Sixth International Symposium on Marine Propulsors
smp'19, Rome, Italy, May 2019

# Automated self-propulsion point search algorithm for ship performance CFD simulations

**Mika Nuutinen**[1]

[1]ABB Marine & Ports, Helsinki, Finland

## ABSTRACT

Determination of the ship self-propulsion point (SPP) is typically achieved in CFD by simulating two or more points around the actual SPP by varying propeller RPM or torque. Optimally, these points lie very close to and on both sides of the SPP so that thrust values larger and smaller than corresponding hull resistance are obtained. Then, approximating the SPP is a matter of interpolation. Hence, two or more considerably long CFD simulations are required for these interpolation points.

An automated self-propulsion point search algorithm is developed to minimize computational time and manual labor. With this algorithm, SPP is found directly with a single simulation and no interpolation is required. Only one input is required from the user. Namely, the relaxation time within which the algorithm tries to match thrust and resistance. The algorithm is derived from the non-dimensional thrust curve, $K_T$, scaling. To avoid confusion with actuator disc self-propulsion simulations, it is emphasized that the algorithm does not utilize a predefined $K_T$ curve. Instead it continually linearizes the relation between propeller RPM and thrust. In case geometric details, e.g., superstructure, appendages, etc. are omitted from the simulation, their estimated extra resistance may be directly added to the simulated hull resistance to produce a new target thrust for the algorithm. Separate versions for fixed pitch propellers (FPP) and contra rotating propellers (CRP) are derived. The CRP version assumes fixed (but adjustable) RPM ratio. The CRP version has application also in case of three propeller ships, with a fixed (but adjustable) side/center RPM ratio. The conventional propeller version works equally well for controllable pitch propellers (CPP). Generally, it can be stated that the automated SPP search algorithm can handle any type of propulsor with smooth and continuous thrust curve $K_T$.

### Keywords

Self-propulsion, CRP, search algorithm, RPM control, CFD

## 1 INTRODUCTION

With increasing computer speeds and advanced CFD software, the performance simulations of full scale ships are becoming a standard tool in ship and propulsor design. Confidence on CFD methods is gained for instance from the research by Ponkratov & Zegos(2015) who validated self-propulsion CFD simulations against full scale sea trial measurements. Self-propulsion CFD simulations with discretized propeller geometry are still very resource intensive computations, typically 10-30 million computational cells are required for good spatial resolution. Moreover, the modeled propeller rotation requires a relatively short timestep (corresponding to 1-2° propeller revolution), typically an order of magnitude shorter than what is required to simulate bare hull resistance (corresponding to Courant number of order 1 for streamwise resolution close to hull surface). The computational burden may be alleviated by utilizing actuator disc model instead of full propeller geometry, as in Pacuraru et al(2011) or by coupled BEM/RANS approach, as in Gaggero et al(2017). However, much of the propeller physics, valuable to the designer, is lost with actuator disc and BEM/RANS approaches.

There are basically two routes to accurately obtain the actual SPP. The first option is to perform two or more long, fixed RPM simulations, where hull resistance and thrust saturate to constant levels on average, and then to interpolate/extrapolate the SPP. The second option is to employ an RPM controller that actively adjusts RPM to drive thrust towards hull resistance. ITTC(2014a) recommends the Proportional-Integral (PI) controller for this purpose. The PI controller has been utilized in self-propulsion simulations by Carrica et al(2010), who chose to control RPM to attain a certain ship speed instead of the conventional method of fixing speed and adjusting RPM to match thrust to resistance. While the PI controller is concise and easily implemented, it comes with two model coefficients that are not known *a priori*. There are systematic ways to determine these coefficients, but they are not straightforward, and appropriate values vary depending on ship/propeller particulars and speed. Wrong choice for these coefficients may lead to strong overshoot and oscillatory solution or very slow convergence towards the target. Other RPM control methods have been proposed by Gustin et al(2018) and Norrison et al(2017), but these methods are rather iterative RPM adjustments than continuous active control.

In present work, the RPM control (the automated self-propulsion point search algorithm) is derived directly from the non-dimensional thrust curve, $K_T$, scaling. Separate versions for conventional propellers and CRP are derived. The CRP version is directly applicable to three propeller ships, with fixed (but adjustable) side/centre RPM ratio. This automated self-propulsion point search algorithm is quite generic and applicable to any type of propulsor with smooth and continuous thrust curve $K_T$. It is emphasized that the algorithm is designed to function with the actual propeller geometry (not actuator disc) and does not require a predefined $K_T$ curve. The algorithm has only one free parameter, the relaxation time, whose appropriate value is easily determined. If the final propeller rotational speed, $n$, in revolutions per second can be roughly estimated, the value $1/n$ always works. Another intuitive value is the estimated time it takes for the unwetted propulsors to accelerate to full power. The former (shorter) estimate gives faster balance between thrust and resistance, while higher values towards the latter estimate are appropriate with high thrust deduction factors, where rapid propeller acceleration has strong effect on the hull resistance and may cause unwanted slowly dissipating waves. Hence, an appropriate relaxation time for the present algorithm is much easier to determine than the two unknown PI controller parameters.

The performance of developed algorithms is demonstrated with self-propulsion simulations of two tanker hulls. One ship hull is fitted with two Azipod units, and the other ship hull is fitted with a CRP system consisting of a larger shaftline propeller and a contra rotating propeller fitted to an Azipod unit. The simulations are performed with Siemens Simcenter Star-CCM+ 13.02.11 and the EHP 13.02.11 add-on. The CFD related details are touched only superficially in this work, and attention is focused on the automated self-propulsion point search algorithm and its performance. Practical guidelines for CFD utilization in ship resistance and self-propulsion simulations are presented, for instance, in ITTC(2014a,b,c). With the tanker hull fitted with two Azipod units, four situations are investigated: i) Recovery to self-propulsion from a grossly underestimated initial RPM. ii) Recovery to self-propulsion after a step change in hull surface roughness. iii) Effect of the relaxation time. iv) Result comparison to the two-point linear interpolation method. With the CRP tanker hull, self-propulsion points are simulated with 3 downstream-to-upstream propeller RPM ratios 1.40, 1.25 and 1.10. Due to strong propeller-hull interaction, the CRP simulations tend to create a low frequency, slowly attenuating wave upon initialization or change of the RPM ratio. This wave requires longer simulations for the hull resistance (and hence the thrust converging towards it) to level out. Ways to tackle this using RPM ratio ramps and suitable relaxation time are elaborated with the CRP results.

