Solving Symbolic and Numerical Problems in the Theory of Shells with *MATHEMATICA*®

Ryszard A. Walentyński

The Department of Building Structures Theory, The Faculty of Civil Engineering,
The Silesian University of Technology, ul. Akademicka 5, PL44-101 Gliwice, Poland,
rwal@kateko.bud.polsl.gliwice.pl

1 Introduction

The theory of shells describes the behaviour (displacement, deformation and stress analysis) of thin bodies (thin walled structures) defined in the neighbourhood of a curved surface in the 3D space. Most of contemporary theories of shells use differential geometry as a mathematical tools and tensor analysis for notations. Examples are [1, 2, 3].

Tensor notation has a lot of advantages, first of all it makes it possible to denote general problems with elegant and short expressions. Therefore it is very suitable for lecturing purposes (from the point of view of the teacher) and writing scientific essays.

However, it seems to be to be very abstract for most of engineers and boring for students. Moreover, it is prone to error especially when summations are done by hand. Therefore most of the publications considering tensor problems are at least suspected to contain errors in the indices. Thus, all of them need careful scrutiny. Despite of verification the error can be repeated. Some theories presumed some a'priori simplification of the expressions, especially in constitutive relations. They make final results shorter but valid under certain circumstances.

The paper will try to show how these problems can be avoided using computer assisted symbolic calculations. ***MathTensor***™ [1], an external package of the computer algebra system *MATHEMATICA*® [2], can be used to solve symbolic tensor problems. The package was written by Parker and Christensen [4].

First the paper presents selections from the *MATHEMATICA*® notebooks[3] illustrating solution of some problems of the theory of shells. The aim is to show that it is possible to proceed from very general equations to ones ready for numerical computations and then solve them within one system. Each step requires specific approach. The translation of the task into the language of *MATHEMATICA*® and ***MathTensor***™ and the information which can be received will be discussed.

The obtained symbolic differential equations that should be approximated numerically are very complex. Moreover due to the essential difference in the shear and bending

[1] ***MathTensor***™ is a registered trademark of MathTensor, Inc.
[2] *MATHEMATICA*® is a registered trademark of Wolfram Research, Inc.
[3] Full notebooks can be requested by email.

stiffness of shells the task is usually numerically ill-conditioned. In a lot of cases it is a boundary layer problem unstable in the Lyapunov sense. Therefore numerical approaches like Finite Differences or Finite Element methods may fail in some cases. There are several engineering two step approaches, represented for example by Bielak [2], which are based on the assumption that the membrane state dominates in shells in most of the domain except boundary layers. It leads to the reduction of the order of differential operator of the problem. It results in better stability but also a loss of generality. Membrane state cannot satisfy all boundary conditions. They are satisfied locally using so called bending theory in the limited boundary layer in the second step. The approach fails in cases when bending of shells cannot be neglected. These problems were investigated in [5].

The Refined Least Squares method (RLS) has been applied to this problem and it has been found a new approach to the solution of the tasks of long cylindrical shells. It has been discovered experimentally that the process can be divided into two steps somehow similar to the [2] approach.

The first step of this approach consists in computing the base solution (approximation of the 10^{th} order differential operator) satisfying only the 4 essential boundary conditions. This solution is feasible for most of the problem domain except the boundary layer. Other conditions are neglected. They are adjusted in the second step where only the boundary layer is considered.

This approach results in better convergence and stability. First of all it is not connected with reduction of the order of the differential operator. Therefore moments and transverse forces are not neglected what makes the method much more general than ones based on the membrane state.

It is hoped that the ideas presented will be impressive, particularly for researchers and engineers who deals with other problems of the Mechanics of Continuum and Mathematical Physics which involve tensor analysis and boundary layer tasks. First of all I tried to present how tools of the computer algebra can be applied effectively and what can be done additionally to moderate simplification process and exploit its symbolic possibilities more deeply, effectively and how to obtain reliable results. The least squares method functional has been appended with terms responsible for boundary conditions. It resulted first of all with simpler algorithm, which has been implemented in *MATHEMATICA*®. The very important matter is the discussed numerical occurrence that the boundary value problem with 10^{th} order differential operator can be approximated with consideration of only 4 boundary conditions. It can be applied for other problems with a boundary layer phenomenon, which are present for many engineering and physical tasks.

The considered symbolic problems are based on the theory of shells by Bielak [2]. However, discussed ideas can be applied for another theory of shells. It is assumed that the reader is familiar with the notations rules used in the *MATHEMATICA*® and ***MathTensor***™ language. Only the crucial aspects are discussed and some input and output information has been intentionally omitted. The system output and formulas derived with *MATHEMATICA*® has been prepared for publication. Despite editorial scrutiny it seems to

be more sensible and effortless to use the presented commands and ideas to receive own output than apply directly the final formula.

Notations are explained in the text, some of them: for tensor problems are collected in the end.

2 Symbolic problems - implementations of *MathTensor*™

Presented are some selections from three *MATHEMATICA*® sessions[4]. The crucial points of the process of the derivation of differential equations are discussed. Starting with the very general relations in tensor notation form, the final equations in terms of displacements are obtained, which are ready for numerical computations.

2.1 Derivation of differential equations from tensor equations of equilibrium

Three steps in the process can be distinguished, each of them is different.

Step 1 - Changing covariant derivatives in equations to (ordinary) partial ones The first equation of shell equilibrium takes the following form:

$$N^{ij}{}_{;i} - b_i{}^j\, Q^i + P^j = 0. \tag{1}$$

The left hand side of the equation can be denoted with the following ***MathTensor***™ definition.

```
In[1]:= CD[n[ui, uj], li] - q[ui] b[li, uj] + p[uj]
Out[1]= N^ij_;i + P^j - (b_i^j) (Q^i)
```

MathTensor™ was written for *MATHEMATICA*® 2. Therefore all symbols in it are denoted with Latin letters. To receive Greek letters the output form have to redefined. For example the symbols of the affine connection **AffineG** have got the standard output form G. This redefines it to Γ.

```
In[2]:= Format[AffineG[a__]] := PrettyForm[Γ, a]
```

The same can be done for Kronecker delta **Kdelta** to be expressed with δ or even indices to be represented with small Greek letters.

That command changes the covariant derivatives to ordinary partial ones. Dummy indices are automatically recognised and denoted with p, q, r . . . letters.

```
In[3]:= CDtoOD[%]
Out[3]= (Γ^j_pq) (N^qp) + (Γ^p_pq) (N^qj) + N^pj_,p + P^j - (b_p^j) (Q^p)
```

[4] The section collects and extends topics which were presented in [8], [9], [10] and [13]

The space is metric so the affine connection can be expressed in terms of a metric tensor. It is done with the **AffineToMetric** function. ***MathTensor***™ function **Tsimplify** will simplify the expression taking into account all symmetries of the tensors involved.

```
In[4]:= Tsimplify[AffineToMetric[%]]
```

$$\begin{aligned}\text{Out[4]}= &\frac{1}{2}\,(g^{pj})\,(N^{qr})\,(g_{pq,r}) + \frac{1}{2}\,(g^{pq})\,(N^{rj})\,(g_{pq,r}) + \\ &\frac{1}{2}\,(g^{pj})\,(N^{qr})\,(g_{pr,q}) - \frac{1}{2}\,(g^{pj})\,(N^{qr})\,(g_{qr,p}) + \\ &N^{pj}{}_{,p} + P^{j} - (b_p{}^{j})\,(Q^{p})\end{aligned}$$

Both *MATHEMATICA*® and ***MathTensor***™ have advanced algebraic simplification tools, but sometimes the use of automatic function does not result in the simplest form. Using more complex instruments better results can be obtained. The set of commands below carry out the following operations. It is easy to find that the expression above contain 3 terms with N^{qr}. Therefore using the **Collect** terms can be collected in the expression with respect to it and then simplify it term by term with **Simplify**. It is done with the so called mapping **Map** function in the infix form **/@**. The result is saved for further use.

```
In[5]:= r1 = Simplify/@Collect[%, n[u2, u3]]
```

It results in the following formula, which contains only "ordinary" partial derivatives.

$$N^{pj}{}_{,p} + \frac{1}{2}\,g^{pq}\,g_{pq,r}\,N^{rj} + \frac{1}{2}\,g^{pj}\,\left(g_{pq,r} + g_{pr,q} - g_{qr,p}\right)\,N^{qr} - b_p{}^{j}\,Q^{p} + P^{j} = 0. \qquad (2)$$

Step 2 - Summations The next step consists in the summation of the expression. The problem of the shell is two dimensional and the system should be informed about it before starting the summation process.

```
In[6]:= Dimension = 2;
```

The first equation of equilibrium in tensor notation produces two partial differential equations. The metric tensor on the reference surface is equal to the first differential form $g_{ij} = a_{ij}$.

```
In[7]:= R1[uj_] = MakeSum[r1]
```

Presentation of the result which contains 21 terms is omitted here.

Step 3 - Transformation to *MATHEMATICA*® differential equations In the end the equation can be transformed into an "ordinary" partial differential equation. First tensor the components have to be represented as the "normal" functions of two variables.

```
In[8]:= n[1, 1] := n11[x, y];
        n[1, 2] := n12[x, y];
        n[2, 1] := n21[x, y];
        n[2, 2] := n22[x, y]; (*and so on*)
```

Next ordinary differentiation has to be turned on.

```
In[9]:= On[EvaluateODFlag]
```

The first differential equation takes the form:

```
In[10]:= eqn[1] = R1[1]
Out[10]= p1[x, y] - bm11[x, y] q1[x, y] - +
           bm12[x, y] q2[x, y] + n21^(0,1)[x, y] +
           1/2 n22[x, y] (ag12[x, y] a22^(0,1)[x, y] + ag11[x, y]
                 (2 a12^(0,1)[x, y] - a22^(1,0)[x, y])) +
           1/2 n12[x, y] (ag11[x, y] a11^(0,1)[x, y] +
                 ag12[x, y] a22^(1,0)[x, y]) +
           1/2 n11[x, y] (2 ag11[x, y] a11^(1,0)[x, y] -
                 ag12[x, y] (a11^(0,1)[x, y] - 4 a12^(1,0)[x, y]) +
                 ag22[x, y] a22^(1,0)[x, y]) +
           1/2 n21[x, y] (2 ag11[x, y] a11^(0,1)[x, y] +
                 ag22[x, y] a22^(0,1)[x, y] + ag12[x, y]
                   (2 a12^(0,1)[x, y] + a22^(1,0)[x, y])) + n11^(1,0)[x, y]
```

We will return to this equation later, in point 2.4.

2.2 Nonlinear equations of the second order theory

The approach can be easily extended to nonlinear problems, this is a simple example.

