Dynamic analysis: Time domain solution

Equation of motion

The equation of motion which OrcaFlex solves in the time domain is

$$M(p,a) + C(p,v) + K(p) = F(p,v,t) \tag{1}$$

where

$M(p,a)$ is the system inertia load

$C(p,v)$ is the system damping load

$K(p)$ is the system stiffness load

$F(p,v,t)$ is the external load

p , v and a are the position, velocity and acceleration vectors respectively

t is the simulation time.

OrcaFlex implements two complementary time domain dynamic integration schemes, **explicit** and **implicit**. Both schemes re-compute the system geometry at every time step and so the simulation takes full account of all geometric nonlinearities, including the spatial variation of both wave loads and contact loads.

Explicit integration scheme

The explicit scheme is semi-implicit Euler with a constant time step. At the start of the time simulation, the initial positions and orientations of all objects in the model, including all nodes in all lines, are known from the static analysis. The forces and moments acting on each free body and node are then calculated. Forces and moments considered include:

- weight
- buoyancy
- hydrodynamic and aerodynamic drag
- hydrodynamic added mass effects, calculated using the usual extended form of Morison's equation with user-defined coefficients
- tension and shear
- bending and torque
- seabed reaction and friction
- contact forces with other objects
- forces applied by links and winches.

The equation of motion (Newton's law) is then formed for each free body and each line node

$$M(p)\,a = F(p,v,t) - C(p,v) - K(p) \tag{2}$$

This is not the system-wide equation of motion, but a *local* equation of motion for each free body and each line node: solving each of these equations of motion merely requires the inversion of a 3×3 or 6×6 mass matrix.

The local equation of motion is solved for the acceleration vector at the beginning of each time step, for each free body and each line node, and then integrated using semi-implicit

Euler integration. Let us denote the position, velocity and acceleration at time step t by p_t , v_t and a_t respectively. Then for a time step dt , the values at the end of the time step, at time $t + dt$, are given by

$$\begin{aligned} v_{t+dt} &= v_t + dt\, a_t \\ p_{t+dt} &= p_t + dt\, v_{t+dt} \end{aligned} \tag{3}$$

At the end of each time step, the positions and orientations of all nodes and free bodies are again known and the process is repeated.

The time step required for stable integration is typically very short and OrcaFlex gives guidance on an appropriate time step. Hydrodynamic and aerodynamic forces typically change little over such a short time interval, and are time consuming to compute. To save computing time, these loads are updated only over a longer outer time step. Both time steps are user-specified and may be set equal for critical cases.

Of the various objects available in OrcaFlex, lines are the most computationally demanding. For most models that include lines, the length of time required for dynamic analysis is approximately proportional to the total number of nodes used multiplied by the total number of inner time steps in the whole simulation. If the time step is maintained at the recommended value and nodes are distributed uniformly along the lines, then the run time is approximately proportional to the square of the number of nodes.

Finite element models may contain spurious high frequency response, a feature inherent in the finite element method. The line target damping data can be used to damp out this high frequency noise.

Implicit integration scheme

For implicit integration OrcaFlex uses the *generalised-α* integration scheme as described by Chung and Hulbert. The forces, moments, damping, mass etc. are calculated in the same way as for the explicit scheme. Then the *system* equation of motion is solved at the *end* of the time step.

Because p , v and a are unknown at the end of the time step an iterative solution method is required. Consequently each implicit time step consumes significantly more computation time than an explicit time step. However, the implicit scheme is typically stable for much longer time steps than the explicit scheme and often this means that the implicit scheme is faster.

Numerical damping of the integration scheme

Finite element models may contain spurious high frequency response, a feature inherent in the finite element method. The generalised-α integration scheme has controllable numerical damping which is desirable since it removes this artificial, non-physical high frequency response. This numerical damping also leads to much more stable convergence and hence allows for longer time steps and much faster simulations.

Any integration scheme which includes numerical damping of high frequency response must be careful to avoid damping response at lower frequencies. The generalised-α integration scheme is designed to minimise the low frequency damping.

The numerical damping is determined by specifying the level of high frequency dissipation, ρ_∞ . OrcaFlex uses a built-in value of 0.4 which has been chosen to give fast simulation run times without compromising accuracy.

Implicit integration and imposed motion

As described, the implicit integration scheme works by solving for the p, v and a values of the free model objects at the end of each time-step. To do this, the integration scheme imposes a functional relationship between the p, v and a values for each degree of freedom, which represents an assumption (constant rate of change of acceleration) about how the position varies over the discrete time-step. This means that the solver is only free to guess the value of one component of the triplet (the other two being enforced by the functional relationship). In contrast, imposed degrees of freedom set all three of the p, v and a values, either as functions of time, or from an external function controller. In this case, the p, v and a values should be interrelated in the standard fashion: v should be the time-derivative, $\dot{p}$, of p and a should be the time-derivative, $\ddot{p}$, of v. Whilst these v and a derivatives may be instantaneously consistent with p at any given time, they will only match the v and a values associated with an equivalent free coordinate (i.e. one which is always at the same position, p) in the limit of an infinitessimally small time-step. At larger time-steps, the solver's assumption about the evolution of the dynamical system (within a time-step) will break down, and the resultant v and a values will differ from the analytic $\dot{p}$ and $\ddot{p}$ values.

The consequences of this discrepancy depend upon the details of the model. Issues can arise in cases where the relative motion between an imposed model object and a free one attracts both stiffness and damping contributions. Consider a model in which a free buoy is attached to an imposed motion vessel by an extremely stiff spring. As the vessel moves, the buoy will track the vessel motion exactly (assuming that the stiffness dominates the buoy inertia). However, the inconsistency between the imposed motion and the free motion will mean a discrepancy between the velocities of the buoy and the vessel, and this discrepancy will increase at larger time-steps. If the spring also attracts a damping contribution, this will lead to artificial noise in the reported spring tension. A typical such case is a Rayleigh damped line attached to an imposed motion vessel, for which spikes in effective tension can sometimes be seen in the first segment of the line.

To overcome this problem, it is possible to force relevant model objects (vessels, constraints and turbines) to **compute imposed motion consistent with the solver**. If this option is selected, then the velocity and acceleration values naturally associated with the imposed motion are disregarded. Instead, OrcaFlex sets v and a from p using the same functional constraint that it assumes when solving for a free coordinate. The result is that the solver will be able to guess p, v and a values that exactly coincide with the imposed motion, and the noise-inducing mechanism described above will no longer apply.

There are pros and cons to computing imposed motion consistent with the solver. The main disadvantage is that the v and a values will not be exactly as one might expect from differentiating the position analytically. In particular, the v and a values returned from any external function imposed motion controllers will be completely ignored. This behaviour can also be advantageous: it may not be easy (or possible) for the external function to compute v and a; having to compute p only provides a significant simplification.

Note: *The **compute imposed motion consistent with the solver** options are only available for implicit integration. Any externally calculated motion controller that is also used with explicit integration will still have to provide the full p, v, a triplet of values.*

In Orcaflex 11.2 and earlier, the option to compute imposed motion consistent with the solver was not available. However, OrcaFlex may still have adopted this policy behind the scenes, e.g. in models with Rayleigh damping on a line or when line feeding from a moving vessel. For backwards compatibility, loading an old OrcaFlex data file will automatically set the relevant data according to what older versions of OrcaFlex would have done internally.

Allowing the user to control this process is more flexible, and avoids OrcaFlex having to make a broad policy decision that may not be appropriate for every model.