## 2 SETUP FOR THE SELF-PROPULSION SIMULATIONS

The EHP add-on to Star-CCM+ creates a proper size computational region for the ship, creates suitable mesh refinements across the water-air surface and around the hull, defines suitable wave damping conditions on the outer domain boundaries as well as sets up most of the models required in a self-propulsion simulation. EHP default realizable k-ε two-layer all-$y^+$ turbulence and volume of fluid (VOF) models are retained. EHP is designed for bare hull performance without appendages or propellers, so it is necessary to create refinement volumes for the Azipod unit(s) and the propellers. Then, the Azipod unit(s) and propeller geometries are fitted to the two hull geometries and sliding mesh rotating propeller regions (with rotating motion definitions) are created for the propellers. The mesh target resolution on/near the hull surface is doubled and the prism layer modified with first prism hight 0.4mm (to accommodate surface roughness up to 150μm) and smooth transition to the core mesh. Finally, the ship geometries are remeshed with the modified settings. The dimensions of the ship hulls and corresponding propellers are given in **Table 1**.

**Table 1. Simulated ship hulls and propellers**

| **Hull** | Hull 1 (2 Azipod units) | Hull 2 (CRP) |
|---|---|---|
| $L_{PP}$ | 282.5m | 260.0m |
| Breadth | 44.0m | 43.0m |
| Draught | 12.0m | 11.0m |
| **Propellers** | Azipod units | Shaftline / Azipod unit |
| Diameter | 7.95m/7.95m | 6.90m/6.10m |
| Number of blades | 4/4 | 5/4 |

The surface mesh on Hull 1 aft and the Azipod unit, as well as cross sections showing the hull prism layers and volumetric refinements around the aft are presented in **Figure 1**. Due to symmetrically placed Azipod units, only half of the ship Hull 1 is modeled. The number of computational cells for Hull 1 is 4.5M + 3.0M for hull- and pod propeller regions, respectively.

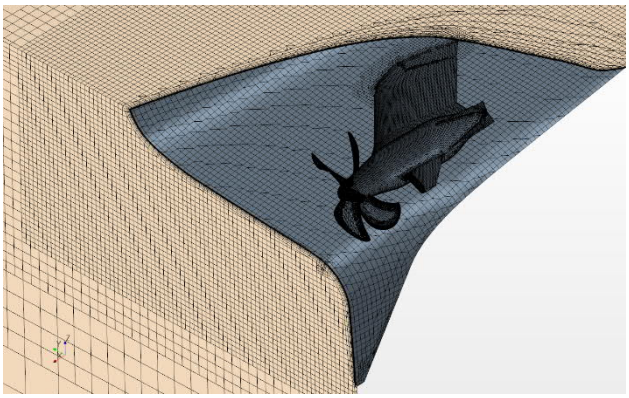

**Figure 1. Computational mesh on Hull 1 aft and Azipod unit**

The surface mesh on Hull 2 aft and the CRP system, as well as a cross section showing the hull prism layer are

presented in **Figure 2**. The CRP system on Hull 2 is asymmetric, so the entire hull is modeled. The number of computational calls for Hull 2 is 9M + 3M +2.5M for hull-, shaftline propeller-, and pod propeller regions, respectively. The mesh refinement levels for Hull 1 wave system at initial water surface level are presented in **Figure 3**. Similar mesh refinements are created for Hull 2 wave system, symmetrically around the entire hull.

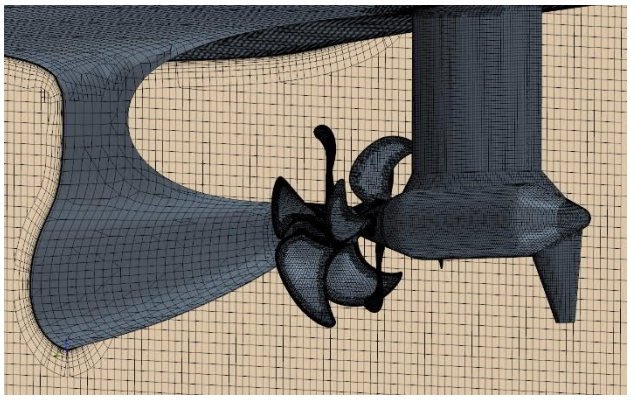

**Figure 2. Computational mesh on Hull 2 aft and CRP system**

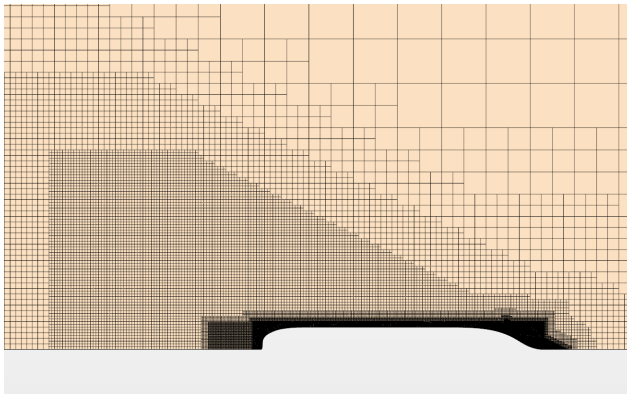

**Figure 3. Hull 1 mesh refinement levels for the wave system**

## 3 DERIVING THE SELF-PROPULSION POINT SEARCH ALGORITHMS

As mentioned in the introduction, ITTC(2014a) recommends the Proportional-Integral (PI) controller to achieve balance between thrust, *T* (positive in the direction of motion), and hull resistance, *R* (positive in the direction opposite to motion). Working with the error

