Computer Assisted Methods in Engineering and Science, **26**: 191–209, 2019, doi: 10.24423/cames.272

Description of large deformations of continuum and shells and their visualisation with *Mathematica*

Ryszard Walentyński

Silesian University of Technology
Faculty of Civil Engineering
Department of Mechanics and Bridges
Akademicka 5, 44-100 Gliwice, Poland
e-mail: Ryszard.Walentynski@polsl.pl

A proper description of large deformation of continuum or shell requires dealing with curved spaces and application of tensor analysis and distinguishing of covariant and contravariant bases. Thanks to symbolic computations and visualization capabilities of the *Mathematica* system, this task can be carried out in a straightforward manner. This has been already discussed in [9] and [10]. This paper is a further extension of these researches. First, it will be shown that the deformation is indeed changing a curvature of the considered space. Next, there will be shown how the Cartesian basis of the undeformed flat space splits into the covariant and contravariant ones and this basis changes in the space. This makes it possible to explain why we have to introduce covariant derivatives and Christoffel symbols, for example. This is important in the case of the optical analysis of large deformations of thin-wall structures. Moreover, it is possible to easily explain that strain tensor is defined with a change of metric tensor. It also helps to show the idea of material (Lagrangian) and spatial (Eulerian) description of the deformation and the motion, and avoid misunderstandings in this matter. Everything is visualised with 3D graphical capabilities and interactive manipulation of the plots provided within the *Mathematica* system. This paper can also be a useful inspiration both in teaching and learning of continuum mechanics, the theory of shells and thin-wall structures. This work has been presented at the conference "4th Polish Congress of Mechanics, 23rd International Conference on Computer Methods in Mechanics" PCM-CMM-2019 in Kraków.

Keywords: continuum mechanics, theory of shells, Mathematica, tensor analysis, thin-wall structures.

1. Introduction

Most of the books on continuum mechanics and the theory of shells contain static drawings describing the problem of the body deformation, which are difficult to understand. An example is presented in Fig. 1 taken from Kiselev *et al.* [2], which nevertheless contains a very clear mathematical description of the problem. Unfortunately, their theoretical introduction to continuum mechanics does not contain an implementation of the *Mathematica* symbolic capabilities.

Another example of static illustrations for the case of the theory of shells is Bielak' work [1]. His approach to the theory of shells was implemented several years ago in my monograph [8]. This was done using external *Mathematica* package *MathTensor* from Parker and Christensen [6]. Unfortunately, *MathTensor* written 25 years ago has some incompatibilities with the current version of *Mathematica*.

Therefore, in this research there are presented some problems in continuum mechanics and the theory of shells using tensor analysis in curved spaces, which can be solved and illustrated with *Mathematica* without use of extra packages. The motivation for that was relatively easy access to *Mathematica* for educational purposes. Students of many universities have campus access to this

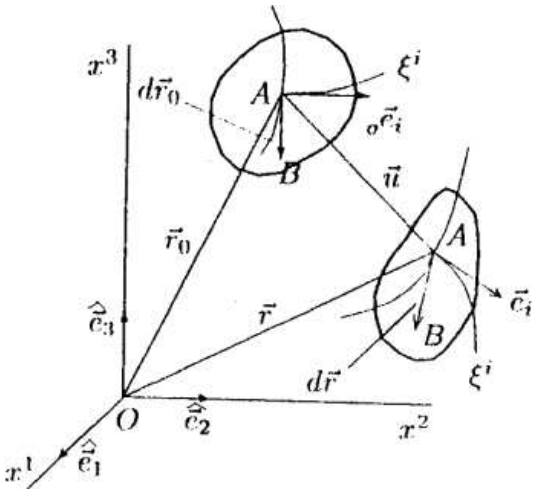

Fig. 1. A traditional illustration of Continuum deformation [2].

software. If not the cheapest, but somehow challenging option is to use Raspberry Pi minicomputer, which is surprisingly inexpensive and its Linux operating system is Raspbian that contains the complementary educational current version of *Mathematica*. Details can be found in [5, 7], for example.

Mathematica contains detailed documentation and access to [11] to learn Wolfram language. Therefore, it is assumed that the reader has at least basic knowledge of this language.

One of the most difficult problems in the education of engineering students is that their understanding of the description of large displacements. The second is the theory of shells, where we must deal with curved spaces. These two fields of mechanics are helpful in understanding of thin-wall structures, especially those with curved geometry. Thus, this paper is divided into two main parts. The next section presents some problems of continuum deformation, and the following section presents some elements of description of bending helicoid to catenoid.

Mathematica, with its symbolic computation and visualization capabilities, can help to explain the difficult problems of large displacements of continuum mechanics and the theory of shells. The presentation is limited to symbolic computations. There is no code for the presented drawings. Unfortunately, it is impossible to present in the printed paper all graphical capabilities of the system shown during the PCM-CMM 2019 conference in Kraków.

2. Description of continuum deformation

The *Mathematica* inputs are denoted with `In[*]:=` and outputs with `Out[*]`, where `*` is a respective number.

First, we format the appearance of the variable ξ[i] in *Mathematica* outputs. This makes it possible to input indices of the spacial (Lagrangian) variables ξ^i with square parantheses and the output is close to the traditional way of notation.

```
In[1]:= ξ[i_] := ξ^i
```

2.1. Reference configuration

To focus our attention [2], we will assume that the analysed undeformed particle is a cube presented in Fig. 2.

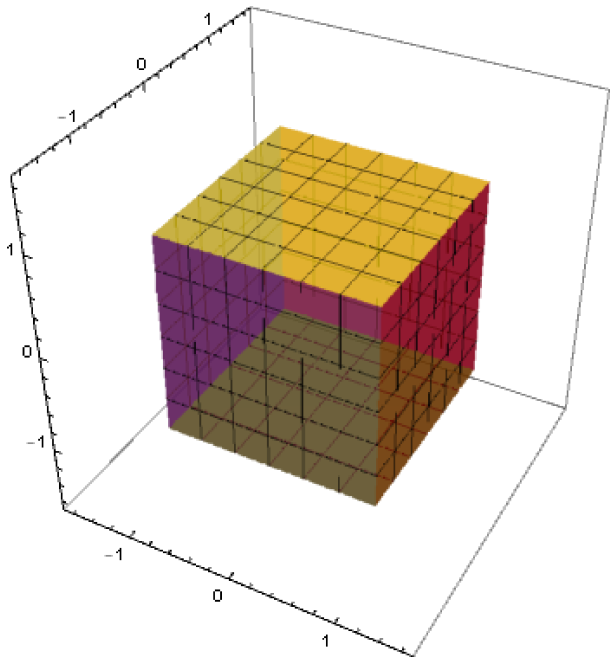

Fig. 2. Reference configuration.

