OPTIMIZATION OF A SELF-TWISTING COMPOSITE MARINE PROPELLER USING GENETIC ALGORITHMS

Mateusz M. Pluciński*, Yin L. Young, Zhanke Liu****
***Department of Operations Research and Financial Engineering, Princeton University**
****Department of Civil and Environmental Engineering, Princeton University**

Keywords: *composite propeller, genetic algorithm, optimization*

Abstract

The design for a self-twisting composite marine propeller is optimized using a genetic algorithm (GA). The objective is to optimize the efficiency of the composite marine propeller for many inflow velocities, an objective unattainable for rigid propellers, and possible only due to the material properties of the self-twisting propeller. Parameters to be optimized are the orientation angles of the fibers in each layer. A generic GA is modified to include a binary-tree memory and local improvement procedures. The GA is also modified to interface with fluid mechanics solvers and ABAQUS. To test the efficacy of a simplified model, two optimizations are performed, one for a simple beam model, and another for the full-fledged propeller model, and these results are compared. Overall, the results for the optimization of the propeller show a significant increase in efficiency for a range of operating conditions.

1 Introduction

One of the difficulties in designing a marine propeller is that the optimal shape of the propeller changes with the inflow velocity, and thus propeller design often requires a propeller that is optimized for only a small range of inflow velocities and works sub-optimally over a larger range of inflow velocities. This paper proposes that the loss of optimality assumed by this approach can be reduced by optimizing for a self-twisting composite laminate propeller whose shape contorts for different inflow velocities. By optimizing the material configuration of the propeller in such a way that the contortions of the propeller work to maximize efficiency, the propeller can approach optimality for an entire spectrum of operating velocities.

The objective is to optimize the efficiency of the composite marine propeller across all possible operating conditions, determined by the advance coefficient *J*. Parameters to be optimized include the orientation angles of the fibers in each layer (discrete variables due to manufacturing constraints). In order to calculate the efficiency of a composite marine propellers given these parameters, a coupled boundary element (BEM) and finite element (FEM) approach is used to predict the performance of self-twisting composite propellers in time-dependent flows [1,2]. The coupled algorithm is able to model complex cavitation patterns on both sides of the blade, and compute the unsteady hydrodynamic load, stress distributions, deflection patterns, and modal characteristics. Both because the coupled algorithm requires a significant amount of computation time (c. 15 minutes), and due to the fact that the parameters are discrete, special care must be given to the choice of the optimization algorithm.

The proposed optimization algorithm is a modified genetic algorithm (GA). Genetic algorithms are a subset of stochastic optimization methods that are modeled after the process of natural selection [3]. Because of the expensive computational times involved in finding the objective function value, several improvements are made to the GA. A generic GA is modified to include both a binary tree memory, which cuts down on the number of function evaluations, and local improvement, a method that locally alters individuals in the GA by estimating the local behavior, and making the requisite adjustments without further function evaluations [4,5]. Finally, the GA is modified to interact with the coupled BEM/FEM algorithm.

Before optimizing the self-twisting composite propeller with the GA, several test cases are considered to study the affect of the modifications to the GA on the performance of the algorithm, and to investigate the feasibility of performing the optimization on simpler structures and applying the results to the more complex structure of the propeller. A composite laminate cantilevered plate under a uniform bending stress is chosen as a test case, and is found to be a fairly good model of the much more complex propeller.

1.1 Background on Propeller Optimization

Previous work on optimization of propeller structure has involved the continuous optimization of the physical parameters of the propeller. Khot and Zweber optimized the structure of a composite wing with respect to the thicknesses of the layers with fixed orientation angles using gradient methods [6]. Cho and Lee looked at the optimal shape of an air propeller by optimizing the twist angle distribution with the span twist angle distributions and spanwise chord length distributions acting as the design variables, utilizing gradient methods with penalty functions [7]. Lee and Lin investigated the possibility of maximizing the efficiency of composite propellers using genetic algorithms, with mixed results [8].

