Unit 3

November 9, 2015

1 System of linear equations

Summary of definitions A system of linear equations is a set of M equations involving N variables, $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ (unknowns). In general a system of linear equations can be written as a matrix equation of the form:

$$A\mathbf{x} = \mathbf{b} \,, \tag{1}$$

where $A_{11}, A_{12}, \ldots, A_{MN}$ are the coefficients of the system (A is a $M \times N$ matrix) and $\mathbf{b} = (b_1, b_2, \ldots, b_M)^T$ are the constant terms:

$$\begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ A_{M1} & A_{M2} & \ldots & A_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_M \end{pmatrix} \,. \tag{2}$$

Clearly, Eqs. (1) and (2) can be equivalently written in terms in its components:

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1N}x_N &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2N}x_N &= b_2 \\ &\ldots \\ A_{M1}x_1 + A_{M2}x_2 + \cdots + A_{MN}x_N &= b_M \,. \end{aligned}$$

A system can either have infinitely many solutions, or one unique solution or else no solution. If the solution to the system exists, then it is given by

$$\mathbf{x} = A^{-1}\mathbf{b} \,, \tag{3}$$

where A^{-1} is the inverse matrix of A (note that if A is a $M \times N$ matrix, then A^{-1} is a $N \times M$ matrix), i.e., is such that $A^{-1}A = \mathbb{I}$ (where $\mathbb{I}$ is here the $N \times N$ identity matrix) and $AA^{-1} = \mathbb{I}$ (while $\mathbb{I}$ is here the $M \times M$ identity matrix). Note also that the elements of A^{-1}, $(A^{-1})_{ij}$, are **not** the inverse of each element of A, A_{ij}^{-1}. Show that explicitly!

For example let's consider the simplest case of $M = N = 2$, a system of 2 equations with two unknowns:

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 &= b_1 \\ A_{21}x_1 + A_{22}x_2 &= b_2 \,. \end{aligned}$$

Each equation is a line in the plane (x_1, x_2), therefore is clear that the solution will be the intersection of these two lines: this can either be a line (infinitely

many solutions), a point (one unique solution) or else the two lines never meet (no solution).

Matlab has built-in commands to solve systems of linear equation and calculate inverse matrices

```
x=mldivide(A,b)
x=A\b
x=inv(A)*b
```

1.0.1 Exercise:

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 3 & 1 \\ 4 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} ,$$

find the solution using the command `mldivide(A,b)` (or equivalently `A\b`) and compare it with the exact solution you find analytically as well as with `inv(A)*b`. Plot the two lines $A_{11}x_1 + A_{12}x_2 = b_1$ and $A_{21}x_1 + A_{22}x_2 = b_2$ in the plane (x_1, x_2) and check they intersect at the point previously found.

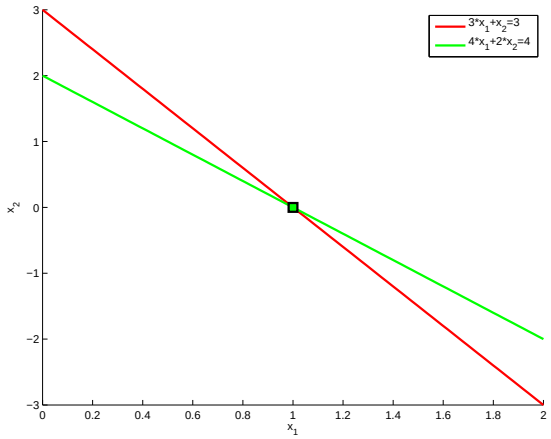


Figure 1: Plotting the two lines $A_{11}x_1 + A_{12}x_2 = b_1$ and $A_{21}x_1 + A_{22}x_2 = b_2$ in the plane (x_1, x_2) of the Ex. 1.0.1.

Similarly in the case of N variables, each equation of the system describes a hyper-plane in the N-dimensional space $(x_1, \ldots, x_N)$ and the solution to the system is the intersection of these hyper-planes.

1.0.2 Exercise:

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 1 & 5 & 4 \\ 7 & 9 & 3 \\ 4 & 5 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 4 \\ 9 \end{pmatrix} ,$$

find the solution using the command `mldivide(A,b)` and compare it with `inv(A)*b` — setting `format long`.