We will assume that in the moment of time $t_o = 0$, the ortho-Cartesian spatial coordinates x^i coincide with material ξ^i. It should be mentioned that unfortunately this is a common cause of misunderstanding of the essence of the problem.

$$x^i\left(\xi^i, 0\right) = \xi^i. \tag{1}$$

The vector pointing to the certain point in space within the particle is equal to:

$$\vec{r} = x^i\,\vec{e}_i. \tag{2}$$

According to our assumption that the space is ortho-Cartesian, we define reference configuration with:

```
In[2]:= reference = {ξ[1], ξ[2], ξ[3]}
```

2.1.1. Base vectors

Vectors of the covariant base in the undeformed configuration are computed from:

$$\mathring{\vec{e}}_i := \frac{\partial \vec{r}}{\partial \xi^i}, \tag{3}$$

```
In[3]:= e0[i_] := ∂ξ[i] reference
```

where $\partial_{\xi[i]}$ stands for $\dfrac{\partial}{\partial \xi^i}$.

These vectors are $\vec{e}_1 = (1,0,0)$, $\vec{e}_2 = (0,1,0)$ and $\vec{e}_3 = (0,0,1)$.

2.1.2. Metric tensor

The space is flat, so the metric tensor is equal to the Kronecker delta:

$$\mathring{g}_{ij} = \mathring{\vec{e}}_i \cdot \mathring{\vec{e}}_j = \delta_{ij}. \tag{4}$$

First we compute the matrix:

```
In[4]:= gLowerMatrix0 = FullSimplify[Table[e0[i].e0[j],{i,3},{j,3}]]

Out[4]= {{1,0,0},
         {0,1,0},
         {0,0,1}}
```

Components of that matrix are components of the metric tensor in the reference configuration.

```
In[5]:= gl0[i_,j_] := gl0[i,j] = gLowerMatrix0[[i,j]]
```

2.2. Vector field of displacement

Now, let us introduce the following vector field u displacement:

```
In[6]:= u = ((ξ[2]^2-ξ[3]) e0[1] + (ξ[2]+ξ[3]^2) e0[2] - ξ[1]^2 e0[3]) t
```

It is a function of parameter t, which models time. All our further plots will be considered for $t \in \langle 0, 1 \rangle$.

```
Out[6]= {t ((ξ^2)^2 - ξ^3), t (ξ^2 + (ξ^3)^2), -t (ξ^1)^2}
```

Its plot is obtained with `VectorPlot3D` for the parameter $t = 1$. The slider in the top is an element of the control provided with `Manipulate` function (Fig. 3).

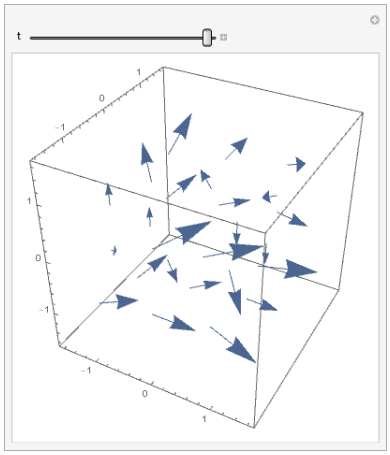

Fig. 3. Displacement vector field.

2.3. Deformed continuum

2.3.1. Spatial coordinates of the curved body

The deformed configuration is defined with:

$$x^i = x^i(\xi^j, t) = x^i(\xi^j, 0) + u^i(\xi^j, t), \tag{5}$$

which we denote with a vector called `actual`.

```
In[7]:= actual = reference + u

Out[7]= {ξ^1 + t((ξ^2)^2 - ξ^3),
         ξ^2 + t (ξ^2 + (ξ^3)^2),
         -t(ξ^1)^2 + ξ^3}
```

The following three `Manipulate` frames (Fig. 4) are obtained with the use of `ParametricPlot3D` to obtain the "walls" of the particle. The first slide is used to control parameter t, the second to control the density of mesh on the "walls", and the third their opacity (the opposite of transparency).

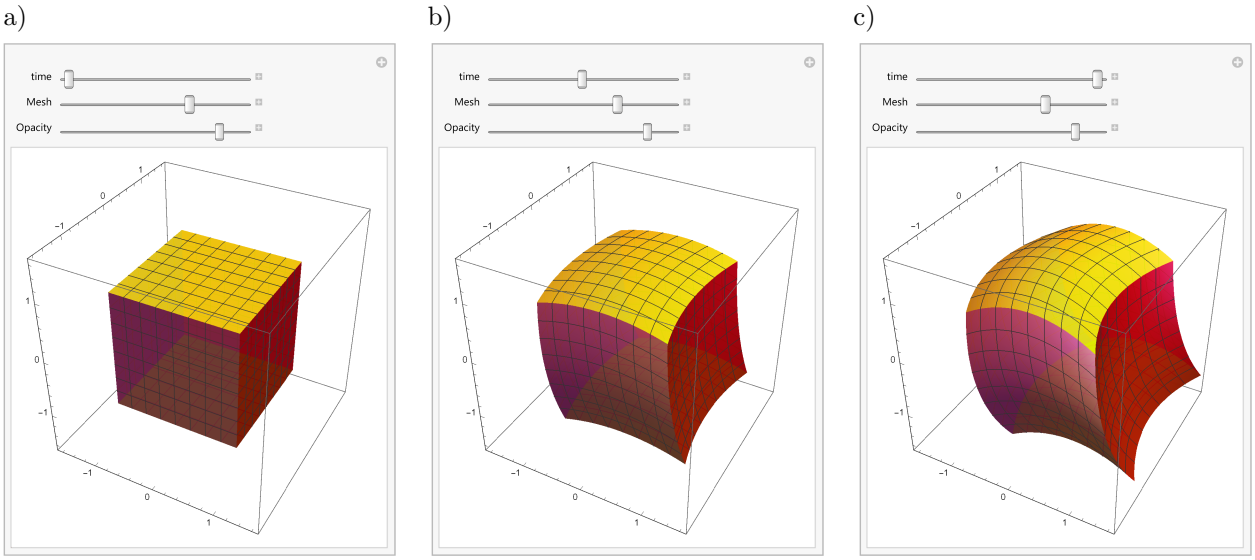

Fig. 4. Deformation progress in time: a) undeformed ($t = 0$), b) deformed ($t = 0.5$), c) deformed ($t = 0.1$).