$$e = R - T \tag{1}$$

the PI controller for the propeller rotation rate, *n*, is written simply as

$$n(t) = Pe(t) + I\int_0^t e(t)\,dt \tag{2}$$

where *P* and *I* are the proportional and integral coefficients of the controller. These numerical values need to be determined by the user. This is by no means a simple task, as appropriate values vary depending on the case details (ship speed, ship length, propeller design). Inappropriate coefficient values may lead to highly oscillating solutions or very slow convergence. This is perhaps the reason this perfectly sound method is seldom applied in practice or presented in publications, and engineers tend to prefer the interpolation method, which always works but requires a larger number of simulations (at least two, in most cases three instead of one). This encourages to develop an automated self-propulsion point search algorithm that is less involved and easier to apply than the PI controller.

### 3.1 A new SPP search algorithm for FPP (and CPP)

Let us begin from the basic definition for non-dimensional thrust curve for propellers

$$K_T = \frac{T}{\rho n^2 D^4} \tag{3}$$

where $\rho$ is the fluid density, *n* (now in rps) and *D* is the propeller diameter. To leading order, $K_T$ is constant around an operating point (*n*, *T*) at constant speed. Linearizing about an operating point, we obtain

$$T = \beta n^2 \tag{4}$$

where the linearization coefficient $\beta$ is now considered constant (within a time step, not otherwise). The trick is that it is not necessary to define $\beta$ *a priori*. Instead, as soon as a single time step has been simulated (with a guessed or estimated *n*), the instantaneous value is obtained directly from the simulated quantities

$$\beta = \frac{T}{n^2} \tag{5}$$

i.e., the CFD software performs the linearization on the fly, and no *a priori* knowledge of propeller thrust curve is required. Differentiating Equation (4) with respect to time, we obtain

$$\frac{dT}{dt} = 2\beta n \frac{dn}{dt} \tag{6}$$

This equation already hints to time evolution of the rotation rate. The final trick is to link the thrust time derivative to the imbalance between resistance and thrust. We simply require that

$$\frac{dT}{dt} = \frac{R - T}{\Delta t^*} \tag{7}$$

where $\Delta t^*$ is the user specified relaxation time. As discussed already in the introduction, the appropriate choice of this parameter is intuitive and easy, e.g., a value of order 1/*n* (*n* in rps) always works. Combining Equations (6) and (7) and rearranging, we obtain the time evolution for the propeller rotation rate

$$\frac{dn}{dt} = \frac{R - T}{2\beta n \Delta t^*} \tag{8}$$

It is implicitly assumed that the user, now simulating self propulsion, initializes the simulation with a positive propeller rotation rate. The choice of initial rotation rate is not critical. The algorithm finds the correct RPM level quickly even from a very poor initial guess (even with negative thrust), as shown in Section 4.1. However, the linearization in Equation (4) makes sense only when also the thrust is positive. The initial rotation rate may easily be

underestimated so that the thrust becomes negative. For instance, a typical propeller with design point advance coefficient $J = V/(nD) = 0.9$ and zero thrust at $J = 1.3$ would produce negative thrust already below 0.9/1.3 = 70% of the design RPM at the design speed. To handle this exception, it is necessary to limit the minimum thrust in Equation (5) to a small value, say $T_{min}$ = 1kN/1N in full/model scale ships. The exact value is not important since the simulation lingers in the negative thrust only a very short time, and Equation (4) becomes physically sound as soon as the thrust turns positive with increasing rotation rate. Furthermore, since both the instantaneous thrust and resistance are always transient values (thrust oscillating with blade frequency, $f_b$, and resistance varying due to interaction with the propeller and perhaps due to transient waves not yet dissipated) it is recommended to use short sliding time averages of $R$, $T$, and $n$ on R.H.S. of Equations (5) and (8) for smoother behavior. An appropriate time scale for the sliding average is, for instance, the blade passing period $1/f_b$. Finally, the self-propulsion point search algorithm for conventional propellers in discretized form reads as

$$\beta_i = \frac{\max(\bar{T}_i, T_{min})}{\bar{n}_i^2} \tag{9}$$

$$n_{i+1} = n_i + \frac{\bar{R}_i - \bar{T}_i}{2\beta_i \bar{n}_i \Delta t^*}\Delta t \tag{10}$$

where subscripts $i$ and $i$+1 refer to time levels, $\Delta t$ is the physical time step and the overbar denotes sliding time average over one blade passing period. This version works equally well for CPP, since adjusting the blade pitch is just a new propeller geometry, and the algorithm adjusts to the new condition through the linearization coefficient $\beta$.

### 3.2 A new SPP search algorithm for CRP

With CRP, the rotation rate of the downstream propeller is assumed proportional to that of the upstream propeller

$$n_2 = \gamma n_1 \tag{11}$$

where subscripts 1 and 2 refer to upstream and downstream propellers, respectively and $\gamma$ is the RPM ratio. Both rotation rates, and hence also $\gamma$, are assumed positive, i.e., their rotation axes are opposite to produce contra rotation. It is assumed that the final SPP is always sought with a fixed $\gamma$, but it may be varied during simulation when shifting from one $\gamma$ value to another (when finding the optimal $\gamma$ for the CRP). Furthermore, it is assumed (and recommended) that should $\gamma$ be changed during simulation, it is done along a linear ramp

$$\gamma = \begin{cases} a, & t < t_1 \\ a + \dfrac{b-a}{t_2 - t_1}(t - t_1), & t_1 \le t < t_2 \\ b, & t \ge t_2 \end{cases} \tag{12}$$