The third equation of shell equilibrium in tensor notation is:

$$M^{ij}{}_{;i} - Q^{j} = 0. \tag{3}$$

In the second order theory the additional moments caused by the stretching (tensile) forces acting on the normal displacements are taken into consideration.

$$M^{ij} \to M^{ij} + N^{ij}\, w^{3}. \tag{4}$$

It is implemented with the following function:

```
In[11]:= m[ui_, uj_] := m1[ui, uj] + n[ui, uj] w3
```

The moment tensor now has the following form:

```
In[12]:= m[ui, uj]
Out[12]= M^ij + w^3 (N^ij)
```

The left hand side (lhs) of the third equation is:

```
In[13]:= CD[m[ui, uj], li] - q[uj]
Out[13]= M^ij;i + w^3 (N^ij;i) + (w^3;i) (N^ij) - Q^j
```

Using a similar approach as that presented in the previous section the following result is obtained:

$$M^{pj}{}_{,p} - \frac{1}{2} M^{pq} g^{rj} \left(g_{pq,r} - g_{pr,q} - g_{qr,p}\right) + \frac{1}{2} M^{pj} g^{qr} g_{qr,p} - Q^{j} + \\ + N^{pj} w^{3}{}_{,p} + \frac{1}{2} w^{3} \left(g^{pq} N^{rj} g_{pq,r} + g^{pj} N^{qr} \left(g_{pq,r} + g_{pr,q} - g_{qr,p}\right) + N^{pj}{}_{,p}\right) = 0. \tag{5}$$

Nonlinear terms are written in the second line.

2.3 Receiving and simplification of the constitutive relations

One of the task of the theory of shells is to reduce the three dimensional problem of the theory of elasticity into a two dimensional one[5]. The analysis of stresses is reduced to the analysis of internal forces: stretching (tensile) forces and moments. The respective tensors are computed from the following integrals:

$$N^{ij} = \int_{-h}^{h} \sqrt{\frac{\boldsymbol{g}}{\boldsymbol{a}}} \left(\delta_r{}^{j} - z\, b_r{}^{j}\right) \tau^{ri}\, dz, \tag{6}$$

$$M^{ij} = \int_{-h}^{h} \sqrt{\frac{\boldsymbol{g}}{\boldsymbol{a}}} \left(\delta_r{}^{j} - z\, b_r{}^{j}\right) \tau^{ri}\, z\, dz. \tag{7}$$

The square root in these formulas is the following function:

$$Z = \sqrt{\frac{\boldsymbol{g}}{\boldsymbol{a}}} = 1 - 2\,H\,z + K\,z^{2}. \tag{8}$$

```
In[1]:= Z := 1 - 2 H z + K z^2;
```

The stress tensor for isotropic material can be derived from the formula:

$$\tau^{ij} = E \left(\frac{\nu}{1-\nu^{2}}\, g^{ij}\, g^{pq} + \frac{1}{1+\nu}\, g^{ip}\, g^{jq}\right) \overset{*}{\gamma}_{pq}. \tag{9}$$

```
In[2]:= tau[ui_, uj_] := (ν e / (1 - ν^2)) Metricg[ui, uj] Metricg[u1, u2] +
                         (e / (1 + ν)) Metricg[ui, u1] Metricg[uj, u2]) gammastar[l1, l2]
```

A strain tensor in 3D space can be expressed in terms of the strain tensor of the reference surface.

$$\overset{*}{\gamma}_{ij} = \gamma_{ij} - 2\,z\,\rho_{ij} + z^{2}\,\vartheta_{ij}. \tag{10}$$

[5] The consideration is made in a new *MATHEMATICA*® session

```
In[3]:= gammastar[li_, lj_] :=
          gamma[li, lj] - 2 z rho[li, lj] + z^2 theta[li, lj]
```

Contravariant components of the metric tensor 3-D shell can be computed from the formula which is a function of the variable z measured along the normal to the reference surface.

$$g^{ij} = \frac{a^{ij}\left(1 - K z^2\right) - 2\left(2 H a^{ij} - b^{ij}\right)(1 - H z)\, z}{Z^2}. \tag{11}$$

```
In[4]:= Metricg[ui_, uj_] :=
          (a[ui, uj] (1 - K z^2) - 2 (2 H a[ui, uj] - b[ui, uj]) (1 - H z) z) / Z^2
```

The integrals (6) and (7) appear to be simple but after substituting into them functions (8), (9), (10) and (11), they become a bit more complicated but using possibilities of the system they can be computed with arbitrary precision with respect to the thickness $2h$ expansion. It can be done with the function:

```
In[5]:= n[ui_, uj_][k_] :=
          Simplify/@Tsimplify[
            AbsorbKdelta[
              integ/@Expand[
                Normal[
                  Series[
                    Z (Kdelta[l3, uj] - z b[l3, uj])
                      tau[ui, u3], {z, 0, k}]]]]]
```

This complex multi-function does the following: the considered integrand – `(Kdelta[l3, uj] - z b[l3, uj]) tau[ui, u3]` is expanded into a power series with **`Series`** and **`Normal`**. The result is algebraically expanded with **`Expand`** and then integrated term by term (using the mapping procedure `/@`) with a predefined function of integration **`integ`**:

```
In[6]:= integ[x_] := ∫_-h^h x dz
```

This approach is necessary as *MATHEMATICA*® is a program and only a program. If the argument of the function is complicated it takes a lot of time to deal with it. Therefore it is usually sensible to divide the task into a set of simpler problems. This approach speeds up computations. Mapping is a very useful tool in this process.

After integration the Kronecker delta is absorbed with **`AbsorbKdelta`** and simplified with the already mentioned functions **`Tsimplify`** and **`Simplify`**.

The parameter **`k`** defines the precision of the computation, for engineering purposes it is sufficient for the calculations to be done with a precision to the third power of the shell thickness. Then the parameter **`k`** in the function should take value 2.

```
In[7]:= noriginal[ui_, uj_] = n[ui, uj][2]
```

The result can be used directly in further computations but is rather long. It contains 24 terms, so its presentation is omitted here. Nevertheless using a set of the *MATHEMATICA*® and ***MathTensor***™ tools it can be presented in a shorter form. Among them we can find: **Dum[%]** for finding pairs of dummy indices, **Expand[%]** for the expression expansion, **Absorb[%, *a*]** and **AbsorbKdelta** for lowering and raising indices, **Canonicalize** for finding the canonical form of tensor expression and the already mentioned **Tsimplify** and **Simplify** for tensor and algebraic simplification.

Automatic simplification does not necessarily give the simplest form. The system can be helped. An example of enforcing the required behaviour is given below. The terms are grouped and each group is simplified.

```
In[8]:= Simplify[%[[1]] + %[[6]]] +
          Simplify[%[[2]] + %[[4]]] +
          Simplify[%[[3]] + %[[5]]]
```

Replacement is a very useful tool for controlling the simplification process. The well known identities:

$$\begin{aligned} b_{pq}\, b_{jp} &\equiv 2\,H\, b_q{}^{j} - K\, a_q{}^{j}, \\ b_p{}^{i}\, b_{jp} &\equiv 2\,H\, b^{ij} - K\, a^{ij}, \end{aligned} \tag{12}$$

$$\gamma^{pj} \equiv \gamma^{pq}\, \delta_q{}^{j} \tag{13}$$

are applied with the following functions, respectively:

```
In[9]:= %/.{b[l1, l2] b[uj, u1] →
              2 H b[l2, uj] - K a[l2, uj],
            b[l1, ui] b[uj, u1] →
              2 H b[ui, uj] - K a[ui, uj]}

In[10]:= %/.gamma[u1, uj] →
            gamma[u1, u2] Kdelta[l2, uj]
```

MATHEMATICA® simplifications tools do not always find the simplest form. Replacement can be very useful in such cases, for example:

```
In[11]:= %/.aa_ ν + bb_ ν → (aa + bb) ν

In[12]:= %/.aa_ b[l1, l2] + bb_ b[l1, l2] →
            (aa + bb) b[l1, l2]

In[13]:= %//.{aa_/(-1 + ν^2) → -aa/(1 - ν^2),
            aa_ (-1 + ν) → -aa (1 - ν),
            aa_ (-1 + h^2 K) → -aa (1 - h^2 K)}
```

At the end the following result is obtained:

$$
\begin{aligned}
N^{ij} = & \frac{2\,E\,h\,\left(1-h^2\,K\right)\,\left(\nu\,a^{ij}\,\gamma_p{}^p+(1-\nu)\;\gamma^{ij}\right)}{1-\nu^2}+\\
& +\frac{4\,E\,h^3\,\gamma^{pq}\left(\nu\,b_{pq}\,\left(H\,a^{ij}+b^{ij}\right)+(1-\nu)\,b_p{}^i\,\left(b_q{}^j+H\,\delta_q{}^j\right)\right)}{3\,\left(1-\nu^2\right)}+\\
& -\frac{8\,E\,h^3\,\left(\nu\,a^{ij}\,b_{pq}\,\rho^{pq}+(1-\nu)\;b_p{}^i\,\rho^{pj}\right)}{3\,\left(1-\nu^2\right)}+\\
& +\frac{4\,E\,h^3\,\left(2\,H\,\delta_q{}^j-b_q{}^j\right)\,\left(\nu\,a^{qi}\,\rho_p{}^p+(1-\nu)\;\rho^{qi}\right)}{3\,\left(1-\nu^2\right)}+\\
& +\frac{2\,E\,h^3\,\left(\nu\,a^{ij}\,\vartheta_p{}^p+(1-\nu)\;\vartheta^{ij}\right)}{3\,\left(1-\nu^2\right)}+O\left(h^5\right).
\end{aligned}
\tag{14}
$$

This is a formula for the tensor of moments. It is received by a similar procedure.

$$
\begin{aligned}
M^{ij} = & \frac{4\,E\,h^3\,\left(\nu\,a^{ij}\,b_{pq}\,\gamma^{pq}+(1-\nu)\;b_p{}^i\,\gamma^{pj}\right)}{3\,\left(1-\nu^2\right)}+\\
& -\frac{2\,E\,h^3\,\left(2\,H\,\delta_q{}^j-b_q{}^j\right)\,\left(\nu\,a^{qi}\,\gamma_p{}^p+(1-\nu)\;\gamma^{qi}\right)}{3\,\left(1-\nu^2\right)}+\\
& -\frac{4\,E\,h^3\,\left(\nu\,a^{ij}\,\rho_p{}^p+(1-\nu)\;\rho^{ij}\right)}{3\,\left(1-\nu^2\right)}+O\left(h^5\right).
\end{aligned}
\tag{15}
$$

The next step is to check if simplified results obtained satisfy the last equation of equilibrium, which has the following form and should be satisfied as an identity.

$$\varepsilon_{pq}\,\left(N^{pq}-b_r{}^p\,M^{qr}\right)=0. \tag{16}$$

The check is carried out by the following function, which shows that it is satisfied. Here an totally antisymmetric object ε_{pq} is denoted with. **`EpsDown[l1, l2]`**. The function contain a lot of simplification tools like tensor simplification **`Tsimplify`**, canonicalization **`Canonicalize`**, absorbtion **`Absorb`** and **`AbsorbKdelta`** and expansion **`Expand`**. Moreover an identity:

$$b_{pq}\,b_r{}^p\equiv 2\,H\,b_{qr}-K\,a_{qr} \tag{17}$$

is applied with **`b[l1, l2] b[l3, u1] → 2 H b[l2, l3] - K a[l2, l3], a`**.

```
In[14]:= Tsimplify[
           Absorb[
             Canonicalize[
               Absorb[
                 AbsorbKdelta[
                   Expand[
                     EpsDown[l1, l2]
                       (nfinal[u1, u2]-
                          b[l3, u1] mfinal[u3, u2] )]
                 ], a
                ]
              ]/.b[l1, l2] b[l3, u1] →
                2 H b[l2, l3] - K a[l2, l3], a
           ]
         ] == 0
Out[14]= True
```

2.4 Description of an arbitrary shell with *MathTensor*™

The problems presented in the previous two sections are general consideration. The next step is to carry out the more detailed calculations for a concrete shell, which is done in a new *MATHEMATICA*® session.