The deformed material space with frozen in Lagrangian coordinates is curved now, whereas spatial (Eulerian coordinates) represented with the outer box remains flat. Now we can see the difference between both coordinates.

2.3.2. Covariant base vectors of the deformed continuum

The vector $\vec{r}$ in the deformed configuration and covariant base vectors are obtained with:

$$\vec{r} = \xi^i \, \vec{e}_i, \tag{6}$$

$$\vec{e}_i := \frac{\partial \vec{r}}{\partial \xi^i}. \tag{7}$$

```
In[8]:= e[i_] := ∂_ξ[i] actual
```

These vectors are $\vec{e}_1 = \{1, 0, -2\,t\,\xi^1\}$, $\vec{e}_2 = \{2\,t\,\xi^2, t + 1, 0\}$ and $\vec{e}_3 = \{-t, 2\,t\,\xi^3, 1\}$.

In the next frames of animation (Fig. 5), we observe that covariant base changes with the deformation of the particle. Base vectors in the material configuration change direction, change value and lose orthogonality, but frozen in coordinates do not change. Spatial coordinates of the deforming body change, but base vectors remain the same.

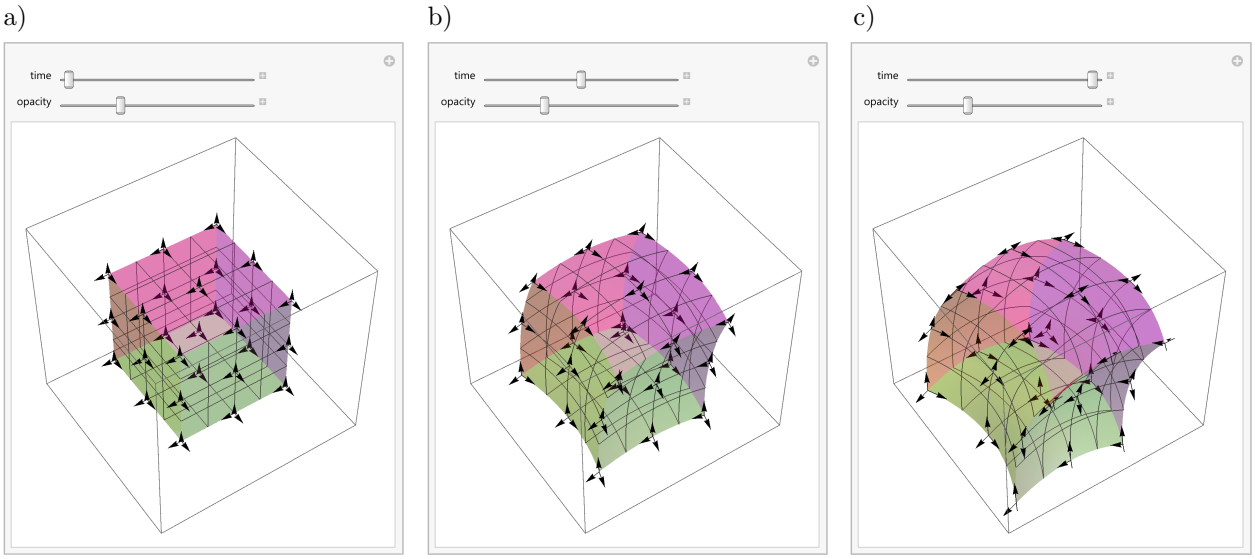

Fig. 5. Change of covariant base along with deformation: a) undeformed ($t = 0$), b) deformed ($t = 0.5$), c) deformed ($t = 0.1$).

2.3.3. Metric tensor after deformation

The components of the metric tensor in the material configuration are not equal to the Kronecker delta any more:

$$g_{ij} = \vec{e}_i \cdot \vec{e}_j. \tag{8}$$

Scalar product of the covariant base builds a matrix `gLowerMatrix` whose elements produce components of the metric tensor.

```
In[9]:= gLowerMatrix = FullSimplify[Table[e[i].e[j],{i,3},{j,3}]]

Out[9]= {{1 + 4 t^2 (ξ^1)^2,  2 t ξ^2,                  -t (1+2 ξ^1)},
         { 2 t ξ^2,           1+t (2+t+4 t (ξ^2)^2),    2 t (ξ^3 + t (-ξ^2+ξ^3))},
         {-t (1 + 2 ξ^1),     2 t(ξ^3+t (-ξ^2 + ξ^3)),  1 + t^2 (1 + 4 (ξ^3)^2)}}

In[10]:= gl[i_,j_] := gl[i,j] = gLowerMatrix[[i,j]]
```

The determinant of the `gLowerMatrix` **g** has to be bigger than 0 in any point of the deformed body to make it possible for the assumed deformation to have physical sense.

```
In[11]:= g = FullSimplify[Det[gLowerMatrix]]

Out[11]= (1 + t - 2 t^2 ξ^1 (1 + t + 4 t ξ^2 ξ^3))^2
```

2.3.4. Contravariant base

Vectors of the contravariant base can be computed from the following definition:

$$\vec{e}^{\,i} := \frac{\vec{e}_j \times \vec{e}_k}{\vec{e}_i \cdot (\vec{e}_j \times \vec{e}_k)}. \tag{9}$$

The code for them is:

```
In[12]:= eu[1] = FullSimplify[e[2]×e[3] / e[1].e[2]×e[3]]

Out[12]= {(1 + t) / (1 + t - 2 t^2 ξ^1 (1 + t + 4 t ξ^2 ξ^3)),
 -(2 t ξ^2) / (1 + t - 2 t^2 ξ^1 (1 + t + 4 t ξ^2 ξ^3)),
 t (1 + t + 4 t ξ^2 ξ^3) / (1 + t - 2 t^2 ξ^1 (1 + t + 4 t ξ^2 ξ^3))}

In[13]:= eu[2] = FullSimplify[e[3]×e[1] / e[2].e[3]×e[1]]

Out[13]= {4 t^2 ξ^1 ξ^3 / (-1 + t (-1 + 2 t ξ^1 (1 + t + 4 t ξ^2 ξ^3))),
 (-1 + 2 t^2 ξ^1) / (-1 + t (-1 + 2 t ξ^1 (1 + t + 4 t ξ^2 ξ^3))),
 2 t ξ^3 / (-1 + t (-1 + 2 t ξ^1 (1 + t + 4 t ξ^2 ξ^3)))}

In[14]:= eu[3] = FullSimplify[e[1]×e[2] / e[3].e[1]×e[2]]

Out[14]= {-(2 t (1 + t) ξ^1) / (-1 + t (-1 + 2 t ξ^1 (1 + t + 4 t ξ^2 ξ^3))),
 4 t^2 ξ^1 ξ^2 / (-1 + t (-1 + 2 t ξ^1 (1 + t + 4 t ξ^2 ξ^3))),
 (1 + t) / (-1 + t (-1 + 2 t ξ^1 (1 + t + 4 t ξ^2 ξ^3)))}
```

