Static Analysis of Eolic Blade through Finite Element Method and OOP C++

Tiago Freires¹, Silvia Caroline¹, Raimundo Menezes Jr² and Moises Meneses Salvino³

1. Alternative and Renewable Energy, Department of Renewable Energy Engineering, Center Federal University of Paraiba, João Pessoa 58051-900, Brazil

2. Laboratory of Numerical Methods Distributed Processing, Department of Renewable Energy Engineering, Federal University of Paraiba, João Pessoa 58051-900, Brazil

3. Energy Efficiency Lab Hydraulics and Sanitation, Federal University of Paraiba, João Pessoa 58051-900, Brazil

Received: July 27, 2014 / Accepted: September 30, 2014 / Published: January 31, 2015.

Abstract: This work deals with a description of an elastic analysis of eolic blade (preprocessing, processing and post-processing stages). The eolic blade geometry is approximated by flat finite elements in which the membrane effects are evaluated using the FF (free formulation) finite element and the flexure effects are calculated using DKT (discrete shear triangle) finite element. The pre-processing stage is implemented using OpenGL library, to provide the graphical construction for geometry, mesh orientation, and other requirements of the finite element model. For the processing stage is built a specific dll (dynamic link library) library implemented in C++ language for the FF and DKT elements analysis. The post-processing stage has been built using specific dialogs to present all results in the graphic interface, where the static displacements of the eolic blade model are shown.

Key words: Flat shell, FEM (finite element method), DKT, FF.

1. Introduction

One of the most important problems involving generation of energy through wind source is the structural study of the blades. These structures have a high degree of complexity and its construction in general is based on international norms (expensive experimental tests) for certification of their safety in use. One of the potential tools to study and design of eolic blades is the numerical methods [1, 2] (FDM (finite difference method), FEM (finite element method), BEM (boundary element method), etc.). In this paper, it is described an attempt to use the object-oriented programming C++ and finite element method numerical capabilities to build a tool related to analysis of eolic blades with geometry approximated by triangular flat elements. In addition, standard drawing functions from OpenGL library are used to provide a more friendly and efficient pre-processing to input element geometry, mechanical properties, element connectivity, and boundary constraints. In the processing stage, the stiffness matrices related to membrane and bending effect and equivalent nodal forces (wind pressure) are evaluated by functions written in oriented object language C++ and compiled in one dll (dynamic link library) called only in the process of analysis. Finally, in the post-processing stage, all results obtained in the analysis process are shown in specific dialogs box.

2. Pre-processing

The pre-processing is the stage where input data (such as geometry, node and element numbering, mechanical properties, boundary conditions, loading) are set to perform the calculation of the discretized
problem. A full discretized model of a wind generator in the tool of analysis is shown in Fig. 1.

The tool provides all drawing capabilities necessary to incorporate remaining FEM input data information (elasticity modulus, poisson ratio, thickness, etc.), and the OpenGL library is used in order to implement the drawing routines and to assign structural analysis data. OpenGL is a cross-language API (application programming interface) for writing applications that produce 2D and 3D computer graphics [3]. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives [4, 5]. In Fig. 2, a piece of source code to draw the triangle finite element is visualized. The parametric tool use OpenGL when the user selects the menu option “Pre-Processing -> Parametric Modeling”, so that this corresponds to execute the instructions in the source code as shown in Fig. 2.

In Fig. 3, the source code of the function to draw the nodes of the finite element is shown.

When the file data with coordinate nodes, elements connections are opened, the mesh of the eolic blade in environment is automatically generated, producing a graphical representation depicted in Fig. 4.

The next step in pre-processing is related to the definition of structural element properties. This procedure can be done in specifics dialogs for the elements. All material properties are set by user and applied to the model. For the nodes, the wind load and boundary conditions that represent the fixed position for the eolic blade must be assigned. This can be done in environment dialog box, where the all six degrees in all nodes of the model can be accessed and one by one conveniently prescribed by user, as shown in Fig. 5.