Geometrical description of the reference surface A surface in 3D space can be parameterised with two variables x^i. This is an example of parameterisation of the catenoide shown in figure 1.

```
In[1]:= r := { Cos[x[2]] √(s_o^2 + x[1]^2),
               Sin[x[2]] √(s_o^2 + x[1]^2), s_o ArcSinh[ x[1] / s_o ] };
```

The components of the covariant curvilinear basis are computed as a derivative of the vector $\boldsymbol{r}$ with respect to the parameter x^i.

$$\boldsymbol{r}_i = \partial_{x^i} \boldsymbol{r}. \tag{18}$$

The condition `NegIntegerQ[i]` restricts the definition to covariant components.

```
In[2]:= ri[i_]/; NegIntegerQ[i] := ri[i] = ∂_x[-i] r
```

Geometrical properties The first differential form of the reference surface is defined by the following scalar product:

$$a_{ij} = \boldsymbol{r}_i \cdot \boldsymbol{r}_j. \tag{19}$$

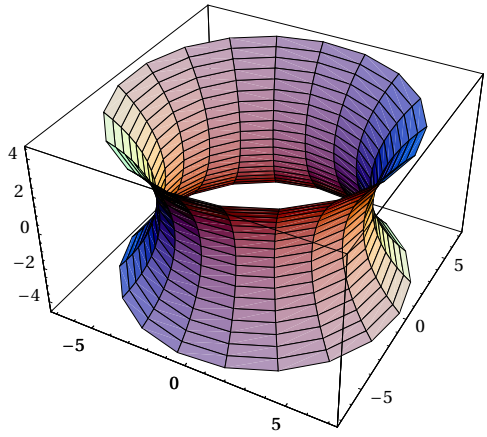

Fig. 1. Catenoide

```
In[3]:= a[i_, j_]/; NegIntegerQ[i]&&NegIntegerQ[j] :=
          a[i, j] = Simplify[ri[i].ri[j]]
```

It is sensible to collect these coefficients into the matrix.

```
In[4]:= aLowerMatrix := aLowerMatrix =
          Table[a[-i, -j], {i, 2}, {j, 2}]
```

This is a definition of the determinant $\boldsymbol{a}$ of this matrix.

```
In[5]:= Deta := Deta =
          Simplify[Det[Table[a[-i, -j], {i, 2}, {j, 2}]]]
```

The third vector of the curvilinear basis can now be computed, it is normal to the mid-surface. It is obtained from the formula:

$$\boldsymbol{r}_3 = \frac{\boldsymbol{r}_1 \times \boldsymbol{r}_2}{\sqrt{\boldsymbol{a}}}. \tag{20}$$

```
In[6]:= ri[-3] := ri[-3] = Simplify[ri[-1] × ri[-2] / PowerExpand[√Deta]]
```

The normal vector is necessary to calculate the second and the third differential form. In this example this computations are omitted because the process is very similar to the first differential form derivation.

Tensor a_{ij} is a metric tensor on the reference surface so its contravariant components can be computed from the inversion of **aLowerMatrix**.

```
In[7]:= aUpperMatrix := aUpperMatrix =
          Simplify[Inverse[aLowerMatrix]]
```

Contravariant coefficients a^{ij} are elements of this matrix and can be computed from the following definition. Here the condition **PosIntegerQ[i]** restricts the definition to the contravariant components.

```
In[8]:= a[i_, j_]/; PosIntegerQ[i]&&PosIntegerQ[j] :=
          a[i, j] = aUpperMatrix⟦i, j⟧
```

Kinematical relations The displacement and rotation vectors can be decomposed in the covariant basis:

$$\boldsymbol{w} = w^k\, \boldsymbol{r}_k + w^3\, \boldsymbol{r}_3, \tag{21}$$

$$\boldsymbol{d} = d^k\, \boldsymbol{r}_k + d^3\, \boldsymbol{r}_3. \tag{22}$$

The last term in the rotation vector is negligible for linear problems. Attention is focused by limiting further consideration to geometrically linear theory.

```
In[9]:= w := w = ww[1] ri[-1] + ww[2] ri[-2] + ww[3] ri[-3]
```

```
In[10]:= d := d = dd[1] ri[-1] + dd[2] ri[-2](* + dd[3][ri[-3]]*)
```

It has to be emphasised that **MakeSum[]** cannot be used in those definitions because it results in an error. It is probably not a bug in the package but is caused by further definitions which express displacements with their physical components. Nevertheless it is a good example of the need to be critical of the results obtained with computer assistance, they need careful scrutiny.

Derivatives of the displacement and rotation vectors are objects of valence one.

$$\boldsymbol{w}_i = \partial_{x^i}\, \boldsymbol{w}, \tag{23}$$

$$\boldsymbol{d}_i = \partial_{x^i}\, \boldsymbol{d}. \tag{24}$$

```
In[11]:= wi[i_]/; NegIntegerQ[i] := wi[i] = Simplify[∂x[-i]w]
```

```
In[12]:= di[i_]/; NegIntegerQ[i] := di[i] = Simplify[∂x[-i]d]
```

The physical components of the displacement and rotation components can be computed from the following formulae:

$$w_i = w^i\, \sqrt{a_{ii}}, \tag{25}$$

$$d_i = d^i\, \sqrt{a_{ii}}. \tag{26}$$

Thus, the following definition is made:

```
In[13]:= ww[i_] := ww[i] = wi[x[1], x[2]] / PowerExpand[√a[-i, -i]]
```

```
In[14]:= dd[i_] := dd[i] = di[x[1], x[2]] / PowerExpand[√a[-i, -i]]
```

Strains The generalised formula for the first strain tensor (containing terms responsible for temperature distortions) is:

$$\gamma_{ij} = \frac{1}{2}\left(\boldsymbol{r}_i \cdot \boldsymbol{w}_j + \boldsymbol{r}_j \cdot \boldsymbol{w}_i\right) - \epsilon\, a_{ij} + (\text{nonlinear terms}). \tag{27}$$

It is denoted by:

```
In[15]:= gamma[i_, j_]/; NegIntegerQ[i]&&NegIntegerQ[j] :=
           gamma[i, j] =
             1/2 ( ri[i].wi[j] + ri[j].wi[i]) - ϵ[x[1], x[2]] a[i, j]
```

The other two strain tensors of the reference surface can be computed with the similar definitions.

Internal forces and equations in terms of displacements The formula for the moments is presented in point 2.3. Having already computed the kinematical relations and strain tensors internal forces can be expressed in terms of displacements. Here is an example of one of the component of moment tensor for the considered parameterisation of the catenoidal shell.

```
In[16]:= mRef[1, 1]
Out[16]= -(2 e h^3 s0 ϵ[x, y])/(3 (-1 + ν) (x^2 + s0^2)) + (2 e h^3 κ[x, y])/(3 - 3 ν) -
         (2 e h^3 x ν d1[x, y])/(3 (-1 + ν^2) (x^2 + s0^2)) +
         (4 e h^3 s0^2 w3[x, y])/(3 (-1 + ν^2) (x^2 + s0^2)^2) +
         (2 e h^3 ν d2^(0,1)[x, y])/((3 - 3 ν^2) Sqrt[x^2 + s0^2]) +
         (2 e h^3 d1^(1,0)[x, y])/(3 - 3 ν^2) + (4 e h^3 s0 w1^(1,0)[x, y])/(3 (-1 + ν) (1 + ν) (x^2 + s0^2))
```

The internal forces are substituted into the differential equations, obtained in point 2.1, to receive them in terms of displacements. They are long, for example the first one for the catenoidal shell contains 17 terms so it will not be presented here.

It can now be stated that it is possible to proceed from the very general equations to very specific ones which are ready for numerical computations. The next section will deal with these problems.

3 Numerical tasks - boundary value problems using the refined least squares method

3.1 Basic features

The least squares method is a well–known, [15], meshless way of finding an approximate solution to a boundary value problem. The classical approach consists in minimising the functional based on algebraic, differential or integral equations, or on a system of equations with a set of independent functions which satisfy boundary conditions.

The refined least squares (RLS) method uses the following approach. The minimized functional is supplemented with the terms responsible for boundary conditions. The approximating functions do not have to satisfy the boundary or initial conditions but they must be linearly independent. The basic features of the RLS method are described in [6] and [7].

Described below is an example of applying the RLS method to the tasks of computing a short and long cylindrical shell, which are specific boundary layer problems. Another example is presented in [12].

The set of equations presented below, functions in boundary conditions have been developed with *MATHEMATICA*® and the ***MathTensor***™ package in a way presented in the previous section.

3.2 Problem description

Physical description Let us consider two steel (Young modulus $E = 2 \cdot 10^8$, Poisson ratio $\nu = \frac{3}{10}$) cylindrical shells of length $l = 4$ m (short) and $l = 60$ m (long), thickness $2h = 10$ mm, and cylinder radius $s_o = 2$ m. It is subjected to the periodical, with regard to the direction of parallel, load normal to the reference surface $p_3 = P_3 \cos(ny)$, $n = 5$, $P_3 = 1$ kN/m^2 normal to the cylinder mid–surface, tangent load components $p_1 = 0$, $p_2 = 0$.

The shapes of the undeformed reference surfaces are presented in the figures 2 and 3.

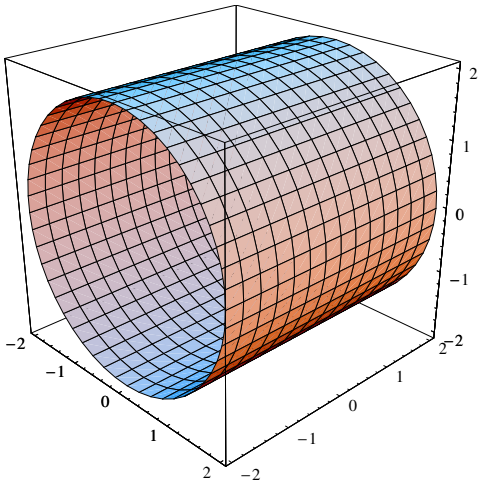

Fig. 2. Short shell

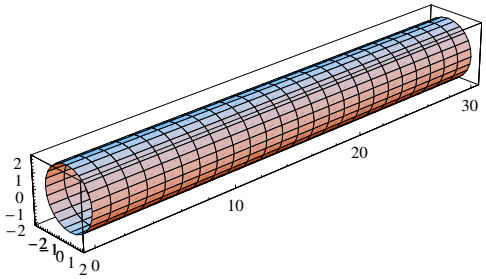

Fig. 3. Long shell, from the plane of symmetry $x = 0$ to the fixed end $x = 30$

The variable $y \in \langle 0, 2\pi \rangle$ is measured along parallel of the cylinder, and $x \in \langle x_a, x_b \rangle$ along the meridian. The cylinder is fixed on both edges $x_a = -\frac{l}{2}$ and $x_b = \frac{l}{2}$.

Due to the axial symmetry of the cylinder the two–dimensional problem can be reduced to one–dimensional task by Fourier expansion.

From the engineering point of view the problem can be considered as a static of a steel pipe loaded with a fifth component of the wind.

Equations The problem is described by a system of five second–order ordinary differential equations obtained within *MATHEMATICA*® notebooks discussed in the previous section. The following notation is used:

$\mathcal{D}_1(x)\ \cos(y)$— meridian rotation component,

$\mathcal{D}_2(x)\ \sin(y)$— parallel rotation component,

$\mathcal{W}_1(x)\cos(y)$— meridian displacement component,

$\mathcal{W}_2(x)\ \sin(y)$— parallel displacement component,

$\mathcal{W}_3(x)\cos(y)$— normal displacement component.

