Aerothermal optimization is a powerful technique for the design of internal cooling passages because it maximizes heat transfer and simultaneously minimizes pressure loss. Moreover, the optimization is fully automatic, which reduces the duration of design process compared with a human-supervised design approach. Existing optimization studies commonly rely on gradient-free methods, which can only handle a small number of design variables. However, cooling passage designs use complex geometry configurations (e.g., serpentine channels with rib-roughened surfaces) to enhance heat transfer; what is needed is to parameterize the passage using a large number of design variables. To address this need, we perform aerothermal optimization using a gradient-based optimization algorithm along with the discrete adjoint method to compute derivatives. The benefit of using the adjoint method is that its computational cost is independent of the number of design variables. In this paper, we focus on optimizing a U-bend duct, which is representative of a simplified, rib-free turbine internal cooling passage. The duct geometry is parameterized using 135 design variables, which gives us sufficient design freedom for geometric modification. We construct a Pareto front for heat transfer enhancement and total pressure loss reduction by running multi-objective optimizations. We also compare our optimization results with those from the gradient-free methods and demonstrate that we achieve better pressure loss reduction and heat transfer enhancement. The above results show that our gradient-based optimization framework functions as desired and has the potential to be a useful tool for turbine aerothermal designs with full internal cooling configurations.

1 Introduction

To increase power and efficiency, gas turbine designs adopt a high turbine inlet temperature that is hundreds of degrees higher than the upper limit turbine materials can withstand [1]. Therefore, turbine cooling systems are critical, as they have a large impact on the safety and durability of gas turbine engines. One common way to reduce the turbine temperature is to inject cooling air into the turbine blades. The cooling air first flows through a serpentine rib-roughened internal cooling passage, and then flows to other cooling systems, e.g., jet impingement cooling or film cooling. The serpentine rib-roughened passage creates strong shear that generates flow separation and vortices to enhance turbulence mixing and heat transfer. However, the separated flow also increases the total pressure loss, which has an adverse impact on the effectiveness of jet impingement and film cooling. Therefore, turbine internal passage design needs to balance heat transfer enhancement and pressure loss reduction, and aerothermal design is required [1].

Owing to the complex geometry in the serpentine cooling passages with full rib configurations, traditional aerothermal designs have relied heavily on experiments [1]. More recently, however, computational fluid dynamics (CFD) has become a powerful tool to simulate the flow and heat transfer characteristics and provide useful design insights. To further increase the CFD as a design tool, we use an optimization algorithm to automate the design process. A recent study of turbomachinery aerodynamics demonstrated that the CFD-based design optimization approach reduced the duration of design process by 85%, while achieving a same quality compared with a sophisticated human-supervised design tool [2]. Given this advantage, CFD-based design optimization is becoming increasingly popular.

There are two main classes of design optimization methods: gradient-free and gradient-based methods. A number of researchers have optimized turbine internal cooling designs using the gradient-free methods such as artificial neural network, genetic algorithms, and design of experiments [3-6]. However, one of the issues with these gradient-free methods is that their computational cost scales exponentially with the number of design variables [7]. Therefore, the above studies only parameterized the cooling configuration using a small number of design variables (ranging from 3 to 26). Given the complex geometry of a serpentine, rib-roughened cooling passage, it is preferable to parameterize the passage using a large number of design variables to achieve the maximum possible performance. We address this limitation by using a gradient-based optimization method together with efficient gradient computation techniques. Instead of evaluating a large number of CFD samples, the gradient-based method starts from the baseline design and utilizes the gradient information to find the most promising direction in the design variable space for improvement.

To efficiently compute the derivatives, we utilize the adjoint method whose computational cost is independent of the
number of design variables \[8, 9\].