The final representation of the model, with the wind load and nodal constraints applied, can be visualized in Fig. 6. In this stage, the model is done for analysis.

3. Processing

The processing is stage of the analysis in which main calculations (such as elemental stiffness matrix evaluation, structural stiffness matrix assembling, nodal equivalent force vector evaluation, algebraic system solution) are done. In present paper, the structural eolic blade problem is analyzed by superposition of bending and membrane effects using FEM. For membrane effects, FF (free formulation) finite element originally developed by Ref. [6] is used.
Fig. 2  Function to draw the triangle finite element.

```cpp
void DrawOpenGIFace()
{
  int i, j;
  float length;
  // Draw the faces using an index to the vertex array
  glBegin(GL_TRIANGLES);
  for (i = 0; i < Glyphs.NumOfFaces - 1; i++)
  {
    //Vertex One
    V1 = Glyphs.Faces[i].Nodes[0];
    g1Color4f(Glyphs.Faces[i].Color.R, Glyphs.Faces[i].Color.G, Glyphs.Faces[i].Color.B, Glyphs.Faces[i].Color.A);
    //Length
    length = sqrt(pow(Glyphs.Nodes[V1].Cx - worldCenter.Cx, 2) + pow(Glyphs.Nodes[V1].Cy - worldCenter.Cy, 2));
    if (length >= 0.0)
    {
      length = 1.0;
      g1Color4f(Glyphs.Nodes[V1].Cx / length, Glyphs.Nodes[V1].Cy / length, worldCenter.Cx, worldCenter.Cy);
    }
    g1Vertex3f(Glyphs.Nodes[V1].Cx, Glyphs.Nodes[V1].Cy, worldCenter.Cx, worldCenter.Cy);
    //Vertex Two
    V2 = Glyphs.Faces[i].Nodes[1];
    g1Color4f(Glyphs.Faces[i].Color.R, Glyphs.Faces[i].Color.G, Glyphs.Faces[i].Color.B, Glyphs.Faces[i].Color.A);
    //Length
    length = sqrt(pow(Glyphs.Nodes[V1].Cx - worldCenter.Cx, 2) + pow(Glyphs.Nodes[V1].Cy - worldCenter.Cy, 2));
    if (length >= 0.0)
    {
      length = 1.0;
      g1Color4f(Glyphs.Nodes[V1].Cx / length, Glyphs.Nodes[V1].Cy / length, worldCenter.Cx, worldCenter.Cy);
    }
    g1Vertex3f(Glyphs.Nodes[V1].Cx, Glyphs.Nodes[V1].Cy, worldCenter.Cx, worldCenter.Cy);
  }
  glEnd();
}
```

Fig. 3  Function to draw nodes of the finite element.

```cpp
void DrawOpenGINodes()
{
  int i;
  g1Vertex3f(GL_POINTS);
  for (i = 0; i < Glyphs.NumOfNodes - 1; i++)
  {
    g1Color4f(0.0, 0.0, 0.0, 0.0);
    g1Vertex3f(Glyphs.Nodes[i].Cx, Glyphs.Nodes[i].Cy, Glyphs.Nodes[i].Cz);
  }
  glEnd();
}
```

Fig. 4  Discretized blade model.
The main characteristics of this element are triangular flat geometry, three DOF (degrees of freedom) by node (two displacements on plane and one drilling rotation perpendicular to the plane) located at each triangle vertex. For bending effect analysis is used the flat triangular element DKT (Discrete Kirchhoff Theory), that have three DOF (one transverse displacement and two slopes) by node located at each triangle vertex. This element was developed to deal with thin bending plate problems and its formulation has been thoroughly discussed in Refs. [7-9]. The eighteen DOF of the...
wind turbine tower analysis element (bending + membrane problems) are shown in Fig. 7.

