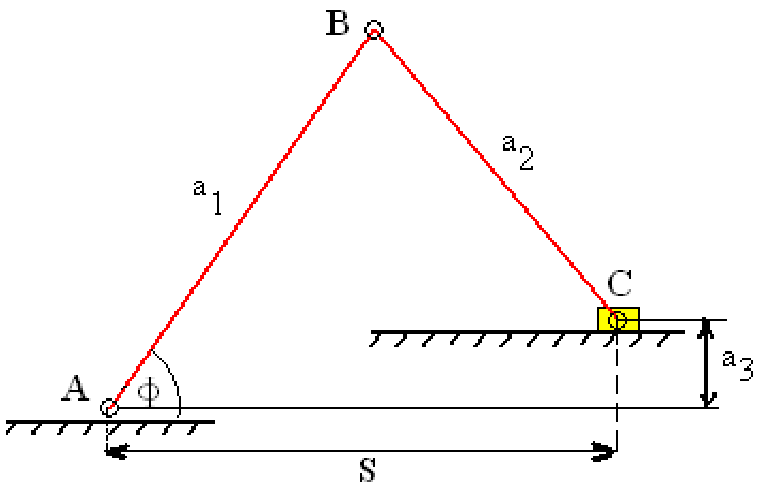
The Numerical Solution of the Nonlinear Systems of Equation

Assume that an articulated structure consists of two rigid rods the lengths of which are denoted by a1 and a2. The two rods are connected to each other by a hinge so the rods can rotate freely round point B. (See the figure below.) We have fixed the end point A to a horizontal platform by a hinge and a solid to the end point C, which can freely slide on the horizontal platform. The vertical distance between the two horizontal platforms are denoted by a3.



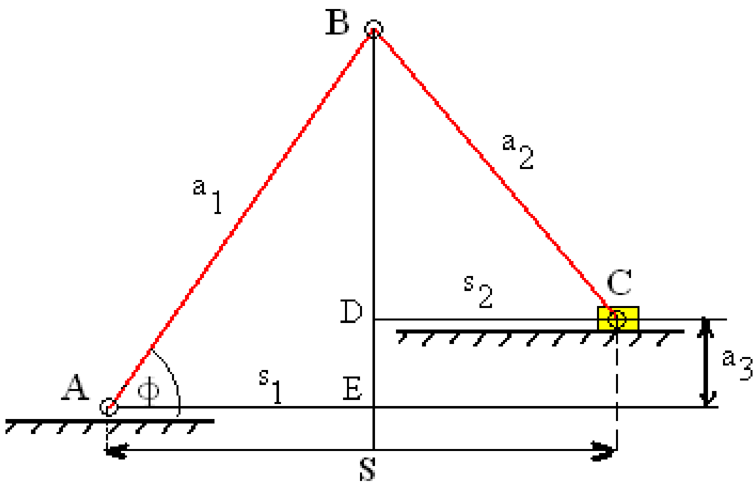
By knowing the a1, a2 and a3 distances and the Phi angle, the S distance can be determined easily, which is the distance measured horizontally between the points A and C. In practice, it is the other way round: the a1, a2 and a3 distances matching the Phi and S values related have to be determined.

We want the construction to be created to meet the following conditions:

1. *ha Phi = 20 `°` , akkor s = 10.5 [cm],*
2. *ha Phi = 45 `°` , akkor s = 8 [cm],*
3. *ha Phi = 60 `°` , akkor s = 5.7 [cm].*

Plan the articulated structure by giving the a1, a2 and a3 distances that satisfies all the three positional conditions.

Let's start the solution by determining the S distance as if we knew the a1, a2 and a3 distances and the Phi angle. For this, divide the horizontal S distance to s1+s2 sum. We can do the division by the E point which is the perpendicular projection of the B apex.



Write the cosine of the Phi angle into the ABE right angle triangle. By rearranging, we get the s1=AE leg because the Phi angle and the a1 hypotenuse are known.

> $restart : s_1 := a_1 \cos(\text{Phi})$

$$s_1 := a_1 \cos(\Phi) \quad \textbf{(1)}$$

Then apply the Pythagorean theorem in the BCD right angle triangle to get the s2 leg. In this formula, the BD leg is given by the difference of the EB=a1sin(Phi) and a3 distances.

> $s_2 := \sqrt{{a_2}^2 - BD^2};$

$$s_2 := \sqrt{a_2^2 - BD^2} \quad \textbf{(2)}$$

> $BD := a_1 \cdot \sin(\text{Phi}) - a_3;$

$$BD := a_1 \sin(\Phi) - a_3 \quad \textbf{(3)}$$

By the s=s1+s2 addition, we can get the equation we were looking for among the s, a1, a2, a3 and Phi variables.

> $egyenlet := s = s_1 + s_2$

$$egyenlet := s = a_1 \cos(\Phi) + \sqrt{a_2^2 - (a_1 \sin(\Phi) - a_3)^2} \quad \textbf{(4)}$$

To make the square root sign disappear, rearrange the equation in a way that the expression containing the root should stand alone on the right side of the equation, then square the equation.