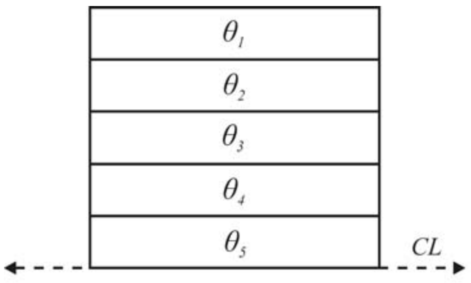


Fig. 1. The material structure

2 Optimization

2.1 Formulation

The objective of the optimization is to maximize the efficiency of the propeller across several inflow velocities, by finding the optimal sequence of fiber orientation angles in a composite self-twisting marine propeller. The structure of composite laminate material is shown in Fig. 1. It consists of 10 layers, with symmetry across the center line. The base propeller geometry is that of propeller 5474 [1,2], and the material used is a graphite epoxy.

Due to manufacturing constraints, the orientation fiber angles can only take on a fixed set of values. Because of this, the problem has a constrained discrete domain:

$$\theta_i = \{-15,0,15,30,45,60,75,90\} \quad i=1\ldots5 \qquad (1)$$

2.2 Objective Function

Although the ultimate objective of the optimization to is to maximize the efficiency of the propeller, the process for calculating the efficiency for a candidate process design is complex, and has not yet been automated. However, the efficiency of a propeller operating at a specific condition (defined by the advance coefficient *J*) is dependent on the pitch angle of the propeller (φ) operating at that advance coefficient. In fact, the analytically optimal efficiency for a propeller for each value of *J* corresponds to an optimal pitch angle at that value of *J*. The plots in Fig. 2 show this calculated set of optimal efficiencies and optimal pitch angles.

These same figures show the limitations of a rigid propeller design; because a rigid propeller maintains a constant pitch angle, independent of the advance coefficient, it can achieve maximum efficiency for only one specific value of the advance coefficient. The advance coefficient itself is a function of the advance speed, or inflow velocity, (*V*), the rotational speed (*n*), and the diameter of the propeller (*D*).

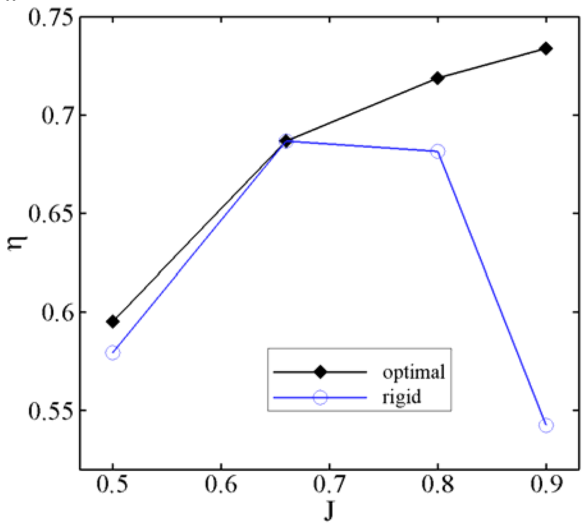
$$J = \frac{V}{nD} \qquad (2)$$

For rigid propellers, the propeller is designed to work optimally for a specific design advance coefficient, and there is a small range of inflow velocities where the propeller is optimal. However, larger deviations from design conditions result in sub-optimal efficiency.

This loss of efficiency could be overcome by a propeller that could change pitch angles according to changes in inflow velocity. Because self-twisting propellers deform in response to changes in inflow velocity, they are not constrained to a single pitch angle and can approach this optimal set of pitch angles. Thus, the optimization problem is formulated as the minimization of the total distance between the optimal theoretical pitch angles and the pitch angles predicted for a self-twisting propeller.

