Gradient Based Optimization of a Sample Z-Flow Cooling System Using Multiple Processors

G. Quinn & G. Venter
Vanderplaats Research and Development, Inc.
Colorado Springs, CO, USA

Abstract

This paper presents and describes a scheme to optimize the cooling efficiency of a contrived two-dimensional Z-flow cooling system. The FLUENT CFD code is used to analyze the steady state flow characteristics of variable flow-hole geometries between the "block" and the "head" portion of the fluid model. The VisualDOC general-purpose optimization code is used to determine the optimum geometry that satisfies minimum fluid velocity constraints at ten fluid locations. A gradient-based optimization algorithm is chosen to investigate its performance as a candidate scheme for CFD problems with larger numbers of design variables. Because optimization algorithms require numerous analyses VisualDOC is used to solve this problem utilizing a distributed[1] heterogeneous[2] system of from one to four CPU's.

Problem Description

Vanderplaats Research and Development (VR&D) has worked closely with FLUENT technical specialists to incorporate geometric design optimization with the two-dimensional Z-flow cooling system shown in Figure 1.

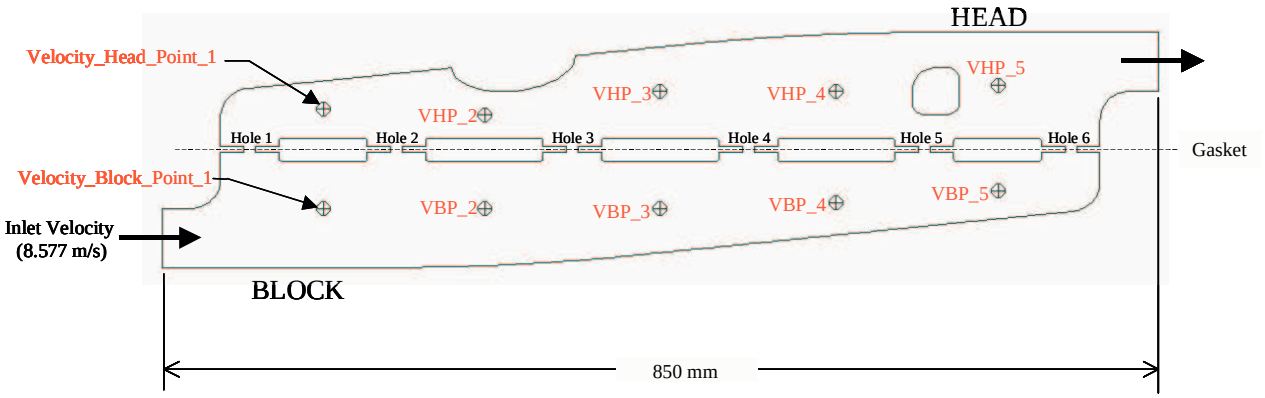


Figure 1 – Z-Flow Cooling System

[1] Distributed computing systems are characterized by the various CPU's residing on different platforms. This is in contrast to parallel computing systems with multiple CPU's residing on the same platform.
[2] Heterogeneous computing systems are distributed systems comprised of different types of platforms (e.g. different hardware and/or operating systems).

Coolant enters the block portion of the model and exits the head as shown by the arrows. The cooling jacket hole geometry in the gasket may be modified to produce an optimum flow condition as calculated by the FLUENT CFD program. Ten locations are chosen to monitor the coolant velocity response. Design constraints are defined such that all response locations must satisfy a minimum velocity requirement of 2.0m/s. The design objective is to maximize the fluid velocity averaged at the ten locations by varying each hole diameter and location (offset from center).

Fluid Meshing

GAMBIT is used to mesh the fluid in the Z-Flow cooling model. Hole zone regions surrounding each hole are re-meshed each time VisualDOC performs an analysis call.