> $egyenlet - 'a_1 \cdot \cos(\text{Phi}) = a_1 \cdot \cos(\text{Phi})'$

$$s - a_1 \cos(\Phi) = \sqrt{a_2^2 - (a_1 \sin(\Phi) - a_3)^2} \quad \textbf{(5)}$$

> $gyoktelen := expand(map(x \to x^2, (5)));$

$$gyoktelen := s^2 - 2\, a_1 \cos(\Phi)\, s + a_1^2 \cos(\Phi)^2 = a_2^2 - a_1^2 \sin(\Phi)^2 + 2\, a_1 \sin(\Phi)\, a_3 - a_3^2 \quad \textbf{(6)}$$

Sort the equation to zero. We can do this by extracting the right side from the left side.

> $nemlinearis := lhs((6)) - rhs((6)) = 0$

$$nemlinearis := s^2 - 2\, a_1 \cos(\Phi)\, s + a_1^2 \cos(\Phi)^2 - a_2^2 + a_1^2 \sin(\Phi)^2 - 2\, a_1 \sin(\Phi)\, a_3 + a_3^2 = 0 \quad \textbf{(7)}$$

We have got the nonlinear equation among the s, Phi, a1, a2 and a3 variables. In fact, we want to determine the a1, a2 and a3 unknowns in a way that we substitute the relating values of Phi and s

Phi $= 20$ \`°\` , $s = 10.5$

Phi $= 45$ \`°\` , $s = 8$

Phi $= 60$ \`°\` , $s = 5.7$

into the equation. We have received three nonlinear equations in this way. Call them e1, e2 and e3.

> $e_1 := nemlinearis \Big|_{s = 10.5,\ \text{Phi} = \frac{\pi}{9}}$

$$e_1 := 110.25 - 21.0\, a_1 \cos\left(\frac{1}{9}\pi\right) + a_1^2 \cos\left(\frac{1}{9}\pi\right)^2 - a_2^2 + a_1^2 \sin\left(\frac{1}{9}\pi\right)^2 - 2\, a_1 \sin\left(\frac{1}{9}\pi\right) a_3 + a_3^2 = 0 \quad \textbf{(8)}$$

> $e_2 := nemlinearis \Big|_{s = 8,\ Phi = \frac{\pi}{4}}$

$$e_2 := 64 - 8\, a_1 \sqrt{2} + a_1^2 - a_2^2 - a_1 \sqrt{2}\, a_3 + a_3^2 = 0 \quad \textbf{(9)}$$

> $e_3 := nemlinearis \Big|_{s = 5.7,\ \text{Phi} = \frac{\pi}{3}}$

$$e_3 := 32.49 - 5.700000000\, a_1 + a_1^2 - a_2^2 - a_1 \sqrt{3}\, a_3 + a_3^2 = 0 \quad \textbf{(10)}$$

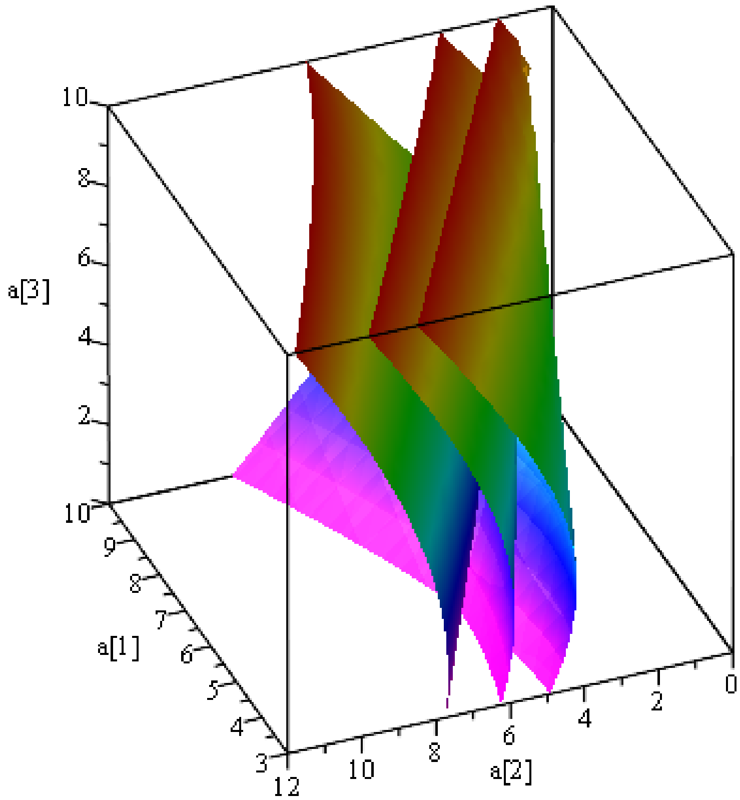
We have to find the solution of the [képlet] system of equation for the [képlet] unknowns.