This combination of gradient-based optimization with adjoint derivative computation has been used in aerodynamic designs for aircraft \[10–15\], cars \[16, 17\], and turbomachinery \[18, 19\], as well as multidisciplinary design optimization \[14, 20\] such as aerostructural \[21–23\] and hydrostructural \[24\] optimization. The adjoint method has also been utilized to optimize the heat transfer performance for turbine blades and fin heat exchangers \[25–26\]. In the context of turbine internal cooling passage design, the adjoint method has not been widely used. We believe this is primarily because the flow structure in the serpentine rib-roughened passages is highly complex, as mentioned above. As a result, the flow converges poorly and the accuracy of adjoint derivatives is degraded. Moreover, when a strong flow separation is present, the adjoint fails to converge and the optimization aborts. Adjoint optimization under poor flow convergence conditions is known to be a challenging issue \[16\], and advanced flow and adjoint stabilization technique is needed.

Despite the above challenge, we believe that the adjoint method is the only hope for the aerothermal optimization of serpentine cooling passages with full rib configurations, where hundreds of design variables are needed. As a first step in this direction, we apply the adjoint method to perform the shape optimization of an internal cooling passage designs. There are two types of adjoint implementations: continuous and discrete \[27\]. We opt to use the discrete adjoint approach because the adjoint derivative is consistent with the flow solution. Moreover, the discrete adjoint implementation is easier to maintain and extend (for example, when adding new objective or constraint functions and boundary conditions). We conduct aerothermal optimizations for a U-bend duct that is representative of a simplified rib-free turbine internal cooling passage. We demonstrate the capability of our adjoint optimization framework by using 135 design variables to parameterize the U-bend duct. Moreover, we evaluate the benefit of the adjoint method by comparing our optimization results with those generated by the gradient-free methods.

The reminder of this paper is organized as follows. In Section II, we introduce the optimization framework and its components. The aerothermal optimization results for the U-bend duct are described in Section III. We end with our conclusions in Section IV.

2 Method

In a previous study, we implemented an aerodynamic design optimization framework using a discrete adjoint method with OpenFOAM \[17\]. In this study, we extend this framework for aerothermal optimization. The central element in our design optimization framework is a discrete adjoint solver for computing the total derivative \(d f / d x\), where \(f\) is the function of interest (objective and constraint functions, e.g., Nusselt number, total pressure loss), and \(x\) is the design variable vector that controls the geometric shape via free-form deformation (FFD) control point movements \[28\].

Our framework also depends on the following external libraries and modules: open field operation and manipulation (OpenFOAM) \[29, 30\], portable, extensible toolkit for scientific computation (PETSc) \[31, 32\], pyGeo \[28\], pyWarp \[28\], and pyOptSparse \[33\]. In this section, we briefly introduce the overall adjoint optimization framework, the flow solution technique, and the adjoint derivative computation method. The detailed theoretical background for the other modules was reported in our previous work \[17\].

1 Discrete Adjoint Optimization Framework

The modules and data flow for the optimization framework are shown in Fig. 1 where we use the extended design structure matrix (XDSM) standard developed by Lambe and Martins \[34\]. The diagonal entries are the modules in the optimization process, while the off-diagonal entries are the data. Each module takes data input from the vertical direction and output data in the horizontal direction. The thick gray lines and the thin black lines denote the data and process flows, respectively. The numbers in the entries represent their execution order.

The framework consists of two layers: OpenFOAM and Python. The OpenFOAM layer consists of a flow solver (buoyantBoussinesqSimpleFoam), an adjoint solver (discreteAdjointSolver), and a graph coloring solver (coloringSolver). The flow solver is based on the OpenFOAM built-in solver buoyantBoussinesqSimpleFoam for steady incompressible turbulent flow with heat transfer. The adjoint solver computes the total derivative \(d f / d x\) based on the flow solution generated by buoyantBoussinesqSimpleFoam. To accelerate the partial derivative computation in the adjoint solver, we developed a parallel graph coloring solver \[17\].