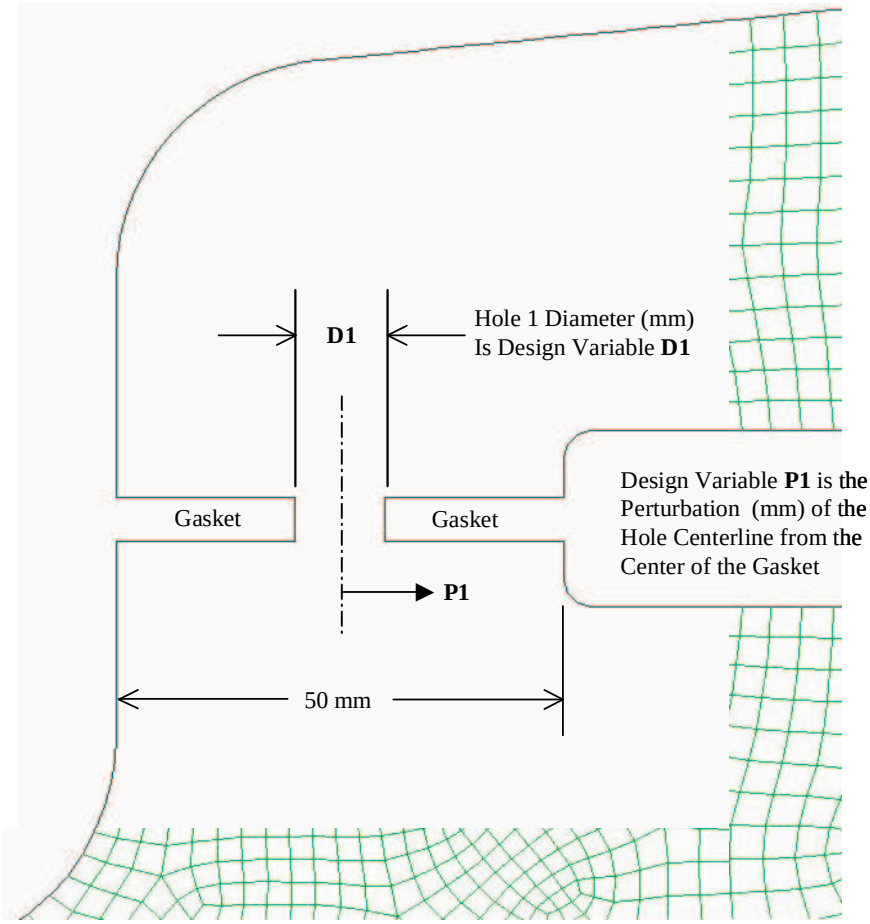


Figure 2 – Design Variable Definition

VisualDOC automatically updates a parameterized gambit journal file to reflect the design variable changes. Figure 2 shows how D1 and P1 are used to adjust the geometry of each hole, which in turn affects the adjacent fluid mesh.

GAMBIT re-meshes the local region surrounding each hole based on twelve updated design variables D1-D6 and P1-P6. The new mesh is then combined with a pre-existing

mesh file of the rest of the model using TMERGE. Figure 3 shows how GAMBIT and TMERGE create an updated mesh file that is ready for FLUENT analysis. Figure 4 shows a close-up of the hole-zone re-mesh along with the adjacent block and head mesh.