The analysis is done using the Saproms.dll library, in which functions and classes are implemented in oriented object language C++. For sake of conciseness, only the principal classes and functions of the Saproms.dll are described in Fig. 8.

The main function that uses methods and objects from Saproms.dll is “solver”. This function works in two steps: in the first step, the objects and functions in Saproms.exe collect the input data assigned in pre-processing stage, and store them in vectors. In the second step, these vectors are processed by objects and methods from Saproms.dll to perform the structural calculations. From the point of view of mathematics, the “solver” function for the static analysis uses the principle of total strain energy. The governing equation of the problem is given by:

$$[K] [D] = [P]$$

where, $$[K] = \sum ([K_F] + [K_{DKT}])$$ is the stiffness matrix of the structure, and $$[D]$$, $$[P]$$ are the vectors of displacement and nodal forces.

The equivalent nodal force vector $$[P]$$ is obtained from the external work done by the wind loads are expressed by:

$$T_e = \int_A g(x, y) w(x, y) dA$$

where, $$w(x, y)$$, $$g(x, y)$$ are the displacement and the wind loading in the element, $$A$$ is the area of the element, $$T_e$$ is the external work done by the wind.

The equivalent nodal force vector is equal to the vector derived from the work of external loads compared to degrees of freedom, so that the establishment of interpolations to $$w(x, y)$$ and $$g(x, y)$$ along the area of the element is necessary, assuming a linear variation in Fig. 9:

$$w = w_i(1 - \xi - \eta) + w_j \xi + w_k \eta$$

$$g = g_i(1 - \xi - \eta) + g_j \xi + g_k \eta$$

where, $$\xi$$ and $$\eta$$ are coordinates of the finite element surface.

Substitute Eqs. (3) and (4) in Eq. (2),

$$T_e = \int_A (g_i(1 - \xi - \eta) + g_j \xi + g_k \eta)(w_i(1 - \xi - \eta) + w_j \xi + w_k \eta)dA$$
\[
\begin{bmatrix}
F_i \\
F_j \\
F_k
\end{bmatrix} = \begin{bmatrix}
\frac{\partial T}{\partial \omega_i} \\
\frac{\partial T}{\partial \omega_j} \\
\frac{\partial T}{\partial \omega_k}
\end{bmatrix} = \int \begin{bmatrix}
(1-\zeta-\eta)^2 & (1-\zeta-\eta)\zeta & (1-\zeta-\eta)\eta \\
(1-\zeta-\eta)\zeta & \zeta^2 & \zeta\eta \\
(1-\zeta-\eta)\eta & \zeta\eta & \eta^2
\end{bmatrix} \, dA \begin{bmatrix}
g_i \\
g_j \\
g_k
\end{bmatrix}
\]

(6)

After calculating the integral, we have the vector of nodal loads that can be given by (see Fig. 10):

\[
\begin{bmatrix}
F_i \\
F_j \\
F_k
\end{bmatrix} = T \begin{bmatrix}
g_i \\
g_j \\
g_k
\end{bmatrix}
\]

where,

\[
T = \frac{A}{12} \begin{bmatrix}
2 & 1 & 1 \\
1 & 2 & 1 \\
1 & 1 & 2
\end{bmatrix}
\]

(6)

Another formulation to the equivalent nodal force vector was proposed by Refs. [9-14], called pseudo-consistent equivalent nodal force vector, where rotational load degrees are calculated considering the external work done by the wind loads expressed by:

\[
T_e = u^T S^T 2A \int_{0}^{1} \int_{0}^{1} \begin{bmatrix}
(1-\zeta-\eta)^2 & (1-\zeta-\eta)\zeta & (1-\zeta-\eta)\eta \\
(1-\zeta-\eta)\zeta & \zeta^2 & \zeta\eta \\
(1-\zeta-\eta)\eta & \zeta\eta & \eta^2
\end{bmatrix} \, d\zeta d\eta \begin{bmatrix}
g_i \\
g_j \\
g_k
\end{bmatrix}
\]