The system of equations represents the equilibrium state of the shell. The right hand sides (rhs) of the equations are equal to zero, and below are their left hand sides (lhs):

$$e_1 := -\frac{E\, n^2\, \mathcal{D}_1(x)\ h^3}{3\, s_o^3\ (1+\nu)} - \frac{2\, E\, \mathcal{D}_1''(x)\ h^3}{3\, s_o\ \left(1-\nu^2\right)} - \frac{E\ \left(h^2+3\, s_o^2\right) n^2\, \mathcal{W}_1(x)\ h}{3\, s_o^4\ (1+\nu)} + \\ + \frac{2\, E\, \mathcal{W}_1''(x)\ h}{1-\nu^2} + \frac{n\, E\, \mathcal{W}_2'(x)\ h}{s_o\ (1-\nu)} - \frac{2\, E\, \nu\, \mathcal{W}_3'(x)\ h}{s_o\ \left(1-\nu^2\right)} + P_1(x)\,, \tag{28a}$$

$$
\begin{aligned}
e_2 :=& \frac{5\,h\,E\,\mathcal{D}_1'(x)}{6\,(1+\nu)} + \frac{h\,n\,E\,\left(4\,h^2+5\,(1-\nu)\,s_o^2\right)\mathcal{D}_2(x)}{6\,s_o^3\,\left(1-\nu^2\right)} + \\
&+ \frac{2\,h\,E\,\nu\,\mathcal{W}_1'(x)}{s_o\,\left(1-\nu^2\right)} + \frac{h\,n\,E\,\left(4\,h^2+(17-5\,\nu)\,s_o^2\right)\mathcal{W}_2(x)}{6\,s_o^4\,\left(1-\nu^2\right)} + \\
&- \frac{E\,h\,\left(4\,h^2+\left(5\,(1-\nu)\,n^2+12\right)s_o^2\right)\mathcal{W}_3(x)}{6\,s_o^4\,\left(1-\nu^2\right)} + \frac{5\,h\,E\,\mathcal{W}_3''(x)}{6\,(1+\nu)} + P_3(x),
\end{aligned}
\tag{28b}
$$

$$
\begin{aligned}
e_3 :=& -\frac{E\,\left(2\,h^2\,n^2+5\,s_o^2\right)\mathcal{D}_1(x)\,h}{6\,s_o^2\,(1+\nu)} + \frac{2\,E\,\mathcal{D}_1''(x)\,h^3}{3\,\left(1-\nu^2\right)} + \frac{n\,E\,\mathcal{D}_2'(x)\,h^3}{3\,s_o\,(1-\nu)} + \\
&- \frac{E\,n^2\,\mathcal{W}_1(x)\,h^3}{3\,s_o^3\,(1+\nu)} - \frac{2\,E\,\mathcal{W}_1''(x)\,h^3}{3\,s_o\,\left(1-\nu^2\right)} - \frac{5\,E\,\mathcal{W}_3'(x)\,h}{6\,(1+\nu)},
\end{aligned}
\tag{28c}
$$

$$
\begin{aligned}
e_4 :=& -\frac{E\,\left(4\,h^2\,n^2+5\,(1-\nu)\,s_o^2\right)\mathcal{D}_2(x)\,h}{6\,s_o^4\,\left(1-\nu^2\right)} - \frac{E\,\mathcal{D}_2''(x)\,h^3}{3\,s_o^2\,(1+\nu)} - \frac{E\,n\,\mathcal{W}_1'(x)\,h}{s_o^2\,(1-\nu)} + \\
&+ \frac{E\,\mathcal{W}_2''(x)\,h}{s_o\,(1+\nu)} - \frac{E\,\left(4\,h^2\,n^2+\left(12\,n^2-5\,\nu+5\right)s_o^2\right)\mathcal{W}_2(x)\,h}{6\,s_o^5\,\left(1-\nu^2\right)} + \\
&+ \frac{n\,E\,\left(4\,h^2+(17-5\,\nu)\,s_o^2\right)\mathcal{W}_3(x)\,h}{6\,s_o^5\,\left(1-\nu^2\right)} + P_2(x),
\end{aligned}
\tag{28d}
$$

$$
\begin{aligned}
e_5 :=& -\frac{E\,n\,\mathcal{D}_1'(x)\,h^3}{3\,s_o^2\,(1-\nu)} - \frac{E\,\left(4\,h^2\,n^2+5\,(1-\nu)\,s_o^2\right)\mathcal{D}_2(x)\,h}{6\,s_o^3\,\left(1-\nu^2\right)} + \frac{E\,\mathcal{D}_2''(x)\,h^3}{3\,s_o\,(1+\nu)} + \\
&- \frac{E\,\left(4\,h^2\,n^2+5\,(1-\nu)\,s_o^2\right)\mathcal{W}_2(x)\,h}{6\,s_o^4\,\left(1-\nu^2\right)} - \frac{E\,\mathcal{W}_2''(x)\,h^3}{3\,s_o^2\,(1+\nu)} + \\
&+ \frac{n\,E\,\left(4\,h^2+5\,(1-\nu)\,s_o^2\right)\mathcal{W}_3(x)\,h}{6\,s_o^4\,\left(1-\nu^2\right)}.
\end{aligned}
\tag{28e}
$$

Boundary conditions It is obvious that the system of 5 second–order differential equations system (28) requires 10 boundary conditions. Two types of boundary conditions can be distinguished: essential conditions and boundary layer conditions.

The **essential conditions** (lhs) for our problem are:

$$
\begin{aligned}
b_1 &:= \mathcal{W}_1(x_a), \\
b_2 &:= \mathcal{W}_2(x_a), \\
b_3 &:= \mathcal{W}_1(x_b), \\
b_4 &:= \mathcal{W}_2(x_b).
\end{aligned}
\tag{29}
$$

Their (rhs) are equal to zero. These boundary conditions are the same as used in the membrane approach to the problem. Here it needs to be stated that the membrane approach is not correct for the considered task as the cylindrical shell is significantly bent by this type of load so the moments and transverse forces cannot be neglected.

The **boundary–layer conditions** (lhs) for our problem are:

$$\begin{aligned} b_5 &:= \mathcal{D}_1(x_a), \\ b_6 &:= \mathcal{D}_2(x_a), \\ b_7 &:= \mathcal{W}_3(x_a), \\ b_8 &:= \mathcal{D}_1(x_b), \\ b_9 &:= \mathcal{D}_2(x_b), \\ b_{10} &:= \mathcal{W}_3(x_b). \end{aligned} \tag{30}$$

3.3 Method description

For the considered system (28) *a–e* of the equations and boundary conditions (29) and (30) the following functional can be built:

$$\mathcal{F} = \int_{x_a}^{x_b} \left(\sum_{n=1}^{5} (\alpha_n e_n)^2 \right) dx + \sum_{k=1}^{10} (\beta_k b_k)^2, \tag{31}$$

where α_n and β_k are scale factors or scale functions.

The RLS method consists in minimising of this functional. If the solution is exact the value of the functional is zero, otherwise it is positive. Minimisation is done by the Ritz method. According to it the approximation of $f(x)$ can be predicted in a form of linear combination of m independent functions $u_i(x)$.

$$f(x) = \sum_{i=1}^{m} C_i u_i(x). \tag{32}$$

Substituting (32) into (31) for each $f(x)$ (it stands here for $\mathcal{D}_1(x)$, $\mathcal{D}_2(x)$, $\mathcal{W}_1(x)$, $\mathcal{W}_2(x)$ and $\mathcal{W}_3(x)$, respectively) one can compute the derivative of $\mathcal{F}$ with respect to C_i. Minimisation of the functional is equivalent to the condition:

$$\frac{\partial \mathcal{F}}{\partial C_i} = 0. \tag{33}$$

Doing it for each unknown C_i a system of algebraic equations is obtained. For a linear problem it is linear one $\left(A_{ij} C_j = B_i\right)$ with a symmetrical and positive definite matrix A_{ij}.

The formulae for developing the terms of the system and the *MATHEMATICA*® implementation are presented below.

3.4 Method implementation

Each unknown function of the system of differential equations (28) can be approximated with a linear combination of monic Chebyshev polynomials. As the considered

problem is symmetrical with regard to the plane $x = 0$, the functions $\mathcal{W}_1(x)$ and $\mathcal{D}_1(x)$ are antisymmetrical and $\mathcal{W}_2(x)$, $\mathcal{W}_3(x)$ and $\mathcal{D}_2(x)$ are symmetrical. The system can be informed about it with the following definitions, for example:

```
In[1]:= w1[x_] :=
          Sum[c[5 i] MonicChebyshevT[2i + 1, (2 (x - bx))/(ax - bx) - 1], {i, 0, PolyDegree}];
In[2]:= w2[x_] :=
          Sum[c[5 i + 1] MonicChebyshevT[2i, (2 (x - bx))/(ax - bx) - 1], {i, 0, PolyDegree}];
```

where **ax** stands for x_a, **bx** for x_b, **c[k]** for C_k and:

```
In[3]:= MonicChebyshevT[n_, x_] := ChebyshevT[n, x]/2^(n-1)
```

It is not difficult to notice that the real domain $\langle x_a, x_b \rangle$ is transformed to the interval $\langle -1, 1 \rangle$.

Each differential equation is scaled and saved into the variable, for example:

```
In[4]:= DifferentialEquation[1] :=
          DifferentialEquation[1] = r1/h;
```

According to this approximation the scaled differential equation $\alpha_k\, e_k$ and boundary condition $\beta_n\, b_k$ can be rewritten in the following form:

$$\alpha_k\, e_k(x) = f_k(x) + \sum_{i=0}^{5(p+1)-1} C_i\, m_{ik}(x)\,, \tag{34}$$

$$\beta_k\, b_k = g_k + \sum_{i=0}^{5(p+1)-1} C_i\, n_{ik}. \tag{35}$$

where p is an assumed polynomial degree.