The Python layer is a high-level interface that takes the user input, as well as the total derivatives computed by the OpenFOAM layer, and calls multiple external modules for optimization. More specifically, these external modules are: “pyGeo” for geometric parameterization, “pyWarp” for volume mesh deformation, and “pyOptSparse” for optimization problem setup. A brief summary of these three modules follows.
Graph Coloring

Figure 1. Extended design structure matrix [34] for the discrete adjoint framework for constrained shape optimization problems. $x$: design variables; $x^{(0)}$: baseline design variables; $x^{(∗)}$: optimized design variables; $x_S$: design surface coordinates; $x_V$: volume mesh coordinates; $w$: state variables; $c$: geometric constraints; $f$: objective and constraint functions.

Figure 2. FFD control points for the U-bend duct. We move only the red points, while the black points remain unchanged during optimization.

**pyGeo:** a surface geometry parameterization module that implements the FFD approach [28]. This approach embeds the geometry into a volume such that the geometry is manipulated by moving points at the surface of that volume (the FFD points). The FFD volume is a tri-variate B-spline volume such that the gradient of any point inside the volume can be easily computed. An example of FFD control points is shown in Fig. 2.

**pyWarp:** a volume mesh deformation module using the analytical inverse distance algorithm [35]. The advantage of this approach is that it is highly flexible and is applicable for both structured and unstructured meshes. In addition, compared with the radial basis function based method, this approach does a better job of preserving mesh orthogonality in the boundary layer.

**pyOptSparse:** an open source, object-oriented Python module, extended from pyOpt [33], for formulating and solving constrained nonlinear optimization problems. pyOptSparse provides a high-level API for defining the design
variables, and the objective and constraints functions. It also provides interfaces for several optimization packages, including some open-source packages. In this study, we choose SNOPT \cite{36} as the optimizer. SNOPT implements a sequential quadratic programming algorithm to solve the constrained nonlinear optimization problem and uses the quasi-Newton method to solve the quadratic subproblem, where the Hessian of the Lagrangian is approximated by using a Broyden–Fletcher–Goldfarb–Shanno update.

2 Flow Solution

In this paper, we simulate three-dimensional steady turbulent flow with heat transfer using the OpenFoam built-in solver buoyantBoussinesqSimpleFoam. The flow is governed by the Navier–Stokes equations and the temperature equation

\[
\nabla \cdot U = 0, \quad (2.1)
\]
\[
\nabla \cdot (UU) + \nabla p - \nabla \cdot (\nu + \nu_t)(\nabla U + \nabla U^T) = 0, \quad (2.2)
\]
\[
\nabla \cdot (TU - \nabla \cdot (\alpha + \alpha_t)\nabla T) = 0, \quad (2.3)
\]

where \( p \) is the pressure, \( T \) is the temperature, \( U = [u, v, w] \) is the velocity vector representing components in the \( x, y, \) and \( z \) directions, respectively, \( \nu \) and \( \nu_t \) are the molecular and turbulent viscosity, respectively, and \( \alpha \) and \( \alpha_t \) are the molecular and turbulent thermal diffusivity, respectively. The viscosity and thermal diffusivity are connected through \( Pr = \frac{\nu}{\alpha} = 0.7 \), and \( Pr_t = \frac{\nu_t}{\alpha_t} = 0.85 \). We ignore any body forces and external or internal heat sources. To connect the turbulent viscosity to the mean flow variables, we use the \( k - \omega \) SST turbulence model. We include all the turbulence variables in our adjoint implementation; we do not assume “frozen turbulence” as is typically done in continuous adjoint implementations \cite{37}.

We use the finite-volume method to discretize the above governing equations. To be more specific, we use the semi-implicit method for pressure-linked equations (SIMPLE) algorithm \cite{38} to solve Eqs. (2.1) to (2.3) in a segregated manner. We first solve the momentum and temperature equations based on the old \( p \) and \( \phi \) fields, where \( \phi \) is the surface flux. Then \( p \) is updated by solving a pressure Poisson equation, followed by an update for \( \phi \) using the Rhie–Chow interpolation scheme \cite{39}. Based on the new \( p \) and \( \phi \), we then update \( U \) such that it satisfies both mass and momentum equations.