(8)

Again, after calculating the integral, we have the vector of nodal loads that can be given by (see Fig. 11):

\[
\begin{bmatrix}
F_i \\
F_j \\
F_k
\end{bmatrix} = S^T \begin{bmatrix}
g_i \\
g_j \\
g_k
\end{bmatrix}
\]

where,

\[
S = G^{-1} Q
\]

(7)

The \( T \) is given by:

Fig. 9  Interpolating for \( w(x,y) \) and \( g(x,y) \).

Fig. 10  Wind loads converted to nodal equivalent forces.

Fig. 11  Wind loads converted to pseudo-consistent nodal equivalent forces.
The $G^{-1}$ and $Q$ in Eq. (9) are matrices given in Eqs. (11) and (12). The explicit values of Eq. (11) can be obtained from Ref. [9].

$$Q = \begin{bmatrix}
    Q_{0,1} & Q_{0,2} & Q_{0,3} & Q_{0,4} & Q_{0,5} & Q_{0,6} & Q_{0,7} & Q_{0,8} & Q_{0,9} \\
    Q_{1,1} & Q_{1,2} & Q_{1,3} & Q_{1,4} & Q_{1,5} & Q_{1,6} & Q_{1,7} & Q_{1,8} & Q_{1,9} \\
    Q_{2,1} & Q_{2,2} & Q_{2,3} & Q_{2,4} & Q_{2,5} & Q_{2,6} & Q_{2,7} & Q_{2,8} & Q_{2,9} \\
    Q_{3,1} & Q_{3,2} & Q_{3,3} & Q_{3,4} & Q_{3,5} & Q_{3,6} & Q_{3,7} & Q_{3,8} & Q_{3,9} \\
    Q_{4,1} & Q_{4,2} & Q_{4,3} & Q_{4,4} & Q_{4,5} & Q_{4,6} & Q_{4,7} & Q_{4,8} & Q_{4,9} \\
    Q_{5,1} & Q_{5,2} & Q_{5,3} & Q_{5,4} & Q_{5,5} & Q_{5,6} & Q_{5,7} & Q_{5,8} & Q_{5,9} \\
    Q_{6,1} & Q_{6,2} & Q_{6,3} & Q_{6,4} & Q_{6,5} & Q_{6,6} & Q_{6,7} & Q_{6,8} & Q_{6,9} \\
    Q_{7,1} & Q_{7,2} & Q_{7,3} & Q_{7,4} & Q_{7,5} & Q_{7,6} & Q_{7,7} & Q_{7,8} & Q_{7,9} \\
    Q_{8,1} & Q_{8,2} & Q_{8,3} & Q_{8,4} & Q_{8,5} & Q_{8,6} & Q_{8,7} & Q_{8,8} & Q_{8,9} \\
    Q_{9,1} & Q_{9,2} & Q_{9,3} & Q_{9,4} & Q_{9,5} & Q_{9,6} & Q_{9,7} & Q_{9,8} & Q_{9,9} \\
    Q_{10,1} & Q_{10,2} & Q_{10,3} & Q_{10,4} & Q_{10,5} & Q_{10,6} & Q_{10,7} & Q_{10,8} & Q_{10,9}
\end{bmatrix}$$

(9)

$$G^{-1} = \begin{bmatrix}
    1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
   -1/2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
   -1/2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
   18 & 0 & 0 & 27 & -9/2 & -9/2 & 9/2 & -45/2 & -45/2 \\
    9 & -9/2 & 0 & 0 & 0 & 0 & 0 & 0 & -45/2 \\
    9 & 0 & -9/2 & 0 & 0 & 0 & 18 & -45/2 & 0 \\
   -27/2 & 0 & 0 & -27 & 27/2 & 0 & 0 & 27/2 & 27 \\
   -27/2 & 0 & 0 & -27 & 27/2 & -27/2 & 27 & 27 & 0 \\
   -9/2 & 9/2 & 0 & 0 & 0 & 0 & 0 & 27/2 & -27/2 \\
   -9/2 & 0 & 9/2 & 0 & 0 & 0 & -27/2 & 27/2 & 0 
\end{bmatrix}$$