Free terms f_k are extracted from (34) with the following functions:

```
In[5]:= DiffEqnFreeTerm[k_] := DiffEqnFreeTerm[k] =
          Expand[-DifferentialEquation[k] /. c[_] → 0];
```

The functions $m_{ik}(x)$ can also be found with the very similar procedure. Standard *MATHEMATICA*® functions could be used but this one is faster.

```
In[6]:= DiffEqnCoefficient[i_, k_] := DiffEqnCoefficient[i, k] =
          Expand[DifferentialEquation[k] +
                 DiffEqnFreeTerm[k] /. c[i] → 1 /. c[_] → 0]
```

Similarly each scaled boundary condition for the task is saved into the variable. Here the full set of *MATHEMATICA*® input for the edge x_a can be seen, the other 5 are similar:

```
In[7]:= BoundaryCondition[1] := BoundaryCondition[1] = w1[ax] e;

In[8]:= BoundaryCondition[2] := BoundaryCondition[2] = w2[ax] e;

In[9]:= BoundaryCondition[5] := BoundaryCondition[5] = d1[ax] e ;

In[10]:= BoundaryCondition[6] := BoundaryCondition[6] = d2[ax] e ;

In[11]:= BoundaryCondition[7] :=
           BoundaryCondition[7] = w3[ax] e;
```

The following commands are applied to make the extractions of $n_{ik}(x)$ and g_k from the boundary conditions (35).

```
In[12]:= BoundCondFreeTerm[i_] := BoundCondFreeTerm[i] =
           Expand[-BoundaryCondition[i] /. c[_] → 0];

In[13]:= BoundCondCoefficient[k_, i_] :=
           BoundCondCoefficient[k, i] =
             Expand[BoundaryCondition[i] +
               BoundCondFreeTerm[i] /. c[k] → 1 /. c[_] → 0]
```

As $x \in \langle x_a, x_b \rangle$, coefficients of the symmetric matrix can be computed with the following formula:

$$A_{ij} = \int_{x_a}^{x_b} \left(\sum_{k=1}^{5} m_{ik}(x)\, m_{jk}(x) \right) dx + \sum_{k=1}^{10} n_{ik}\, n_{jk}. \tag{36}$$

It is done with:

```
In[14]:= MatrixCoefficient[i_, j_] :=
           MatrixCoefficient[i, j] = MatrixCoefficient[j, i] =
             Expand[integ1[Expand[
           Sum[DiffEqnCoefficient[i, k] DiffEqnCoefficient[j, k], {k, 5}]]] +
             Expand[
           Sum[BoundCondCoefficient[i, k]
             BoundCondCoefficient[j, k], {k, 10}]]
```

where **integ1** is a function for computing a definite integral. There are predefined formulas for integrating monomials and dealing with sums, it makes calculations faster.

```
In[15]:= integ1[z_Plus] := xxinteg1 /@ z;

In[16]:= integ1[z_] := xxinteg1[z];

In[17]:= xxinteg1[z_] := integx[z, ax, bx];

In[18]:= integx[a_ x^n_., ax_, bx_] := a (-ax^(1+n) + bx^(1+n)) / (1 + n) /; FreeQ[a, x];
```

```
In[19]:= integx[x^n_., ax_, bx_] := (bx^(1+n) - ax^(1+n))/(1+n);
In[20]:= integx[a_, ax_, bx_] := a (bx - ax)/; FreeQ[a, x];
In[21]:= integx[z_, ax_, bx_] := ∫_ax^bx z dx;
```

System matrix is built from the matrix coefficients. The matrix is symmetrical so only the lower triangle should be saved.

```
In[22]:= SystemMatrix := SystemMatrix =
           Table[Table[
             MatrixCoefficient[j, i],
               {i, 0, j}], {j, 0, NumberOfEqns}]
```

where, for our task:

```
In[23]:= NumberOfEqns = 5 (PolyDegree + 1) - 1
```

Analogously elements of the free vector can be obtained:

$$B_i = \int_{x_a}^{x_b} \left(\sum_{k=1}^{5} m_{ik}(x) \; f_k \right) dx + \sum_{k=1}^{10} n_{ik} \, g_k. \tag{37}$$

It is done with:

```
In[24]:= FreeVecCoefficient[i_] :=
           FreeVecCoefficient[i] =
             Expand[integ1[Expand[
               Sum[DiffEqnCoefficient[i, k] DiffEqnFreeTerm[k], {k, 1, 5}]]]
             +Expand[
               Sum[BoundCondCoefficient[i, k]
                 BoundCondFreeTerm[k], {k, 1, 10}]]]
In[25]:= FreeVector := FreeVector =
             Table[FreeVecCoefficient[i],
               {i, 0, NumberOfEqns}]
```

As the matrix A_{ij} is positive definite and symmetrical the Cholesky-Banachiewicz method is used for the solution of the linear system of equations. This is the only part of the procedure done numerically. The results of the approximation are functions.

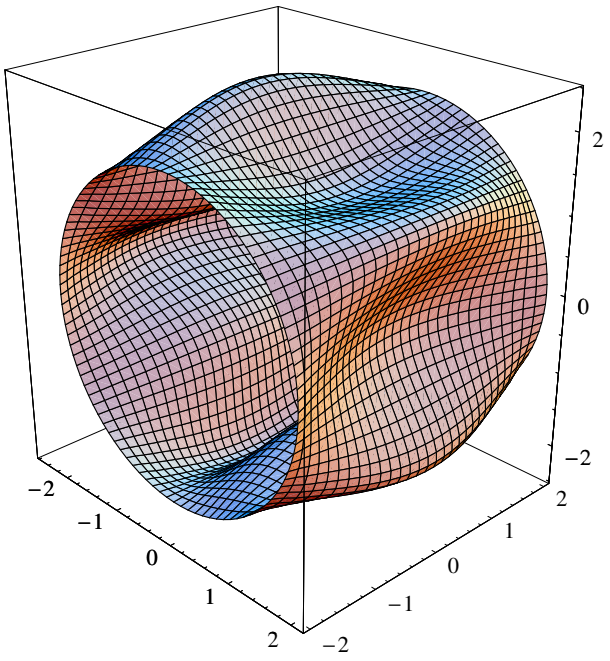

Fig. 4. Short shell: deformation (exaggeration 5000 times)

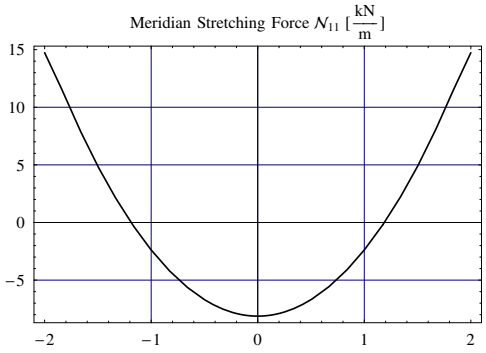

Fig. 5. Short shell approximation

3.5 One step approach

Short shell The approximation for the short shell ($l = 4$ m) is not difficult. The deformed shape of it is presented in figure 4.

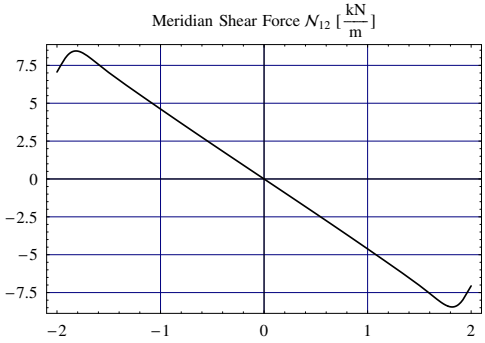

Fig. 6. Short shell approximation

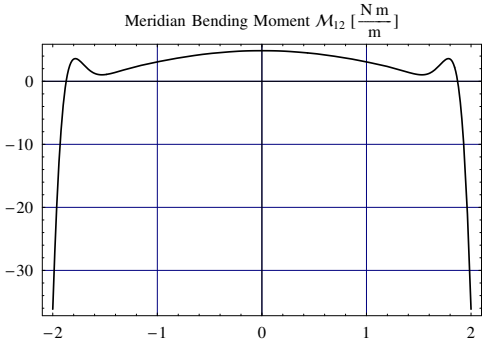

Fig. 7. Short shell approximation

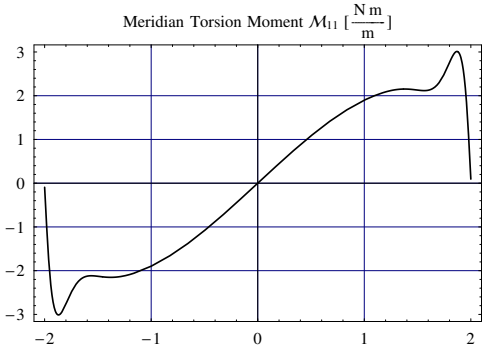

Fig. 8. Short shell approximation

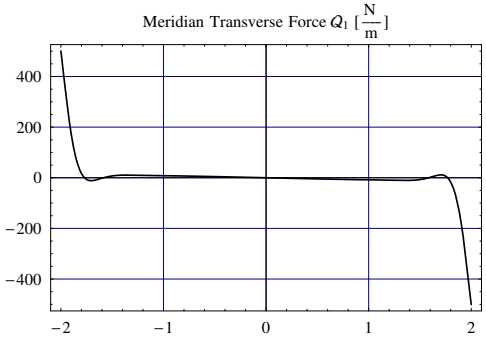

Fig. 9. Short shell approximation

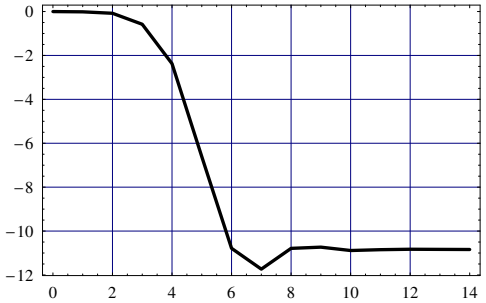

Fig. 10. Short shell approximation: convergence analysis

The figures 5, 6, 7, 8 and 9 show diagrams of some physical internal forces for the considered case. Here $\mathcal{N}_{11}(x)\ \cos(y)$ is a meridian stretching force and $\mathcal{N}_{12}(x)\ \sin(y)$ is a meridian shear force, $\mathcal{M}_{11}(x)\ \sin(y)$ is a meridian torsion moment, $\mathcal{M}_{12}(x)\cos(y)$ is a meridian bending moment, and $\mathcal{Q}_1(x)\ \cos(y)$ is a meridian transverse force. They are defined as follows:

$$\mathcal{N}_{11}(x) = -\frac{2\,E\,\mathcal{D}_1'(x)\,h^3}{3\,s_o\,\left(1-\nu^2\right)} + \frac{2\,E\,\mathcal{W}_1'(x)\,h}{1-\nu^2} + \frac{2\,n\,E\,\nu\,\mathcal{W}_2(x)\,h}{s_o\,\left(1-\nu^2\right)} - \frac{2\,E\,\nu\,\mathcal{W}_3(x)\,h}{s_o\,\left(1-\nu^2\right)}, \quad (38)$$

$$\mathcal{N}_{12}(x) = -\frac{E\,\mathcal{D}_2'(x)\,h^3}{3\,s_o\,(1+\nu)} - \frac{E\,n\,\mathcal{W}_1(x)\,h}{s_o\,(1+\nu)} + \frac{E\,\mathcal{W}_2'(x)\,h}{1+\nu}, \quad (39)$$

$$\mathcal{M}_{11}(x) = \frac{n\,E\,\mathcal{D}_1(x)\,h^3}{3\,s_o\,(1+\nu)} - \frac{E\,\mathcal{D}_2'(x)\,h^3}{3\,(1+\nu)} + \frac{E\,\mathcal{W}_2'(x)\,h^3}{3\,s_o\,(1+\nu)}, \quad (40)$$

$$\mathcal{M}_{12}(x) = \frac{2\,E\,\mathcal{D}_1'(x)\,h^3}{3\,\left(1-\nu^2\right)} + \frac{2\,n\,E\,\nu\,\mathcal{D}_2(x)\,h^3}{3\,s_o\,\left(1-\nu^2\right)} - \frac{2\,E\,\mathcal{W}_1'(x)\,h^3}{3\,s_o\,\left(1-\nu^2\right)}, \quad (41)$$