First, let's start with the geometrical illustration of the task. The e1, e2 and e3 equations determine a second-order surface in the R3 3D plane. The implicitplot3d instruction of the plots package was designed to display the points of the surfaces given as an implicit.

> $with(plots):$

> $implicitplot3d(\{e_1, e_2, e_3\}, a_1 = 3..10, a_2 = 0..12, a_3 = 0..10, grid = [15, 15, 15], axes = boxed, style = patchnogrid, orientation = [158, 56], shading = ZHUE);$

```
Warning, the name changecoords has been redefined
```



The three surfaces proceed close to each other like the petals of a rose. We can see that the surfaces cross each other. But is there only one point in the plane that all the three surfaces cross? It is impossible to decide based on the graph.

Let's see the numerical approach. At first, we cannot give a search range for the fsolve procedure. In this case, it gives the approximate value of a root, if such a root exists.

> $fsolve(\{e_1, e_2, e_3\}, \{a_1, a_2, a_3\});$

$$\{a_2 = 5.182580039,\ a_1 = 5.838867375,\ a_3 = 0.6830978044\} \tag{11}$$

It seems we have received the root we wanted. But how can we decide if there are more roots? Shall we look for a procedure that operates based on another solution algorithm? We can quickly find the answer. Since the equations are the second degree polynomials of the a1, a2 and a3 variables, we can use the Homotopy procedure of the Rootfinding package for the solution of the system of equation. This procedure was designed for the numerical determination of all the roots of the polynomial system of equation. The procedure should be given the polynomials but the variables need not be listed because it considers the symbols variables. The procedure looks for the solution in the whole complex plane.

> $RootFinding_{Homotopy}([lhs(e_1), lhs(e_2), lhs(e_3)]);$

$$[[a_1 = 5.838867373 - 0.\ I,\ a_3 = 0.6830977967 - 0.\ I,\ a_2 = -5.182580045 + 0.\ I],\ [a_1 \tag{12}$$

$= 5.838867373 - 0.\ I, a_3 = 0.6830977967 - 0.\ I, a_2 = 5.182580045 - 0.\ I]]$

The procedure returned three real roots. It is interesting that all the roots appear in a complex form although their imaginary unit is 0. The difference between the two solutions is that the value of the a2 appearing in one of the roots is the minus one fold of the one located in the other root.

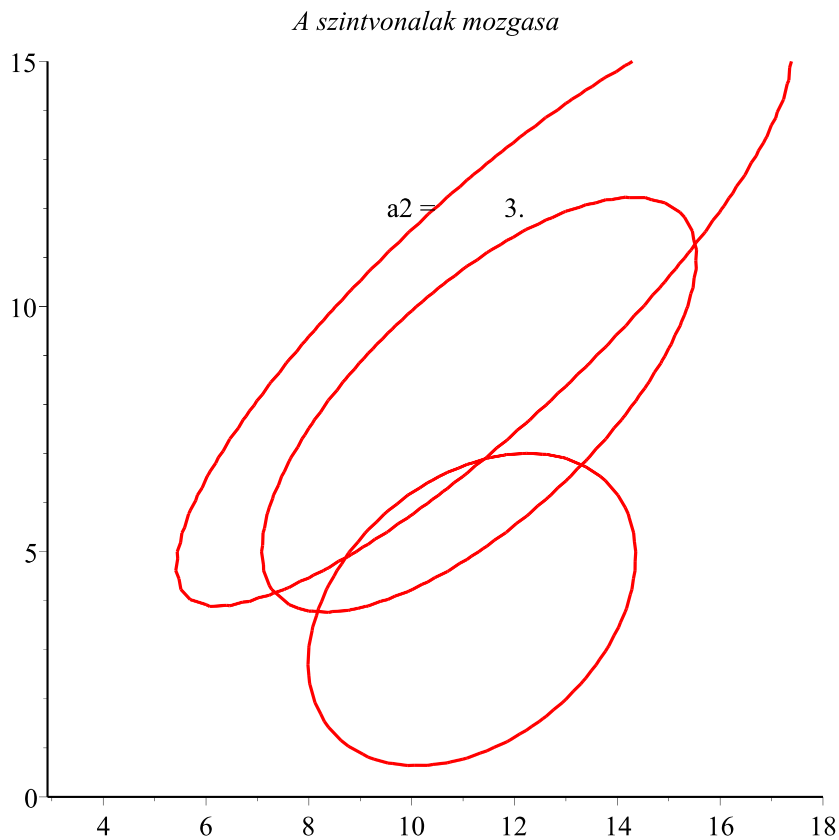
But the original task cannot have two negative a2 solutions because a2 denotes a distance. Notice that the a2 variable is rooted in each of the e1, e2 and e3 equations. Thus if a triple [képlet] satisfies the [képlet] system of equation then the [képlet] does so.

The positive root coincides with the roots found thus we can be sure that the approximation value of the only positive root is [képlet].