This can further be simplified since it is sufficient to only maximize twist in a candidate propeller design, as any increase in twist will increase the slope of the φ-J curve (Fig 2b), hence approaching the set of pitch angles that maximizes the efficiency for all inflow velocities.

a



b

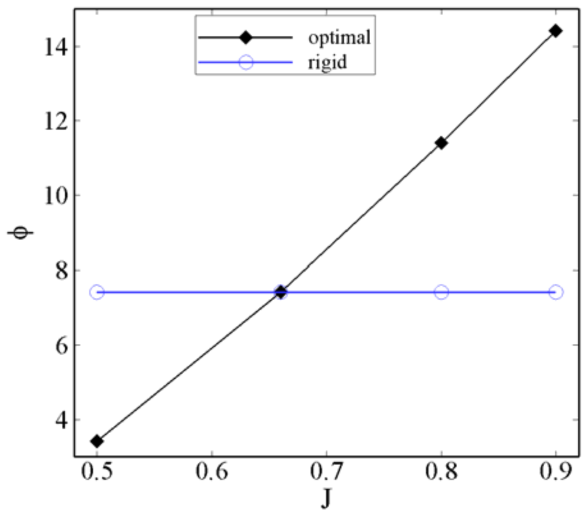


Fig. 2. The efficiency (a) for a rigid propeller is optimal for only one value of *J* because it has a constant pitch angle (b) that can match the optimal pitch angle at only one value of the advance coefficient

Thus, the maximization problem can be written as the maximization of the twist as a function of the orientation angles:

$$\max_{\theta} \varphi\left(\vec{\theta}\right) \qquad (3)$$

However, increasing the twist reduces the bending stiffness of the propeller. To limit the extent of this undesirable outcome, a constraint must be introduced:

$$\frac{k\left(\vec{\theta}\right)}{k_{\max}} \geq \alpha \qquad (4)$$

Here, k is the bending stiffness of the candidate propeller design, $k_{\max}$ denotes the maximum bending stiffness achievable for the material, and α is between 0 and 1. In the optimization runs described here, α is set to 0.5, corresponding to maintaining at least 50% of the maximum bending stiffness. In addition, in order to limit the chance of delamination failure, a constraint is implemented to limit the number of consecutive layers (m) sharing the same fiber orientation angle:

$$m\left(\vec{\theta}\right) \leq m_{\max} \qquad (5)$$

The maximum number of consecutive layers with the same fiber orientation is set to 2. Finally, the constraints are incorporated into the original maximization problem,

$$\max_{\theta} \left\{ \varphi\left(\vec{\theta}\right) \lambda_1^{\left(m\left(\vec{\theta}\right) - m_{\max}\right)} \left[\frac{k\left(\vec{\theta}\right)}{k_{\max}} \right]^{\lambda_2} \right\} \qquad (6)$$

Here, λ_1 and λ_2 are parameters that determine the severity of the penalty functions, and are 1 and 0, respectively, if the constraints are not violated. When the constraints are violated, $\lambda_1 < 1$ and $\lambda_2 > 0$.

2.3 Optimization Algorithm

Genetic algorithms [3], first developed at the University of Michigan by John Holland, are a subset of stochastic optimization methods. Genetic algorithms are specifically modeled after the process of natural selection, a process that itself is a stochastic form of optimization (over genes in this case). Natural selection guarantees that out of a population that is constantly varying through genetic mutations, the most-fit individuals have a higher chance of reproduction, and are therefore more likely to pass on their genes to the next generation. Genetic algorithms copy this type of evolutionary pressure, but instead of working with individuals differentiated by genes, they guide the evolution of populations of individuals that represent possible solutions to an optimization problem. Each individual is simply a collection of parameters that defines a certain point in the optimization domain, and these parameters are encoded essentially as binary chromosomes that can mimic real chromosomal operations such as mutation and cross-over.

Genetic algorithms begin by creating an initial population of random individuals, each representing a possible solution as a set of parameters. They then assign each individual a fitness, which is a measure of how well the particular set of parameters satisfies the objective function, with a higher fitness corresponding to a higher objective value. In order to model the process of natural selection, the genetic algorithm must make sure that the individuals that are most fit have the highest chance of reproducing and having children in the next population. Each subsequent population is determined by pairing off individuals to mate and have children, and the exact mechanism that chooses the mates varies. It can either be as simple as making the chance of having children proportional to the relative fitness of an individual, or can involve individuals competing for mates in a tournament where a higher fitness means a higher probability of winning. Regardless of the exact mechanism, the most-fit individuals have a higher chance of mating, and hence a higher chance of passing on their beneficial genes. Two individuals have children by simulating crossing-over, which is the process by which chromosomes exchange genes in real-world genetics. In genetic algorithms, a cross-over signifies that each child's chromosomes consist of a continuous series of a random number of bits from the father, with the remainder coming from the mother. Also, there exists a probability that a child may have a mutation in its genetic code, which is modeled by reversing one bit in a chromosome. This adds further randomness to a genetic algorithm, and prevents the algorithm from being trapped in a globally non-optimal local maximum.