The RPM ratio could as well be changed stepwise but that causes an unnecessary disturbance in both thrust and resistance, which in turn considerably delays convergence towards the self-propulsion point. Equivalent to Equation (4), the linearized thrust now reads as

$$T_1 + T_2 = \beta_1 n_1^2 + \beta_2 n_2^2 = \beta_1 n_1^2 + \beta_2 \gamma^2 n_1^2 \tag{13}$$

The linearization coefficients $\beta_1$ and $\beta_2$ for the propellers are obtained directly from simulated quantities identically to Equation (9) with the same limitation applied to minimum thrust. Differentiating Equation (13), keeping Equation (12) in mind, yields

$$\frac{d(T_1 + T_2)}{dt} = 2n_1(\beta_1 + \beta_2\gamma^2)\frac{dn_1}{dt} + 2n_1^2\beta_2\gamma\frac{d\gamma}{dt} \tag{13}$$

where the $\gamma$ time derivative is now a constant, (b-a)/($t_2$-$t_1$), during the ramp and zero otherwise. With the same trick used in Equation (7), the time evolution for the upstream propeller rotation rate becomes

$$\frac{dn_1}{dt} = \frac{R - (T_1 + T_2) - 2n_1^2\beta_2\gamma\dot{\gamma}\Delta t^*}{2n_1(\beta_1 + \beta_2\gamma^2)\Delta t^*}\Delta t \tag{14}$$

where the dot above symbol denotes time derivative. An appropriate value for the relaxation time is for instance $1/n$ ($n$ in rps) of the slower propeller. The time evolution for the downstream propeller rotation rate is immediately obtained from Equation (11). For smoother behavior, the short sliding time averages are again recommended for $R$, $T_1$, $T_2$, $n_1$, and $n_2$ on the R.H.S. of the linearization coefficient equations and Equation (14). An appropriate time scale for the sliding average is the blade passing period $1/f_b$ of either of the propellers. Finally, the self-propulsion point search algorithm for CRP in discretized form reads as

$$\beta_{1,i} = \frac{\max(\bar{T}_{1,i}, T_{min})}{\bar{n}_{1,i}^2} \tag{15}$$

$$\beta_{2,i} = \frac{\max(\bar{T}_{2,i}, T_{min})}{\bar{n}_{2,i}^2} \tag{16}$$

$$n_{1,i+1} = \left(n_1 + \frac{\bar{R} - (\bar{T}_1 + \bar{T}_2) - 2\bar{n}_1^2\beta_2\gamma\dot{\gamma}\Delta t^*}{2\bar{n}_1(\beta_1 + \beta_2\gamma^2)\Delta t^*}\right)_i \Delta t \tag{17}$$

$$n_{2,i+1} = \gamma_{i+1} n_{1,i+1} \tag{18}$$

The procedure for three propeller systems controlled by side/center propeller RPM ratio is identical. This is because all the propeller physics, propeller-propeller interaction or lack of it are implicitly given in the linearization coefficients $\beta_1$ and $\beta_2$, solved each time step by the CFD software on the fly.

## 4 ALGORITHM PERFORMANCE

The two versions of the automated self-propulsion point search algorithm are tested with two ship hull geometries. Hull 1 with two Azipod units with FPP (half of the ship and one Azipod unit modeled due to symmetry) and Hull 2 with a CRP system (whole hull modeled). The ship and propeller dimensions are given in Section 2, **Table 1**. The sink and trim are fixed to zero for both ships to reduce the computational time in these demonstrative simulations. Both ships are simulated at 20knots.

### 4.1 Hull 1 – two Azipod units with fixed pitch propellers

As a first test, the initial propeller RPM is intentionally set to an overly low value, 10rpm, to show how the algorithm manages the thrust zero crossing. The simulation is quickly initialized for the algorithm by gradually decreasing the time step from 0.5s towards the target corresponding to a

2° propeller rotation until the resistance and thrust settle to approximately constant levels. After this, a variable time step to maintain a 2° propeller rotation per step is employed. This makes it easier to set the correct number of time steps for the short sliding averages used in the algorithm. For a four-blade propeller the blade passing period corresponds to 45 time steps, and this is set as the length of the short sliding averages of $R$, $T$ and $n$. The expected propeller speed at SPP is above 60rpm (1rps), so the relaxation time for the algorithm is set to $\Delta t^* = 1$s. The algorithm is activated at t = 40s. The relaxation of thrust towards resistance is presented in **Figure 4**, the corresponding propeller RPM in **Figure 5**, and the propeller power in **Figure 6**. The propeller quickly gains speed, while it takes around 1 second for the thrust to react. The propeller power drops from -2MW to -4MW within the first second (this is how the propeller behaves in the "generator" mode below positive thrust producing rotational speeds). As the propeller speed still increases, the thrust begins to rise with a very steep slope, has a sharp overshoot around 2 seconds after algorithm activation and attains equilibrium with the resistance just 4 seconds after algorithm activation. After this, the thrust obediently follows the resistance. The effect of thrust deduction is also clear in **Figure 4**, as the resistance increases approximately 200kN when the propeller is powered up. Although a longer simulation is required for the resistance and the thrust to level out to final SPP values, the simulation is paused at t = 50s, and the hull surface roughness is changed from 0 to 150μm. After restart, the resistance immediately shoots up from around 950kN to 1200kN, and it takes less than 2 seconds for the thrust to catch up with the new resistance level. After this, resistance, thrust, propeller RPM and power continue to level out towards the final undisturbed SPP. Similar effect would result from adding appendages to the hull (and remeshing) or from adding an estimated superstructure resistance to target resistance in Equation (10). Final convergence to SPP is presented further in this section. The internal accuracy of the algorithm is demonstrated in **Figure 7** with the relative error between resistance and thrust averaged over 6 propeller revolutions. The relative error drops below 0.05% in these short simulations with 0 and 150μm hull surface roughness. Note that this is just a measure of how closely the thrust follows the resistance, not the accuracy of SPP which is recovered after the curves level out (then the relative error is the accuracy of SPP).