3 Discrete Adjoint Derivative Computation

As mentioned above, we need to compute the total derivative \( \frac{df}{dx} \) to perform gradient-based shape optimization, where \( f \) depends not only on the design variables, but also on the state variables that are determined through the solution of governing equations. Thus,

\[
f = f(x, w), \quad (3.1)
\]

where \( w \) is the vector of state variables. Applying the chain rule for the total derivative, we obtain

\[
\frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial w} \frac{dw}{dx} \quad (3.2)
\]

A naive computation of \( \frac{dw}{dx} \) via finite-differences requires solving the governing equations \( n_x \) times, where \( n_x \) is the number of design variables. This operation is computationally expensive for a large number of design variables. We avoid this issue by using the fact that the derivatives of the residuals with respect to the design variables must be zero for the governing equations to remain feasible with respect to variations in the design variables. Thus, applying the chain rule to the residuals, we can write

\[
\frac{dR}{dx} = \frac{\partial R}{\partial x} + \frac{\partial R}{\partial w} \frac{dw}{dx} = 0, \quad (3.3)
\]

where \( R \) is the vector of flow residuals. Substituting Eq. (3.3) into Eq. (3.2) and canceling out the \( \frac{dw}{dx} \) term, we get

\[
\frac{df}{dx} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial w} \left( \frac{\partial R}{\partial w} \right)^{-1} \frac{\partial R}{\partial x} \quad (3.4)
\]

Considering the combination of the \( \frac{\partial R}{\partial w} \) and \( \frac{\partial f}{\partial w} \) terms in Eq. (3.4), we solve a linear equation

\[
\frac{\partial R^T}{\partial w} \psi = \frac{\partial f^T}{\partial w} \quad (3.5)
\]
to obtain the adjoint vector \( \psi \). Then, this adjoint vector is substituted into Eq. (3.4) to compute the total derivative

\[
\frac{df}{dx} = \frac{\partial f}{\partial x} - \psi^T \frac{\partial R}{\partial x}.
\]

(3.6)

Since the design variable \( x \) does not explicitly appear in Eq. (3.5), we need to solve Eq. (3.5) only once for each function of interest, and thus the computational cost is independent of the number of design variables. This is an advantage for three-dimensional shape design optimization problems, since the number of functions of interest is usually less than 10 but the number of design variables can be a few hundred.

A successful implementation of adjoint-based derivative computation requires an efficient and accurate computation for the partial derivatives \( \frac{\partial R}{\partial w}, \frac{\partial R}{\partial x}, \frac{\partial f}{\partial w}, \text{and} \frac{\partial f}{\partial x} \) in Eqs. (3.5) and (3.6). In this paper, we use the finite difference to compute the partial derivatives. To accelerate the finite-difference based partial derivative computation, we utilize a heuristic parallel graph coloring algorithm [17]. We partition all the columns of the Jacobians into different structurally orthogonal subgroups (colors), such that, in one structurally orthogonal subgroup, no two columns have a nonzero entry in a common row. With this treatment, we simultaneously perturb multiple columns that have the same colors. The number of residual evaluation is reduced to \( O(1000) \), independent of the mesh size and number of CPU cores [17].

As mentioned above, in this paper, we extend our adjoint-based aerodynamic optimization framework [17] for aerothermal optimization. Similar to our previous aerostructural adjoint implementation [21], we formulate the aerothermal adjoint system using the multidisciplinary feasible (MDF) approach [14]. To be more specific, we first solve an aerothermal problem using the segregated method showed in Section 2. Based on the converged flow, we assemble the coupled adjoint system including all state variables, i.e., \( w=[U, p, T, k, \omega, \phi] \). We then solve the coupled adjoint system using the generalized minimal residual method. Compared with other multidisciplinary design optimization architectures such as individual discipline feasible (IDF) and simultaneous analysis and design (SAND), the benefit of using the MDF approach is that the multidisciplinary system has a minimal number of control variables (i.e., design variables, objective and constraint functions) for the optimizer. Moreover, it always returns a system that satisfies the governing equations and constraints, even the optimization terminates early [14].