When analysing the above expressions, we can easily notice that, for example:

$$\vec{e}_1 \cdot (\vec{e}_2 \times \vec{e}_3) = \sqrt{\mathbf{g}}. \tag{10}$$

It is connected with a property of the absolute antisymmetric Levi-Civita tensor:

$$\varepsilon_{ijk} := \vec{e}_i \cdot (\vec{e}_j \times \vec{e}_k).$$

2.4. Contravariant components of metric tensor

The contravariant components of the metric tensor can be obtained from the scalar product of contravariant base vectors:

$$g^{ij} = \vec{e}^i \cdot \vec{e}^j, \tag{11}$$

however there is an alternative way.

The metric tensor is a special one, and for inversion of `gLowerMatrix` it computes contravariant components of it.

```
In[15]:= gUpperMatrix = FullSimplify[Inverse[gLowerMatrix]];

In[16]:= gu[i_,j_] := gu[i,j] = gUpperMatrix[[i,j]]
```

A sample component of that tensor is:

```
In[17]:= gu[1,1]
```

$$\text{Out[17]}= \frac{1 + t\,(2 + t\,(2 + 4\,(\xi^2)^2 + t\,(1 + 4\,\xi^2\,\xi^3)\,(2 + t + 4\,t\,\xi^2\,\xi^3)))}{(-1 + t\,(-1 + 2\,t\,\xi^1\,(1 + t + 4\,t\,\xi^2\,\xi^3)))^2}$$

2.5. Rising indices

The metric tensor can be used for the tensor operation called "rising" or "lowering" indices. It makes it possible to find a contravariant component.

Making a tensor sum according to Einstein's rule, we have the following formula:

$$\vec{e}^{\,j} = \vec{e}_i\, g^{ij}. \tag{12}$$

Practically its computation is carried out with the following function:

```
In[18]:= eu[j_] := eu[j]=FullSimplify[∑_i^3 e[i] gu[i,j]]
```

It can be easily checked that it produces the same formulas as the ones obtained above with Eq. (9).

2.6. Covariant base versus contravariant one

Now we can observe the difference between the two bases. In undeformed state $t = 0$ (Fig. 6) both sets of base vectors coincide despite the position within the particle controlled with slides 3, 4 and 5.

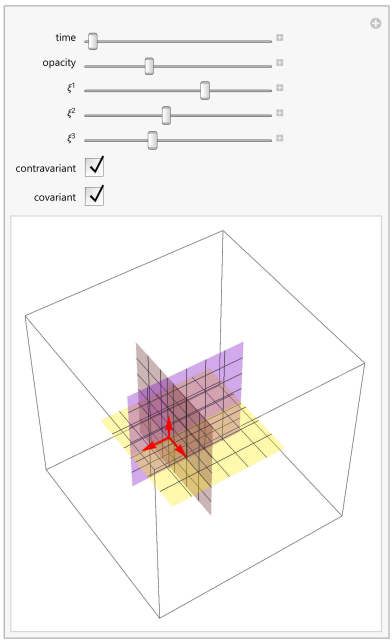

Fig. 6. Base vectors in the undeformed configuration.

When we set parameter $t = 1$ (Fig. 7) for two different positions within the particle, we can see that both bases split.

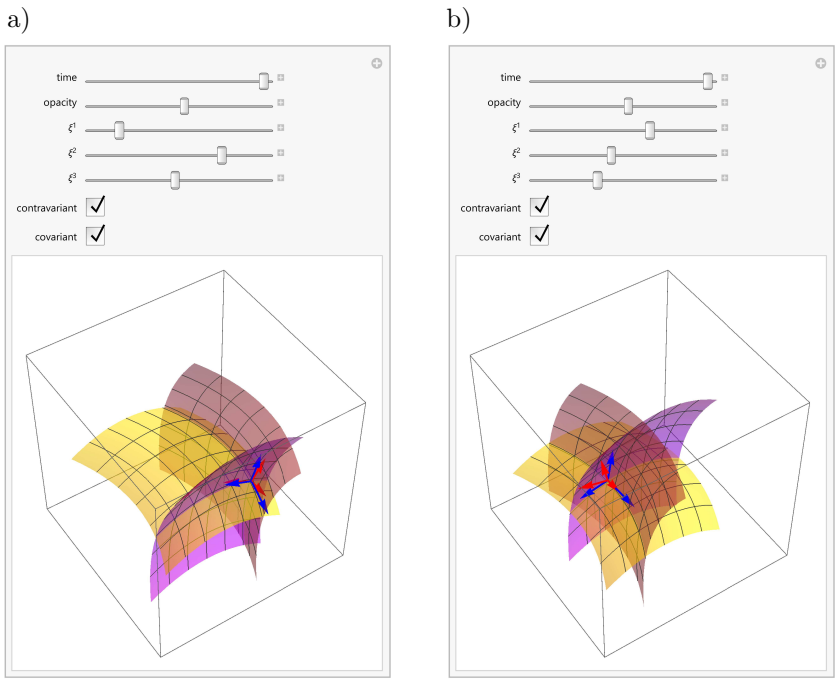

Fig. 7. Splitting of base vectors to covariant (blue) and contravariant (red) ones in the deformed configuration.