Before introducing the Newton-Raphson interation algorithm and how the solution can be found with this method, we are going to show another interesting way to find the intersection. If we fix one of the variables in the three equations, e.g. the value of the a2, then only the a1 and a3 variables will be unknown in the [képlet] equations. Thus based on the three implicit equations, we can plot three plane curves with the 2D implicitplot procedure. After this, we change the value of the a2 and watch the alternation in an animation window. If we are lucky, we can see the three curves crossing one point in the case of an a1, a2 or a3 value. We have written an animation program for this.

```
> nivok := NULL;
  for k from 0 to 20 do
   nivo:=implicitplot(subs(a[2] = 3+1/8*k,[e[1],e[2],e[3]]),
                            a[1]=3..18,a[3]=0..15,grid = ([40,40]));

   szoveg:=textplot([[10,12,"a2 ="],[12,12,evalf(3+1/8*k,3)]]);
   nivok:=nivok, display([nivo,szoveg])
  od;
  display([nivok],insequence=true,title=`A szintvonalak mozgasa`)
```



We had already known the approach of the solution above when we did the animation thus we were able to choose the right value domains of the variables. The animation perfectly illustrates that all three contours cross one point in the graph belonging to a2=5 and this point is approximately the [képlet] coordinate point.

We are deducing a calculation procedure that approximates in second order to find the location of the root. This procedure is the generalisation of the Newton tangent method, known for the functions with one variable, for the functions with more variables. We give the description only for three variables but the formulas are similar in the case of an arbitrary n variable. Consider the

$$\begin{bmatrix} f_1(x, y, z) \\ f_2(x, y, z) \\ f_3(x, y, z) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

nonlinear system of equation. Create an F vector-vector function from the functions located on the left side of the equations.

$$F(X) = \begin{bmatrix} f_1(x,y,z) \\ f_2(x,y,z) \\ f_3(x,y,z) \end{bmatrix}, \text{ ahol } X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Thus the syntax of the system of equation is F(x)=0 where the 0 on the right side is the 3D null vector. The so-called Newton-Raphson iteration method is based on the Taylor polynomial approximation of the equations. Its syntax is the following:

$$f_1(x+\text{delta}_x, y+\text{delta}_y, z+\text{delta}_z) = f_1(x,y,z) + \left(\frac{\partial}{\partial x} f_1(x,y,z)\right)\text{delta}_x + \left(\frac{\partial}{\partial y} f_1(x,y,z)\right)\text{delta}_y + \left(\frac{\partial}{\partial z} f_1(x,y,z)\right)\text{delta}_z + \textit{magasabb rendü tagok}$$

$$f_2(x+\text{delta}_x, y+\text{delta}_y, z+\text{delta}_z) = f_2(x,y,z) + \left(\frac{\partial}{\partial x} f_2(x,y,z)\right)\text{delta}_x + \left(\frac{\partial}{\partial y} f_2(x,y,z)\right)\text{delta}_y + \left(\frac{\partial}{\partial z} f_2(x,y,z)\right)\text{delta}_z + \textit{magasabb rendü tagok}$$

$$f_3(x+\text{delta}_x, y+\text{delta}_y, z+\text{delta}_z) = f_3(x,y,z) + \left(\frac{\partial}{\partial x} f_3(x,y,z)\right)\text{delta}_x + \left(\frac{\partial}{\partial y} f_3(x,y,z)\right)\text{delta}_y + \left(\frac{\partial}{\partial z} f_3(x,y,z)\right)\text{delta}_z + \textit{magasabb rendü tagok}$$

We did not write but only indicated the second or higher order elements of the approximation because we would not use them. If the deltax, deltay and deltaz denote such values for which the

$$f_1(x+\text{delta}_x, y+\text{delta}_y, z+\text{delta}_z) = 0,$$

$$f_2(x+\text{delta}_x, y+\text{delta}_y, z+\text{delta}_z) = 0 \text{ és}$$

$$f_3(x+\text{delta}_x, y+\text{delta}_y, z+\text{delta}_z) = 0$$

equations are true, that is, we have found the location of the root, then all the left sides are zeros. Furthermore, if we disregard the higher order elements on the right side then we get the following linear system of equation for the deltax, deltay and deltaz variables.

$$-f_1(x,y,z) = \left(\frac{\partial}{\partial x} f_1(x,y,z)\right)\text{delta}_x + \left(\frac{\partial}{\partial y} f_1(x,y,z)\right)\text{delta}_y + \left(\frac{\partial}{\partial z} f_1(x,y,z)\right)\text{delta}_z,$$

$$-f_2(x,y,z) = \left(\frac{\partial}{\partial x} f_2(x,y,z)\right)\text{delta}_x + \left(\frac{\partial}{\partial y} f_2(x,y,z)\right)\text{delta}_y + \left(\frac{\partial}{\partial z} f_2(x,y,z)\right)\text{delta}_z,$$

$$-f_3(x,y,z) = \left(\frac{\partial}{\partial x} f_3(x,y,z)\right)\text{delta}_x + \left(\frac{\partial}{\partial y} f_3(x,y,z)\right)\text{delta}_y + \left(\frac{\partial}{\partial z} f_3(x,y,z)\right)\text{delta}_z.$$