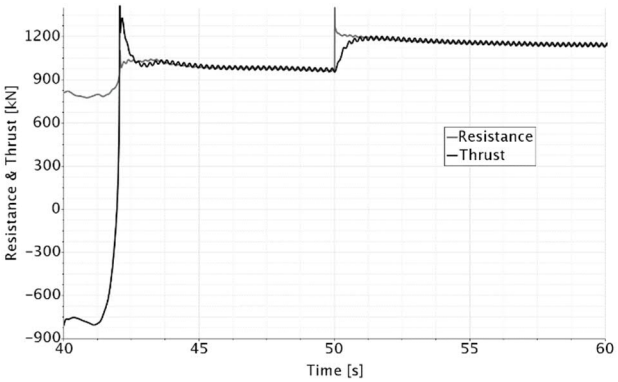


**Figure 4. Thrust relaxing towards resistance with the SPP algorithm. Hull roughness changed 0 → 150μm at t = 50s**

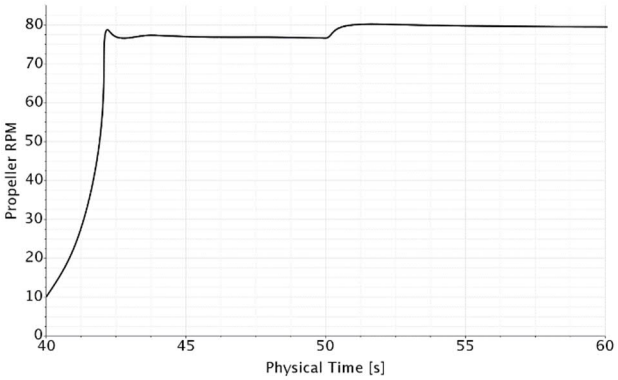


**Figure 5. Propeller RPM with the SPP algorithm. Hull roughness changed 0 → 150μm at t = 50s**

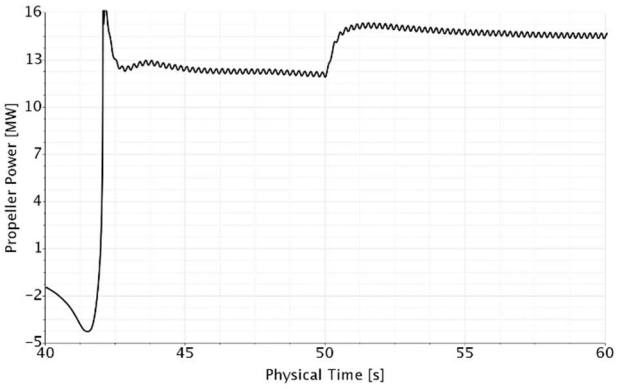


**Figure 6. Propeller power with the SPP algorithm. Hull roughness changed 0 → 150μm at t = 50s**

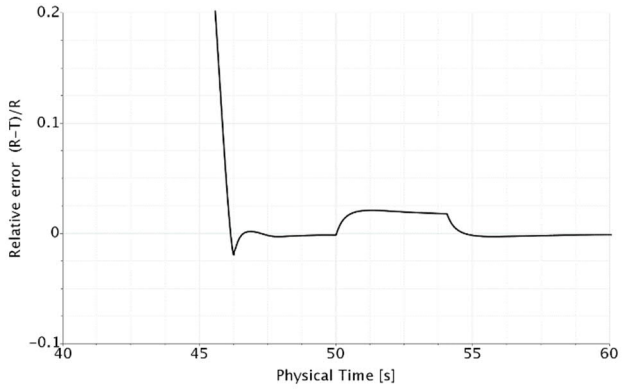


**Figure 7. Relative error. *R* and *T* averaged over 6 propeller revolutions with the SPP algorithm. Hull roughness changed 0 → 150μm at t = 50s**

The effect of the relaxation time is studied by continuing the previous smooth hull simulation to t = 60s and repeating the simulation from t = 40s with the relaxation time doubled to $\Delta t^* = 2$s. The corresponding thrust curves are presented in **Figure 8**. As expected, the longer relaxation time results in slower response and less overshoot but eventually the curves converge together. Hence, the converged SPP solution is independent of the chosen relaxation time.

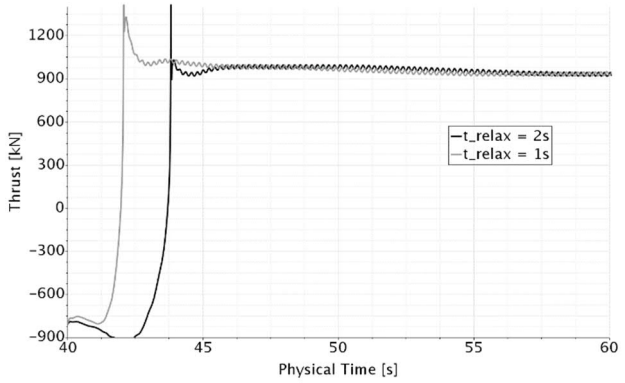


**Figure 8. Effect of relaxation time on thrust with the SPP algorithm**

As the final FPP test, the previous 150μm rough hull simulation is continued to t = 90s to obtain the converged SPP. For comparison, two constant RPM simulations, starting from the t = 50s state, are performed. The constant propeller speeds are chosen by selecting two random points from the SPP algorithm RPM curve, adding 1rpm to the larger and subtracting 1rpm from the smaller value to get some asymmetry as is probable in a real case. The sliding average window is increased to 10 propeller revolutions in all three simulations for more accurate final values. Simulation from 50s to 90s is just enough for the constant RPM relative resistance-thrust differences to level out, as presented in **Figure 9**. Note that off SPP, these curves should not go to zero but level out instead. The performance data of the constant RPM simulations and the linearly interpolated SPP values are compared to the SPP algorithm results in **Table 2**. The resistance-thrust errors in the table are relative to the SPP algorithm resistance. Both the interpolated and the SPP algorithm relative errors are below 0.1%, accurate enough for any engineering purpose. This example also shows that the two-point interpolation method is sufficient if propeller speeds can be estimated within, say ±2%.