Next, Fig. 8 shows that vectors of the covariant base are tangent to the lines of curvilinear coordinates, and contravariant ones are normal to the respective surfaces. Moreover, it can be seen that if a vector $\vec{e}_i$ becomes longer, the respective vector $\vec{e}^{\,i}$ becomes shorter and opposite. This is

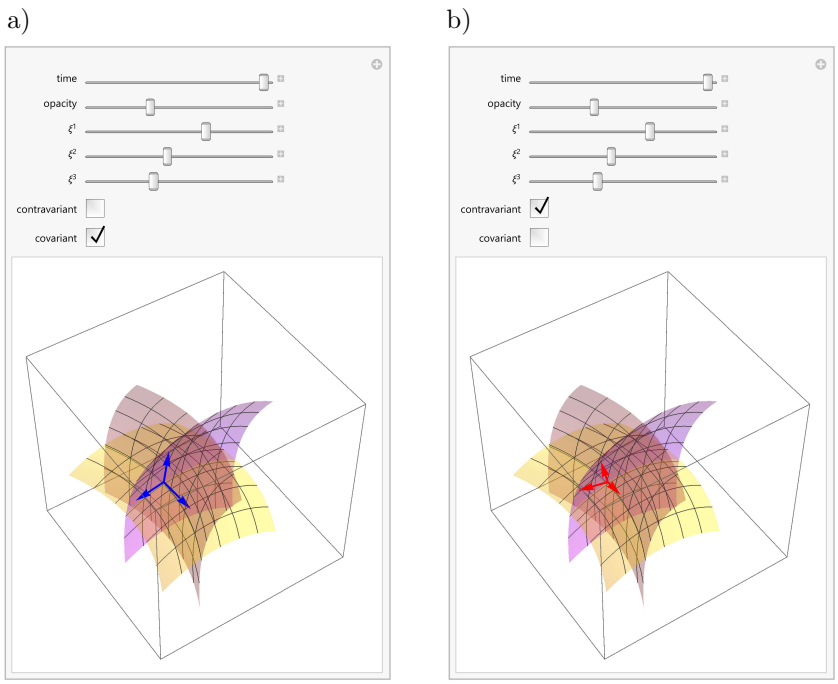

Fig. 8. Covariant base vectors versus contravariant ones.

connected with relation (9) and the fact that the scalar product of them is equal to the Kronecker delta:

$$\vec{e}^{\,j} \cdot \vec{e}_i = \delta_i^j. \tag{13}$$

2.7. Transformation between coordinate systems

Relation of components of the tensors in Eulerian (spatial) and Lagrangian (material) systems can be done with the aid of the Jacobian matrix.

2.7.1. Jacobian matrix

The Jacobian matrix is defined with:

$$J_i^j = \frac{\partial x^j}{\partial \xi^i}. \tag{14}$$

It is implemented with the function:

```
In[19]:= JacobianMatrix = Table[∂ξ[i]actual[[j]],{i,3},{j,3}];
```

It is easy to check that the Jacobian matrix has a connection:

$$\mathscr{J} = \begin{pmatrix} \vec{e}_1 \\ \vec{e}_2 \\ \vec{e}_3 \end{pmatrix}. \tag{15}$$

```
In[20]:= JacobianMatrix == {e[1],e[2],e[3]}

Out[20]= True
```

The determinant of that matrix is called Jacobian:

$$J := \det \mathscr{J}. \tag{16}$$

```
In[21]:= jacobian = FullSimplify[Det[JacobianMatrix]]

Out[21]= 1 + t - 2 t^2 ξ^1 (1 + t + 4 t ξ^2 ξ^3)
```

Jacobian has the following connection with the determinant of metric tensor:

$$J^2 = g. \tag{17}$$

```
In[22]:= FullSimplify[jacobian^2 == g]

Out[22]= True
```

The inversion of the Jacobian matrix has the following definition:

$$I_i^j = \frac{\partial \xi^i}{\partial x^j}. \tag{18}$$

It is computed by the inversion of the Jacobian matrix.

```
In[23]:= InvertedJacobianMatrix = FullSimplify[Inverse[JacobianMatrix]];
```

This matrix has the following relation with contravariant base vectors:

$$\mathscr{I} = \begin{pmatrix} \vec{e}^1 \\ \vec{e}^2 \\ \vec{e}^3 \end{pmatrix}^T . \tag{19}$$

```
In[24]:= FullSimplify[
           InvertedJacobianMatrix == Transpose[{eu[1],eu[2],eu[3]}]]

Out[24]= True
```

2.8. Strain tensor

Strain tensor is defined as the change of the metric tensor due to deformation:

$$\gamma_{ij} := \frac{1}{2}\left(g_{ij} - \mathring{g}_{ij}\right). \tag{20}$$

Its evaluation is implemented with:

```
In[25]:= gamma[i_,j_] := gamma[i,j] = 1/2 (gl[i,j] - gl0[i,j])
```

$$\text{Out[25]=}\ \begin{Bmatrix} \{2\,t^2\,(\xi^1)^2, & t\,\xi^2, & -\frac{1}{2}\,t\,(1+2\,\xi^1)\}, \\ \{t\,\xi^2, & \frac{1}{2}\,t\,(2+t+4\,t\,(\xi^2)^2), & t\,(\xi^3+t\,(-\xi^2+\xi^3))\}, \\ \{-\frac{1}{2}\,t\,(1+2\,\xi^1), & t\,(\xi^3+t\,(-\xi^2+\xi^3)), & \frac{1}{2}\,t^2\,(1+4\,(\xi^3)^2)\} \end{Bmatrix}$$

Now we can show how the transformation to another system of coordinates looks like. This tensor can be transformed into the Eulerian coordinate system with the following formula:

$$\varepsilon_{ij} := I_i^k\, I_j^l\, \gamma_{kl}. \tag{21}$$

Implementation of this formula:

```
In[26]:= Table[FullSimplify[
           Sum[Sum[InvertedJacobianMatrix[[i,k]]
             InvertedJacobianMatrix[[j,l]] gamma[k,l]],{k,3}],{l,3}],{i,3},{j,3}]
```

The output matrix has a very long form. For example, the component ε_{33} is equal to:

$$\frac{t^2\left((t+1)^2\left(4\,\xi^1\left(t^2\,\xi^1-1\right)-1\right)+a^*+b^*\right)}{2\left(-2\,t^2\,\xi^1\left(4\,t\,\xi^2\,\xi^3+t+1\right)+t+1\right)^2}, \tag{22}$$

where

$$a^* = 8\,t\,(t+1)\xi^2\,\xi^3\left(-2\,\xi^1+4\,t^2\,(\xi^1)^2-1\right),$$

$$b^* = 4(\xi^3)^2\left(4\,t^2(\xi^2)^2\left(4\,t^2(\xi^1)^2-1\right)-1\right).$$

Due to the length of the formula, the font size is being reduced. The plot of this function in 3D space can be done with `ContourPlot3D` function (Fig. 9).

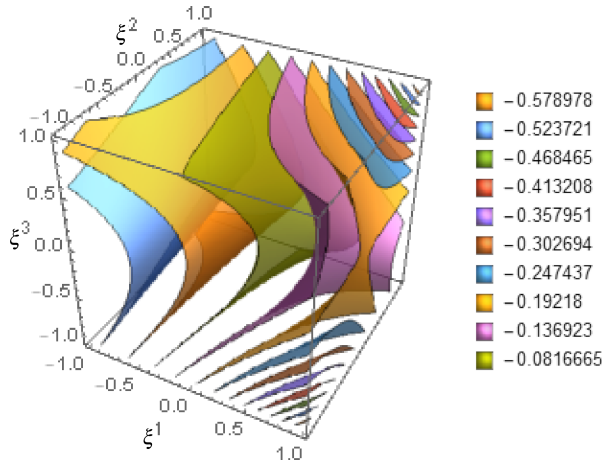

Fig. 9. 3D contour plot of ε_{33} strain tensor component for $t = .25$.