The matrix of the system of equation is the Jacobian matrix

$$Jacobi = \begin{bmatrix} \frac{\partial}{\partial x} f_1(x,y,z) & \frac{\partial}{\partial y} f_1(x,y,z) & \frac{\partial}{\partial z} f_1(x,y,z) \\ \frac{\partial}{\partial x} f_2(x,y,z) & \frac{\partial}{\partial y} f_2(x,y,z) & \frac{\partial}{\partial z} f_2(x,y,z) \\ \frac{\partial}{\partial x} f_3(x,y,z) & \frac{\partial}{\partial y} f_3(x,y,z) & \frac{\partial}{\partial z} f_3(x,y,z) \end{bmatrix}$$

of the $F(X) = \begin{bmatrix} f_1(x,y,z) \\ f_2(x,y,z) \\ f_3(x,y,z) \end{bmatrix}$ vector-vector function.

Based on this, the algorithm is the following. Let’s start with an (x,y,z) approximation value of the root location. Solve the

$$\begin{bmatrix} f_1(x,y,z) \\ f_2(x,y,z) \\ f_3(x,y,z) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f_1(x,y,z) & \frac{\partial}{\partial y} f_1(x,y,z) & \frac{\partial}{\partial z} f_1(x,y,z) \\ \frac{\partial}{\partial x} f_2(x,y,z) & \frac{\partial}{\partial y} f_2(x,y,z) & \frac{\partial}{\partial z} f_2(x,y,z) \\ \frac{\partial}{\partial x} f_3(x,y,z) & \frac{\partial}{\partial y} f_3(x,y,z) & \frac{\partial}{\partial z} f_3(x,y,z) \end{bmatrix} . \begin{bmatrix} \mathrm{delta}_1 \\ \mathrm{delta}_2 \\ \mathrm{delta}_3 \end{bmatrix}$$

linear system of equation for the unknown $\begin{bmatrix} \mathrm{delta}_1 \\ \mathrm{delta}_2 \\ \mathrm{delta}_3 \end{bmatrix}$ vector. With the $\begin{bmatrix} \mathrm{delta}_1 \\ \mathrm{delta}_2 \\ \mathrm{delta}_3 \end{bmatrix}$ vector received,

create the new (“generally better”) approximation of the $\begin{bmatrix} x - \mathrm{delta}_1 \\ y - \mathrm{delta}_2 \\ z - \mathrm{delta}_3 \end{bmatrix}$ root location. Then continue

the iteration with the writing and the solution of the new system of equation while using the new $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

values. The iteration should continue until the appropriate approximation of the root is not reached.
J(-1) denotes the inverse of the Jacobian matrix, that is,

$$\left(J^{(-1)}\right)(X) = \begin{bmatrix} \frac{\partial}{\partial x} f_1(x,y,z) & \frac{\partial}{\partial y} f_1(x,y,z) & \frac{\partial}{\partial z} f_1(x,y,z) \\ \frac{\partial}{\partial x} f_2(x,y,z) & \frac{\partial}{\partial y} f_2(x,y,z) & \frac{\partial}{\partial z} f_2(x,y,z) \\ \frac{\partial}{\partial x} f_3(x,y,z) & \frac{\partial}{\partial y} f_3(x,y,z) & \frac{\partial}{\partial z} f_3(x,y,z) \end{bmatrix}^{(-1)} .$$

So the syntax of the iteration in short is

$$X_{n+1} = X_n - \left(J^{(-1)}\right)\left(X_n\right) F\left(X_n\right) \text{ , ahol } X_0 \text{ adott.}$$

where the X0 is given.

The reciprocal of the derivative is its Jacobian inverse **in** the case of one variable thus the

$$\left(J^{(-1)}\right)(X_n) = \frac{1}{\left(\frac{d}{dX} F(X)\right)\Big|_{X=X_n}}$$

formula has to be applied. This is the formula of the well-known Newton tangent method.

For the convergence of the iteration, it is needed that the [képlet] matrix could be inverted in the environment of the root location. It is fulfilled if [képlet].

After such a long theoretical preparation create the [képlet] functions from the left sides of the polynomial equations received. Then create the F(X) vector-vector function from the functions.

