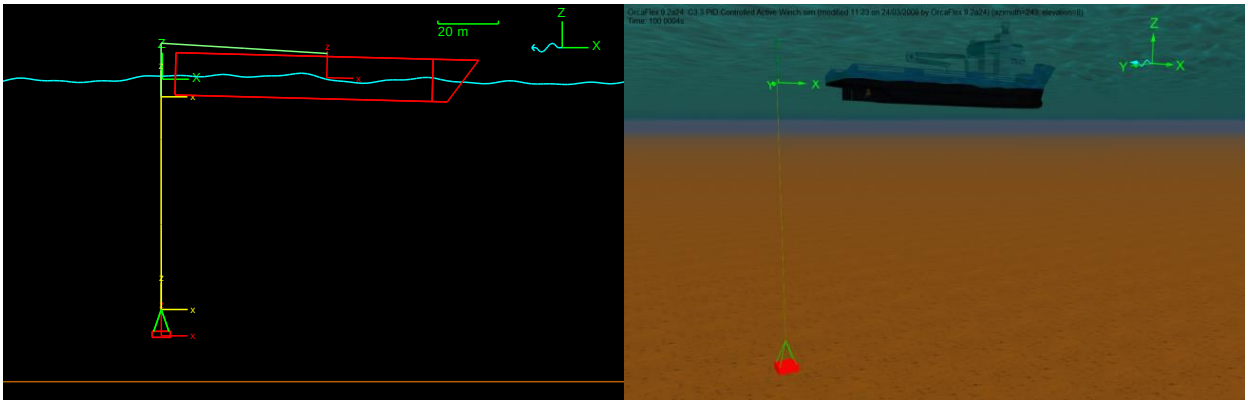
F03 PID controlled active winch



This package-lowering example uses a winch to model an active heave compensation device.

Introduction

An external function written in Python is used to control the length of a winch, which is holding an object at midwater. An OrcaFlex line object is used to model the majority of the work wire; using a line means that the wire physical and hydrodynamic properties are accounted for. The winch between the vessel and work wire is controlled with a PID controller via the OrcaFlex programming interface, OrcFxAPI.

Python is used to write the external code in this example; however C++ or Delphi could have been used instead.

Building the model

The package being lowered is a simple 6D lumped buoy, with links used as rigging. An OrcaFlex line is used between the links and the lowering winch. Links and winches have no mass or hydrodynamic forces, so to use a winch for the entire work wire would mean that we miss the forces on the wire in the water.

The vessel uses displacement RAOs, and so moves in response to the wave loading imposed on the model. Our aim is to use the winch to compensate for the vessel motion and maintain the package at a constant depth, i.e. active heave compensation.

To control the winch, a PID controller (proportional – integral – derivative controller) was chosen. This is a generic control-loop feedback mechanism, which is widely used in control systems. The PID control algorithm has been coded in Python, a dynamic scripting language, and interfaces with the model via the OrcFxAPI. The Python code, contained in the file *PIDController.py*, can be viewed with any text editor, however we recommend Notepad++ as it presents files in an easily readable format.

Full details of the Python Interface are not included here; instead, the user is directed to the help file for the OrcaFlex API, which can be found on the OrcaFlex documentation page of the Orcina website.

In the model, OrcaFlex is made aware of the Python code on the *variable data form*, under the *external functions* heading. Click on *PIDWinchControl* in the model browser to see this. Once the *PIDController.py* file has been identified as the *file name*, the name of the function to be used to control the winch is then selected from a drop-down list in the *function name* box (OrcaFlex queries the contents of the Python file automatically, and lists the available functions).

We also need to define an *initial value* (i.e. the value passed to the external function the first time it is called), and how regularly the external function should be called. In this case, we are using the implicit integration scheme, so the function will be called on each iteration of each timestep regardless of what is entered in the *time step* box.

Once the external function is defined, we can use it to control the winch. This is done on the winch data form. The name of the external function, *PIDWinchControl*, is used as a variable data item for the winch payout rate throughout the simulation. On the *tags* page, additional information that needs to be passed to the external function can be defined.

In this example, we use tags to tell the function the following information:

- the name of the object to be controlled (*Template*)
- the name of the object result to be controlled (*Z*)
- the control start time, which is set at 20s so that the motion of the package with and without heave compensation can be compared
- the target depth at which the package should be held (85m), so Z = -85
- a min and max payout rate for the winch (optional)
- PID control parameters, k0, kP, kI and kD

Results

Opening the simulation file opens up the default workspace file, which displays some important results. The variation in winch tension is shown (top right) to illustrate the loads experienced during the motion. The template weight in water is 19.75 te, and so a mean tension of approx. 200 kN is reasonable. There is a shock load on the winch when it begins to control the package depth at 20s. This is because of a step change in acceleration.

Time histories of winch *length* and template *Z* position are shown below the tension plot; the latter clearly showing the effect of the PID controller holding the template close to the required depth. Run the replay to see this in action in the wireframe model view. Vessel motion near the stern is also plotted (top left), and has a range of up to ±5 m. The corresponding range in template depth, once the heave compensation is operating, is less than ± 0.5 m. The effectiveness of the compensation depends on the PID controller parameters; the maximum capability of the real equipment should be considered when the model is built i.e. the parameters must be 'tuned' to reflect the performance of the real system. If this is not done then it is possible to achieve better or even worse performance in a simulation than may be realistic.

Note: Further external function examples in both Python and C++ and available for download from the resources section of the Orcina website: http://www.orcina.com.