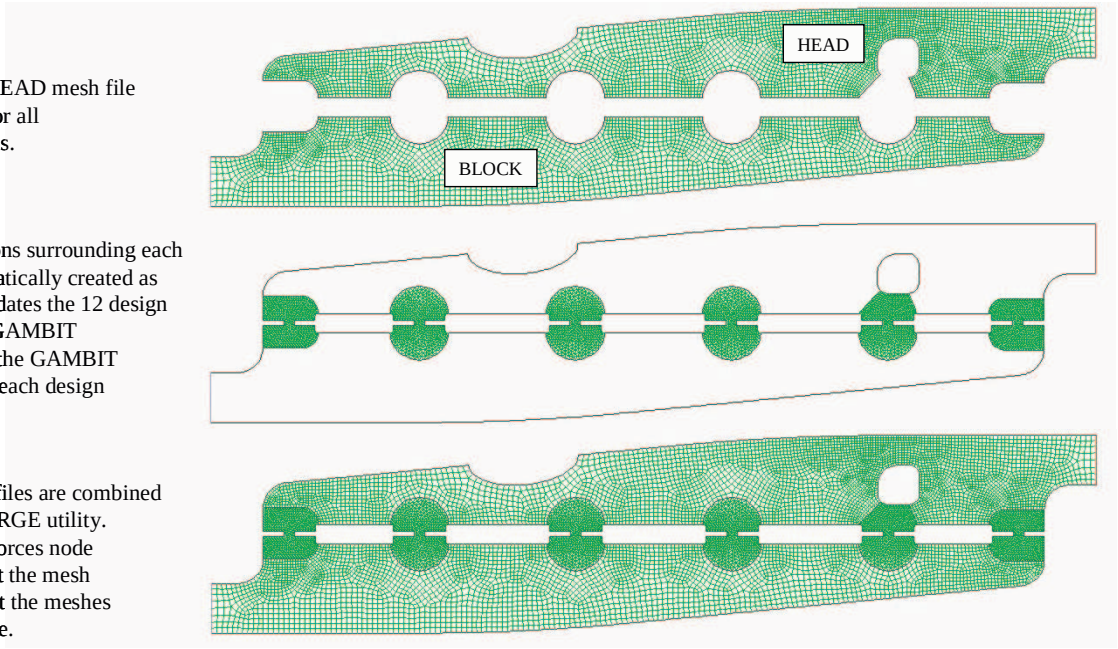


Figure 3 - Meshing Scheme Used for Optimization Iterations

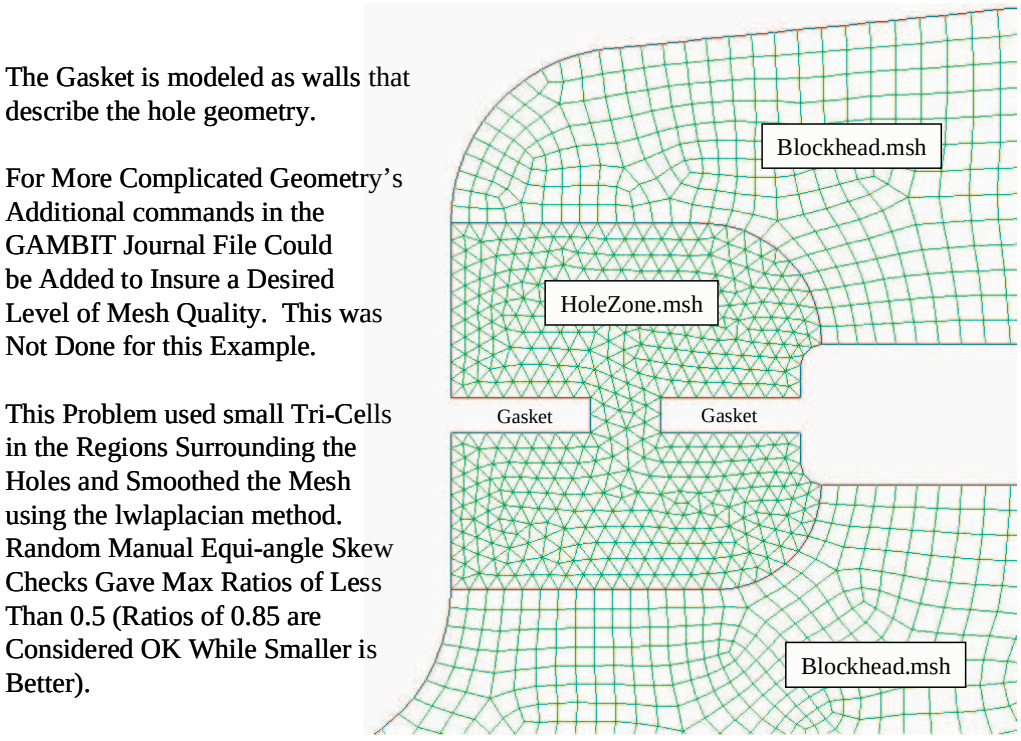


Figure 4 - Close-Up of Hole 1 Mesh

FLUENT CFD Analysis

VisualDOC automatically executes FLUENT using a journal file that reads in the updated Z-Flow mesh file. An interpolation file containing the FLUENT solution for the initial configuration is used to initialize the FLUENT solver to speed up solution convergence. Both first and second order upwind schemes for discretization of field variables are run as well as two passes of velocity gradient mesh adaptation. This is shown graphically below in Figure 5.

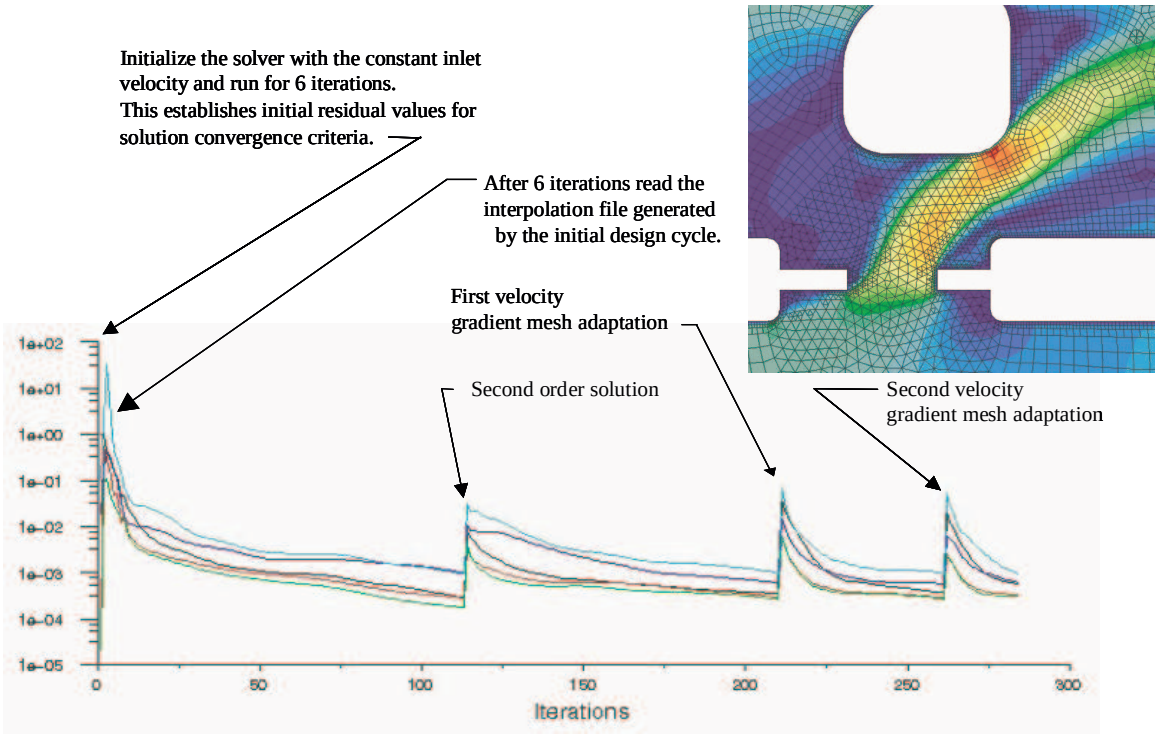


Figure 5 - FLUENT Residuals versus Solver Iterations

The goal of the VisualDOC optimization is to ensure that the coolant velocity at ten locations all have velocities of at least 2.0m/s. These are design constraints and as such, their response values must be passed to the optimizer. For this problem, the velocity response is defined as FLUENT point-surface entities (FLUENT => Surface => Point). The point-surface response is exported as a Tecplot formatted file from which VisualScript may extract the FLUENT velocities.

```
TITLE = ”Sample Tecplot File"
VARIABLES  = X, Y, "Velocity Magnitude"
ZONE T="block_point_1", N=1, E=1, ET=TRIANGLE, F=FEBLOCK

  8.80000e-02

 -5.00000e-02

  3.62145e+00
1
```

VisualScript Job Control

VisualScript is a stand-alone program provided by VR&D to allow easy coupling between VisualDOC and one or more existing analysis programs, as required to perform design optimization. The control and execution for this problem is graphically shown in Figure 6.

