COMPUTATIONAL PLATFORM DESIGNED TO STREAMLIME THE MODELING, PROCESSING AND THE ANALYSIS OF OFFSHORE RISER SYSTEMS

Celso Kazuyuki Morooka, morooka@dep.fem.unicamp.br
Dustin Michael Brandt, dustin@dep.fem.unicamp.br
Department of Petroleum Engineering, State University in Campinas, Campinas, São Paulo, Brazil

Wander Fernandes Junior, wander@dep.fem.unicamp.br
Department of Petroleum Engineering, State University in Campinas, Campinas, São Paulo, Brazil

Abstract. *The design of modern offshore riser systems in deep and ultra deep water present many challenges both technologically and computationally. There are various types of riser systems further adding to the complexity of the analysis of the system as a whole, which often times requires the use of various numerical solvers. The need to analyze riser behavior under various sets of environmental loads is a critical part of the design process. The combination of various wave and ocean current configurations increases the simulations required exponentially. This process produces a large computational load and is often labor intensive. The current work presents a computational platform designed in C++ to unify the modeling, processing and analysis of complete riser systems using various numerical solvers. This platform was developed to help manage the analysis of a complete offshore petroleum riser system. This work will also introduce a parallelization scheme designed to speed up and simplify analysis along with a 3-D visualization system to review simulation results in real time.*

Keywords: *Offshore Petroleum System, Visualization, Parallelization, Computational platform*

1. INTRODUCTION

The Design of modern offshore petroleum production systems presents many technological and computational challenges. These challenges are greatly amplified in deep water scenarios (1000+ meters). With millions of US dollars of investment at risk, adequate testing and analysis of independent components and the global system is necessary. Beyond hazards such as collision, environmental and human interactions must also be considered. Environmental variables include waves, ocean currents and wind along with the consideration of extreme situations such as hurricanes and tsunamis. Human interactions could include fishing activities, merchant shipping, sinking ships, dropped objects and a wide range of other scenarios. Another very important concern is material fatigue of the components. The current software platform is being developed to aid the design engineer in taking into account as many of these considerations as possible using a wide variety of numerical tools. This paper will give a brief summary of typical components of an offshore petroleum production system with a focus on production risers.

The current platform's purpose is to unify the modeling, processing and the analysis of a complete riser system. The platform includes a visualization system in order to view the riser system at a component or global level. Another important part is the use of modular programming concepts to enable the use of various numerical solvers and tools independently. In order to aid in the processing of the massive amount of data, simple parallelization schemes coupled with a computer cluster can also be used within the platform.

There are currently many active projects in Brazil and the world concerning visualization of offshore petroleum systems. One Brazilian project provides a 3D immersive tool designed for offshore petroleum production systems (Nishimoto, 2002). Bernardes et. al. (2004; 2006) have also worked on the 3D visualization of riser behavior and developed a program that offers data analyses tools and visualization of risers. Other initiatives also can be found in the literature (Morooka, 2006). Commercial software is also available with the similar goal. The focus of this paper is on the integration of many different and independent tools under one application.

2. SUMMARY OF MAIN COMPONENTS OF AN OFFSHORE PETROLEUM PRODUCTION SYSTEM

A deepwater offshore petroleum system is made up of many components. A basic offshore petroleum system is generally comprised of a floating petroleum facility, sub-sea wells, manifolds, risers and possibly export and transportation pipelines. There exists a multitude of configurations.

A riser is a tube or set of tubes that transports fluids and tools from the sub-sea manifolds or wells to and from the offshore petroleum facility. In one system there can be anywhere from one to hundreds of risers. It is a critical component of the system and it is exposed to many environmental and man-made hazards. It can be exposed to ocean currents, waves, forces induced by internal fluid flow, floating facility motions and vortex induced vibrations (Morooka et al, 2003). Vortex induced vibrations (VIV) is caused by alternating vortices that are emitted from an immersed body when fluid passes by it. These vortices induce an oscillating force perpendicular to the direction of the flow which could cause material fatigue (Morooka et al, 2005). It is critical that a component be designed to withstand the worst case scenario. Many times the worst case is not known and must be determined by running a myriad of scenarios varying environmental loads. This idea will be covered later.