The process of mate selection, crossing-over, and mutation is repeated for each new generation, and with time each population has more and more highly-fit individuals, and the process halts after a certain amount of generations, with the process hopefully having found the global optimum in the form of the most-fit individual over-all.

2.2.1 Memory

As the genetic algorithm progresses, various individuals in different generations begin to appear repeatedly, especially as the algorithm is beginning to converge to its final optimum. This is particularly relevant for optimization problems with discrete domains (as in the case of the problem at hand), as the likelihood of reappearing individuals is larger than for problems with continuous domains. Normally, the genetic algorithm does not account for repeated individuals across generations and evaluates the same individuals each time, which leads to inefficiencies and a slower process. To avoid these unnecessary redundancies, genetic algorithms have been augmented with binary tree memories [5]. The memory holds the fitness value for each individual encountered, effectively pairing each orientation angle combination with a fitness value.

A binary tree, a self-referential structure, is suited as a memory system as it provides quick insert and search capabilities. The tree is constructed by recursively entering new entries (the orientation angles coupled with the fitness) using the principle that all the branches to the left of a node are smaller than its value, and all the branches to the right are larger than its value. In the case of a set of orientation angles, the method, in determining whether a value is smaller or larger, first looks at the first orientation angle and then the next orientation angle. The search function is also recursive, and returns the required data (the fitness value) once it has found a matching node.

2.2.2 Local Improvement

Local improvement is a method by which one can speed up a genetic algorithm without requiring more function evaluations. The idea underlying local improvement is to optimize each individual for each generation by making slight alterations (the idea of "local" improvement) and estimating which slight alteration will produce the local optimum on the basis of regression analysis. However, it is difficult to estimate what affect slight alterations will have on sets of discrete variables, and two distinct methods have been developed to address this challenge.

The first method depends on the calculation of a secondary continuous variable unique to each set of discrete variables that are correlated to fitness, and which can aid in the estimation of fitness for new designs. Kogiso et al., for example, calculated the bending lamination parameters W_1^* and W_2^* for each individual and stored these in the binary-tree memory [4]. For local improvement, the algorithm perturbs each individual by making every possible swap of two orientation angles. The algorithm then looks up the five "closest" individuals to the individual that is to be optimized (in the W_1^* and W_2^* plane), and fits a curve to the relationship between the secondary variables and fitness. Now, all that is left is to calculate W_1^* and W_2^* for each new "altered" individual and estimate its new fitness value. By this process, it is possible to estimate which of the altered individuals is the best improvement over the "nominal" design. This new individual then replaces the nominal design in the breeding process of the