> $f[1] := unapply\left(evalf\left(lhs\left(e_1\right)\right), a_1, a_2, a_3\right)$

$$f_1 := (a\_1, a\_2, a\_3) \rightarrow 110.25 - 19.73354504\, a\_1 + 1.000000000\, a\_1^2 - 1.\, a\_2^2 - 0.6840402866\, a\_1\, a\_3 + a\_3^2 \quad \mathbf{(13)}$$

> $f_2 := unapply\left(evalf\left(lhs\left(e_2\right)\right), a_1, a_2, a_3\right)$

$$f_2 := (a\_1, a\_2, a\_3) \rightarrow 64. - 11.31370850\, a\_1 + a\_1^2 - 1.\, a\_2^2 - 1.414213562\, a\_1\, a\_3 + a\_3^2 \quad \mathbf{(14)}$$

> $f_3 := unapply\left(evalf\left(lhs\left(e_3\right)\right), a_1, a_2, a_3\right)$

$$f_3 := (a\_1, a\_2, a\_3) \rightarrow 32.49 - 5.700000000\, a\_1 + a\_1^2 - 1.\, a\_2^2 - 1.732050808\, a\_1\, a\_3 + a\_3^2 \quad \mathbf{(15)}$$

> $F := \langle f_1(x, y, z), f_2(x, y, z), f_3(x, y, z)\rangle;$

$$F := \begin{bmatrix} 110.25 - 19.73354504\, x + 1.000000000\, x^2 - 1.\, y^2 - 0.6840402866\, x\, z + z^2 \\ 64. - 11.31370850\, x + x^2 - 1.\, y^2 - 1.414213562\, x\, z + z^2 \\ 32.49 - 5.700000000\, x + x^2 - 1.\, y^2 - 1.732050808\, x\, z + z^2 \end{bmatrix} \quad \mathbf{(16)}$$

With the help of the Jacobian procedure of the VectorCalculus package, we create the 3x3 Jacobian matrix because it will be the matrix of the system of equation.

> $J := VectorCalculus_{Jacobian}(F, [x, y, z])$

$$J := \begin{bmatrix} -19.73354504 + 2.000000000\, x - 0.6840402866\, z & -2.\, y & -0.6840402866\, x + 2\, z \\ -11.31370850 + 2\, x - 1.414213562\, z & -2.\, y & -1.414213562\, x + 2\, z \\ -5.700000000 + 2\, x - 1.732050808\, z & -2.\, y & -1.732050808\, x + 2\, z \end{bmatrix} \quad \mathbf{(17)}$$

Let's examine from where we should not start the iteration procedure. For this, calculate the determinant of the Jacobian matrix.

> $LinearAlgebra_{Determinant}(J) = 0; solve(\%, \{x, y, z\})$

$$2.84568453\, y\, x - 1.\, 10^{-9}\, z\, y\, x = 0$$

$$\{y=0.,z=z,x=x\},\ \{x=0.,y=y,z=z\},\ \{z=2.845684530\ 10^{9},y=y,x=x\} \quad \textbf{(18)}$$

It returned that neither the x nor the y variables can be zero and the variable z cannot be 2845684530. Start the iteration from the x=y=z=1 start value. Let's list the steps of the operation then, when reaching the end of the iteration, start again from the first step eight times. In every step, we collect the values of each approximation in the variable bolyong. At the end, we can examine the speed of the convergence.

Step 0: The set of the start value

Initialisation: the set of the x,y,z start values

```
> xk, yk, zk, step := 1, 1, 1, 0;
  bolyong := [xk, yk, zk]
```

$$xk, yk, zk, step := 1, 1, 1, 0$$

$$bolyong := [1, 1, 1] \quad \textbf{(19)}$$

Step 1: The calculation of the Jacobian matrix

The evaluation of the Jacobian matrix in the P(xk,yk,zk) points.

```
> step := step + 1;
  Jacobi := subs(x = xk, y = yk, z = zk, J);
```

$$step := 8$$

$$Jacobi := \begin{bmatrix} -8.523076706 & -10.36516060 & -2.627824858 \\ -.6020199459 & -10.36516060 & -6.891209782 \\ 4.794574621 & -10.36516060 & -8.747019309 \end{bmatrix} \quad \textbf{(20)}$$

Step 2: The solution of the system of equation

Solve the [képlet] linear system of equation with the LinearSolve procedure of the LinearAlgebra package.

```
> delta := LinearAlgebra[LinearSolve](Jacobi, subs(x = xk, y = yk, z = zk, F))
```

$$\delta := \begin{bmatrix} 1.62344743016201612\ 10^{-8} \\ 2.51363147307478495\ 10^{-7} \\ 4.23593448944508702\ 10^{-8} \end{bmatrix} \quad \textbf{(21)}$$

Step 3: The modification of the start value

Modify the xk, yk and zk values according to the

$$xk = xk - \text{delta}_1,\ yk = yk - \text{delta}_2,\ zk = zk - \text{delta}_3$$

formula.

```
> xk := xk - delta[1]; yk := yk - delta[2]; zk := zk - delta[3]; bolyong := bolyong, evalf([xk, yk, zk], 6)
```

$$xk := 5.838867369$$
$$yk := 5.182580051$$
$$zk := 0.6830977880$$
$$bolyong := [1, 1, 1], [5.83887, 0.638588, -.850350], [5.83887, 19.5083, 0.683098], [5.83887, 10.4425, 0.683098], [5.83887, 6.50731, 0.683098], [5.83887, 5.31742, 0.683098], [5.83887, 5.18429, 0.683098], [5.83887, 5.18258, 0.683098], [5.83887, 5.18258, 0.683098] \quad (22)$$

4. lépés: Feltétel vizsgálat

Calculate the sum of the absolute values of the function values at the

$$|f_1(xk, yk, zk)| + |f_2(xk, yk, zk)| + |f_3(xk, yk, zk)| = összeg$$

(xk,yk,zk) approximation location. We can recognise that we are near the root if the value of this sum is almost zero. If [képlet], then we can finish the calculation procedure because we have found the root with epsilon exactness. Otherwise, we execute another iteration step from step 1.

