Object Oriented System as Applied to Jack-Up Analysis

Phuor Ty, Indra S.H. Harahap and Ng Cheng Yee
Universiti Teknologi PETRONAS, Bandar Seri Iskandar, Tronoh 31750, Perak, Malaysia; Ty168.itc@gmail.com, indrasati@petronas.com.my, chengyee.ng@petronas.com.my

Abstract
Jack-up is a type of movable offshore structure that comprises of foundation, leg and hull. Its analysis requires finite element analyses to generate a reliable structural responses under different geometrical configurations and loading types, especially during the conceptual phase. In this phase, the objective is to seek a vigorous and economical structural configuration. This paper presents the application of an object-oriented approach in developing finite element software for jack-up analysis within the MATLAB environment. The objective is to update the cycle of data preparation and analysis. In this finite element tool, with a simple command, any changes in the jack-up element such as material, basic geometrical dimensions and loading types are easily changed. This paper also provides a further illustration of the usefulness and potential of the object oriented perspective. A detail discussion of the structural analyzer sub system with regard to structural optimization and structural design has been additionally specified. Examples are given for application for creation of different configurations of jackup structures.

Keywords: Finite Element Method, Jack-Up, Object-oriented Approach

1. Introduction
Mobile jack-up rigs are utilized generally as a part of the offshore oil and gas industry for installing platforms, maintenance work and drilling and notwithstanding for action for fields of limited life\textsuperscript{1,2}. It commonly comprises of a triangular operation platform and three or four independent retractable lattice legs, each of which is jacked on a saucer-shaped footing known as a “spud-can”. Jack-up rigs are self-installing. Its legs are lifted out of water when it has been towing to the site. On site, their legs are jacked down to the seabed. Once the jack-up has been situated, the spud-can is jacked into the seabed until a satisfactory bearing capacity exists for the hull to be lifted out of the water. For jack-up, static load and variable loads (wind, wave and current load) will be taken into account to design the system. So the concept to design the nonlinear system is very suitable for jack-up analysis. By separately establishing the three components of the model such as hull, leg and spud-can, the design system will provide faster speed in design as well as optimization of structure when there are the changes in the system.

Structural engineer attempts to build up the optimal structural system by considering each conceivable configuration during the early stage of design of structure. For instant material to be utilized, structural system to be utilized (tubular steel vs. conventional steel section), column layout, floor to floor distance, cost comparison and constructability. In this stage, the significant members of analysis are performed and data preparation become very cumbersome if advanced computer method such as Finite Element (FE) program is used\textsuperscript{3-6}. This paper considers the application of an object-oriented approach in developing finite element software for jack-up analysis under the MATLAB environment. The objective is to make straightforward to update the cycle of data preparation and analysis. With a simple command, any changes in the jack-up element, it will be easily achieved the modification in the system such as material properties of the elements, basic geometrical dimensions and loading types. A basic introduction to object oriented approach in this paper is given by\textsuperscript{4,7-11}. The application of the new approach in this work is to eliminate the need to restart overall problem definition if some parts of data change. With this tactic, jack-up
structure is assumed as one entity. Its basic geometrical dimensions (leg, hull, spud-can), type of structural members (beam, column, plate slab), type of materials (steel, concrete, wood) and type of loads are illustrated as attributes to the entity. Hence, by changing those attributes, the change of overall structural configuration and performance can be achieved. To achieve this objective SDT (Structural Dynamic Toolbox), the finite element program develop by INRIA3, is adopted for the finite element engine. This paper consists mainly of 6 sections are: 1. Introduction, 2. Finite element program, 3. Object oriented approach is developed in MATLAB environment12, 3.1 System, 4. Jack-up configuration, 5. Results and discussions and 6. Conclusion.

2. Finite Element Program

The relevant characteristics of SDT program is presented in depth in3. FE methodology of the structural analysis consists of three generic phases:

2.1 Preprocessing phase

The model is able to build up by either manually model generation (nodal coordinate and element connectivity) or automatically model generation. With femesh command, the node and element information of the model could be assigned. The assignment of material properties to each element will be done in this stage. The available keywords for the different purpose of the building of the model can be also obtainable such as copy, revolt, divide and rotate a particular element.