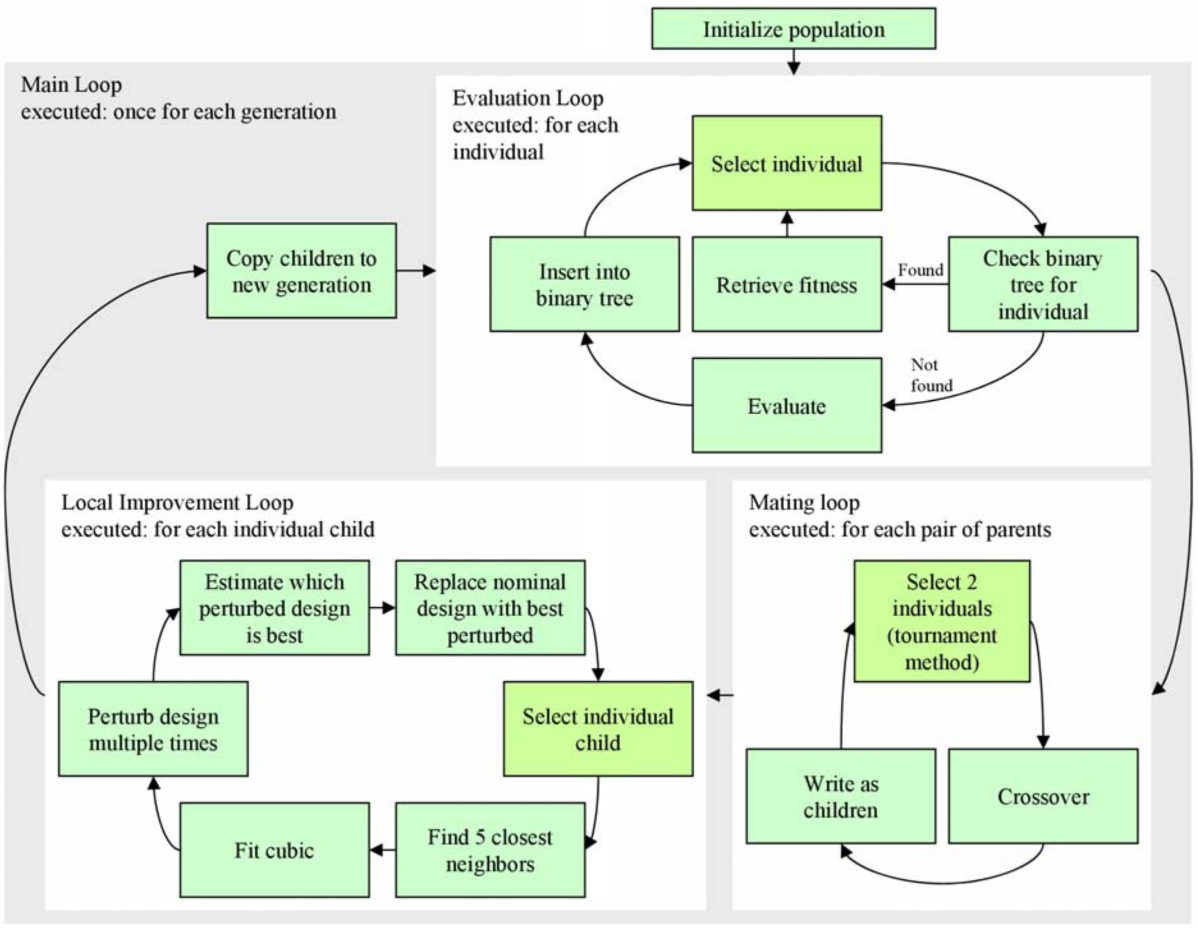

Initialize population
Main Loop
executed: once for each generation
Evaluation Loop
executed: for each individual
Select individual
Copy children to new generation
Insert into binary tree
Retrieve fitness
Found
Check binary tree for individual
Not found
Evaluate
Local Improvement Loop
executed: for each individual child
Mating loop
executed: for each pair of parents
Estimate which perturbed design is best
Replace nominal design with best perturbed
Select 2 individuals (tournament method)
Perturb design multiple times
Select individual child
Write as children
Crossover
Fit cubic
Find 5 closest neighbors

Fig. 3. A Genetic Algorithm with Memory and Local Improvement

genetic algorithm, and ensures that the genetic algorithm advances without new and expensive function evaluations.

The other method is to fit a curve to the discrete set of orientation angles instead of fitting a curve to secondary variables. Lin and Lee showed that regression analysis with various trigonometric functions can successfully approximate the response surface of composite laminated structures [9]. Later, they showed that this model can be used as a local improvement procedure when optimizing over orientation angles [10]. Their algorithm runs a regression analysis on the previous individuals, and finds the estimated fitness for locally perturbed individuals, replacing the worst individual in a generation with the best perturbed individual, whose fitness is exactly calculated.

2.4 Implementation

A generic FORTRAN GA [11] is modified to include a binary tree memory and a local improvement scheme that attempts to predict optimal swaps based on secondary variables. It is also modified to interact with a coupled BEM/FEM solver [1,2]. Fig 3 shows the overall structure of the genetic algorithm.