3. Bending helicoid to catenoid

3.1. Definitions

First, like in the previous section, we format the appearance of the variable $u[i]$ to appear ξ^i in outputs.

```
In[27]:= u[i_] := u^i
```

The main difficulty of understanding the theory of shells by students of engineering is differential geometry.

We will observe the application of *Mathematica* tools to the analysis of the well-known phenomenon of bending helicoid to catenoid.

Definition of helicoid:

$$hel := \left\{\sin\left(u^2\right)\sinh\left(u^1\right), -\cos\left(u^2\right)\sinh\left(u^1\right), u^2\right\}. \tag{23}$$

```
In[28]:= hel = {Sin[u[2]] Sinh[u[1]], -Cos[u[2]] Sinh[u[1]], u[2]};
```

```
Out[28]= Sin[u^2] Sinh[u^1],-Cos[u^2] Sinh[u^1],u^2}
```

Definition of catenoid:

$$kat := \left\{\cos\left(u^2\right)\cosh\left(u^1\right), \sin\left(u^2\right)\cosh\left(u^1\right), u^1\right\}. \tag{24}$$

```
In[29]:= kat = {Cos[u[2]] Cosh[u[1]], Sin[u[2]] Cosh[u[1]], u[1]}
```

```
Out[29]= {Cos[u^2] Cosh[u^1],Cosh[u^1] Sin[u^2],u^1}
```

Bending of helicoid to catenoid can be described with the following equation:

$$het2kat := hel\,\cos(t) + kat\,\sin(t). \tag{25}$$

```
In[30]:= r = hel Cos[t] + kat Sin[t];
```

Using the `Manipulate` function with control parameter $t \in \measuredangle 0, 2\pi\rangle$, we can see how drawing of the surface produced with `ParametricPlot3D` changes (Fig. 10). In our analysis, $u^1 \in \langle -1, 1\rangle$ and $u^2 \in \langle 0, 2\pi\rangle$.

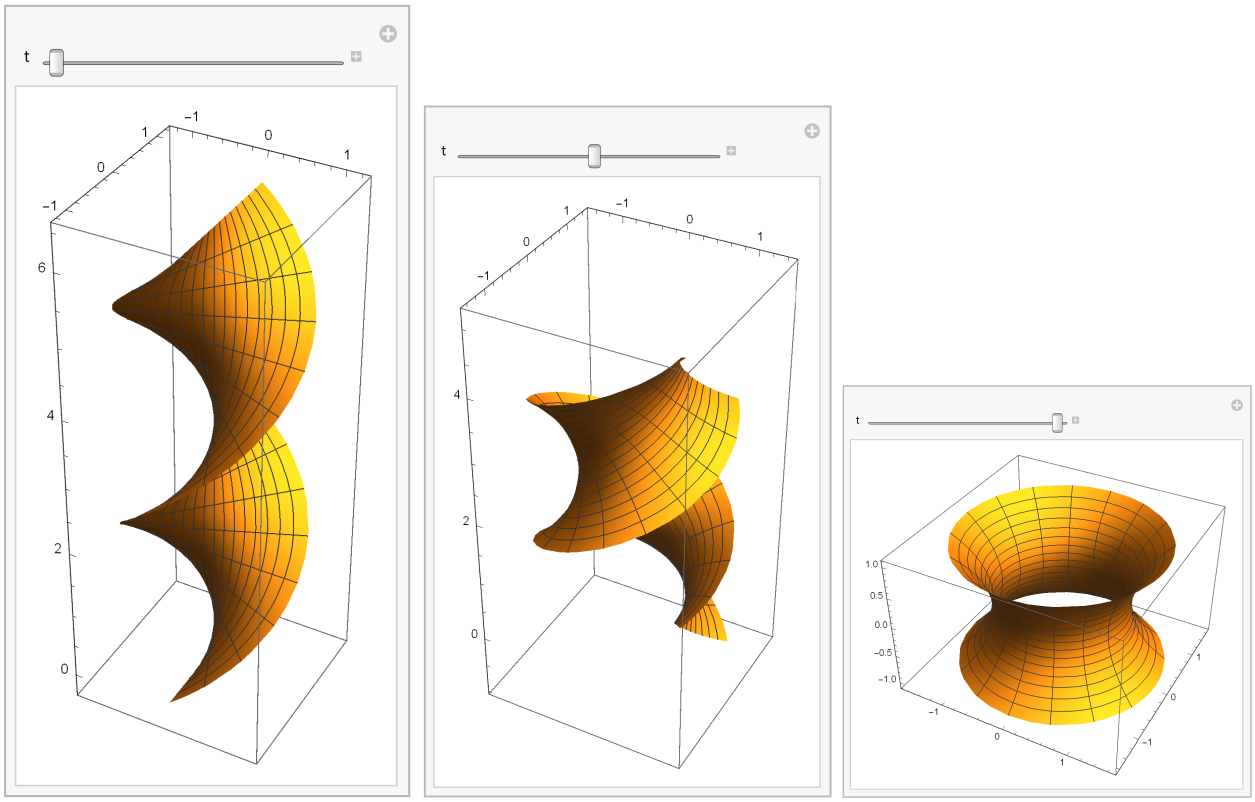

Fig. 10. Bending helicoid to catenoid.