2.2 Program Processing

Boundary condition will be defined in this stage. To solve the linear algebraic equation in order to obtain model displacement, assembly of stiffness matrix and load vector are here performed by fe_mk (or fe_mknl) and fe_load, respectively.

2.3 Program Post Processing

Stresses, weight and graphical presentation of displacements are the secondary quantities. These quantities will be calculated in this phase by using fe_stress command.

3. Object Oriented Approach

The system of this object oriented approach consists of two basic components: 1. Structural analyzer and 2. Data distributor as shown in Figure 1. The interactions and behavior of the data distributor and structural analyzer are presented in the form of class and inheritance diagram and sequence diagram as shown in Figure 2 and Figure 3, respectively.

3.1 System

The relationship between users and the system, the interaction between the sub system and the system and the relationship amongst the sub systems themselves are
demonstrated in Figure 1. The principal controller of the framework system is called “data distribution centre” which is the only place that user interacts with the system. Data distribution centre is functioning to accept data from the user and to control all works of the system.

### 3.1.1 Data Distributor

Figure 2 illustrates Main class diagram for the system. Data distribution centre class is functioning as the execution and flow program controller. The successions of undertaking execution are: 1. To receive and spread out data for each class and 2. To build up the structure. The relation between data distribution centre and other sub systems is by means of interface functions such as receiveData and createStr function. Other functions for their class are private, so for that reason they are not available from other classes. Figure 3 illustrates Data Distributor class that it is the application of data distributor sub system. Relationship between classes and super class is aggregation similar to “consist-of” type relationship. Therefore, the relationship will be presented in term of soil parameters (SOILData), the environmental loads (LOAData), and structure data (STRData).”

### 3.1.2 Structural Analyzer

Figure 4 shows application of the sub system as Structural Analyzer class, its inheritance and interaction with other sub systems. As has been mentioned before, the total system and the high fidelity models will work together. The building of structural object is achieved by in

**Table 1.** Identified method for structural analyzer

<table>
<thead>
<tr>
<th>N</th>
<th>Technique</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>createStr</td>
<td>Create structure using the minimal geometric parameter</td>
</tr>
<tr>
<td>2</td>
<td>changeM</td>
<td>Change material properties</td>
</tr>
<tr>
<td>3</td>
<td>assemblStr</td>
<td>Assemble stiffness matrix and load vector</td>
</tr>
<tr>
<td>4</td>
<td>solveStr</td>
<td>Solve algebraic equation</td>
</tr>
<tr>
<td>5</td>
<td>calcWeight</td>
<td>Calculate the total weight of structure</td>
</tr>
<tr>
<td>6</td>
<td>calDisp</td>
<td>Calculate the displacements of the element structures</td>
</tr>
<tr>
<td>7</td>
<td>calStress</td>
<td>Calculate the stresses of the element structures</td>
</tr>
<tr>
<td>8</td>
<td>checkStress</td>
<td>Check stresses of structure</td>
</tr>
</tbody>
</table>

DATADISTCENTER class through the interface function createStr which interacts with HIGHFIDELITY class with the similar interface function. Table 1 shows the identified method and job function for structural analyzer performance.

### 4. Jack-up Configuration

Jack-up structure is divided into 3 parts are foundation (spud-can), leg and hull structure. Each part is presented below:

#### 4.1 Geometries and Dimension of Spud-can

The geometries of the spud-cans, which vary with the constructors, are cylindrical, conical or in the form of a hyperboloid of revolution. The spud-cans of new jack-ups have an overall conical shape, with a base angle (not including the point) of 120 to 150 degrees (making an angle θ of 15 to 30 degrees with the horizontal). Most spud-cans have a tough, projecting taper point serving essentially to reduce the risk of sliding of the structures in the case of low soil penetration (in a sand or very stiff clay). Spud-can is for the most part hexagonal or octagonal in plan view and its diameter ordinarily extends from 5 m to 20 m³. Figure 5 shows the configuration of spud-can.