The optimization is performed in two stages, both sharing the same material design of Fig. 1 and the same objective function. However, the first stage involves the calculation of the twisting rate for a simple cantilevered beam, for which the computationally expensive BEM/FEM solver is not used. Instead, a relatively simple beam model based on classical composite plate theory is used to find the twisting rate of the cantilevered beam, in the hope that there is some correspondence between the material performance in the simple cantilevered beam, and the much more complex propeller.

This model test case also allows for the tweaking of the genetic algorithm to judge and evaluate its performance, and to find the optimal values for the parameters controlling the genetic algorithm, something that would be wasteful to do on the GA executing the coupled BEM/FEM evaluations.

The second stage is the full GA, making use of the accurate twist calculations coming from the BEM/FEM solver. This optimization run is the same as the tweaked model test case optimization run, with the exception that now, the twist values are obtained for the actual propeller design.

3 Results

3.1 Beam Code Run

The convergence plot for the beam code optimization run is shown in Fig 4. The GA convergences after about 60 generations, and the final optimal configuration angles (in degrees) are found to be:

$$\theta=15/15/0/0/90 \tag{7}$$

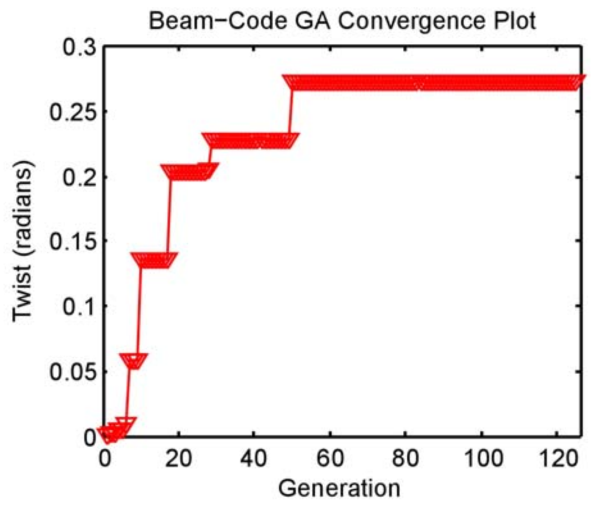


Fig. 4. Convergence of the genetic algorithm for simple beam code case

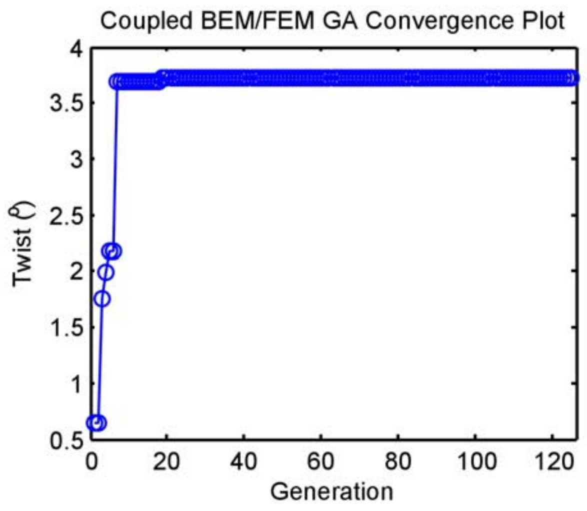


Fig. 5. Convergence of the genetic algorithm with the BEM/FEM solver

Notably, both the bending stiffness and number of consecutive orientation angle constraints are binding. Because of the symmetry, no non-consecutive layers exist. Similarly, the bending stiffness ratio of the optimal solution is 0.51, only slightly exceeding the minimal required. When the algorithm is allowed to run without the bending constraints, it can be seen that the optimal twist with the bending constraints is only c. 40% of the twist occurring without the bending constraints. However, the decrease in efficiency due to the number of consecutive orientation angles is less significant.

Another important insight from the beam-code run is that local improvement provides only minimal enhancement to the algorithm, speeding up convergence in only select runs, suggesting that this sort of "guided" approach fails when the secondary variables are only moderately linked to the objective; due to poor performance of local improvement in testing beam-code runs, it is not used in the runs utilizing the BEM/FEM solver to calculate twist. However, local improvement could potentially be useful when directly optimizing the efficiency of the propeller, using twist as a predictive secondary variable.