The offshore petroleum facility serves many purposes. One of the primary purposes is to process the reservoir fluids often including a mixture of water, oil, solids and gas. These facility can also be capable of well-testing and interventions. If secondary recovery methods are used such as water or gas injection, the facility is responsible for compressing and pumping fluids into the formation. There are various types of platforms used in deepwater, only three of the most common will be discussed. A Tensioned Leg Platform (TLP) consists of a platform fixed to a template which is secured to the sea floor by piles. Tensioned pipes, called tendons, connecting the platform to the template greatly reduce the heave induced by waves and keep the platform in place. Another type of offshore petroleum facility is the Semi-submersible Platform. This type of platform is usually anchored by a set of cables to the sea floor. The platform is designed to keep the wave induced heave to a minimum. The Floating Production Storage and Offloading (FPSO) facility is basically a converted oil tanker which is equipped with petroleum processing equipment and equipment used to connect the hull to the riser system. It is usually anchored by a group of cables. This type of platform is sensitive to vertical wave induced motion but has many benefits including a reduction of cost. In many riser configurations, the risers are connected to the platform deck through a turret or moon pool.

Sub-sea pipelines are used to transport fluids and tools to wells, manifolds, templates or even to processing plants on or offshore. They are exposed to many hazards including trawl impacts, corrosion and general impacts (Bai, 2001). Free spanning sections, or when the pipeline is not directly supported by the soil, have shown to be a concerned based on VIV caused fatigue (Morooka et al, 2006).

3. PRODUCTION RISERS

The software platform has been designed for the analysis of production risers. As stated previously, modern production risers come in many configurations. Every type of riser configuration has its own particular advantages, disadvantages and design complexities. Steel Catenary Risers (SCRs) are comprised of steel tubing that forms a catenary shape in the water and runs along the seafloor for a distance in order to absorb heave motion. This configuration has the advantage of cost and simplicity. Fatigue induced by VIV and heave is a serious concern in the design of this system. Another alternative is flexible risers which are comprised of layers of rubber and steel that create a flexible, fatigue resistant riser. This design suffers from high costs and lack of adequate suppliers in the global market. Top Tensioned Risers (TTR) consists of steel tubing tensioned by the platform or an undersea buoy forming a vertical tube. The bottom is secured using a piled template. This configuration when tensioned by an undersea buoy modified by bundling many tubes together is called a Compliant Riser Tower. Mourelle (1995) has presented work on flexible riser analyses among many others.

The analysis of riser systems under environmental loads needs to consider a wide range of scenarios that could possibly occur in the field. Many riser systems such as SCRs and flexible risers are often highly affected by the direction of the environmental loads. Another extremely important consideration is the motion of the platform because many riser configurations are sensitive to heave motion. Each riser systems can have a multitude of numerical solvers, both commercial and academic, which can be used to predict the behavior of the system under environmental loads. The current work seeks to unify the solvers using a common application.

Currently, the application accommodates for the design and analyses of rigid risers and risers in catenary in various configurations with or without the effects of VIV. Static and dynamic results can be analyzed. Current rigid riser configurations include free-span pipeline, TTR, hanging and Self-Standing Hybrid Riser. Further information concerning riser dynamics that are featured in this project can be found in Ferrari (1998), Morooka et. al. (2003) and Morooka et. al. (2005). These numerical implementations are packaged into solvers. In time other numerical solvers and capabilities will be accommodated.

4. COMPUTATIONAL PLATFORM OVERVIEW

4.1 Modular Programming Structure

The current computational platform is being designed with production riser systems in mind but could easily be expanded to other components. In many cases the analysis of complete riser systems can be extremely time-consuming. The analysis of individual risers is done separately often requiring manipulation of text files. The analysis of the results is then performed utilizing spreadsheets. This paper will present an approach that greatly facilitates the completion of global riser system analysis along with aiding the design engineer with powerful visualization tools.

Using modular programming techniques, capabilities can be expanded or updated without the need to modify the system as a whole. To do this we implement modular programming concepts. In modular programming, the various software components (modules) communicate using well-defined communication interfaces. These interfaces act as bridges connecting the two components (Lieberherr et al, 2001). The advantage is that components using the same interface can be interchanged at will. In our application the solvers, capabilities and elements can be thought of as components. This technique relies heavily on runtime libraries, noted in this work as plug-ins, and abstract class definitions. Figure 1 shows a diagram of the basic software elements used and their particular scopes. When the program is loaded, there is a search for all runtime libraries. When one is found, new capabilities are identified according to the interfaces they use. Each type of interface identifies a type of capability. Currently there are three types of interfaces. The first defines a visual element; this could be a petroleum platform, sub-sea component, sea surface, sea floor, remote operated vehicles (ROV) or any visual object that does not use a numerical solver. The second interface defines an element with a numerical solver attached; it includes the same functionality of the visual element but contains many functions to configure the solver and manage results. The third interface defines a data analysis or result presentation capability. This could include fast Fourier transforms, spectral analysis, data analysis or graphing capability.