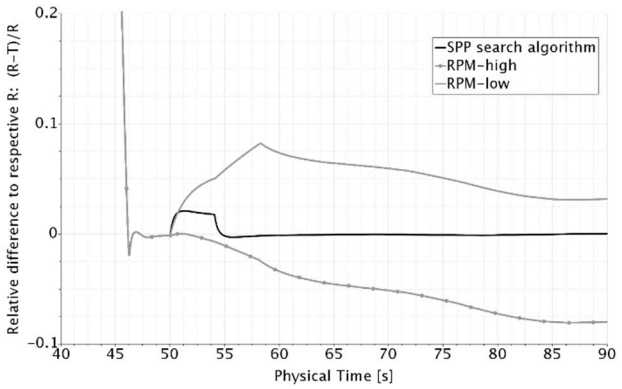


**Figure 9. Relative difference of thrust to resistance. SPP algorithm and constant RPM simulations**

**Table 2. Self-propulsion point comparison. SPP algorithm vs. two-point interpolation method**

| Case | RPM-low | RPM-high | Interpol. | SPP algorithm |
|---|---|---|---|---|
| RPM | 78.14 | 80.32 | 78.75 | 78.77 |
| R [kN] | 1114.26 | 1124.63 | 1117.18 | 1116.42 |
| T [kN] | 1078.99 | 1214.64 | 1117.18 | 1116.22 |
| P [MW] | 13.57 | 15.52 | 14.12 | 14.10 |
| $\lvert\varepsilon_{rel}\rvert$ | - | - | 0.07% | 0.02% |

### 4.2 Hull 2 – CRP

As the first test, the RPM ratio for the CRP is set to $\gamma = 1.40$ with the initial propeller speeds more carefully estimated. The flow field is quickly initialized for the SPP algorithm in the same manner as for the Hull 1 case. Now, the variable time step is set corresponding to 2° revolution of the faster Azipod unit propeller. The expected slower propeller speeds above 60rpm suggest $\Delta t^* = 1$s as a suitable relaxation time. For this four-blade propeller, the blade passing period corresponds to 45 time steps, and this is set as the length of the short sliding averages of $R$, $T_1$ $T_2$, $n_1$, and $n_2$, used in the algorithm. The resulting resistance, total and individual propeller thrust curves are presented in **Figure 10**. The total thrust catches up with the resistance in less than 3 seconds. After this, the total thrust faithfully follows the resistance. However, due to strong interaction between the CRP system and the hull, a slowly decaying disturbance is generated with all practical values of $\gamma$, suggesting that a longer relaxation time is appropriate to reduce this interaction induced disturbance. The corresponding propeller RPMs and propulsion power are presented in **Figure 11** and **Figure 12**, respectively.

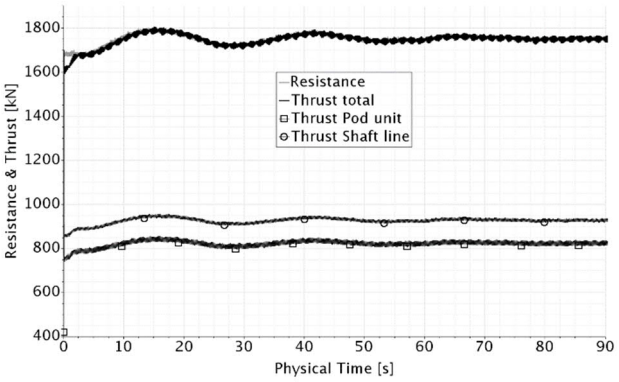


**Figure 10. Thrust relaxing towards resistance with the SPP algorithm CRP version, γ = 1.40**

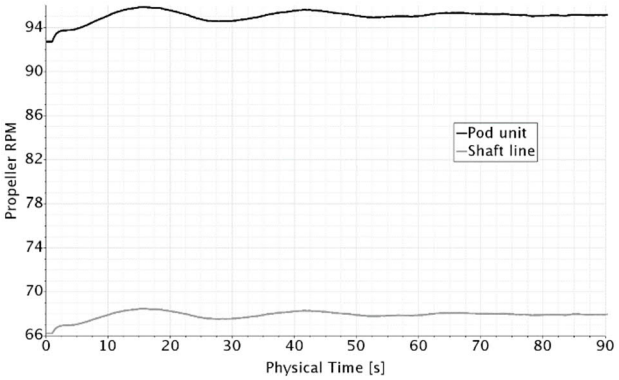


**Figure 11. Propeller RPMs with the SPP algorithm CRP version, γ = 1.40**

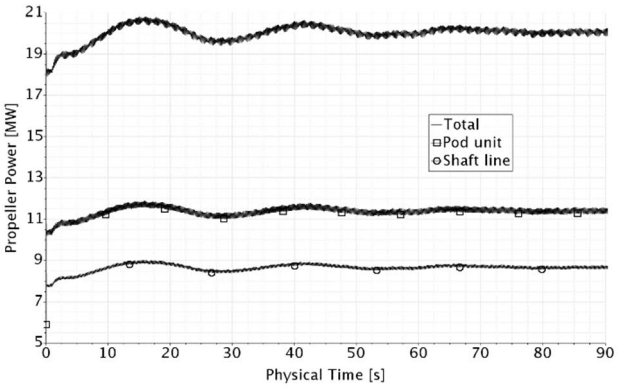


**Figure 12. Propulsion power with the SPP algorithm CRP version, γ = 1.40**

As the second test, the time history of the previous simulation is zeroed and the RPM ratio of the CRP system is ramped down from 1.40 to 1.25 in 2 seconds starting from t = 1s. The resulting resistance, total and individual propeller thrust curves are presented in **Figure 13**. The individual propeller thrusts change almost linearly during this ramp, and there is just a small, approximately 30kN, bend in the total thrust where it drops below the resistance. After the ramp, the total thrust catches up with the resistance in a couple of seconds. The corresponding propeller RPMs and propulsion power are presented in **Figure 14** and **Figure 15**, respectively. It is noteworthy that the disturbance is now considerably smaller although the individual propeller speeds and thrusts change much more and faster than in the previous simulation. This is due to the $\gamma$ time derivative term included in the algorithm, see Equation (17). Changing the RPM ratio stepwise would result in a disturbance even stronger than in the previous simulation as the Azipod unit (and total) thrust would immediately drop by some 200kN (approximating from **Figure 13**).