1.0.3 Exercise:

Find the solution of the system of linear equations $\mathbf{x}^T A = \mathbf{b}^T$, where A and $\mathbf{b}$ are given in the Ex. 1.0.1. Compare the solution with the one obtained by plotting the lines defined by each one of the two equations of the system. Check that `(b/A)'=A'\b'`, i.e. $(\mathbf{b}^T A^{-1})^T = (A^{-1})^T \mathbf{b}$.

In general, for M **linearly independent** equations[1] (i.e. none of the equations can be derived from the others) we can have the following situations

1. **Undetermined system:** If $M < N$ there are less equations than unknowns and the system has infinitely many solutions. For example when $N = 2$ and $M = 1$, the solutions lie on the line defined by the single equation in two variables. In this case the dimension of the solution is equal to $2 - 1 = 1$ — and in general by $N - M$;

2. There is a **unique solution** when $N = M$ and when the square matrix A has an inverse A^{-1}; the solution is given by $\mathbf{x} = A^{-1}\mathbf{b}$;

3. **Overdetermined system:** There are **no solutions** when instead $M > N$, i.e. there are more equations than unknowns. For example, if $N = 2$ and $M = 3$, then each equation of the system describes different lines and they can meet in three different points.

1.0.4 Exercise:

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 1 & -2 \\ 2 & -4 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 5 \\ 6 \end{pmatrix} .$$

Discuss why there are no solutions to this system of linear equations. What does it mean that $\det(A) = 0$?

[1] One can make sure this is the case by checking that $\det(A)$ is finite. For a singular matrix A, i.e. $\det(A) = 0$, at least two equations of (1) are linearly dependent.

1.0.5 Exercise:

Consider the system of linear equations (1) with

$$A = \begin{pmatrix} 1 & 2 \\ 3 & -4 \\ 3 & 2 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 5 \\ 6 \\ 8 \end{pmatrix} .$$

Explain why there is no solution to such a system of linear equations by plotting the three lines it describes. What happens if you evaluate `A\b`? What is Matlab finding? What does it minimise? (Hint: type `help mldivide` and `doc mldivide`).

Solving systems of linear equations can also be useful for non-linear functions, as long as these functions have the same structure, as in the following exercise.

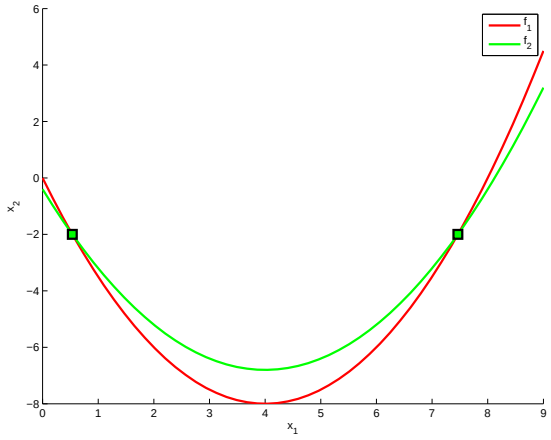


Figure 2: Plotting the two functions $f_1(x) = \left[3 - \left(3 + 8x - x^2\right)\right]/2$ and $f_2(x) = \left[4 - 2\left(3 + 8x - x^2\right)\right]/5$ of the Ex. 1.0.6.

1.0.6 Exercise:

Consider the following non-linear system of two equations in two unknowns:

$$2y + \left(3 + 8x - x^2\right) = 3$$
$$5y + 2\left(3 + 8x - x^2\right) = 4\ .$$

1. Plot the two functions $y = f_1(x) = \left[3 - \left(3 + 8x - x^2\right)\right]/2$ and $y = f_2(x) = \left[4 - 2\left(3 + 8x - x^2\right)\right]/5$ and establish graphically the values of the two intersection points (x_1, y_0) and (x_2, y_0);
2. introduce a new variable, $z = 3 + 8x - x^2$ and solve the system of two linear equations for y_0 and z_0:

$$2y + z = 3$$
$$5y + 2z = 4\ .$$

3. Now solve $z_0 = 3 + 8x_{1,2} - x_{1,2}^2$ and verify that the solutions you found above, (x_1, y_0) and (x_2, y_0) coincide with the intersection points found in the first point 1. Plot the two points on the graph as in Fig 2.