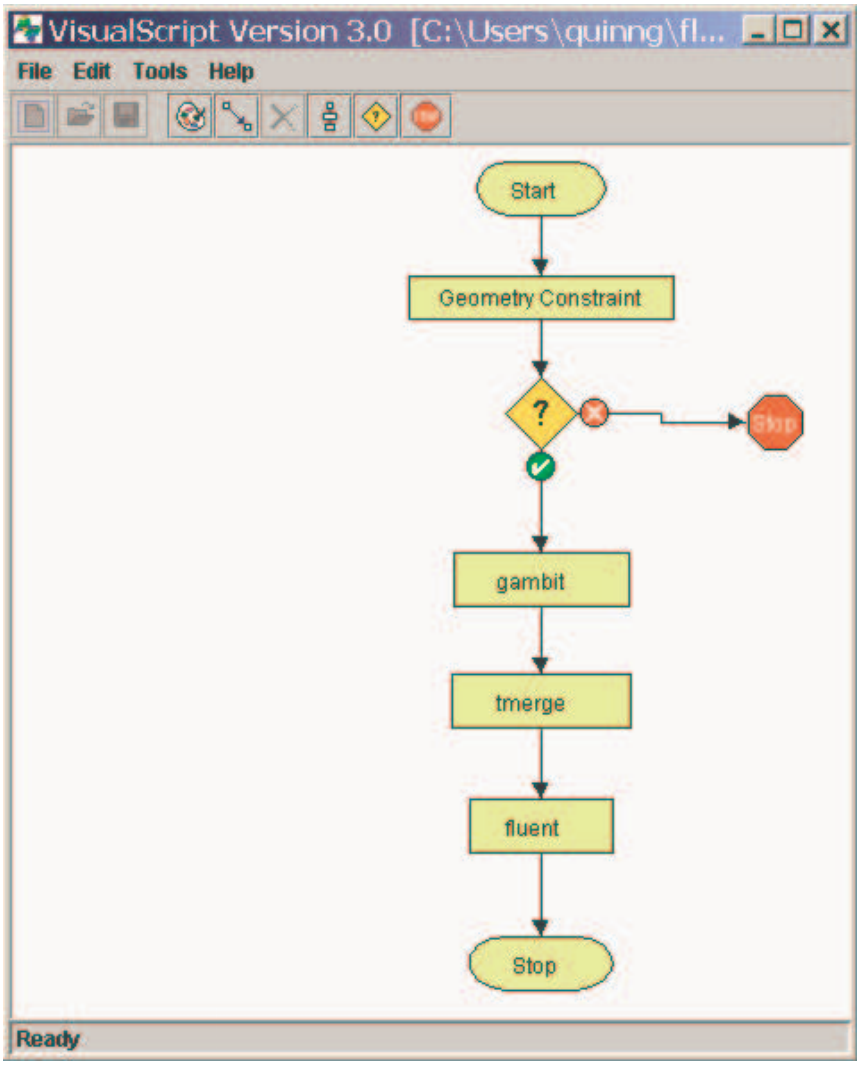
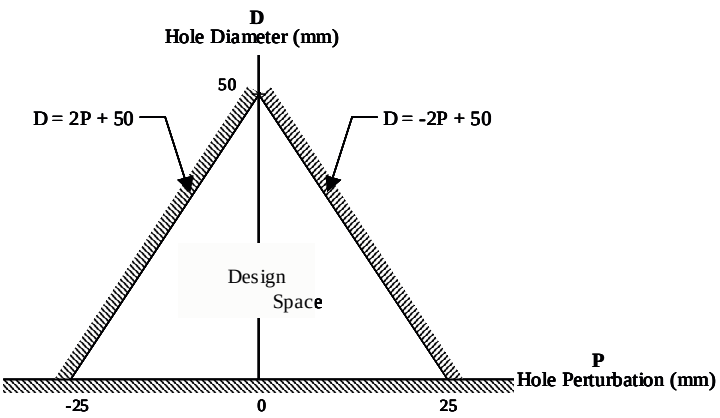


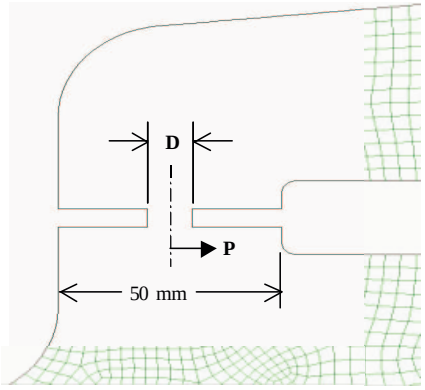
Figure 6 - VisualScript Flow Diagram

As previously described, the parameterized GAMBIT journal file is modified with perturbed VisualDOC design variables (e.g. D1-D6, & P1-P6). TMERGE combines the GAMBIT re-meshed hole zones with the unchanged portion of the fluid mesh. The FLUENT journal file execution in this flow exports the velocity response file.

The geometry constraint requires explanation that is more detailed. The relationship between each hole diameter and its X-perturbation from the gasket centerline must

constrain improbable geometry's (e.g. a positive perturbation plus the radius of the hole must be less than 25mm). This geometric constraint is shown in Figure 7.





$$D \le -|2P| + 50 \;\Rightarrow\; D + |2P| - 50 \le 0 \quad \text{Geometric Constraint}$$

$$0 \le D \le 50 \quad \text{Design Variable D Bounds}$$

$$-25 \le P \le 25 \quad \text{Design Variable P Bounds}$$

Figure 7 - Geometric Constraint

Normally, design variables that fall between their upper and lower bounds do not pose an operational problem if they violate the geometric constraint above (e.g. D1=40, & P1=20). The resulting design is analyzed and is simply deemed infeasible due to the violated constraint. The optimizer continues with its design variable search strategy to find a feasible design. However, for this particular problem GAMBIT will fail to re-mesh the regions surrounding the holes if the geometric constraint is violated. This is why the "Geometry Constraint" block exists in the flow diagram of Figure 6. It allows VisualScript to exit the script if this constraint is violated, and thus avoid a GAMBIT hard failure. When this happens, VisualDOC revises its search strategy accordingly.

Optimization Results

All optimization results in this report have the following characteristics in common:

```
                Optimization Method : SQP(Sequential Quadratic Programming)
                          Objective : Maximize
                 Constraint Tolerance : -0.03
       Violated Constraint Tolerance :  0.003
             Gradients Calculated By : First Forward Difference
    Relative Finite Difference Step :  0.25
     Minimum Finite Difference Step :  3.00
```

The optimization statement for this problem is: **Maximize the fluid velocity averaged at ten response points by varying each hole diameter and lateral displacement subject to minimum velocity constraints of 2.0 m/s at each response location.**

Figure 8 lists the FLUENT velocity magnitude at ten points that are required to have a minimum coolant velocity. Design cycle 0 shows the results that were calculated using the initial design variable values[3] (which are also shown in the Figure). Note that two points have fluid velocities that are less than 2.0 m/s, which means this initial design is not feasible.