3.2 BEM/FEM Run

While the beam-code optimization run lasts only a few seconds, the full BEM/FEM run for the self-twisting propeller lasts around 50 hours of CPU time. The convergence plot for the algorithm is shown in Fig 5, and the optimal configuration is remarkably close to the one found in the primary test run:

$$\theta=15/15/0/0/30 \tag{8}$$

The algorithm follows a slightly different path than the one for the beam code, despite having the same algorithm parameters. The steeper rate of descent, and quicker leveling-off are evidence of a smaller pool of feasible configurations (that satisfy the constraints), suggesting that the bending constraints, although the same, are tighter in the case of the realistic propeller.

Given the optimal configuration determined by the GA coupled with the BEM/FEM process, a no-load propeller is designed with this material configuration, using the procedure outlined in [12]. The result of the design process yields the propeller shown in Fig. 6, in which the undeformed geometry corresponds to the designed no-load geometry, and the deformed geometry corresponds to the shape of the propeller when in operation.

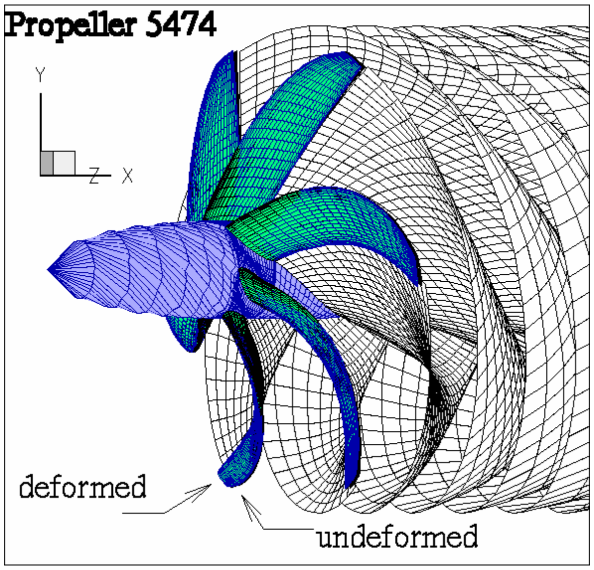


Fig. 6. The geometry of the new self-twisting propeller

The pitch angles for this optimized self-twisting propeller (Fig. 7) bridge the gap between the rigid propeller and the theoretical optimal, with the resultant increases in efficiency (Fig. 8). Overall, the self-twisting propeller shows significant efficiency gains ranging from 1 to 7 percentage points.

4 Conclusion

A genetic algorithm used to optimize the material fiber orientations of a 10-layered symmetric composite laminate propeller to maximize twist, calculated with a coupled BEM/FEM solver, proposes a new design for a self-twisting composite marine propeller that shows efficiency improvements over a rigid propeller for several distinct operating conditions.

However, for composite materials with more fiber layers, even an optimization algorithm like the GA would be prohibitively computationally expensive. Encouragingly, the same genetic algorithm, when calculating twist for a simplified beam model (instead of the BEM/FEM propeller model), yields results not much different from the results using the more expensive calculations. This suggests that the genetic algorithm utilizing the beam code can be used instead of the fuller BEM/FEM variant in more complex cases without much loss in optimality. Alternately, the results of the genetic algorithm using the simpler twist calculations could provide a good initial starting point (population) for a genetic algorithm (or any