2 Application: Electrical circuits

Currents I_i and potential differences V_j in electrical circuits are ruled by the Kirchhoff's circuit laws. The first (current) law states that, at any node in an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing away from that node:

$$\sum_{i \in \text{into the node}} I_i = \sum_{k \in \text{out from the node}} I_k\ .$$

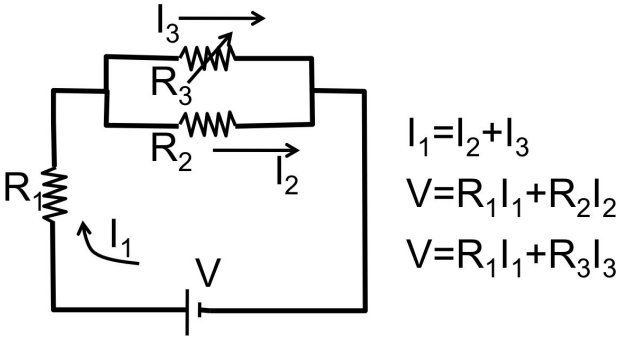
This law reflects the principle of conservation of electric charge. On the contrary, Kirchhoff's second law applies to closed loops in an electrical circuit and states that the sum of the electrical potential differences (i.e., the voltages) around any closed network is zero,

$$\sum_{j \in \text{network}} V_j = 0\ .$$

This law follows from the principle of conservation of energy.

2.0.7 Exercise:

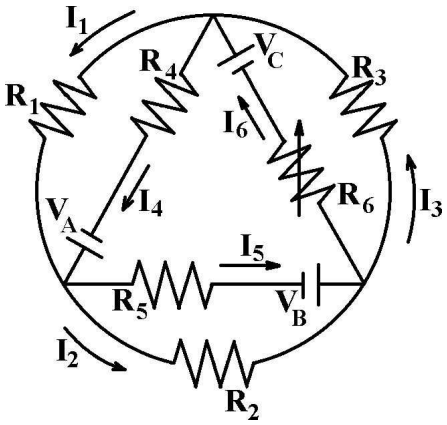
Applying the two Kirchhoff's circuit laws, find the three currents, I_1, I_2, and I_3, of the following electric circuit:



by knowing the values of the voltage $V = 10$ V, and the resistances $R_1 = 2\ \Omega$, $R_2 = 4\ \Omega$, and $R_3 = 3\ \Omega$. By assuming that R_3 is a variable electric resistance, plot the three currents as function of $R_3 \in [0, 100]\ \Omega$.

2.0.8 Exercise:

Applying the two Kirchhoff's circuit laws, find the six currents, $I_1, \ldots, I_6$ of the following electric circuit:



Plot all currents knowing the values of the voltages $V_A = 3$ V, $V_B = 2$ V, and $V_C = 3$ V, and the resistances $R_1 = 2\ \Omega$, $R_2 = 1\ \Omega$, $R_3 = 2\ \Omega$, $R_4 = 2\ \Omega$, $R_3 = 1\ \Omega$, as a function of the variable resistance $R_6 \in [0, 3]\ \Omega$.

3 Zeros of a function: bisection method

Summary of definitions The bisection method is the easiest algorithm one can think of that finds the root of a function, $f(x) = 0$, in a given interval $[a, b]$. The interval has to be chosen so that one knows for sure in advance that one of the zeros lies inside it. The algorithm repeatedly divides the interval in half and selects the subinterval in which the root must lie. Despite its simplicity this method is quite slow.