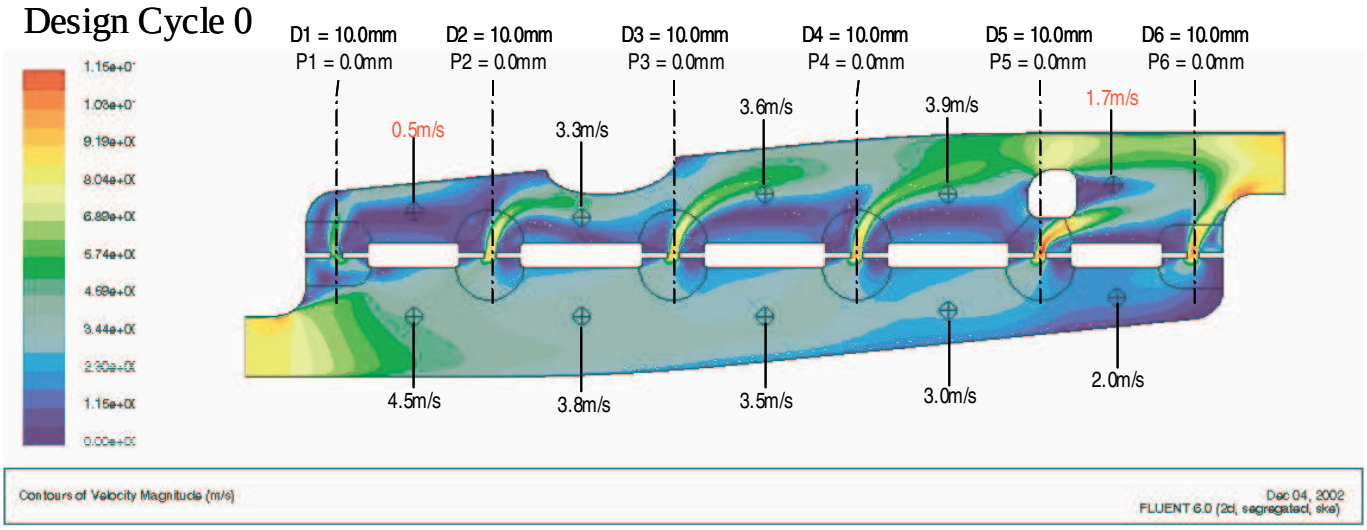


Figure 8 - 3-Proc (Pluto, Neptune, Hyperion) Results; Design cycle 0

Figure 9 shows the optimized design for this same case. The hole diameters and positions have been changed such that the coolant velocity constraints are all satisfied, and their average velocity has been maximized. The optimization took three design cycles with a total of 42 FLUENT analyses. This is a feasible design. The design

[3] The initial solution results are brought into all subsequent FLUENT analyses as an initial conditions interpolation file. This helps to speed up solution convergence.

objective, velocity response, P design variable, and D design variable histories are plotted in Figure 10, Figure 11, Figure 12, and Figure 13 respectively.