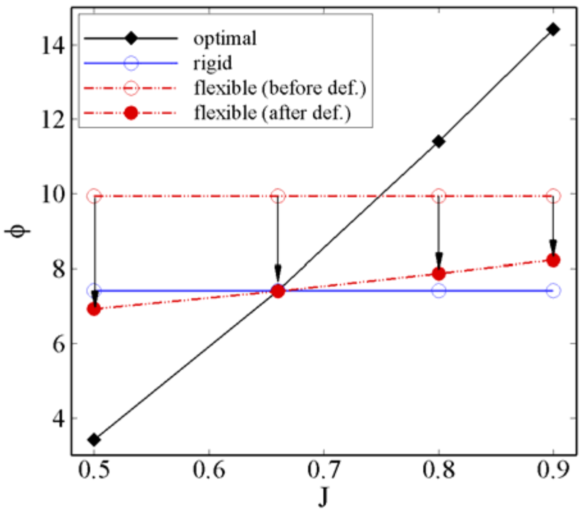


Fig. 7. Pitch angles for the new self-twisting propeller design

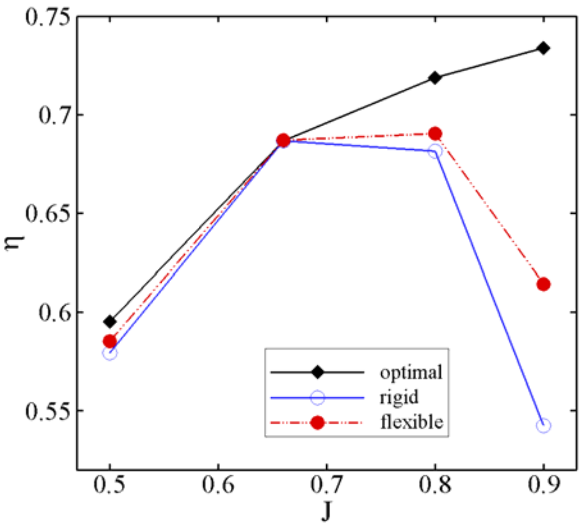


Fig. 8. Efficiency of the new self-twisting propeller design

other optimization algorithm) implementing the more accurate, but slower, twist calculations.

Constraints on both the number of consecutive layers and the bending stiffness limit the efficiency of the self-twisting propeller, although the bending stiffness constraint is much more limiting. This places an importance on material choice. Materials with strong twisting abilities that do not imply significant losses of bending stiffness and strength would decrease the effect of this constraint, and further increase the efficiency gains of the self-twisting composite propeller.

References

[1] Young Y. "Hydroelastic response of composite marine propeller". *The SNAME Propeller/Shafting Symposium,* Williamsburg, VA. 2006

[2] Young Y. "Numerical and experimental investigations of composite marine propellers". *Twenty-Sixth Symposium on Naval Hydrodynamics*, Rome, Italy. 2006.

[3] Goldberg D. "*Genetic Algorithms in Search, Optimization, and Machine Learning*". Addison-Wesley, 1989.

[4] Kogiso N., Watson L., Gurdal Z., Haftka R., "Genetic algorithms with local improvement for composite laminate design". *Tech. Rep. 93-17*. Department of Computer Science, Virginia Polytechnic Institute and State University. 1993.

[5] Kogiso N., Watson L., Gurdal Z., Haftka R., Nagendra S., "Design of composite laminates by a genetic algorithm with memory". *Tech. Rep. 94-02*. Department of Computer Science, Virginia Polytechnic Institute and State University. 1994.

[6] Khot N.S., Zweber J.V., "Design of flutter characteristics of composite wings using frequency constraint optimization". *Journal of Aerospace Engineering*, Vol. 16, pp 19-30, 2003.

[7] Cho J., Lee S.-C., "Propeller blade shape optimization for efficiency improvement". *Computers and Fluids*. Vol. 27, No. 3, pp 407-419, 1998.

[8] Lee Y.-J., Lin C.-C. "Optimized design of composite propeller". *Mechanics of Advanced Materials and Structures*. Vol. 11, pp 17-30, 2004.

[9] Lee Y.-J., Lin C.-C., "Regression of the response surface of laminated composite structures". *Composite Structures*. Vol. 62, pp 91-105, 2003.

[10] Lin. C.-C., Lee Y.-J., "Stacking sequence optimization of laminated composite structures using genetic algorithm with local improvement". *Composite Structures*. Vol. 63, pp 339-345, 2004.

[11] Carroll D.L., "Genetic algorithms and optimizing chemical oxygen-iodine lasers". *Developments in Theoretical and Applied Mathematics.* Vol. 18, pp 411-424, 1996.

[12] Young Y.L., Liu Z., "Hydroelastic tailoring of composite naval propulsors". *26th International Conference on Offshore Mechanics and Arctic Engineering.* San Diego, CA. 2007.