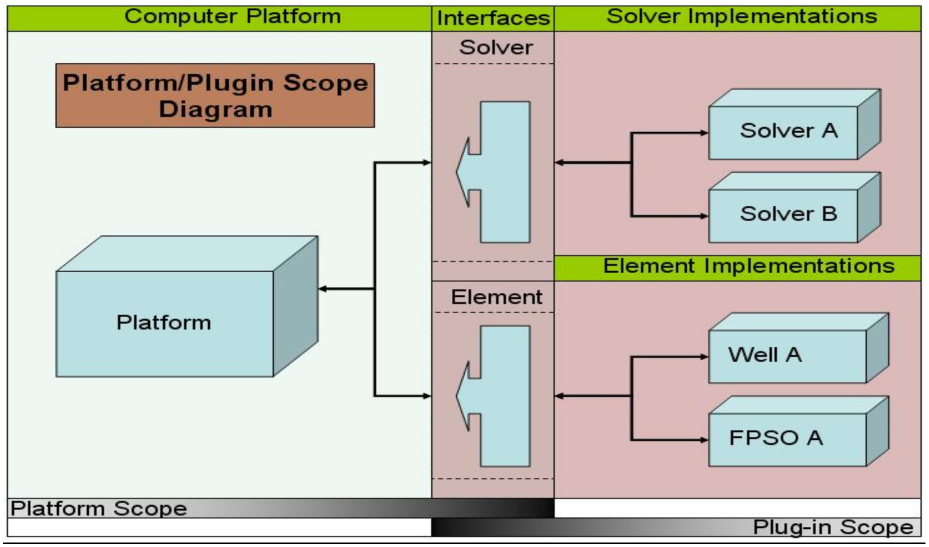


Figure 1. Basic diagram of the plug-in mechanism with scopes

The interfaces are carefully developed to include a complete list of functions that can be used to interact with the main module (Platform). The interfaces contain all necessary functions needed to aid in the modeling, displaying and running of the solver. The actual solver specific code is contained inside the implementation of a class that inherits the interface capabilities. The numerical solver's plug-in interface class will implement exactly how basic tasks will be performed but the interface, or communication protocol, is generic. These tasks include opening and processing configuration files, configuration windows, saving files and processing results. The same abstractions can be used for the other types of interfaces.

4.2. Global Numerical Solver Capabilities

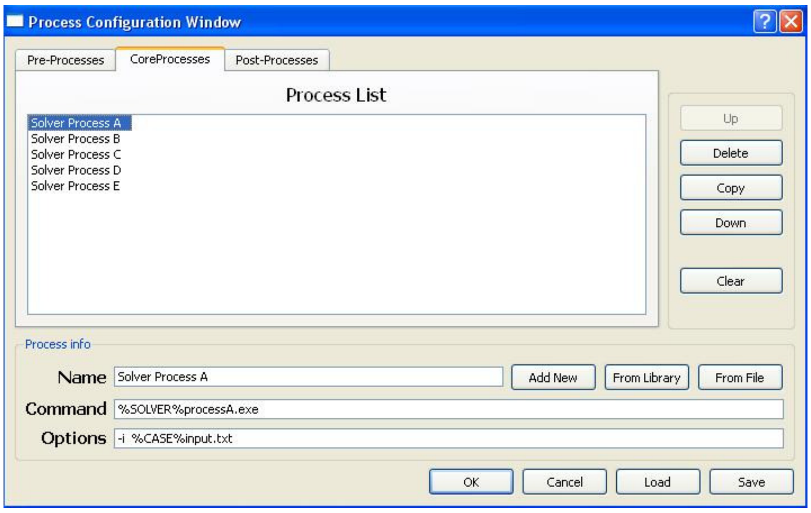
The previous section has defined the method in which numeric solvers can be divided and contained in runtime libraries through the application of common interfaces. The next problem is how to complete a global analysis, that is, how can a riser systems be tested against many different sets of environmental loads. A system could be modeled then the environmental loads could be manually changed for each riser. This process is practical if the system is comprised of a few risers and a limited set of environmental load combinations. If the system has hundreds of pipelines, risers and thousands of environmental load combinations, this process quickly becomes impractical. This creates a need to automate the process. The platform currently has limited capabilities to specify the environmental conditions globally and automatically apply them to the risers depending on solver dependent restrictions.