(10)

Fig. 12  Displacement results.
4. Post-Processing

After discussing the mathematical aspects of the wind load model, this section finally shows the post-processing results. In user-friendly environment, the displacements results can be accessed by graphical outputs as viewed in Fig. 12.

The graphical results for the displacement variation on the length of the blade are shown in Fig. 13. The mechanical properties and other input data considered for the model analysis are presented in Fig. 5. The load for the wind pressure in each node in X direction on surface of the eolic blade is 1.000e-02 N.

5. Conclusion

In this paper, a tool for analysis of eolic blades was presented. The main attractive feature of this structural analysis tool is the dll program for the processing stage implemented in C++ language for the FF and DKT finite elements used to discretize the eolic blade. In addition, a user-friendly environment was implemented using OpenGL library to provide the graphical construction for geometry, mesh orientation, and other requirements of the finite element model.

References

[1] Menezes Junior, R. A., Mendonça, A. V., Paiva, J. B., and Mendonça, A. V. 2010. “A User-Friendly Environment for Planar and Space Frames Using the Boundary Element Method.” In Proceedings of Ect 2010 the Seventh International Conference on Engineering Computational Technology, 94-115.

[2] Zienkiewics, O. C. 1977. The Finite Element Method, 2nd ed.. New York: Mcgraw-Hill.

[3] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. 1995. Computer Graphics: Principles and Practice in C. Upper Saddle River: Addison-Wesley.

[4] Shreiner, D., Woo, M., Neder, J., and David, T. 2005. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2, 5th ed.. Upper Saddle River: Addison-Wesley.

[5] Hearn, D. D., Baker, M. P., and Carithers, W. 2010. Computer Graphics with Open GL. Upper Saddle River: Prentice Hall.

[6] Bergman, P. G., and Felippa, C. A. 1985. “A Triangular Membrane Element with Rotational Degrees of Freedom.” Computer Methods in Applied Mechanics and Engineering 50 (1): 25-69.

[7] Batoz, J. L., Bathe, K. J., and Ho, L. W. “A Study of Three-Node Triangular Plate Bending Elements.” International Journal for Numerical Methods in Engineering 15 (12): 1771-812.

[8] Batoz, J. L., and Lardeur, P. 1989. “A Discrete Shear Triangular Nine d.o.f. Element for the Analysis of Thick to Very Thin Plates.” International Journal for Numerical Methods in Engineering 28 (3): 533-60.

[9] Batoz, J. L., and Dhatt, G. S. 1992. Modélisation des Structures par éléments Finis. Vol 3: Coques. Paris: Hermes, 448-55.

[10] Vlasov, V. Z. 1961. Thin-Walled Elastic Beams.
Washington: National Science Foundation.

[11] Taranah, B. S. 1968. “Torsional Behaviour of Open Section Shear Wall Structures.” Ph.D. thesis, University of Southampton.

[12] Onu, G. 1990. “Inclusion of Warping Shear Effect in the Thin-Walled Core Element for Multistory Building.” Computers & Structures 35 (2): 175-82.

[13] Cantin, G., and Clough, R. W. 1968. “A Curved Cylindrical Shell Finite Element.” AIAA Journal 6 (6): 1057-62.

[14] Hrennikof, A., and Tezcan, S. S. 1966. “Analysis of Cylindrical Shells by the Finite Element Method.” Presented at the Symposium of Problems of Interdependence of Design and Construction of Large Span Shells for Industrial and Civic Buildings, Leningrad, Russia.