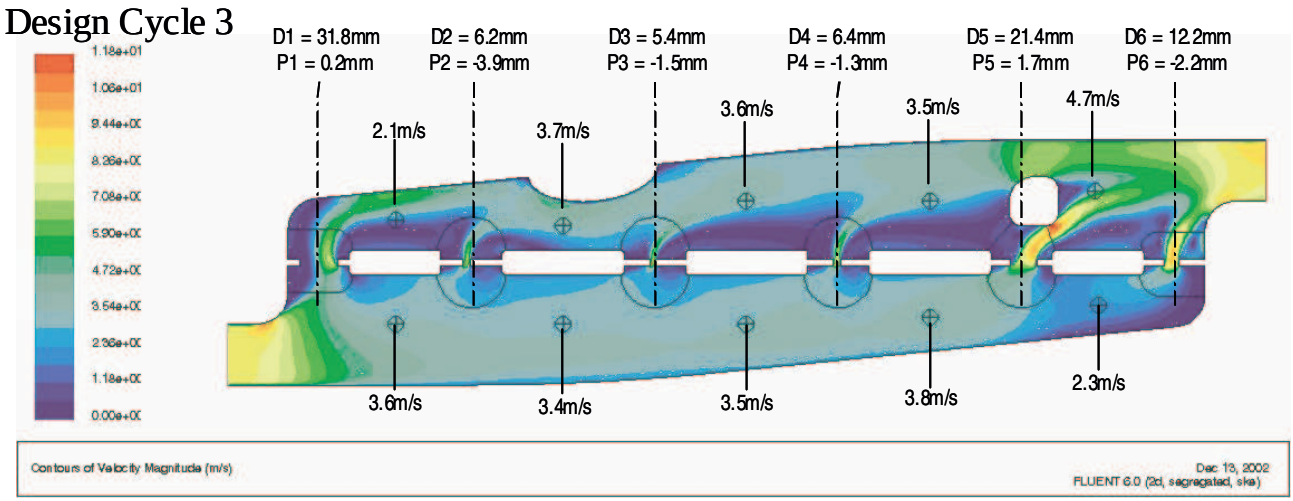


Figure 9 - 3-Proc(Pluto, Neptune, Hyperion) Results Design cycle 3

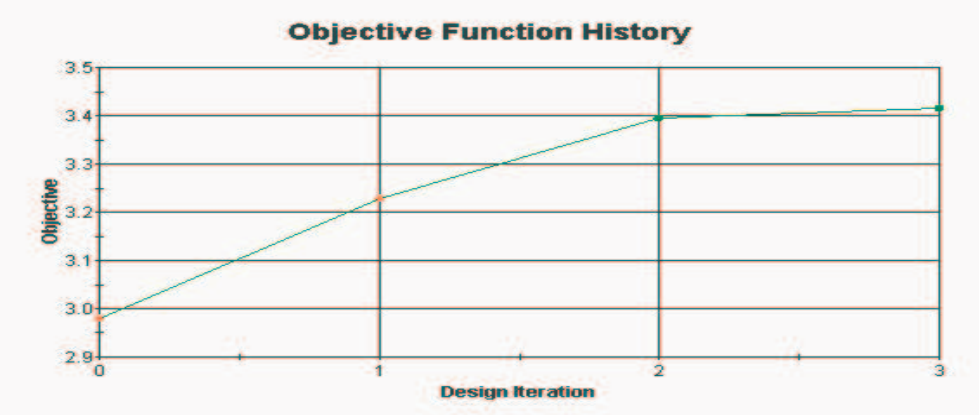


Figure 10 - 3-Proc(P-N-H) Avg. Vel. Objective[4] Function

[4] For the objective function plot VisualDOC displays a red asterisk when one or more design constraints are violated and a green asterisk when all the design constraints are satisfied.

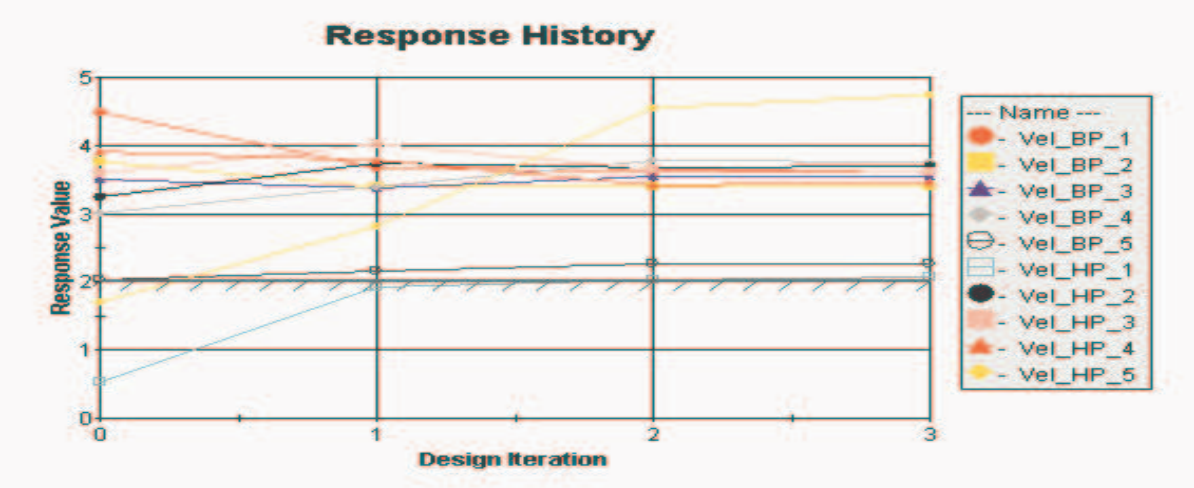


Figure 11 - 3-Proc(P-N-H) Velocity Response

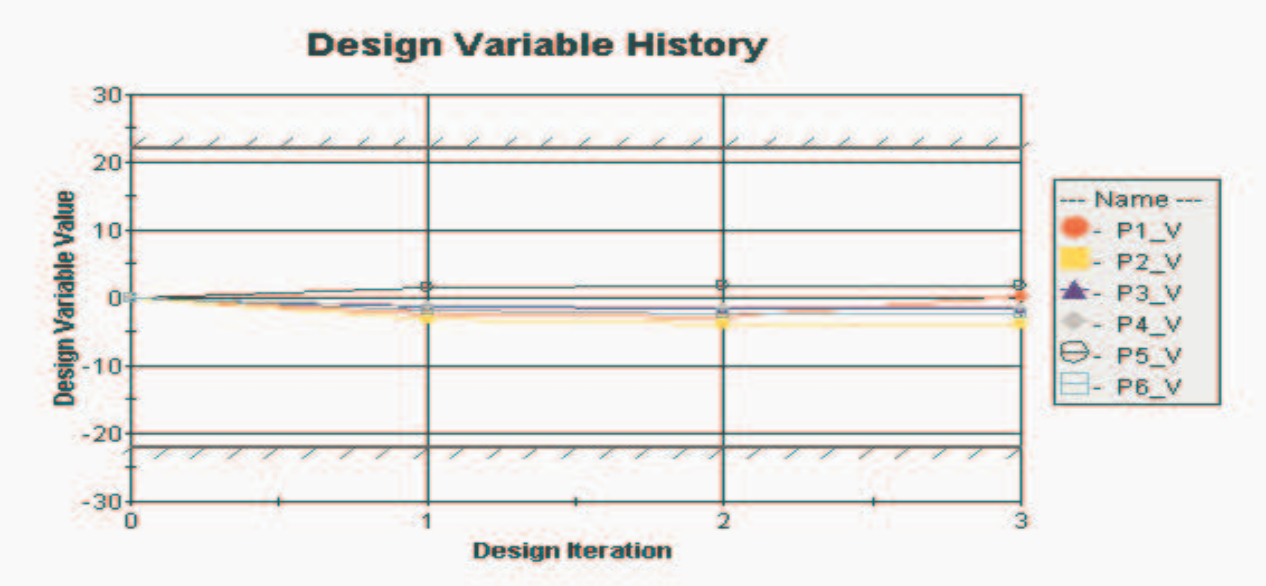


Figure 12 - 3-Proc(P-N-H) Hole Perturbations (mm)