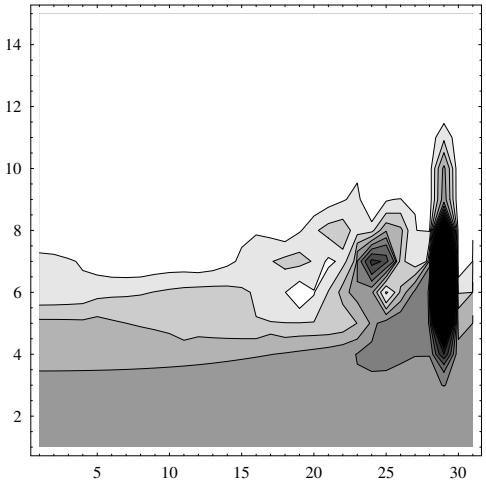

Fig. 11. Short shell: relative error of the convergence

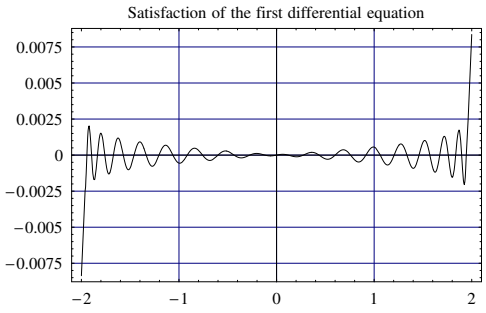

Fig. 12. Short shell approximation

$$Q_1(x) = \frac{5\,h\,E\,\mathcal{D}_1(x)}{6\,(1+\nu)} + \frac{5\,h\,E\,\mathcal{W}_3'(x)}{6\,(1+\nu)}. \tag{42}$$

The figures 10 and 11 present analysis of the approximation convergence. The figure 10 shows convergence of the function value $\mathcal{M}_{12}(-1.933\,\mathrm{m})$ with respect to the value of **`PolyDegree`**. The figure 11 demonstrates function value $\mathcal{M}_{12}$ in 31 point in the interval $x \in \langle 0, \frac{l}{2} \rangle$; variable on the vertical axis is **`PolyDegree`**, lighter colour represent smaller relative error with respect to the approximation for the highest value of **`PolyDegree`** used in computations. It can be seen that the approximation close to exact solution is reached when value of the variable **`PolyDegree`** is equal to 14. It means that the degrees of the approximating polynomials attain value 28 for even functions and 29 for odd ones.

The obtained functions can be substituted to the approximated equations to observe the error. The figure 12 presents (dis)satisfaction of the first differential equation for the considered case.

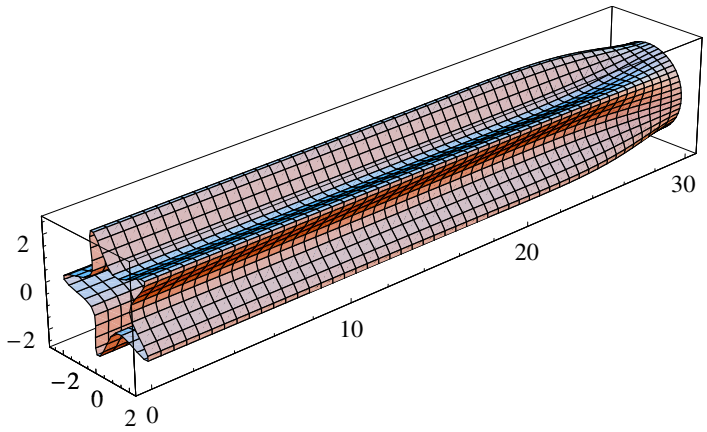

Fig. 13. Long shell: deformation from the plane of symmetry $x = 0$ to the fixed end $x = 30$ (exaggeration 500 times)

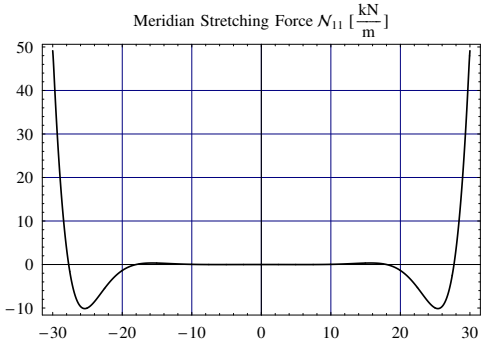

Fig. 14. Long shell, one step approximation approach

Long shell If the approximation of the problem of a long cylindrical shell had been done with a full set of boundary conditions, then difficulties would have been experienced with convergence and stability. This is a high-order differential operator and therefore the problem is ill-conditioned. The degrees of the approximating polynomials

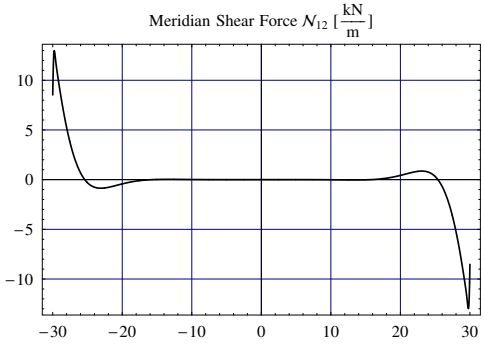

Fig. 15. Long shell, one step approximation approach

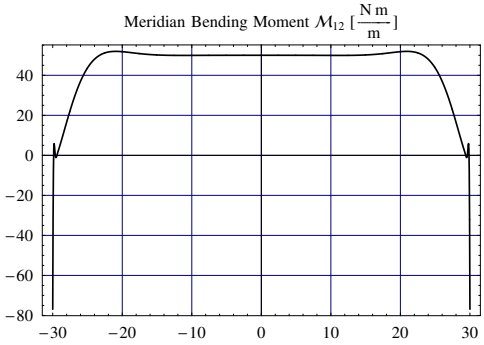

Fig. 16. Long shell, one step approximation approach

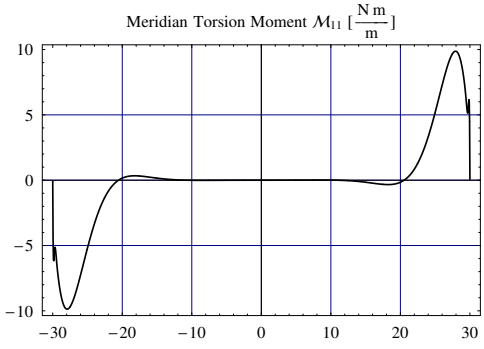

Fig. 17. Long shell, one step approximation approach

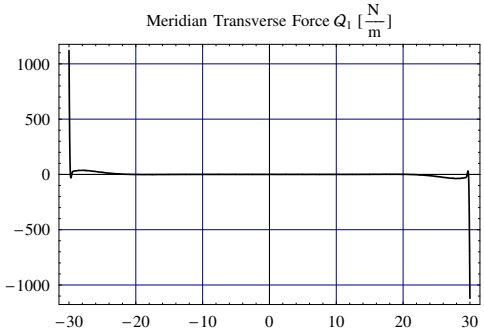

Fig. 18. Long shell, one step approximation approach

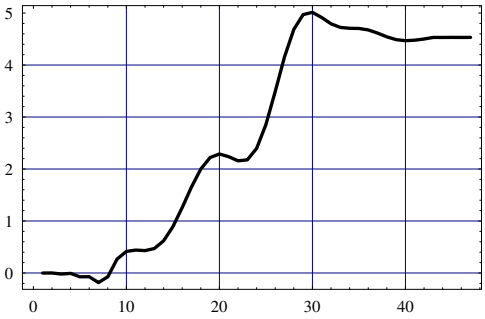

Fig. 19. Long shell, one step approximation approach: convergence analysis

have to be quite large number and the working precision of the computations have to be equal to 512. Despite of this the loss of the precision has been over 300 digits.

The results of this approximation are presented in figures 14, 15, 16, 17 and 18.

The figures 19 and 20 show analysis of the approximation convergence. The figure 19 shows convergence of the function value $\mathcal{M}_{12}(-29\,\mathrm{m})$ with respect to the value of **`PolyDegree`**. The figure 20 demonstrates function value $\mathcal{M}_{12}$ in 31 point in the interval $x \in \langle 0, \frac{l}{2} \rangle$; variable on the vertical axis is **`PolyDegree`**, lighter colour represent smaller relative error with respect to the approximation for the highest value of **`PolyDegree`** used in computations. It can be seen that the approximation close to exact solution is reached when value of the variable **`PolyDegree`** is equal to 46. It means that the degrees of the approximating polynomials attain value 92 for even functions and 93 for odd ones. The computational process becomes very long because the system has a lot integrals to be computed symbolically.

The figure 21 show (dis)satisfaction of the first differential equation for the considered case.

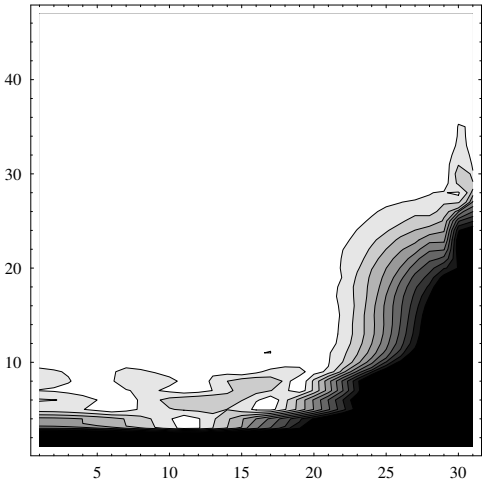

Fig. 20. Long shell, one step approximation approach: relative error of the convergence

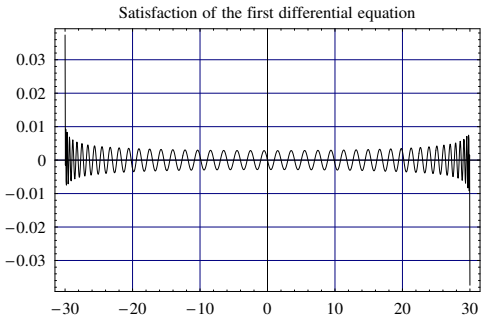

Fig. 21. Long shell, one step approximation approach

3.6 Two step approach

One of the most important features of the RLS method is the scale factors by which better or worse satisfaction a selected equation or boundary condition can be enforced. The idea of the base solution presented below has been developed by experimenting with scale factors. When decreasing a scale factor in some boundary conditions a zero has been put in instead of a very small number and it has turned out to still produce a very good approximation.