3.2. The covariant base of the surface

Covariant base vectors, which are tangent to the surface, are computed with:

$$\vec{r}_i := \frac{\partial \vec{r}}{\partial u^i}. \tag{26}$$

```
In[31]:= r_i_ := r_i = FullSimplify[∂_u_i r]
```

The result of the symbolic computation is:

```
In[32]:= r_1

Out[32]= {Cos[t] Cosh[u^1] Sin[u^2]+Cos[u^2] Sin[t] Sinh[u^1],
          -Cos[t] Cos[u^2] Cosh[u^1]+Sin[t] Sin[u^2] Sinh[u^1],
           Sin[t]}

In[33]:= r_2

Out[33]= {-Cosh[u^1] Sin[t] Sin[u^2]+Cos[t] Cos[u^2] Sinh[u^1],
          Cos[u^2] Cosh[u^1] Sin[t]+Cos[t] Sin[u^2] Sinh[u^1],
          Cos[t]}
```

3.3. First differential form, metric tensor

The components of the first differential form are computed as a scalar product of base vectors

$$a_{ij} := \vec{r}_i \cdot \vec{r}_j. \tag{27}$$

```
In[34]:= a_i_,j_ := a_i,j = FullSimplify[r_i.r_j]
```

We can define a matrix `gLowerMatrix` from the elements of this tensor.

```
In[35]:= aLowerMatrix = Table[a_i,j,{i,2},{j,2}]
```

This tensor is a metric tensor, so we can evaluate its contravariant components from the inverted matrix `gLowerMatrix`.

```
Out[35]= {{Cosh[u^1]^2,0         },
          {0,          Cosh[u^1]^2}}
```

```
In[36]:= aUpperMatrix = Inverse[aLowerMatrix]
```

```
Out[36]= {{Sech[u^1]^2,0         },
          {0,          Sech[u^1]^2}}
```

```
In[37]:= au_i_,j_ := au_i,j = aUpperMatrix[[i,j]]
```

It is worth to note that the metric tensor does not depend on parameter t.
Next, we compute the determinant of the `gLowerMatrix`:

$$\mathbf{a} := \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}. \tag{28}$$

```
In[38]:= a = FullSimplify[Det[aLowerMatrix]]
```

Having this determinant, we can compute the area of the surface.

```
Out[38]= Cosh[u^1]^4
```

```
In[39]:= ∫_-1^1 ∫_0^2π √a du[1] du[2]
```

```
Out[39]= 1/2 (4 π + Sinh[4 π])
```

Since the metric tensor does not change with the deformation, the surface area is constant.

3.4. Normal vector

Normal vector to the surface is defined with:

$$\vec{r}_3 := \frac{\vec{r}_2 \times \vec{r}_2}{\sqrt{\mathbf{a}}}. \tag{29}$$

```
In[40]:= r3 = FullSimplify[PowerExpand[(r1×r2)/√a]]

Out[40]= {-Cos[u^2] Sech[u^1],-Sech[u^1] Sin[u^2],Tanh[u^1]}
```

This vector is unit.

```
In[41]:= FullSimplify[r3.r3]

Out[41]= 1
```

Having a definition of tangent and normal vectors, we can build a function where we can observe how the covariant base and shape of the surface changes with parameters t and u^i within the surface particle deformation (Fig. 11).

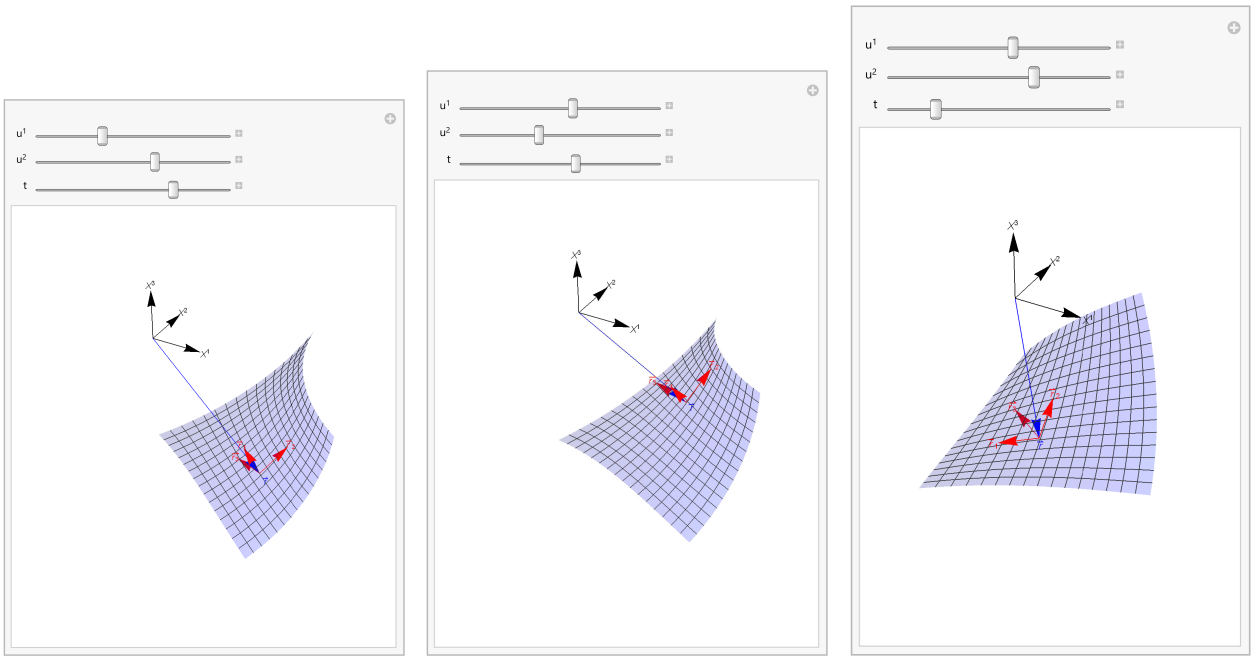

Fig. 11. Deformation of the mid-surface particle in time.

3.5. Second and third differential forms

To compute components of the second of the first and the second differential forms, we should define vectors $\vec{m}_i$, which are tangent to the surface

$$\vec{m}_i := \frac{\partial \vec{r}_3}{\partial u^i}. \tag{30}$$

```
In[42]:= m_i_ := m_i = FullSimplify[∂_u[i] r3]

In[43]:= m1

Out[43]= {Cos[u^2] Sech[u^1] Tanh[u^1],Sech[u^1] Sin[u^2] Tanh[u^1],Sech[u^1]^2}

In[44]:= m2

Out[44]= {Sech[u^1] Sin[u^2],-Cos[u^2] Sech[u^1],0}
```

Components of the second differential form are:

$$b_{ij} := -\vec{r}_i \cdot \vec{m}_j. \tag{31}$$

```
In[45]:= b_i_,j_ := b_i,j = FullSimplify[-m_i.r_j]
```

These coefficients can be grouped in `bLowerMatrix`

```
In[46]:= bLowerMatrix = Table[b_i,j,{i, 2},{j,2}]

Out[46]= {{-Sin[t],-Cos[t]},
          {-Cos[t], Sin[t]}}
```

The determinant of this matrix:

$$\mathbf{b} := \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{vmatrix} \tag{32}$$

is equal to:

```
In[47]:= b = FullSimplify[Det[bLowerMatrix]]

Out[47]= -1
```

Components of the third differential form are:

$$c_{ij} := \vec{m}_i \cdot \vec{m}_j. \tag{33}$$

```
In[48]:= c_i_,j_ := c_i,j = FullSimplify[m_i.m_j]
```