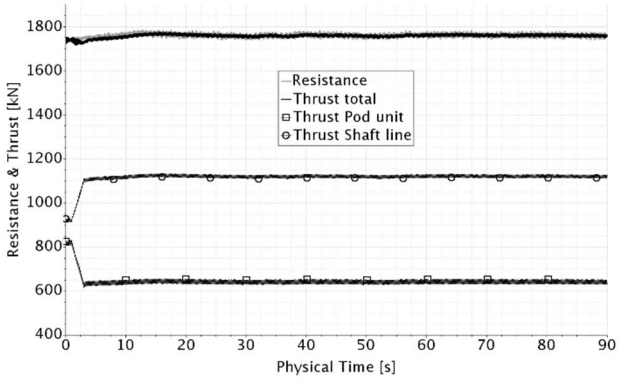


**Figure 13. Thrust relaxing towards resistance after $\gamma$ ramp from 1.40 to 1.25 with the SPP algorithm CRP version**

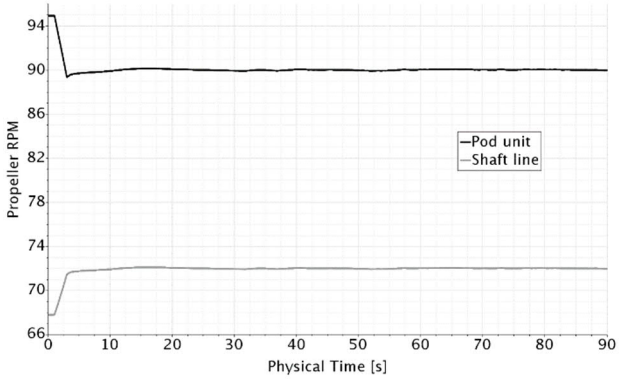


**Figure 14. Propeller RPMs after $\gamma$ ramp from 1.40 to 1.25 with the SPP algorithm CRP version**

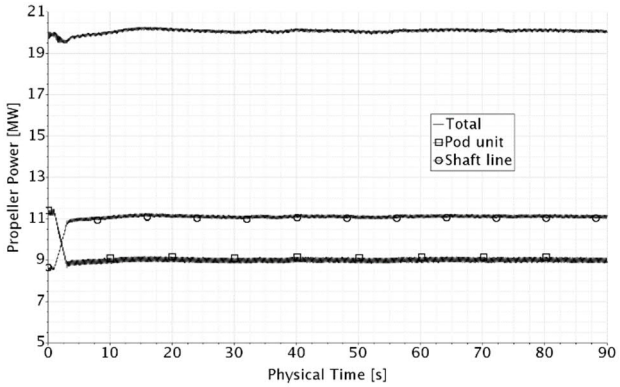


**Figure 15. Propulsion power after $\gamma$ ramp from 1.40 to 1.25 with the SPP algorithm CRP version**

As the third test, the time history of the previous simulation is again zeroed, and the RPM ratio of the CRP system is ramped further down from 1.25 to 1.10, now in 4 seconds starting from t = 1s. The resulting resistance, total and individual propeller thrust curves are presented in **Figure 16**. Again, the individual propeller thrusts change almost linearly during the ramp, and the total thrust bend is now barely visible due to the more gently sloped RPM ratio ramp. Moreover, the disturbance found in the first simulation is now almost completely gone, the total thrust remains on top of the resistance curve, and the SPP result is complete already at t = 60s. The corresponding propeller RPMs and propulsion power are presented in **Figure 17** and **Figure 18**, respectively. The SPP results with the three RPM ratios are presented in **Table 3**. It is noted that the hull resistance (and hence thrust deduction) slightly increases with decreasing RPM ratio as more power is drawn from the upstream propeller closer to the hull. The total required power is almost unchanged with RPM ratios 1.40 and 1.25 although the distributions of thrust and power between the propellers change considerably. A notable increase in required power is observed with $\gamma$ = 1.10. Finally, it is concluded that in a series of CRP simulations with a set of RPM ratios, the first simulation (with probably the largest initial resistance-thrust imbalance) should be simulated with a longer relaxation time, say 2-3 times that suggested by the slower propeller speed to reduce the strong propeller-hull interaction induced disturbance. In subsequent simulations, a suitable RPM ratio ramp prevents the disturbance.

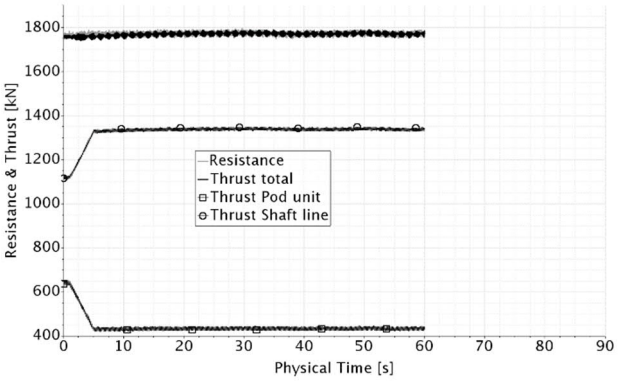


**Figure 16. Thrust relaxing towards resistance after $\gamma$ ramp from 1.25 to 1.10 with the SPP algorithm CRP version**