3 Results and Discussion

In this section, we first validate our flow solver by comparing our simulation results with experiments. We then perform a series of optimizations for different objectives and discuss the optimization results.

1 Validation

As mentioned in the introduction, we use the U-bend duct as our benchmark case, whose geometry is shown in Fig. 2. Since the duct is symmetric with respect to the \( z=0 \) plane, we simulate only half of the geometry. We generate a
Figure 4. The simulation results qualitatively agree with the experiments [40]. However, the velocity is underestimated at the symmetry plane, and $N_u$ is overestimated at the upper and outer surfaces.
structured mesh with 0.25 million cells, as shown in Fig. 3. The average $y^+$ is 2.0. Following Coletti et al. [40], we set the inlet velocity to 8.4 m/s with a turbulence intensity of 5%. The Reynolds number is $4.2 \times 10^4$ based on the hydraulic radius ($D_h=0.075$ m). The inlet temperature is 293.15 K, while the wall temperature is fixed at 303.15 K with a 10 K temperature difference driving the heat transfer. The symmetric boundary condition is imposed at the $z=0$ plane. The turbulence model is $k-\omega$ SST. As mentioned in the introduction, large regions of separated flow are typically present in the U-bend duct. Therefore, to ensure a good flow and adjoint convergence during the optimization process, we use the first-order upwind scheme to discretize the inviscid terms, and central differencing for the viscous terms. From a CFD perspective, the first-order scheme introduces strong numerical dissipation and is not recommended. However, from an optimization standpoint, it is hopeful that the adjoint derivatives based on the first-order scheme qualitatively capture the correct trend for improvement. Adjoint optimization using the first-order discretization scheme has been used in previous studies where large flow separation is present [41, 42].

Next, we validate our CFD solver by comparing the simulated velocity field with the experimental data of Coletti et al. [40], as shown in Fig. 4. Note that our definition of $z/D_h$ differs from Coletti et al. [40] due to the use of the symmetry plane. The velocity distribution at the symmetric ($z/D_h=0$) and upper ($z/D_h=0.94$) planes qualitatively agree with the experiments. However, the flow separation and velocity magnitude are underestimated, especially at the symmetry plane.

Next, we compare the distribution of normalized Nusselt number ($Nu/Nu_0$). $Nu$ is defined as

$$Nu = \frac{QD_h}{(T_w - T_B)k},$$

where $Q$ is the heat flux, $k$ is the thermal conductivity, and $T_w$ and $T_B$ are the wall temperature and bulk temperature, respectively. To compute the bulk temperature, we assume a linear increase in mainstream temperature from the inlet to outlet. We then linearly interpolate $T_B$ based on the normalized streamwise location. The reference Nusselt number $Nu_0$ is computed based on the Dittus–Boelter correlation ($Nu_0 = 0.023Re^{0.8}Pr^{0.4}$). As shown in Fig. 4, the distribution of simulated Nusselt number qualitatively agrees with the experiments. We observe high $Nu$ in the downstream region of U-bend, which is due to the flow separation and enhanced turbulence mixing. However, the simulation overestimates the magnitude of $Nu$ at both top and outer walls.

## 2 Aerothermal Optimization

To consider both aerodynamics and heat transfer in the U-bend duct design, we define an objective function using a weighted sum of $Nu$ and $C_{PL}$,

$$f = w_1 C_{PL} - w_2 Nu,$$

where $w_1$ and $w_2$ are weights ($w_1 + w_2 = 1$), and $C_{PL}$ is defined as

$$C_{PL} = \frac{p_0^{in} - p_0^{out}}{0.5 \rho U_0^2}.$$  

Here, $p_0^{in}$ and $p_0^{out}$ are the total pressure at the inlet and outlet, respectively, $\rho$ is the density, and $U_0$ is the inlet velocity. In Eq. (2.1), we use relative values for $Nu$ and $C_{PL}$; $Nu$ and $C_{PL}$ are normalized with respect to the baseline design values.