As has been mentioned in Finite Element Program and Object Oriented Approach session, with the create Spudcan command and some specific parameters, spudcan could be generated as follow:

Object 1= createSpud-can (a, a’, b, b’, c, c’, d, d’, e, e’, f, f’, g) where a, b, c, d, e, and f are length of spudcan element shown in Figure 5 and a’, b’, c’, d’, e’, and f’ are the thickness coincide with their element. g is the radius of spigot.
4.2 Geometries and Dimension of Leg

Jack-up leg component is usually designed with three cylindrical shape chords or four triangular shape chords, which are connected through bracing members. Mainly, there are three truss-work configurations for the bracing members, namely, X type, K type and diamond shape type. In this paper, K type is selected to present as shown in Figure 6.

Similar to the spud-can, with create Leg command and relevant parameters, jack-up leg could be created:

\[
\text{Object 2} = \text{createLeg} (L, a, a', b, b', c', d')
\]

where L, a, and b are total height leg, length of horizontal leg element, length of vertical leg element, respectively. a', b', c', and d' are the cross section of each member shown in Figure 6.

5. Geometries and Dimension of Hull

Same way as spud-can and leg, jack-up hull is generated as below:

\[
\text{Object 3} = \text{createHull} (a, a', b, b', c, c', d, d', e, f)
\]

where a, b, c and d are length of hull elements (see Figure 7). a', b', c' and d' are the cross section of each element of hull (see Figure 7). e and f are the number of floor and height of each floor (distance from floor to floor), respectively.

6. Outcomes and Discussion

In this paper, two examples of the jack-up creation are given to illustrate the proposed idea. From Table 1, only create Structure function can be revealed with examples. Figure 5, Figure 6 and Figure 7 also show basic geometrical properties of spud-can defining by 13 parameters, leg defining by 7 parameters and hull defining by 10 parameters, respectively. The model is created in MATLAB script file utilizing the commands of SDT.

Example 1: The given commands are created with simple Matlab type statements. Object 1, Object 2, Object 3 and Object are the commands using to create the spud-can, leg, hull and jack-up, respectively. With these commands, jack-up could be created and shown in Figure 8.

\[
\text{Object1} = \text{createSpud-can} (2, 0.1, 2, 0.1, 6, 0.1, 3, 0.1, 1, 0.1, 1.5, 0.1, 1.5)
\]

\[
\text{Object2} = \text{createLeg} (112, 8, 0.05, 6, 0.03, 0.015, 0.02)
\]
Example 2: Similarly, some parameters are changed in commands createSpudcan, createLeg, and createHull. Therefore, the jack-up structure is created once and illustrated in Figure 9.

From this, it has been raised that to generate a specific object, the function needs only the parameters along with that object to start the process. It is possible in object oriented system because both of the information and techniques are encapsulated within the object. Therefore, to generate the spud-can, leg, and hull structure require totally different processes but they are still using the same interface function. Lastly, those parts are all combined to form jack-up structure.

In the forthcoming paper, employment of other methods in Table 1 will be exposed with regard to soil-structure interaction, structural optimization, and structural design. The structural optimization system will be presented to optimize the relevant parameters of jack-up such as \( a, a', b, b', c, c', d, d', e, e', f, f', g \).

7. Conclusion

The paper has presented the design for finite element program within object oriented system by providing complementary command, object wrapper, and the application of jack-up analysis. With this approach, any changes in the jack-up element, it will be easily achieved the modification in the system such as the material properties of the elements, basic geometrical dimensions, and loading types.

This paper also provides a further illustration of the usefulness and potential of the object oriented perspective. The discussion of the structural analyzer sub system with regard to structural optimization and structural design has been additionally specified. Jack-up structure creation are illustrated in Section 5.

7. Acknowledgement

The authors are grateful to Universiti Teknologi PETRONAS for supporting the research study through the Graduate Assistance scheme.

8. References