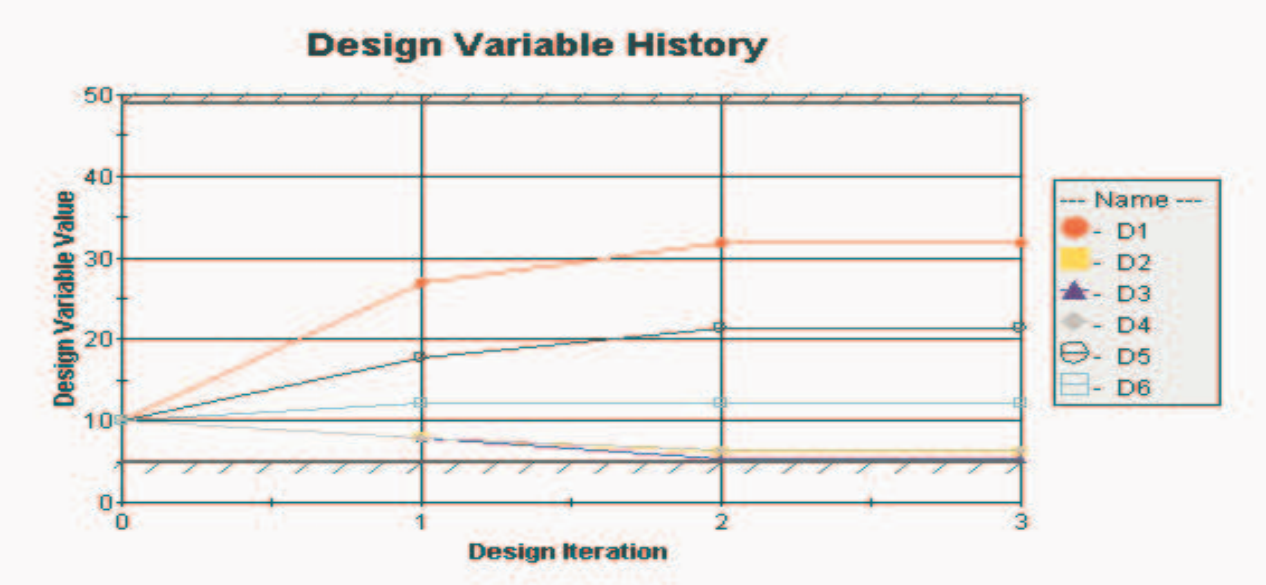


Figure 13 - 3-Proc(P-N-H) Hole Diameters (mm)

This case shows that changes only to the gasket portion of the geometry will achieve the desired flow conditions. Most importantly all ten sensor points in both the head and block show coolant velocities at or above the minimum 2.0m/s. The average velocity design objective of these ten points has risen from 3.0m/s to 3.4m/s as well.

Multi-Processor Implementation

Design optimization of a FLUENT simulation requires multiple analyses. For large CFD models, this will require vast amounts of CPU time. Utilization of the FLUENT parallel solver certainly helps, but since the gradient calculations during optimization require numerous prescribed analyses these discrete jobs may be run in parallel as well to further increase efficiency. Combining these two approaches provides the user with an opportunity to use large numbers of compute nodes, if available. In such a scheme, the finite difference calculations of the optimization algorithm will be parallelized, potentially using as many compute nodes as the number of design variables. However, each one of these compute nodes then becomes a master node from which the FLUENT simulation will parallelize to sub-nodes, thus providing a two-level parallel implementation.

As stated in Chapter 28.1 of the FLUENT users guide: *"The parallel process methods integral to FLUENT can automatically partition the fluid grid into multiple sub-domains such that the number of partitions is an integral multiple of the number of compute nodes available to the user. These processes can be compute nodes on a massively parallel computer, processes on a multiple-CPU workstation, or processes on a network of heterogeneous workstations running UNIX or on a network of workstations running Windows. In general, as the numbers of compute nodes increases, turnaround time for the solution will decrease. However, parallel efficiency decreases as the ratio of communication to computation increases, so there is a point of diminishing returns when considering model size versus number of compute nodes."* Since the Z-Flow FLUENT model is small, this category of parallel processing was not used for this task.

Parallel processing has the potential to reduce the time requirements such that general-purpose optimization becomes practical for a wide range of industrial applications. As the number of design variables increases, the bulk of the computational time required completing the optimization is consumed by the finite difference gradient calculations. Each design iteration calculates a set of gradients used to determine a search direction for the optimization algorithm during the one-dimensional search. The default in VisualDOC is to use forward finite difference calculations, which requires as many analyses as there are independent design variables for each set of gradient calculations. Since the analyses required during the finite difference calculations are independent of each other, these calculations are good candidates for parallel computing. Indeed, when the number of parallel processors is greater than or equal to the number of independent design variables, each set of finite difference calculations may be performed in the same time it takes to complete a single analysis.

The freely available LAM[5] system is a set of programs and libraries that allows a cluster of workstations connected with a local area network to be used as a parallel processing

[5] Local Area Multicomputer (LAM) is developed and maintained by Ohio State University. (http://www.lam-mpi.org)

computer. LAM contains an implementation of the MPI[6] standard that allows for dynamic load balancing. The LAM system was used to implement a parallel version of VisualDOC using existing UNIX and Linux workstations available at VR&D.

The VisualDOC analysis module is parallelized using a master-slave paradigm where the master process allocates the tasks to all available slave processors (see Venter & Watson[1], and Smith[2]). When a slave finishes its current task, it immediately becomes available so that another task may be allocated to it. This paradigm is ideally suited to a heterogeneous parallel environment, such as a local area network of workstations, because it is intrinsically load balanced. That is to say, faster processors will be allocated more tasks. Additionally, this scheme requires only minimal inter-processor communication. The design variables are sent to the slaves, and the response values are sent back to the master. A single slave process running on the same processor as the master process performs all the remaining analyses required during the optimization (e.g. one-dimensional search analyses).