The platform allows the generation of a parametric study, which is a group of configurations (cases) with incrementally varying variables. This allows the engineer to define a range of wave heights and periods, for example, and the platform will generate configurations equally covering the given ranges of the variables. The set can then be processed and analyzed together. This allows for a quick method to test for extremes or analyze the system's sensitivity to changes in properties or loads. These studies can create three dimensional graphs. This clearly aids the analysis of results and pursuit of system optimization.

A time consuming part of any global analysis of any system is documentation. The current platform is being designed to give a quick and highly customized tool for creating template documentations for cases. This allows a uniform documentation strategy to be applied to a wide variety of studies. The documentation system can even be applied to whole entire scenarios or a parametric study. This includes an automatic graph generator that can be applied to different sets of solver results.

4.3. Processing Capabilities

Each numerical solver implementation should have a scheme to execute the solver correctly. An additional mechanism is built-in to allow custom modification to this order referred to as an execution stack. Each solver contains a list of processes to run in order to obtain results. This list can then be changed to include or exclude processes. The order in which the processes are run can be changed. The modification of processes allows the inclusion of varying versions of a common solver (one that uses the same inputs and outputs) or inclusion of other processes to the execution stack. This list of processes can be saved and loaded. The processes are divided into three sections; pre-processes, core processes and post-processes. Processes placed in the pre-processes stack will be ran before the numerical solver. These could modify input files or any number of functions. Core processes represent the basic processes comprising the solver. Post-processes are run after the solver which could be data filters or result analysis tools. Figure 3 shows a diagram of how these stacks work while Fig. 2 shows the platform's execution stack user interface.


Process Configuration Window

Pre-Processes | CoreProcesses | Post-Processes

Process List

Solver Process A
Solver Process B
Solver Process C
Solver Process D
Solver Process E

Process info

Name: Solver Process A

Command: %SOLVER%processA.exe

Options: -i %CASE%input.txt

Figure 2. Process configuration window

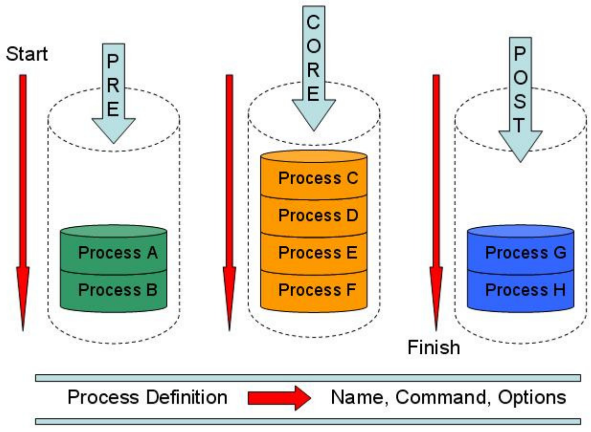


Figure 3. Process queue diagram

Cases can be analyzed in three different manners; cluster, local computer, and remote computer. This allows a great deal of flexibility. If the local computer is selected, the engineer can choose how many cases to run at any given time on the computer. When one case is completed, another case will automatically be started until all cases are finished. The case can also be run on a remote computer. The engineer can send the case data to a remote computer that has the capabilities (executables) that are needed to successfully complete the task. This harnesses unused clock cycles of other computers to accelerate the processing of the global system. The third alternative is the computer cluster. This will be explained more in depth later. It is basically a large group of computers that work together to solve complex computationally intense problems such as riser systems under environmental loads. An engineer can send one or more cases to the cluster. The cluster keeps its own queue and processes cases in the order in which they arrive.

These tools are meant to manage and accelerate the processing of cases and make the process as efficient as possible. The ability to change the execution stack provides the designer with a quick way to highly customize the numerical solver. The ability to have cases processed in more then one way allows a great amount of flexibility and freedom.

4.4. Cluster

A cluster can be defined as "a collection of closely related computer systems, providing a common service or running a common parallel application" (Lucke, 2001). Clusters are powerful tools and can greatly improve efficiency when processing riser system data. The cluster is comprised of a group of highly connected computers or nodes. For a traditional cluster, one or more master nodes control the rest of the computers in the clusters. The current cluster implementation is limited in the fact it is based on a one master paradigm.