We set 63 FFD control points to manipulate the shape of U-bend, as shown in Fig. 2. Only the red FFD points are selected as the design variables, while the black FFD points remain unchanged during the optimization process. The FFD points on the upper surface move in the $z$ direction, while the points on the inner and outer surfaces move in the $x$ and $y$ directions. In total, we have 135 design variables. To ensure a smooth geometric transition at the symmetry plane, we enforce a zero-slope constraint for the first and second layers of FFD points in the $z$ direction. The optimization problem is summarized in Table 1.

To construct a Pareto front for $Nu$ and $C_{PL}$, we run optimizations using six combinations of weights, as shown in Table 2. From Opt0 to Opt5, we gradually increase the weights for $C_{PL}$. Opt5 is equivalent to a min-$C_{PL}$ case. For the max-$Nu$ case (Opt0), if we set $w_1=0$, the optimizer tends to generate large flow separation to enhance heat transfer. As a result, the flow convergence is poor and the adjoint fails to converge. Therefore, we give a 2% weight for $C_{PL}$. This allows the optimizer to maximize $Nu$ while maintaining the flow separation at a level such that the optimization can proceed.

We run all the optimizations on TACC Stampede2 using SKX nodes. The SKX nodes are equipped with Intel Xeon Platinum 8160 CPU running at 2.1 GHz. For each optimization, we use 96 CPU cores with 4 SKX nodes. The Opt0 to Opt5 cases converge in 41, 46, 30, 25, 16, and 24 iterations. On average, each iteration takes 300 s. The
Table 1. Optimization configuration for the U-bend duct. We use 135 design variables to parameterize the duct geometry.

<table>
<thead>
<tr>
<th>Function or variable</th>
<th>Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimize ( f = w_1 C_{PL} - w_2 Nu )</td>
<td>weighted ( Nu ) and ( C_{PL} )</td>
<td></td>
</tr>
<tr>
<td>with respect to ( \Delta x )</td>
<td>displacement of FFD points</td>
<td>135</td>
</tr>
<tr>
<td>subject to ( y_u^{\text{symm}} = 0 )</td>
<td>( y )-direction slope at symmetry plane is zero</td>
<td>18</td>
</tr>
</tbody>
</table>

\(-0.05 \text{ m} < \Delta x < 0.05 \text{ m}\)
design-variable bounds

Table 2. Weights for multi-objective optimization.

<table>
<thead>
<tr>
<th></th>
<th>Opt0</th>
<th>Opt1</th>
<th>Opt2</th>
<th>Opt3</th>
<th>Opt4</th>
<th>Opt5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1 )</td>
<td>0.02</td>
<td>0.05</td>
<td>0.10</td>
<td>0.20</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>( w_2 )</td>
<td>0.98</td>
<td>0.95</td>
<td>0.90</td>
<td>0.80</td>
<td>0.50</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 5. Pareto front for \( Nu \) and \( C_{PL} \). For the min-\( C_{PL} \) case (Opt5), we obtain 52.1% \( C_{PL} \) reduction, while for the max-\( Nu \) case, we increase \( Nu \) by 40.3%.

optimization requires more iterations to converge if a higher weight is given to \( Nu \). This is expected, as the optimizer tends to create flow separation to enhance heat transfer. As a result, the accuracy of adjoint derivatives degrades. This eventually increases the number of optimization iterations. We observed a similar behavior in our previous study [17].

Figure 5 shows the Pareto front generated from the multi-objective optimization runs. For the Opt5 case, we obtain 52.1% \( C_{PL} \) reduction. Our adjoint-based \( C_{PL} \) reduction is much higher than the value (36%) reported by Verstraete et al. [43], who used a gradient-free method (combined ANN and kriging). This extra \( C_{PL} \) reduction is primarily attributed to the use of a large number of design variables. Since the turbulence mixing is suppressed in the duct, \( Nu \) decreases by 23.8%. When increasing \( w_2 \) to 0.98 (Opt0), we obtain a 40.3% increase in \( Nu \). Again, we obtain much more \( Nu \) increase, compared with the value (16.9%) reported by Verstraete et al. [4]. However, this design is not favorable, since it generates excessive flow separation. As a result, \( C_{PL} \) increases by 438.5%. For the Opt3 case, we obtain a 3.6% increase in \( Nu \) and a 20.6% reduction in \( C_{PL} \); a favorable design for both aerodynamics and heat transfer.