> $osszeg := |f_1(xk, yk, zk)| + |f_2(xk, yk, zk)| + |f_3(xk, yk, zk)|$

$$osszeg := 1.70\ 10^{-8} \quad (23)$$

> $if\ (osszeg > 10^{-6})\ then\ print(`Folytassuk\ az\ 1.\ lépésnél`)\ else\ print(`Vége\ az\ iterációnak!`)\ end\ if$

Vége az iterációnak! **(24)**

If the sum received is larger than 10-6 then return to step 1 because we have not reached the appropriate and exact approximation of the root.

The outputs above show the results after the 8th iteration step. We have got the 10-6 exact numerical approximation of the root which we compare with the result received earlier by the homotopy procedure.

> $`Valós\ megoldások` = map(Re, subs(op(2, \mathbf{(12)}), [a_1, a_2, a_3]));$
> $Elteresek = rhs(\%) - [xk, yk, zk];$

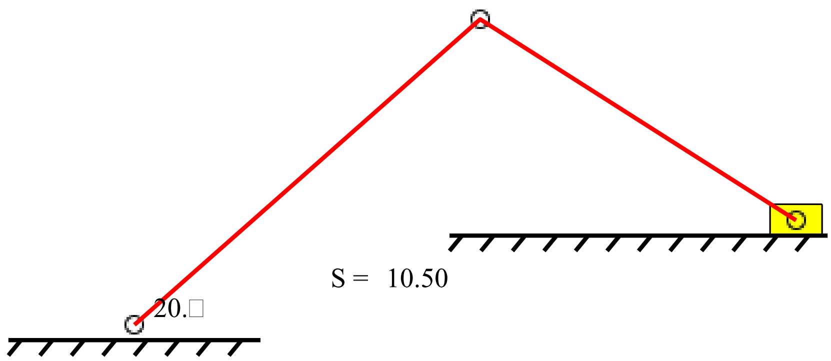
$$Valós\ megoldások = [5.838867373, 5.182580045, 0.6830977967]$$
$$Elteresek = [4.\ 10^{-9}, -6.\ 10^{-9}, 8.7\ 10^{-9}] \quad (25)$$

As we can see, the exactness is in the given tolerance after the 8th iteration step.

Using the solution of the task, we have made an animation. We have written the matching values of the Phi angle and the S distance in the graphs. Thus we can see if the given conditions are satisfied in case the construction is made up from the solution received.

```
> restart;
  a1,a2,a3:= 5.838867375,5.182580039,.6830978044:
> Ax:=t->a1*cos(t):Ay:=t->a1*sin(t):
> Bx:=t->Ax(t)+sqrt(a2^2-(Ay(t)-a3)^2):By:=t->a3:
```

```
> d:=0.1:
> talaj1:=plot([[[5,a3-d],[11,a3-d]],seq([[5+k/2,a3-2*d],[5+k/2+2*
  d,a3-d]],k=0..11)],
                    color=black,thickness=2):
> talaj2:=plot([[[-2,-d],[2,-d]],seq([[-2+k/2,-2*d],[-2+k/2+2*d,-d]
  ],k=0..7)],
                    color=black,thickness=2):
>  rajzok:=NULL:
>  for k from 8 to 24 do
>    t:=k*Pi/72:fok:=evalf(180*t/Pi,3);
>    rudak:=plot(evalf([[0,0],[Ax(t),Ay(t)],[Bx(t),By(t)]]),
  thickness=2):
>    csuklok:=plots[pointplot]({[0,0],[Ax(t),Ay(t)],[Bx(t),By(t)]},
  symbol=circle,symbolsize=18);
>    szoveg1:=plots[textplot]([[1,0.1,`°`],[0.6,0.1,fok]]):
>    szoveg2:=plots[textplot]([[3.5,0.3,`S = `],[4.5,0.3,evalf(Bx
  (t),4)]]):
>    test:=plottools[rectangle]([Bx(t)-4*d,By(t)-d], [Bx(t)+4*d,By
  (t)+d], color=yellow):
>    rajzok:=rajzok,plots[display]([csuklok,rudak,talaj1,talaj2,
  szoveg1,szoveg2,test]):
>  od:
>  plots[display]([rajzok],insequence=true,axes=none);
```



With the run of the animation we can check if the construction satisfies the 3 conditions specified for the matching (Phi,s) pair of values.

$\mathrm{Phi} = 20$ `°` , $s = 10.5$

$\mathrm{Phi} = 45$ `°` , $s = 8$

$\mathrm{Phi} = 60$ `°` , $s = 5.7$

The conditions are satisfied thus we have solved the task. □

>

Mit tanultunk Maple-bl?