Base solution It has been found that it is possible to approximate the system (28) with the RLS method, taking into account only the essential boundary conditions (29) and

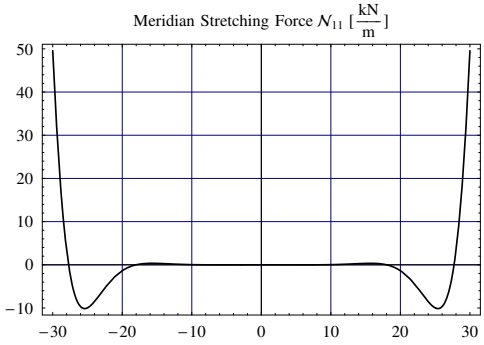

Fig. 22. Long cylindrical shell, ”base” approximation

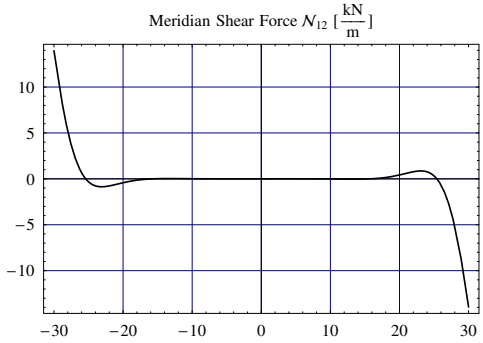

Fig. 23. Long shell, ”base” approximation

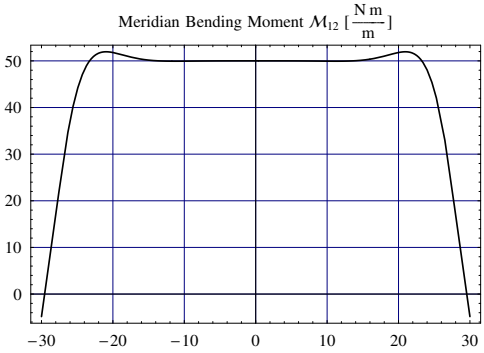

Fig. 24. Long shell, ”base” approximation

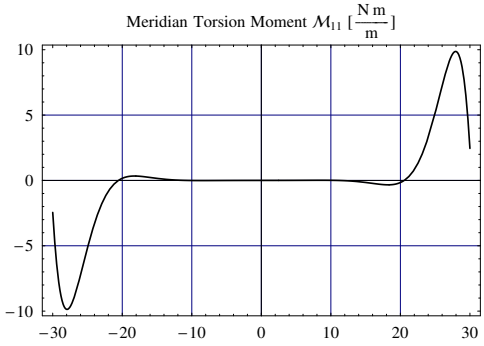

Fig. 25. Long shell, ”base” approximation

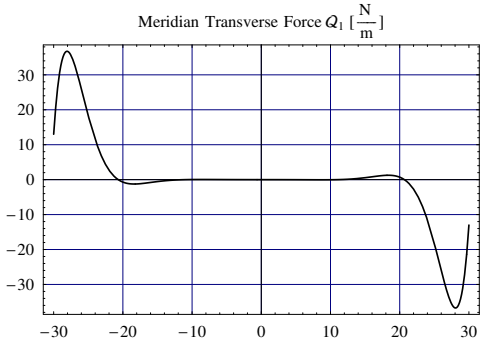

Fig. 26. Long shell, ”base” approximation

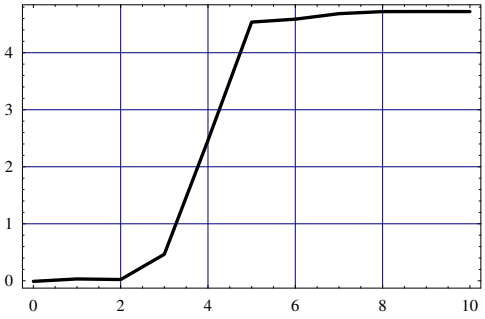

Fig. 27. ”Base” approximation: convergence analysis

neglecting the boundary–layer ones (30). As it has been already mentioned they are simply multiplied by zero, for example:

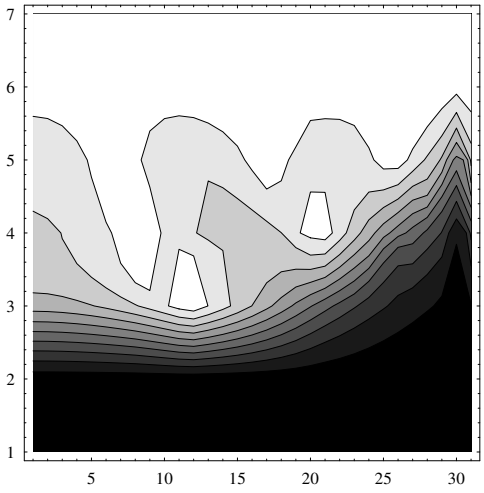

Fig. 28. "Base" approximation: relative error of the convergence

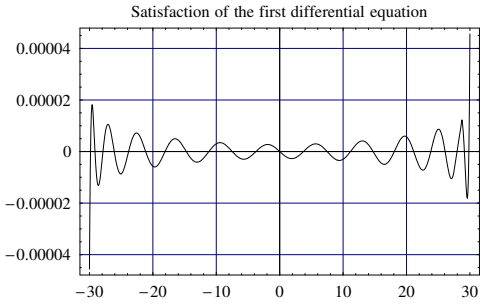

Fig. 29. Long shell, "base" approximation

```
In[26]:= BoundaryCondition[5] :=
           BoundaryCondition[5] = D1[ax] e 0 ;
```

It is an unexpected situation but it can be interpreted physically - the problem is statically indeterminate if all boundary conditions on fixed edge are satisfied. Some the boundary conditions which are not essential for the overall stability can be released. It has been found that it is enough to apply polynomials much lower degree than for one step approach to obtain quite good results. The results of this approximation are shown in figures 22, 23, 24, 25 and 26.

The figures 27 and 28 show analysis of the approximation convergence. The figure 27 shows convergence of the function value $\mathcal{M}_{12}\,(-29\,\mathrm{m})$ with respect to the value of **PolyDegree**. The figure 28 demonstrates function value $\mathcal{M}_{12}$ in 31 point in the

interval $x \in \langle 0, \frac{l}{2} \rangle$; variable on the vertical axis is **PolyDegree**, lighter colour represent smaller relative error with respect to the approximation for the highest value of **PolyDegree** used in computations. It can be seen that the approximation close to exact solution is reached when value of the variable **PolyDegree** is equal to 10. It means that the degree of the approximating polynomial attains value 20 for even functions and 21 for odd ones.

The figure 29 shows (dis)satisfaction of the first differential equation for the considered case.

The solution is feasible on the most of the domain except boundary layers which sizes are limited to about 1.2 m from each edge. The actual functions are highly oscillating but they quickly decay to the base solution, which is smooth. Hence, it is practically impossible to satisfy all the boundary conditions in one step, as the problem becomes ill–conditioned in the Lyapunov sense and is slowly convergent. The base solution is not a membrane because the moments and shear forces are not zero functions. Note that neglecting of some boundary conditions in other methods usually results in a singular system. Here a well–conditioned and non–singular problem is obtained.

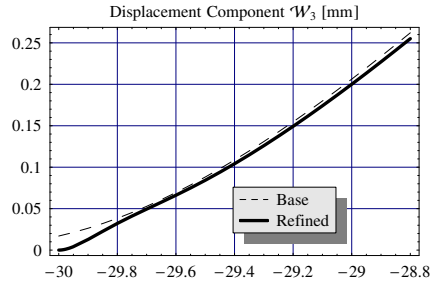

Fig. 30. Boundary layer refinement of the ” base” approximation

Boundary–layer refinement The base solution allows refinement of the approximation at the boundary layers.

We are taking into account limited domain so the parameters describing the position of the boundaries take values:

```
In[27]:= ax = -30;
```

```
In[28]:= bx = -288/10;
```

As none symmetry can be expected the approximation polynomials should contain both odd and even functions.

$$\mathit{In[29]:=}\ \mathcal{W}_1[x\_] := \sum_{i=0}^{\mathrm{PolyDegree}} c[5\,i]\ \mathrm{MonicChebyshevT}\left[i, \frac{2\,(x - bx)}{ax - bx} - 1\right];$$

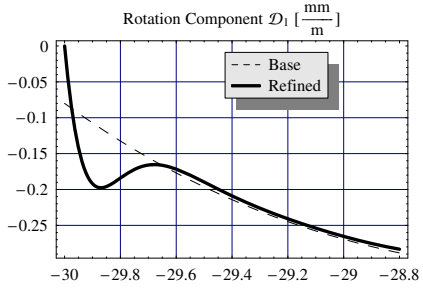

Fig. 31. Boundary layer refinement of the " base" approximation

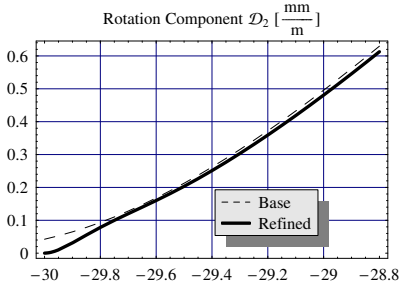

Fig. 32. Boundary layer refinement of the " base" approximation

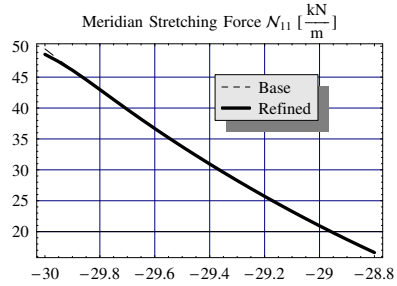

Fig. 33. Boundary layer refinement of the " base" approximation

```
In[30]:= w2[x_] :=
          PolyDegree
            ∑        c[5 i + 1] MonicChebyshevT[i, (2 (x - bx))/(ax - bx) - 1];
           i=0
```

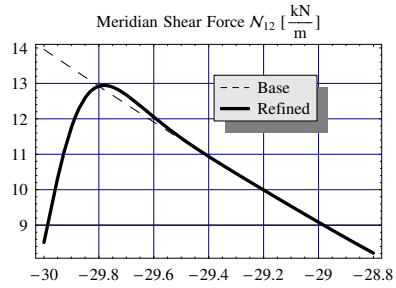

Fig. 34. Boundary layer refinement of the ” base” approximation

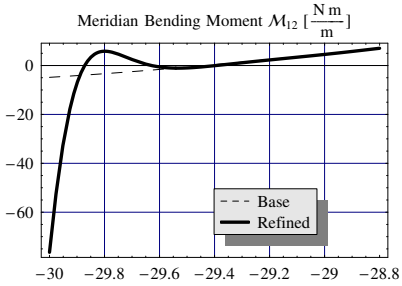

Fig. 35. Boundary layer refinement of the ” base” approximation

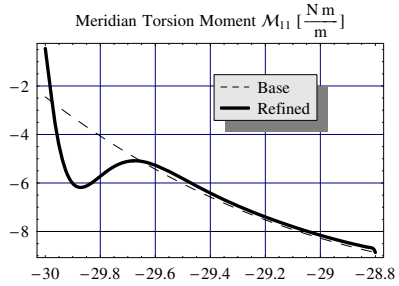

Fig. 36. Boundary layer refinement of the ” base” approximation

A full set of boundary conditions are now taken into account, for the fixed edge the boundary conditions are expressed in displacements.

```
In[31]:= BoundaryCondition[1] :=
           BoundaryCondition[1] = 𝒟₁[ax] e;
```

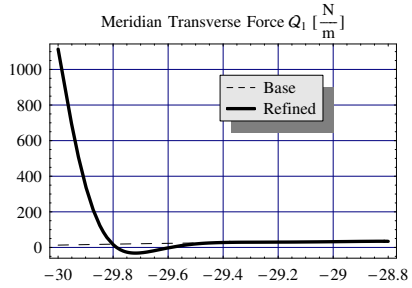

Fig. 37. Boundary layer refinement of the " base" approximation

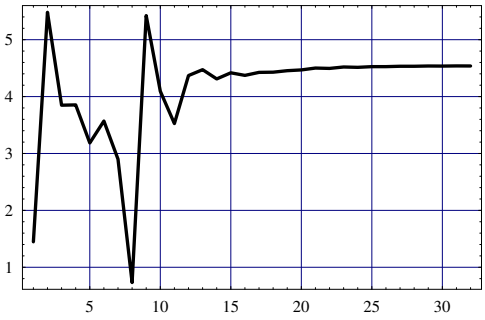

Fig. 38. Boundary layer refinement of the " base" approximation: convergence analysis

```
In[32]:= BoundaryCondition[2] :=
           BoundaryCondition[2] = 𝒟₂[ax] e;
In[33]:= BoundaryCondition[3] :=
           BoundaryCondition[3] = 𝒲₁[ax] e;
In[34]:= BoundaryCondition[4] :=
           BoundaryCondition[4] = 𝒲₂[ax] e;
In[35]:= BoundaryCondition[5] :=
           BoundaryCondition[5] = 𝒲₃[ax] e;
```