The master node is responsible for managing and monitoring the activities of the other nodes commonly referred to as slave nodes. The master is the only computer that can receive cases, send progress and case status or transmit results to the user. The master also keeps a log of currently running cases, which cases are waiting and where all case results are located. A slave node is responsible for accepting cases from the master, processing them, transmitting status and saving the results for later retrieval.

The server/client software was developed to accept cases and new numerical solver executables. Figure 4 illustrates a basic flowchart of communication between the platform running on the engineer's desk, the master node and the slave node. All communication is done utilizing the tc/ip protocol.

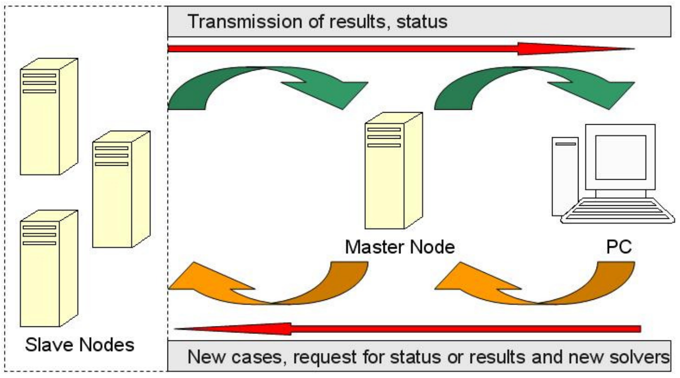

Transmission of results, status
Master Node
PC
Slave Nodes
New cases, request for status or results and new solvers

Figure 4. Flowchart of communication with the cluster

Parallelization is an application specific task. It involves breaking a large program into independent computational blocks. These blocks can then be processed on individual processors and the results of the calculations can be passed to other processors that need them. This is extremely useful when dealing with large matrixes. Due to the fact that this operation is solver specific and requires the actual code of the solver, the platform cannot automatically perform the parallelization. Instead parallelization is done in the scope of the global system. All cases are sent to the cluster and are distributed among the slave nodes.

4.5. 3-D Visualization

An important part of the riser design process is the visualization of the components. This allows the engineer to inspect the system in three dimensions. To do this a graphic engine was designed. This engine utilizes OpenGL and is capable of using the OpenGL Shading Language (GLSL). This engine is responsible for managing and drawing all objects in the system. It is also designed to allow the engineer to freely move around the system in the three-dimensional environment.

Three dimensional models can be created in a wide variety of modeling programs and imported as a visual element into the platform. Elements can then be collected into a library allowing the engineer a wide amount of flexibility. Additional elements can be added simply by putting the library into the platform path. The visual element, as discussed in section 4.1, contains many common functions in which to communicate with the platform. These functions allow the simple three-dimensional mesh to interact with the global system. This could include induced motion due to waves on a platform or any number of things. A collision system is also integrated into the functions of the visual element. This could allow for detection of collisions or areas of high risk of collision.

The engine allows the use of multiple textures either through texture images or shader implementations. Shaders are small subprograms created using GLSL to specify visualization effects. These functions can be used on the visual element level to apply complex visual effects to elements including smoke, fire or even mist. On the system level, this ability is used to produce shadows and fog.

Figure 5 shows an image of a simple top tensioned riser and a visual element (shark). The sea bottom utilizes a set of texture maps while the sea surface is generated using a Phillips spectrum and is drawn using a custom shader. The reader can see the global effects including shadow and fog.

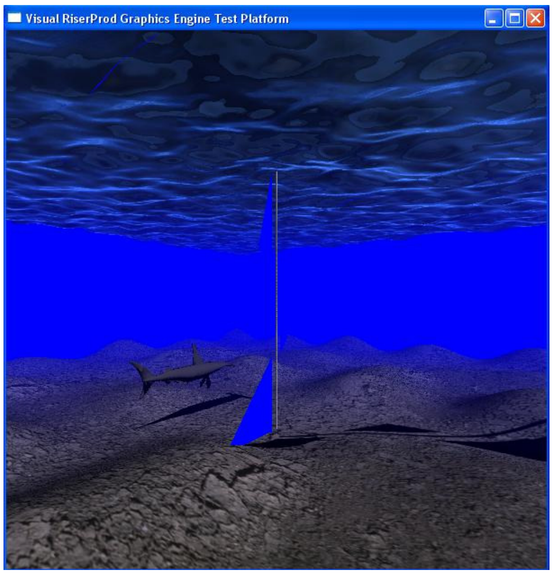


Figure 5. 3-D Graphics Engine

5. CONCLUSION

The current platform represents an advance in the management and modeling of a global riser system but requires a vast amount of work. The project is still a work in progress. There are currently only a few solver modules that have been developed. In order to analyze a complete system, many more solvers for various other components must be added. The same problem exists with visual elements. In addition, more result analysis tools are needed, along with many improvements to the currently existing ones.