- The implicitplot3d instruction plots the set of the (x,y,z) 3D points satisfying the F(x, y,z,)=0 equation in a specific [képlet] rectangular domain. The points usually determine a coherent surface. The procedure is in the plots package and its simplest call is:

$$implicitplot3d(F(x, y, z) = 0, x = a..b, y = c..d, z = e..f, \textit{egyéb opciók});$$

- We can find the numerical solution to the systems of equation with the fsolve procedure. If we did not give a search range then it looks for the solution in the whole interpretation domain of the equations and picks one out of those. In this case the call is the following:

$$fsolve(\{egyenlet_1, egyenlet_2, ..., egyenlet_n\}, \{változó_1, változó_2, ..., változó_k\}).$$

If there is no solution then the response is empty. If we limit the search range of the variables then it looks for the solution only in this domain and if there is a solution it returns it. In this case the call is the following:

$$fsolve(\{egyenlet_1, egyenlet_2, ..., egyenlet_n\}, \{változó_1 = a..b, változó_2 = c..d, ..., változó_k = e..f\})$$

- The Homotopy procedure of the Rootfinding package determines the numerical approximation of all the roots of the polynomial system of equation. The procedure has to be given the polynomials but the variables need not be listed because it considers the symbols variables. The procedure looks for the solution in the whole complex plane. Its call is:

$$Homotopy([polinom_1 = 0, polinom_2 = 0, ..., polinom_n = 0]).$$

The procedure gives the solution in a complex syntax, that is, in the list of lists.

- With the help of the Jacobian procedure of the VectorCalculus package we can determine the 3x3 Jacobian matrix
- $F := \begin{bmatrix} x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} f_1(x, y, z) \\ f_2(x, y, z) \\ f_3(x, y, z) \end{bmatrix}$ *that consists of the partial derivatives of the vector-vector function.*

$$Jacobi = \begin{bmatrix} \frac{\partial}{\partial x} f_1(x,y,z) & \frac{\partial}{\partial y} f_1(x,y,z) & \frac{\partial}{\partial z} f_1(x,y,z) \\ \frac{\partial}{\partial x} f_2(x,y,z) & \frac{\partial}{\partial y} f_2(x,y,z) & \frac{\partial}{\partial z} f_2(x,y,z) \\ \frac{\partial}{\partial x} f_3(x,y,z) & \frac{\partial}{\partial y} f_3(x,y,z) & \frac{\partial}{\partial z} f_3(x,y,z) \end{bmatrix}$$

In the case of the $F = \begin{bmatrix} x \\ y \end{bmatrix} \to \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix}$ vector-vector function the procedure returns the following 2x2 Jacobian matrix.

$$Jacobi = \begin{bmatrix} \frac{\partial}{\partial x} f_1(x,y) & \frac{\partial}{\partial y} f_1(x,y) \\ \frac{\partial}{\partial x} f_2(x,y) & \frac{\partial}{\partial y} f_2(x,y) \end{bmatrix}.$$

Coinciding with the dimensions, the call sequence of the procedure is $Jacobian([f_1(x, y), f_2(x, y)], [x, y])$.

- The Determinant procedure of the LinearAlgebra package calculates the determinant of an nxn M matrix. The long syntax of the call is

$$LinearAlgebra[Determinant](M)$$

- We can get the x solution of the A.x=b system of equation with the LinearSolve procedure of the LinearAlgebra package. The procedure has to be given the A mxn coefficient matrix and the b mxk matrix. In this case the procedure gives the nxk matrix of the x solution. Its call sequence is:

$$LinearAlgebra(A, b).$$

Exercises

Solve the following nonlinear systems of equation and illustrate the solution in 2 and 3D! Use the procedures and methods mentioned in this worksheet.

1.	$x+y-\sqrt{y}-\frac{1}{4}=0$	$8x^2+16y-8xy-5=0$
2.	$e^x+y=0$	$\cosh(y)-x=3.5$

3.	$x_1+x_2+x_3+x_4=5$	$x_1^2+x_2+x_3^2+x_4=12$	$x_1x_2+x_2x_3+x_4=5$	$x_1x_3+x_2x_4+x_4^2=9$
4.	$5x_1+3x_2+x_3+x_4=16$	$x_1x_2+x_2x_3+x_3x_4=17$	$x_1^2+x_2^2+x_3^2-x_4^2=9$	$x_1x_3+x_2x_4+x_1^3=8$
	(megoldás $x_1=1,\ x_2=1,\ x_3=4, x_4=3$)			

5.	$5x_1+3x_2+x_3+x_4=31$	$x_1x_2+x_2x_3+x_4x_5=58$	$x_1^2+x_3x_4-x_2^2+x_1x_5=79$	$x_1-x_2x_4+x_3^2+x_5^3=17$	$x_1x_3-x_2^3x_5-x_5x_2+x_3^2x_4=234$