000
Step  2: root 1.00000; fnc    0.00000  drv   12.00000  new root 1.00000
```

```
Enter initial root value: 5

Step  0: root 5.00000; fnc   32.00000  drv   36.00000  new root 4.11111
Step  1: root 4.11111; fnc   10.44719  drv   14.07407  new root 3.36881
Step  2: root 3.36881; fnc    4.50840  drv    3.02899  new root 1.88040
Step  3: root 1.88040; fnc    4.95366  drv    0.80345  new root-4.28513
Step  4: root-4.28513; fnc -610.06866  drv  274.72760  new root-2.06449
Step  5: root-2.06449; fnc -178.85222  drv  123.50767  new root-0.61639
Step  6: root-0.61639; fnc  -51.35739  drv   56.77127  new root 0.28825
Step  7: root 0.28825; fnc  -13.82147  drv   27.85107  new root 0.78451
Step  8: root 0.78451; fnc   -3.02379  drv   16.15740  new root 0.97166
Step  9: root 0.97166; fnc   -0.34739  drv   12.51499  new root 0.99942
Step 10: root 0.99942; fnc   -0.00702  drv   12.01053  new root 1.00000
Step 11: root 1.00000; fnc   -0.00000  drv   12.00000  new root 1.00000
```

To Question 2.5: This function is smooth with no local extrema (its first derivative is positive for $x > 0$). Thus the search converges monotonously in 4 – 5 steps to the goal root:

```
Enter initial root value: 1.

Step  0: root 1.00000; fnc   -1.00000  drv    2.00000  new root 1.50000
Step  1: root 1.50000; fnc   -0.26120  drv    1.11111  new root 1.73508
Step  2: root 1.73508; fnc   -0.02529  drv    0.90851  new root 1.76292
Step  3: root 1.76292; fnc   -0.00027  drv    0.88901  new root 1.76322
Step  4: root 1.76322; fnc   -0.00000  drv    0.88879  new root 1.76322
```

```
Enter initial root value: 2.

Step  0: root 2.00000; fnc    0.19315  drv    0.75000  new root 1.74247
Step  1: root 1.74247; fnc   -0.01859  drv    0.90326  new root 1.76306
Step  2: root 1.76306; fnc   -0.00015  drv    0.88891  new root 1.76322
Step  3: root 1.76322; fnc   -0.00000  drv    0.88879  new root 1.76322
```

To Question 2.6: LU decomposition is the matrix form of Gaussian elimination, which converts a non-singular linear system, $\mathbf{Ax} = \mathbf{b}$, of m equations with m unknowns into the equivalent linear system $\mathbf{Cx} = \mathbf{c}$ with the upper triangular $m \times m$ matrix $\mathbf{C} = \mathbf{M}_{m-1} \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A}$. Because all the eliminating matrices $\mathbf{M}_i$; $i = 1, \ldots, m-1$, are non-singular (invertible), the elimination implicitly builds the LU decomposition:

$$\mathbf{A} = \underbrace{\mathbf{M}_1^{-1}\mathbf{M}_2^{-1} \cdots \mathbf{M}_{m-1}^{-1}}_{\mathbf{L}} \underbrace{\mathbf{C}}_{\mathbf{U}}$$

In our Example 2.3,

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}^{-1}}_{\mathbf{M}_1^{-1}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.2 & 1 \end{bmatrix}^{-1}}_{\mathbf{M}_2^{-1}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_1^{-1}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.2 & 1 \end{bmatrix}}_{\mathbf{M}_2^{-1}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0.2 & 1 \end{bmatrix}}_{\mathbf{L}}$$

and the initial matrix $\mathbf{A}$ has the following LU decomposition:

$$\underbrace{\begin{bmatrix} 6 & 2 & -3 \\ 12 & -1 & 4 \\ 18 & 5 & -5 \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0.2 & 1 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 6 & 2 & -3 \\ 0 & -5 & 10 \\ 0 & 0 & 2 \end{bmatrix}}_{\mathbf{U}(\equiv\mathbf{C})}$$

To Question 2.7: $\begin{bmatrix} 1 & 4 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} 1 & 4 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \quad \Rightarrow$

$y = \frac{-2}{-2} = 1$

$x = \frac{3-4 \cdot 1}{1} = -1$

$$\Leftrightarrow \quad \underbrace{\begin{bmatrix} 1 & 4 \\ 2 & 6 \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 1 & 4 \\ 0 & -2 \end{bmatrix}}_{\mathbf{U}}$$