VR&D Multiple Processors

Three different computing platforms with a combined total of four processors were used to optimize the Z-Flow cooling model.

- Neptune – SGI Octane Workstation with IRIX OS, 1 CPU
- Pluto – HP C3600 Workstation with HP-UX OS, 1 CPU
- Hyperion – Windows NT Pentium III Workstation with LINUX OS, 2-733MHz CPUs

Multi-Processor Results

The following results show solution differences due only to the number and type of processors used. All VisualDOC optimization and FLUENT set-ups are identical. The results are listed in Table 1 and presented graphically in Figure 14.

Each of the three different types of processors was run separately to establish CPU characteristics for each processor. These are runs 1-3 for Neptune, Pluto, and Hyperion respectively. It is interesting to note the differences for those design cycles listed as well as the total number of design cycles at convergence for exactly the same optimization problem. Even though there is only about a 3% difference between the final optimized objectives, the analysis path is different depending on the processor combination chosen. The analysis path variation is even more pronounced for the multi-processor optimizations. VR&D believes this is due to the different machine precision of the

[6]The Message Passing Interface (MPI) standard means true portability for parallel programs. It defines the necessary infrastructure for third party software products and will enable a wider proliferation of parallel technology.

processors used. The difference in the solutions due to the machine precision is just slightly less than the actual solution difference due to small perturbations of the design variables. This "solution noise" affects the gradient calculations, which in turn affects the one-dimensional search results. The VisualDOC finite difference move limits for this problem were chosen based on considerations such as these.

Run	1	2	3	4	5	6
Processor 1	Neptune	Pluto	Hyperion	Hyperion	Pluto	Pluto
Processor 2				Hyperion	Neptune	Neptune
Processor 3					Hyperion	Hyperion
Processor 4						Hyperion
Objective (DC 0)	2.96	2.98	2.96	2.96	2.98	2.98
Objective (DC 1)	3.39	3.24	3.34	3.34	3.23	3.26
Objective (Maximum)	3.41	3.34	3.41	3.41	3.42	3.45
No. Design Iterations	5	5	16	16	3	6
Total Analysis Calls	75	72	235	235	42	89
Total Time (Sec)	23153	18114	40274	26932	3485	5891
Time/Iteration	4631	3623	2517	1683	1162	982

Table 1 - VisualDOC Multi-Processor Summary Table

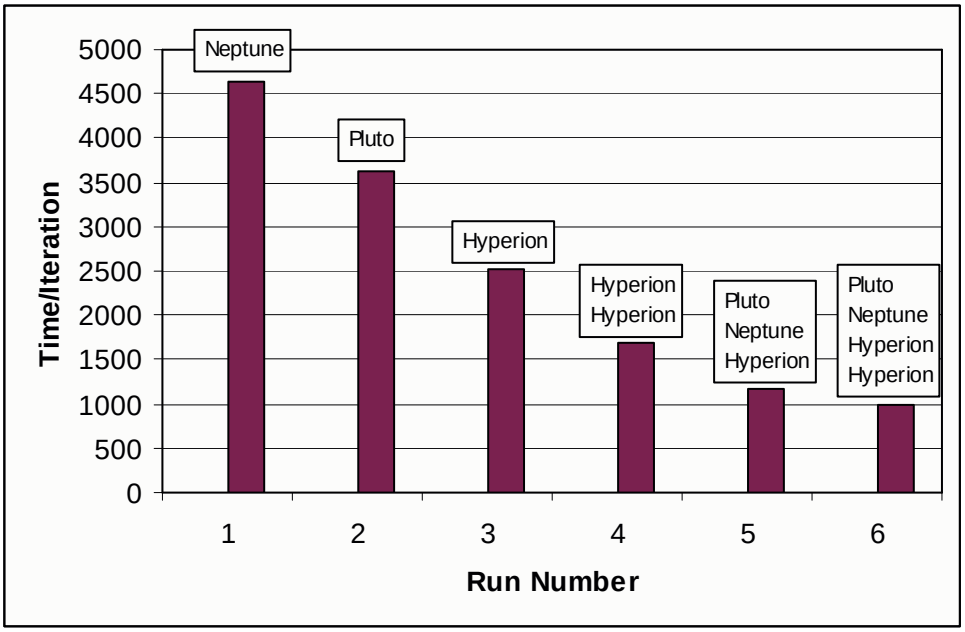


Figure 14 - VisualDOC Multi-Processor Summary Chart

Conclusions

The goal of this paper is two-fold: demonstrate a method to optimize a FLUENT CFD simulation using VisualDOC, and demonstrate the efficiency of a virtual parallel processor using VisualDOC.

Initially the Z-Flow design did not satisfy the minimum fluid velocity requirements at two of the ten response points. After optimization, all velocity constraints are satisfied and the average velocity of the ten points went up by 15%.

The Z-Flow CFD optimization used 12 design variables and up to 4 parallel processors. Figure 14 clearly shows the benefits realized when the finite difference gradient calculations are parallelized. For this optimization problem, the slowest time per design iteration is 4631 seconds using a single processor on Neptune. The fastest time per design iteration is 982 seconds using four processors. The virtual 4-CPU processor exhibits a factor of five enhancement in time over the single CPU on Neptune.

References

[1] G. Venter & B. Watson, "Exploiting Parallelism in General Purpose Optimization", Vanderplaats Research and Development, Colorado Springs, CO.

[2] Smith, S.L., and Schnabel, R.B., "Centralized and Distributed Dynamic Scheduling for Adaptive, Parallel Algorithms", *Unstructured Scientific Computation on Scalable Multiprocessors*, eds. P. Mehrotra, J. Saltz, and R. Voigt, MIT Press, Cambridge, MA, pp.301-322, 1992.