The structure of the algorithm (see Fig. 3) is the following one:

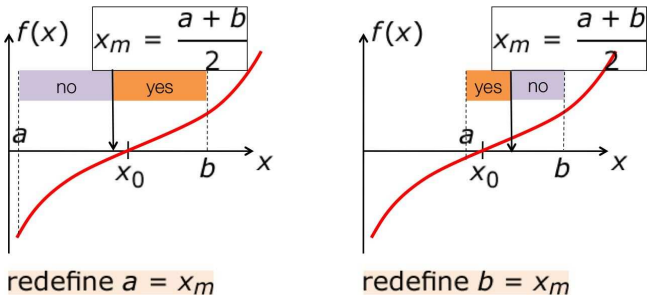


Figure 3: Schematic representation of the way the bisection method works.

```
a=...; b=...; fa=...; fb=...;
while b-a>small number
           xm=(a+b)/2; fm=...;
           if (fa*fm)>0
              a=xm
           else
              b=xm
           end
end
```

For the *small number* you can choose whatever number you want the accuracy of your answer to be in the . Type `help eps` and `doc eps`.

3.0.9 Exercise:

Find the two zeros of the function $U(x) = -3x^2 + x^4 + x^5/10$ in the two intervals $[1, 2]$ and $[-3, -1]$ by writing a bisection algorithm. Compare the results you find in this way with the ones you obtain graphically by plotting the function (see Fig. 2). Finally find the zeros of $U(x)$ by using the built-in Matlab routine `fzero(f,[1,2])` — N.B. you need to first write an anonymous function, i.e. `f = @(x)-3*x^2+x^4+x^5/10`.

3.0.10 Exercise:

Use the bisection method in order to find the zero of the function $f(x) = x^3 - 7x^2 + 14x - 6$ in within the interval $[0, 1]$, with an accuracy of 10^{-2}.

4 Roots of a function: Newton-Raphson method

A better (i.e., converging faster) algorithm to find the zeros (or roots) of a function than the bisection method is the Newton-Raphson method. In the root-finding process, this method uses not only the actual values of the function but also its first derivatives. It is based on the fact that, if we Taylor expand the function $f(x)$ up to the first order term at the point x_0, which is close to

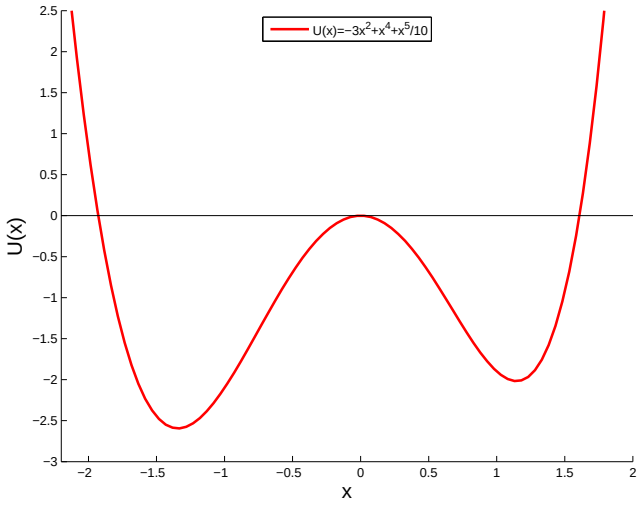


Figure 4: Plot of the function $U(x) = -3x^2 + x^4 + x^5/10$.

the root of $f(x)$,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + O((x - x_0)^2) \,, \tag{4}$$

as we are looking for the solution to $f(x_1) = 0$, then, ignoring higher order terms, we get that

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \,. \tag{5}$$

The point x_1 found as described above is not the real root of $f(x)$ because we have truncated the Taylor expansion, but if we apply iteratively this formula, expanding now around x_1 and so on and so fort, we can converge quickly to the real zero of the function. This method can converge very quickly, and faster than the bi-section method, if the initial guess of the root is quite close to the one we are looking for. In addition, the derivative of $f(x)$ should neither be zero nor infinite in the region of interest.