The velocity distribution is closely correlated with the pressure loss and heat transfer. As the flow accelerates, the friction loss increases. In the meantime, the increased velocity also increases the convective heat transfer. With this in mind, we now compare the baseline and optimized shapes for Opt0, Opt3, and Opt5, as shown in Fig. 6. The corresponding velocity contour and streamline at the symmetry planes are shown in Fig. 7. For Opt5, the U-bend expands, and the velocity decreases. Therefore, the friction loss is reduced. On the other hand, for Opt0, the duct shrinks before the U-bend section, forcing the flow to accelerate. This also generates a large separation region in the
Figure 6. Comparison between the baseline and optimized shapes. For the min-$C_{PL}$ case (Opt5), the bend section expands, while for the max-$Nu$ case (Opt0), the duct shrinks before the bend section.

Figure 7. Velocity contour and streamline at the symmetry planes. For the min-$C_{PL}$ case (Opt5), velocity decreases due to the duct expansion, while for the max-$Nu$ case (Opt0), velocity increases and a large separation is found downstream of the bend section.
Figure 8. Comparison of $Nu$ between the baseline and optimized shapes. For the min-$C_{PL}$ case (Opt5), $Nu$ decreases due to the reduced velocity in the duct, while for the max-$Nu$ case (Opt0), $Nu$ is large on the top and outer surfaces due to the increased velocity.

downstream of U-bend, as shown in Fig. 7. For Opt3, the U-bend also expands, which reduces the flow separation compared with the baseline case. However, the velocity magnitude is maintained.

Figures 8 and 9 show the $Nu$ contour on the duct surfaces. The average streamwise $Nu$ is plotted in Fig. 10 for reference. As shown in Fig. 10, $Nu$ increases in the U-bend section and then gradually decreases for all configurations. For Opt0, due to the flow acceleration, $Nu/Nu_0$ increases up to 4.2 in the U-bend section. The increased $Nu$ is most evident on the top and outer walls in the downstream region of U-bend (Figs. 8 and 9). For Opt5, $Nu$ decreases compared with the baseline case at all streamwise locations. As discussed before, this is because the duct expands and the velocity reduces. The streamwise $Nu$ distribution for Opt3 is similar to the baseline design, with an average $Nu$ increment of 3.6%.

4 Conclusion

In this paper, we develop the capability to perform aerothermal optimization for internal cooling passages using a gradient-based optimization algorithm along with the discrete adjoint method for computing derivatives. Compared with approaches based on gradient-free methods, our approach can handle a much larger number of design variables, and thus gives us more design freedom for geometric modification. We run multi-objective optimizations for a U-bend duct representative of a simplified rib-free turbine internal cooling passage. We consider both heat transfer enhancement and pressure loss reduction, and explore different combinations of weights for these two objectives during the optimizations, varying from max-$Nu$ to min-$C_{PL}$. The duct geometry is parameterized using 135 design variables. We demonstrate our optimization capability by comparing our optimization results with the ones based on the gradient-free methods. We obtain 52.1% $C_{PL}$ reduction for the min-$C_{PL}$ case, which is higher than the previously reported value (36%), where a gradient-free optimization method was used. For the max-$Nu$ case, we achieve a 40.3% increase in $Nu$. This value is also higher than the previously reported value (16.9%). The above results highlight that
Figure 9. Same as Fig. 8 from a different view.

Figure 10. Average $N_u$ along the streamwise location. For the max-$N_u$ case (Opt0), $N_u/N_{u0}$ increases to 4.2 in the duct section.
our adjoint optimization framework is efficient and effective. Our optimization framework has the potential to be a useful tool for aerothermal designs of turbine internal cooling passages with full rib configurations.

Acknowledgments

The computations were done in the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation Grant No. ACI-1548562.

References