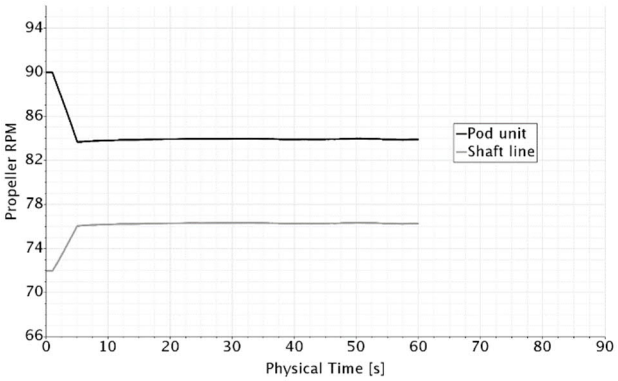


**Figure 17. Propeller RPMs after $\gamma$ ramp from 1.25 to 1.10 with the SPP algorithm CRP version**

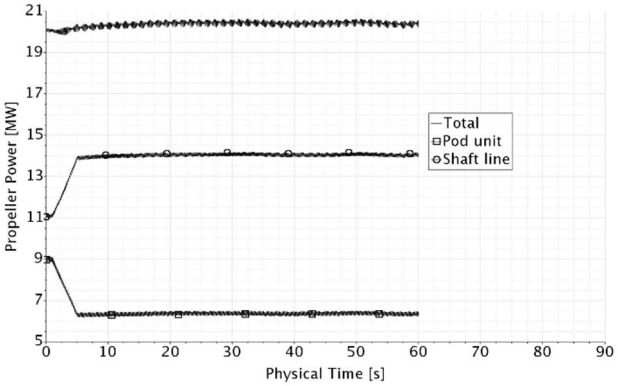


**Figure 18. Propulsion power after $\gamma$ ramp from 1.25 to 1.10 with the SPP algorithm CRP version**

**Table 3. Self-propulsion points with the SPP algorithm CRP version**

| $\gamma$ | 1.40 | 1.25 | 1.10 |
|---|---|---|---|
| $RPM_1$ | 67.96 | 72.00 | 76.27 |
| $RPM_2$ | 95.15 | 90.00 | 83.90 |
| R [kN] | 1752 | 1760 | 1771 |
| $T_1$ [kN] | 928 | 1119 | 1337 |
| $T_2$ [kN] | 824 | 641 | 434 |
| $T_{tot}$ [kN] | 1752 | 1760 | 1771 |
| $P_1$ [MW] | 8.68 | 11.09 | 14.05 |
| $P_2$ [MW] | 11.41 | 8.99 | 6.36 |
| $P_{tot}$ [MW] | 20.09 | 20.08 | 20.41 |

## 5 CONCLUSIONS

Separate versions of the self-propulsion point search algorithm for fixed pitch propellers and contra rotating propellers derived in this work are shown to perform excellently in the two self-propulsion point demonstrations presented. Both algorithm versions are easily implemented to any CFD software allowing user programming. With the algorithms, the required simulation time and number of simulations can be reduced at least by 50% compared to the standard two-point interpolation technique since the algorithms find the self-propulsion point accurately in a

single simulation. The savings are even more pronounced if the estimated propeller speeds for the interpolation are not sufficiently close to the self-propulsion point RPM, and further points are required. Of known techniques, only the Proportional-Integral (PI) controller achieves comparable performance. However, the PI controller contains two model coefficients whose optimal values vary depending on the case and are not straightforward to evaluate. The present algorithms contain only one user defined parameter, the relaxation time, whose value is intuitive and easy to choose: With a ballpark estimation of the correct propeller speed $n$ (in rps), the choice $1/n$ always works. The FPP version of the algorithm works directly with CPP, as a new pitch setting is automatically detected by the algorithm's linearization part. In fact, the continuous linearization via simulated quantities (thrust and rotational speed) is the key component that supplies the propeller physics to the algorithm without involvement from the user's side. With the CRP version of the algorithm, it is possible to quickly change from one RPM ratio to another almost without deflecting the total thrust curve from the resistance curve, albeit the individual propeller thrusts undergo substantial changes. The CRP version of the algorithm is also directly applicable to self-propulsion point simulation of three propeller ships operated with a fixed (but adjustable) RPM ratio.

## REFERENCES


Carrica, P., Castro, A. and Stern, F. (2010), 'Self-propulsion computations using speed controller and discretized propeller with dynamic overset grids', Journal of Marine Science and Technology 15: 316-330.

Gaggero, S., Villa, D., and Viviani, M. (2017), 'An extensive analysis of numerical ship self-propulsion prediction via a coupled BEM/RANS approach', Applied Ocean Research 66: 55-78.

Gustin, G., Lavini, G., and Mocnik, F. (2018). 'CFD Automated Self Propulsion Test', Technology and Science for the Ships of the Future: Proceedings of NAV 2018: 19th International Conference on Ship and Maritime Research, Marinò, A., and Bucci, V. (Eds.), IOS Press Ebooks: 753-761.

ITTC (2014a), 'Practical Guidelines for Ship Self-Propulsion CFD (7.5-03-03-01)'.

ITTC (2014b), 'Practical Guidelines for Ship CFD Applications (7.5-03-02-03)'.

ITTC (2014c), 'Practical Guidelines for Ship Resistance CFD (7.5-03-02-04)'.

Norrison, D., Petterson, K., and Sidebottow, W. (2017). 'Numerical Study of Propeller Diameter Effects for a Self-Propelled Conventional Submarine', Fifth International Symposium on Marine Propulsors SMP'17, Espoo, Finland.

Pacuraru, F., Lungu, A., and Marcu, O. (2011). ' Self-Propulsion Simulation of a Tanker Hull', AIP Conference Proceedings 1389: 191-194, Halkidiki, Greece.

Ponkratov, D., and Zegos, C. (2015). 'Validation of Ship Scale CFD Self-Propulsion by the Direct Comparison with Sea Trial Results', Fourth International Symposium on Marine Propulsors SMP'15, Austin, Texas.