The implementation of the method is quite easy. Let's suppose we want to find numerically the root of the function $f(x) = \exp(x) - x - 4$. By plotting $f(x)$ (see Fig. 3), we can see the root is around $x \simeq 1.7$. We start giving a first guess of the root, say $x_0 = 2.5$; evaluating the tangent line to $f(x)$ at $x_0 = 2.5$,

$$f(x) = (e^{2.5} - 1)(x - 2.5) + e^{2.5} - 2.5 - 4 + O((x - 2.5)^2) \,, \tag{6}$$

we can find an approximation of the root of $f(x)$ better than $x_0 = 2.5$ by finding the root x_1 of $f(x)$ approximated as in the above expression around $x_0 = 2.5$, which is $x_1 \simeq 1.99$ (see Fig. 3). If we now implement to the next step the above procedure, by defining as new initial guess $x_0 = x_1 \simeq 1.99$, then the new approximated value of the zero will give $x_1 \simeq 1.78$, which is already very close (after two steps only!) to the real zero of the function $f(x) = \exp(x) - x - 4$, which is $x \simeq 1.75$ (check this yourself).

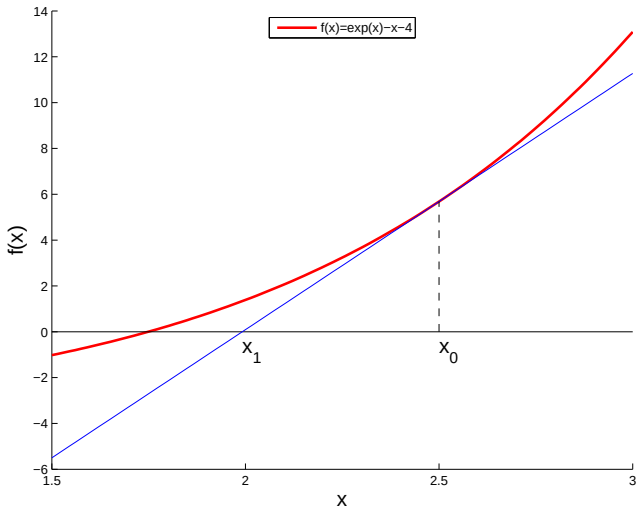


Figure 5: Function $f(x) = \exp(x) - x - 4$ and first step of the Newton-Raphson method if we use as initial guess $x_0 = 2.5$.

4.0.11 Exercise

Write a routine which reproduces Fig. 3 for $f(x) = \exp(x) - x - 4$, plot $f_1(x)$ for $x_0 = 2.5$ and find x_1. (Optional: Export the figure into an .eps file).

4.0.12 Exercise

Find the zero of $f(x) = \exp(x) - x - 4$ by using the built-in Matlab routine `fzero(f,2)` — N.B. you need to first write an anonymous function, i.e. `f = @(x)exp(x)-x-4`.

4.0.13 Exercise

Find the zero of the function $f(x) = \exp(x) - x - 4$ by writing a Newton-Raphson algorithm. Use as initial guess $x_0 = 2.5$. Set `format long` and compare the result obtained in this exercise with the one obtained in the previous exercise.

Hints to solve Exercise 4.0.13

- Given the initial guess x_0, evaluate the derivative of $f(x)$ in x_0, $f'(x_0) \simeq [f(x_0) - f(x_0 - \delta x)]/\delta x$, where δx is a small number;
- once you know the derivative, use Eq. (5) to find the next guess x_1;
- build a `while` loop which runs until the condition $f(x_1) >$ eps is satisfied.

4.0.14 Exercise

Solve the previous exercise by writing a bisection algorithm. Set `format long` and compare the result obtained in this exercise with the ones obtained in the previous two exercises.

4.0.15 Exercise

Count the number of times the loop `while` is called in the Newton-Raphson algorithm you developed in Ex. 4.0.13 and how many times instead is called in the bisection algorithm you developed in Ex. 4.0.14. For the particular case of the function $f(x) = \exp(x) - x - 4$, which algorithm is more efficient and why?

4.0.16 Exercise

Write a routine using the Newton-Raphson algorithm which evaluates the minimum of the function $f(x) = -3x^2 + x^4 + x^5/10$ (shown, together with its derivative in Fig. 4) close to $x_0 = 1$. Compare the result you get this way with the one you obtain by writing a bisection method. Indicate which method is more efficient and why.

Hints to solve Exercise 4.0.16

- Remember that at the maximum or at a minimum of a function, $f'(x_0) = 0$ — and in this specific case, $f'(x) = -6x + 4x^3 + x^4/2$;
- apply the Newton-Raphson algorithm to find the root of $f'(x)$ close to $x_0 = 1$ — i.e., Eq. (5) now reads $x_1 = x_0 - f'(x_0)/f''(x_0)$, where $f''(x) = -6 + 12x^2 + 2x^3$.