Operational risk analysis is a very important part of the design process. The current platform does not support any form of standard risk analysis. Future work is needed to add risk analysis as a tool that can be used in the overall design. This could include the implementation of consequence modeling or fault tree analysis to name a few.

The current collision system in the visualization process is very primitive. More work is required to create quick algorithms to check for collision, and a procedure to estimate the severity of the impact. Another alternative is to create safety zones around components and check for collision of the safety zones. This creates a quick way to find trouble spots in the design.

In many cases, results using different solvers modeling the same riser are performed for numerical verification. This process involves the modeling of the riser using each solver separately. This could be a time consuming and error prone process. Future work could be invested in creating mechanisms that can convert generic riser and environmental properties into acceptable input data for various solvers.

ACKNOWLEDGEMENTS

The authors would like to thank CNPq and FINEP (CTPetro), Petrobras and Capes for supporting the present development. The authors would also like to thank the team at the *Tanque de Provas Numéricos* (Naval Engineering Department, Polytechnic School, University of São Paulo) for their time and guidance.

REFERENCES

Bai, Y., “Pipelines and Risers” Elsevier Ocean Engineering Book Series Vol.3 Elsevier, Oxford UK 2001.

Bernardes Jr., J. L., Silveira, L. M .Y., Martins, C. A., “Riserview: an Open Source, Multiplatform Post-Processing Tool for the Visualization and Analysis of Riser Dynamics”, 25th International Conference on Offshore Mechanics and Arctic Engineering (OMAE), ASME, Hamburg, 2006.

Bernardes Jr., J. L., Martins, C. A., “Development of An Environment for 3D Visualization of Riser Dynamics”, VII Symposium on Virtual Reality, SVR 2004, pg 88-99, São Paulo, 2004.

Ferrari Jr, J.A., “Hydrodyanmic Loading and Response of Offshore Risers”, Published Doctoral Dissertation, Imperial College of Science, London, 1998.

Lieberherr K., Ovlinger, J., Mezini, M., and Lorenz, D., "Modular Programming with Aspectual Collaborations”, College of Computer Science, Northeastern University , Tech report NUCCS-2001-04, March 2001.

Lucke, R., “Building Clustered Linux Systems”, Prentice Hall, NJ 2005.

Morooka, C.K. ; Martins, F.P.; Kubota H.Y. ; Ferrari Jr., J.A. ; Ribeiro E.J.B ., “A Study on Inline and Transverse Dynamic Behavior of a Vertical Riser in Time Domain”, 17th International Congress of Mechanical Engineering (COBEM), São Paulo, 2003.

Morooka, C.K., Coelho, F.M., Ribeiro, E.J.B., Ferrari Jr., J.A., Franciss, R., “Dynamic Behavior of a Vertical Riser and Service Life Reduction” 24th International Conference on Offshore Mechanics and Arctic Engineering (OMAE), Halkidiki (Greece), 2005.

Morooka, C.K.;Brandt, D.M.;Fernandes Jr., W.; Coelho,F.M., "A User-friendly Graphical Tool to Help Visualize the Analysis of Production Risers", 27th Iberian Latin-American Congress on Computational Methods in Engineering, CILAMCE, Belém, 2006.

Morooka, C.K., Idehara, A.Y., Matt, C.G., “Behavior of a Pipeline with Free Span under Steady Current”, 27th Iberian Latin-American Congress on Computational Methods in Engineering, CILAMCE, Belém, 2006.

Mourelle, M. M., Gonzalez, E. C. and Jacob, B. P., “ANFLEX - Computational System for Flexible and Rigid Riser Analysis”, 9th International Symposium on Offshore Engineering, Brazil Offshore 95, pp. 441-457, Rio de Janeiro, September 1995.

Nishimoto, K., Russo Neto, A.A., Taniguchi, D., “TPNView - Visualização dos Resultados da Simulação em Realidade Virtual”, 19º Congresso Nacional de Transporte Marítimos, Construçao Naval e Offshore, Rio de Janeiro, 2002.

RESPONSIBILITY NOTICE