4. Curvatures

The Gaussian curvature does not depend on parameter t

$$K := \frac{\mathbf{b}}{\mathbf{a}}. \tag{34}$$

```
In[49]:= K = FullSimplify[b/a]

Out[49]= -Sech[u^1]^4
```

The deforming surface is minimal since its curvature $H = 0$,

$$H := \frac{1}{2}\, b_{ij}\, a^{ij}. \tag{35}$$

```
In[50]:= H = FullSimplify[Sum[a u_i,j b_i,j, {i, 2}, {j, 2}]]

Out[50]= 0
```

4.1. Strain tensor

The first strain tensor is defined with the change of the first differential form:

$$\gamma_{ij} := \frac{1}{2}(a_{ij} - \mathring{a}_{ij}). \tag{36}$$

```
In[51]:= γ_{i_,j_} := γ_{i,j} = 1/2 FullSimplify[(a_{i,j} - (a_{i,j} /. t → 0)]
```

In the considered case, it does not change.

```
In[52]:= Table[γ_{i,j},{i,2},{j,2}]

Out[52]= {{0,0},
          {0,0}}
```

The second strain tensor is defined with the change of the second differential form:

$$\rho_{ij} := \frac{1}{2}(b_{ij} - \mathring{b}_{ij}). \tag{37}$$

```
In[53]:= ρ_{i_,j_} := ρ_{i,j} = 1/2 FullSimplify[b_{i,j} - (b_{i,j} /. t → 0)]

In[54]:= 2 Table[ρ_{i,j},{i,2},{j,2}]

Out[54]= {{-Sin[t],  1-Cos[t]},
          { 1-Cos[t],Sin[t]}}
```

Components of this tensor are not equal to zero. Therefore we call the considered deformation as a pure bending.

The third strain tensor is defined with the change of the third differential form:

$$\vartheta_{ij} := \frac{1}{2}(c_{ij} - \mathring{c}_{ij}). \tag{38}$$

```
In[55]:= ϑ_{i_,j_} := ϑ_{i,j} = 1/2 FullSimplify[c_{i,j} - (c_{i,j} /. t → 0)]

In[56]:= Table[ϑ_{i,j},{i,2},{j,2}]

Out[56]= {{0,0},
          {0,0}}
```

4.2. "Parallel" surface

A shell is defined with reference to a certain surface, usually in the equal distance from the bounding surfaces. Any point of the shell is defined with:

$$\vec{R} := \vec{r} + z\,\vec{r}_3. \tag{39}$$

```
In[57]:= R = r + z r_3;
```

The result of this definition is illustrated in Fig. 12.

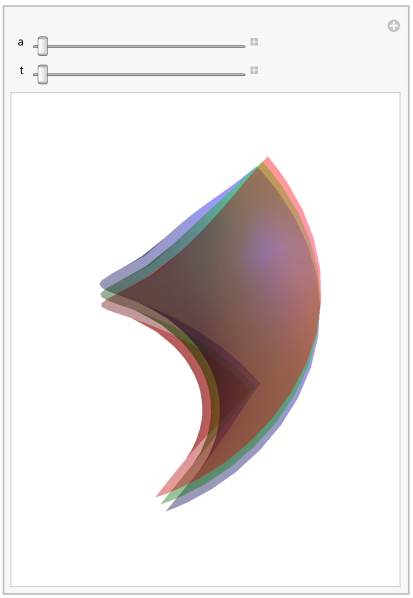

Fig. 12. Construction of the "parallel" surface with regard to mid-surface.

5. Conclusions

Mathematica is an effective tool to compute and visualize large displacements of continuum, shells and thin-wall structures.

It is an effective aid in teaching and learning the continuum mechanics and the theory of shells, and makes it possible to explain crucial differences between Lagrangian and Eulerian systems of coordinates.

Mathematica graphical tools, especially:

- `Manipulate`,
- `ContourPlot3D`,
- `ParamentricPlot3D`

help to effectively illustrate the results of theoretical analysis and thus to explain to students difficult issues.

Acknowledgments

The author would like to acknowledge J. Kuczmarski for his packages [3, 4], which made it possible to introduce *Mathematica* inputs and outputs in this paper.

References

[1] S. Bielak. *Theory of Shells. Part II, Theory and Applications* [in Polish]. Civil Engineering. Silesian University of Technology, 2nd ed., 1993.

[2] S.P. Kiselev, E.V. Vorozhtsov, V.M. Fomin. *Foundations of Fluid Mechanics with Applications. Problem Solving Using Mathematica*. Modelling and Simulation in Science, Engineering and Technology. Birkhäuser Basel, 1st ed., 1999, doi: 10.1007/978-1-4612-1572-1.

[3] J. Kuczmarski. Cells to TeX, Jan 2019. https://github.com/jkuczm/MathematicaCellsToTeX.
[4] J. Kuczmarski. *Mathematica* cells in TeX, January 2017. https://github.com/jkuczm/mmacells.
[5] S. McManus, M. Cook. *Raspberry Pi for Dummies*. John Wiley & Sons, 3rd ed., 2017.
[6] L. Parker, S.M. Christensen. *MathTensor: A System for Doing Tensor analysis by Computer*. Addison-Wesley, 1994.
[7] UpSkill Learning. *Raspberry Pi 3: Get Started with Raspberry Pi 3 a Simple Guide to Understanding and Programming Raspberry Pi 3 (Raspberry Pi 3 User Guide, Python Programming, Mathematica Programming)*. CreateSpace Independent Publishing Platform, 2016.
[8] R. Walentyński. Application of computer algebra in symbolic computation and boundary-value problems of the theory of shells. *Zeszyty Naukowe. Budownictwo*, **100**: 13–198, Silesian University of Technology, 2003.
[9] R. Walentyński. Description and visualization of large deformations of continuum with *Mathematica*. In: *Proceedings of 12th International Mathematica Symposium (IMS 2015)*, Prague, January 12–14, 2015.
[10] R. Walentyński. Lecturing continuum mechanics with *Mathematica*. In: *Computer Algebra Systems in Teaching and Research, Mathematical Modelling in Physics, Civil Engineering, Economics and Finance*, L. Gadomski *et al.* [Eds], pp. 184–193. Collegium Mazovia, Siedlce, 2011.
[11] S. Wolfram. *An Elementary Introduction to the Wolfram Language*. Wolfram Media, Inc., 2015. Available online.