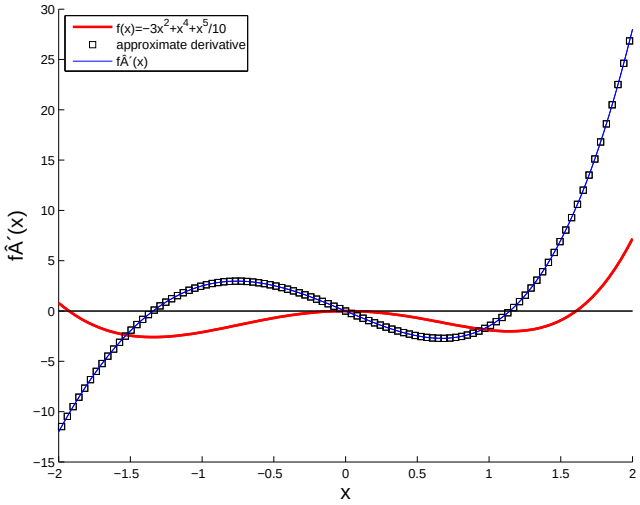
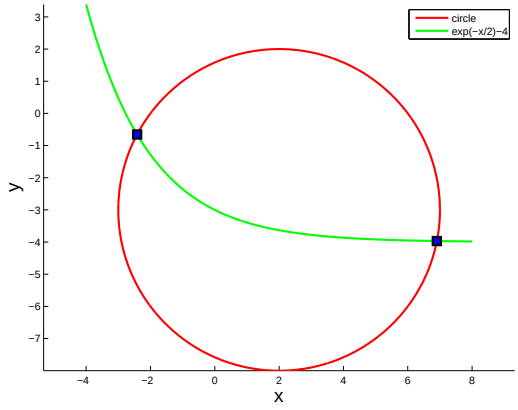


Figure 6:

4.0.17 Exercise

Find the crossing points between a circle centered in $(x_0, y_0) = (2, -3)$ and radius $R = 5$ and the exponential function $f(x) = e^{-x/2} - 4$ (answer: $x_1 \simeq -2.42$ and $x_2 \simeq 6.91$).



5 Application: energy conservation

In mechanics, the conservation of energy means that the sum of the kinetic energy T and the potential energy U is a constant, E. In case of a particle moving in one dimension, the conservation of energy reads:

$$E = T + U(x) = \frac{1}{2} m \left(\frac{dx}{dt}\right)^2 + U(x) \,, \tag{7}$$

where $v = dx/dt$ is the velocity of the particle and m its mass. As a consequence, the velocity of the particle can be evaluated as a function of the position only,

$$\frac{1}{2} m \left(\frac{dx}{dt}\right)^2 = E - U(x) \,,$$

which means that the motion is restricted only to the region where $U(x) \leq E$. At the points x_i for which $E = U(x_i)$, the particle has zero velocity — points of motion inversion. Finally, the force in terms of the potential energy reads:

$$F(x) = -\frac{dU(x)}{dx} \,. \tag{8}$$

5.0.18 Exercise

Consider the case of the energy potential $U(x)/E_c = -3x^2 + x^4 + x^5/10$ plotted in Fig. 5. Evaluate the points of motion inversion x_1 and x_2 for the total energy $E/E_c = -1$.

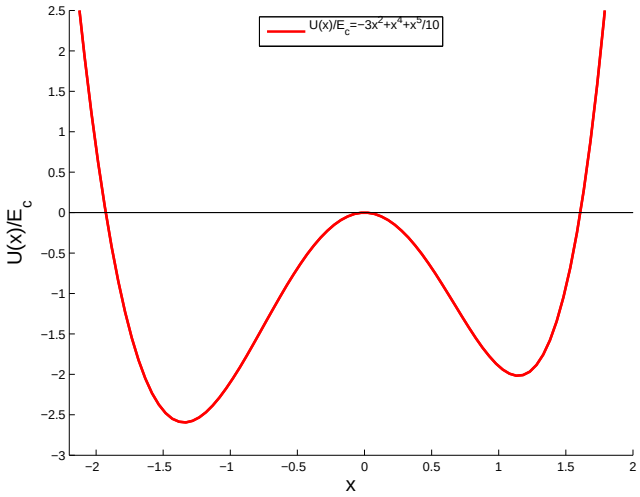


Figure 7:

6 Example of external function routine for the bisection method

We remember that in order to create an external function script we have to use the following syntax,

$$[\texttt{a}, \texttt{b}, \texttt{c}] = \texttt{your\_function}(\texttt{x}, \texttt{y}, \texttt{z}, \texttt{t})$$

and that you need to write a separate script file and name it your_function.m. In this file, `x, y, z, t` are INPUT variables, while `a, b, c` are OUTPUT variables, i.e., what the function returns. To use this function in a script, you then call the function by `[a, b, c] = your_function(x,y,z,t)`. Note that the names of the variables inside the function do not need to be the same as the names of the variables you pass to the function!

The following example (see Fig. 8) creates a function script zero_bisection.m that evaluates the zero of a function in a given interval `(a, b)` and with a given accuracy `err`. Thus `a, b, err` are input values, while `x_0, fx_0, iter` are output values: `x_0` is the approximated value of the zero, `fx_0` the value of the function at this point, `iter` the number of iterations the programme needed to find the zero with the accuracy `err` initially required. This external function can be used in the following script

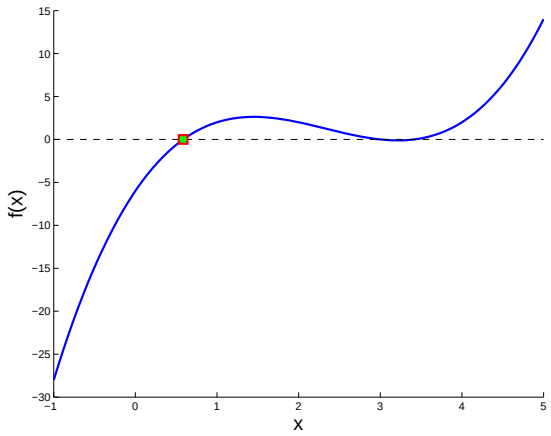


Figure 8: Result of the script that uses the function script zero_bisection.m, finding the zero of the function $f(x) = x^3 - 7x^2 + 14x - 6$.

```
clear all
close all
clc
% example on how to use the external function zero_bisection.m
fun=@(x)(x.^3-7*x.^2+14*x-6);

a=-1; b=1.5; err=2*eps;
[ x_0, fx_0, iter ] = zero_bisection( fun, a, b, err );
x_0
fx_0
iter

a=-1; b=5; N=100;
x=linspace(a, b, N);
hold on
plot(x,fun(x),'Linewidth', 2)
plot(x,zeros(length(x),1),'k--')
plot(x_0,fx_0,'sr')
hold off
```

```
function [ x_0, fx_0, iter ] = zero_bisection( fun, a, b, err )
% bisection method in order to find the root of the function f
% INPUT
% fun function
% a
% b interval [a, b]
% OUTPUT
% x_0 root
% fx_0 value of the function at x_0
% iter number of iterations used

fa=fun(a);
fb=fun(b);
i=0
while b-a>2*err
      %define the middle point
      x=(a+b)/2;
      fx=fun(x);
      if (fa*fx)>0
           %eliminate left half interval
           a=x;
      else
           %eliminate right half interval
           b=x;
      end
      i=i+1;
end
x_0=x;
fx_0=fun(x_0);
iter=i;
end
```

6.0.19 Exercise

By writing external function routines for both the bisection method as well as the Newton-Raphson method, find the zero of the function $f(x) = \exp(x) - x - 4$ with both methods.

6.0.20 Exercise

By writing an external function routine, find the crossing points between a circle centered in $(x_0, y_0) = (2, -3)$ and radius $R = 5$ and the exponential function $f(x) = e^{-x/2} - 4$ (answer: $x_1 \simeq -2.42$ and $x_2 \simeq 6.91$).