The boundary conditions on the "artificial" edge $x = 28.8$ are expressed in forces. It is assumed that the forces are equal to those obtained from the base approximation. To avoid further loss of precision we can change numerical values of base approximation with a **Rationalize** function to "exact" fractions. It is a kind of cheating but ensures symbolic computations of the coefficients of the system of algebraic equations. Moreover the calculations are quicker.

```
In[36]:= rd1 = Rationalize[n11[bx], 10^-Precision[n11[bx]]];
```

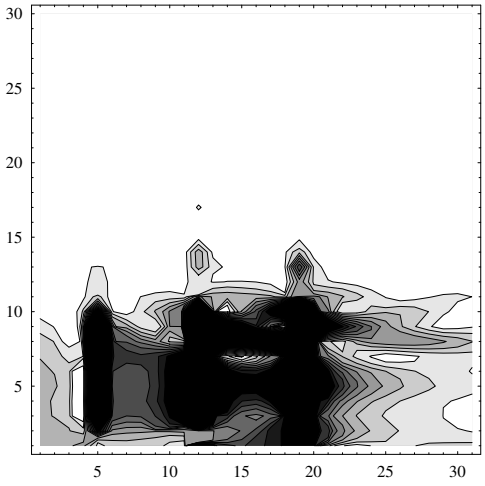

Fig. 39. Boundary layer refinement of the ” base” approximation: relative error of the convergence

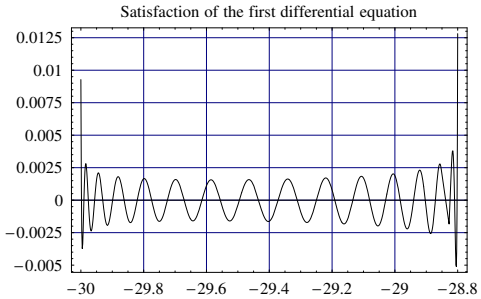

Fig. 40. Boundary layer refinement of the ” base” approximation

```
In[37]:= rd2 = Rationalize[n12[bx], 10^-Precision[n12[bx]]];
In[38]:= rd3 = Rationalize[m11[bx], 10^-Precision[m11[bx]]];
In[39]:= rd4 = Rationalize[m12[bx], 10^-Precision[m12[bx]]];
In[40]:= rd5 = Rationalize[q1[bx], 10^-Precision[q1[bx]]];
In[41]:= BoundaryCondition[6] :=
           BoundaryCondition[6] = (N11[bx] - rd1)/h;
In[42]:= BoundaryCondition[7] :=
           BoundaryCondition[7] = (N12[bx] - rd2)/h;
```

```
In[43]:= BoundaryCondition[8] :=
           BoundaryCondition[8] = (M11[bx] - rd3)/h^3;
In[44]:= BoundaryCondition[9] :=
           BoundaryCondition[9] = (M12[bx] - rd4)/h^3;
In[45]:= BoundaryCondition[10] :=
           BoundaryCondition[10] = (Q1[bx] - rd5)/h^3;
```

Satisfying all boundary conditions on the fixed edge layer conditions results in the refined functions which diagrams are shown in figures 30, 31, 32, 33, 34, 35, 36 and 37 compared to the base solution. It can be observed that the taking into account additional boundary conditions results in a small decrease of displacements. It can be interpreted physically as a local increase of the shell stiffness connected with taking into consideration more constrains in the boundary conditions. The obtained approximation error is allowable for engineering purposes.

The figures 38 and 39 show analysis of the approximation convergence. The figure 38 demonstrates convergence of the function value $\mathcal{M}_{12}(-29\,m)$ with respect to the value of **PolyDegree**. The figure 39 demonstrates function value $\mathcal{M}_{12}$ in 31 point in the interval $x \in \langle 0, \frac{l}{2} \rangle$; variable on the vertical axis is **PolyDegree**, lighter color represent smaller relative error with respect to the approximation for the highest value of **PolyDegree** used in computations. It can be seen that the approximation close to exact solution is reached when value of the variable **PolyDegree** is equal to 25. It means that the degree of the approximating polynomial attains the same value.

The figure 40 presents (dis)satisfaction of the first differential equation for the considered case.

4 Conclusions and final remarks

4.1 Symbolic tasks

It has been shown that it is possible to solve the linear and nonlinear tensor symbolic problems in different aspects with the ***MathTensor***™ package and formulate equations from the very general formulas to the ones ready for numerical computations. Each task requires different tools. There has been shown how the built in functions of the system and the package can be used efficiently. The tools are intuitive and relatively simple, nevertheless the full responsibility for the results obtained belongs to the user of the system and package.

4.2 Numerical tasks

The RLS method allows simpler algorithm to be built if there is no need to use functions that have to satisfy boundary conditions. Therefore it is applicable to more general purposes like multidimensional problems with discontinuities of boundary conditions.

The results of approximation are functions, not numbers. Therefore the results can be used directly in further computations without interpolation. The quality of the approximation can be evaluated by substituting functions into approximated equations.

Moreover it has been shown that there is the possibility of the neglecting of some boundary conditions for boundary layer problems and satisfy them locally in the next step. This phenomenon for the problem being considered has a physical interpretation. The mathematical interpretation seems to be an open problem.

According to that the two step approach to the boundary layer problems of theory of shells has been proposed. The first step consists in approximation with negligence of some boundary conditions. They are adjusted in the second step when only boundary layer is considered. This approach can be applied to other boundary layer tasks.

MATHEMATICA® as a (fully) integrated environment of symbolic and numerical computation and graphics is a very effective tool for the complex analysis of symbolic (tensor) calculations and can be applied to in the entire computational and publication process. Its external package ***MathTensor***™ is an effective tool of tensor analysis with the theory of shells. However, its long expected upgrade should be better adopted for the current possibilities of *MATHEMATICA*® 4.1.

5 Acknowledgements

The author would like to express gratitude to organisers of the SNSC'2001 conference, especially Professor Franz Winkler and Mrs. Ramona Poechinger, for the kind invitation and nice welcome.

The grant from Wolfram Research, Inc., has supported the author with a free upgrade of the system *MATHEMATICA*® 4.1.

Some notations

There are some symbols used for tensor problems discussed in the section 2.

$\boldsymbol{a}$ — determinant of the first differential form

a_{ij} — the first differential form of the reference surface, metric tensor on the reference surface

b_{ij} — the first differential form of the reference surface, curvature tensor

$\boldsymbol{d}$ — rotation vector

$\boldsymbol{d}_i$ — derivative of rotation vector

d_i — physical component of the rotation vector

d^i — contravariant component of the rotation vector in the curvilinear basis

$\boldsymbol{g}$ — determinant of the metric tensor

g_{ij} — metric tensor

$2\,h$ — shell thickness

H — mean curvature

K — Gaussian curvature

N^{ij} — tensor of stretching (tensile) forces

M^{ij} — tensor of moments

P^i — vector of loads

Q^i — vector of transverse forces

$\boldsymbol{r}$ — parameterisation vector

$\boldsymbol{r}_i$ — component of the curvilinear basis

$\boldsymbol{w}$ — displacement vector

$\boldsymbol{w}_i$ — derivative of displacement vector

w_i — physical component of the displacement vector

w^i — contravariant component of the displacement vector in the curvilinear basis

w^3 — normal displacement component

$\delta_i{}^j$ — Kronecker delta

$\overset{*}{\gamma}_{ij}$ — strain tensor in 3D space

γ_{ij} — the first strain tensor of the reference surface

ϵ — coefficient of the thermal expansion

ε_{ij} — antisymmetric object under interchange of any indices

ρ_{ij} — the second strain tensor of the reference surface

ϑ_{ij} — the third strain tensor of the reference surface

τ^{ij} — stress tensor

Other notations are explained in the text.

References

[1] Başar Y., Ding Y. (1990): Theory and finite-element formulation for shell structures undergoing finite rotations. In: Voyiadjis G.Z., Karamanlidis D. (eds) Advances in the theory of plates and shells. Elsevier Science Publishers B.V., Amsterdam, 3–26

[2] Bielak S. (1990): Theory of Shells. Studies and Monographs, **30**, Opole University of Technology, Opole

[3] Naghdi P.M. (1963): Foundations of Elastic Shell Theory. chapter 1, North-Holland Publishing CO., Amsterdam, 2–90

[4] Parker L., Christensen S.M. (1994): ***MathTensor***™: A System for Doing Tensor Analysis by Computer. Addison-Wesley, Reading, MA, USA

[5] Walentyński, R.A. (1995): Statics of Sloped Shells of Revolution, Comparison of Analytic and Numerical Approaches. PhD Thesis, Silesian University of Technology, Gliwice

[6] Walentyński, R.A. (1997): A least squares method for solving initial-boundary value problems. In: Keränen V., Mitic P., Hietamäki A. (eds) Innovations in Mathematics, Proceedings of the Second International Mathematica Symposium. Rovaniemen Ammattikorkeakoulun Julkaisuja, Computational Mechanics Publications, Boston, MA; Southampton, 483–490

[7] Walentyński R.A. (1997): Computer assisted analytical solution of initial-boundary value problems. In: Garstecki A., Rakowski J., (eds) Computer Methods in Mechanics, Proceedings of the 13^{th} PCCMM. Poznań University of Technology, Poznań, 1363–1400

[8] Walentyński R.A. (1998): Refined constitutive shell equations. In: Chróścielewski J. et al., (eds), Shell Structures, Theory and Applications, Proceedings of the 6^{th} SSTA Conference. Technical University of Gdańsk, Gdańsk–Jurata, 275–276

[9] Walentyński R.A. (1999): Geometrical description of shell with computer algebra system. In Proceedings of the 11th International Scientific Conference. Brno, Technical University of Brno, 51–54

[10] Walentyński R.A. (1999): Computer assisted refinement of shell's constitutive equations. In: Łakota W. et al. (eds) Computer Methods in Mechanics, Proceedings of the 14^{th} PCCMM. Rzeszów University of Technology, Rzeszów, 381–382

[11] Walentyński R.A. (1999): Refined constitutive shell equations with ***MathTensor***™. In: Keränen V. (ed) Mathematics with Power: Proceedings of the Third International *MATHEMATICA*® Symposium'99. Research Institute for Symbolic Computations, Hagenberg-Linz, `http://south.rotol.ramk.fi/~keranen/IMS99/`

[12] Walentyński R.A. (2001): Refined least squares method for shells boundary value problems. In: Tazawa Y. et al. (eds) Symbolic Computation: New Horizons, Proceedings of the Fourth International Mathematica Symposium, Tokyo Denki University Press, Tokyo, 511–518, electronic extended version on the conference CDROM

[13] Walentyński R.A. (2001): Solving symbolic tasks in theory of shells of shell with computer algebra system. In: Kralik J. (ed) Proceedings of the 1^{st} International Scientific Seminar: New Trends in Statics and Dynamics of Buildings, STU, Bratislava, 183–188

[14] Wolfram S. (1999): The *MATHEMATICA*® book. Cambridge University Press and Wolfram Research, Inc., New York, Champaign, fourth edition

[15] Zwillinger D. (1989): Handbook of differential equations. Academic Press Inc., New York, second edition