Advances in Aircraft and Spacecraft Science, *Vol. 4, No. 3 (2017) 000-000*
DOI: http://dx.doi.org/10.12989/aas.2017.4.3.000

Service ORiented Computing EnviRonment (SORCER) for deterministic global and stochastic aircraft design optimization: part 2

Chaitra Raghunath[*1], Layne T. Watson[1,2,3a], Mohamed Jrad[3b], Rakesh K. Kapania[3c] and Raymond M. Kolonay[4d]

[1]*Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061, USA*
[2]*Department of Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061, USA*
[3]*Department of Aerospace & Ocean Engineering, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061, USA*
[4]*AFRL/RQVC, 2210 8th Street, Bldg. 146, Wright-Patterson Air Force Base, Dayton, OH 45433, USA*

(Received May 18, 2016, Revised August 8, 2016, Accepted September 20, 2016)

Abstract. With rapid growth in the complexity of large scale engineering systems, the application of multidisciplinary analysis and design optimization (MDO) in the engineering design process has garnered much attention. MDO addresses the challenge of integrating several different disciplines into the design process. Primary challenges of MDO include computational expense and poor scalability. The introduction of a distributed, collaborative computational environment results in better utilization of available computational resources, reducing the time to solution, and enhancing scalability. SORCER, a Java-based network-centric computing platform, enables analyses and design studies in a distributed collaborative computing environment. Two different optimization algorithms widely used in multidisciplinary engineering design-VTDIRECT95 and QNSTOP-are implemented on a SORCER grid. VTDIRECT95, a Fortran 95 implementation of D. R. Jones' algorithm DIRECT, is a highly parallelizable derivative-free deterministic global optimization algorithm. QNSTOP is a parallel quasi-Newton algorithm for stochastic optimization problems. The purpose of integrating VTDIRECT95 and QNSTOP into the SORCER framework is to provide load balancing among computational resources, resulting in a dynamically scalable process. Further, the federated computing paradigm implemented by SORCER manages distributed services in real time, thereby significantly speeding up the design process. Part 1 covers SORCER and the algorithms, Part 2 presents results for aircraft panel design with curvilinear stiffeners.

Keywords: service-oriented computing; federated computing; deterministic global optimization; stochastic optimization; multidisciplinary design

∗Corresponding author, M.Sc., E-mail: chaitra@vt.edu
[a]Professor, E-mail: ltwatson@computer.org
[b]Post-doctoral Fellow, E-mail: jmohamed@vt.edu
[c]Mitchell Professor, E-mail: rkapania@vt.edu
[d]Ph.D., E-mail: raymond.kolonay@us.af.mil

http://www.techno-press.org/?journal=aas&subpage=7 ISSN: 2287-528X (Print), 2287-5271 (Online)

1. Optimization of curvilinear blade-stiffened panels

1.1 Problem description

Advances in manufacturing technology, computational science, and material science have produced a new generation of custom built so-called unitized structures that have multifunctionality tailored to design requirements. With additive manufacturing technology such as electron beam free-form fabrication (Taminger and Hafley 2003), it is possible to produce arbitrary curved metallic structures using aerospace alloys like titanium and aluminum. The possibility of curved stiffening members enlarges the design space and leads to the possibility of a more efficient aircraft design. Previous research (Mulani *et al*. 2010, Jrad *et al*. 2014) has shown that curvilinear stiffeners can improve the buckling resistance of local panels. Locatelli *et al*. (2013) demonstrated a savings in weight from the structural optimization of an aircraft wing using curvilinear spars and ribs (SpaRibs). The aircraft wing can be decomposed into multiple local panels bordered with the spars and ribs. Therefore, minimizing the structural weight of these panels reduces the overall wing weight.

Powerful computational environments have been developed in order to make use of such flexibility and build optimal light weight structures (Mulani *et al*. 2013), (Liu *et al*. 2015). The framework *EBF3PanelOpt* described here (Fig. 1) facilitates the structural optimization of curvilinearly stiffened panels by considering a number of constraints that have to be satisfied (buckling, von Mises stress, and crippling constraints). The framework, written in the scripting language Python, interacts with the commercial software MSC Patran (for geometry and mesh creation) and MSC Nastran (for finite element analysis). Given the input parameters and design

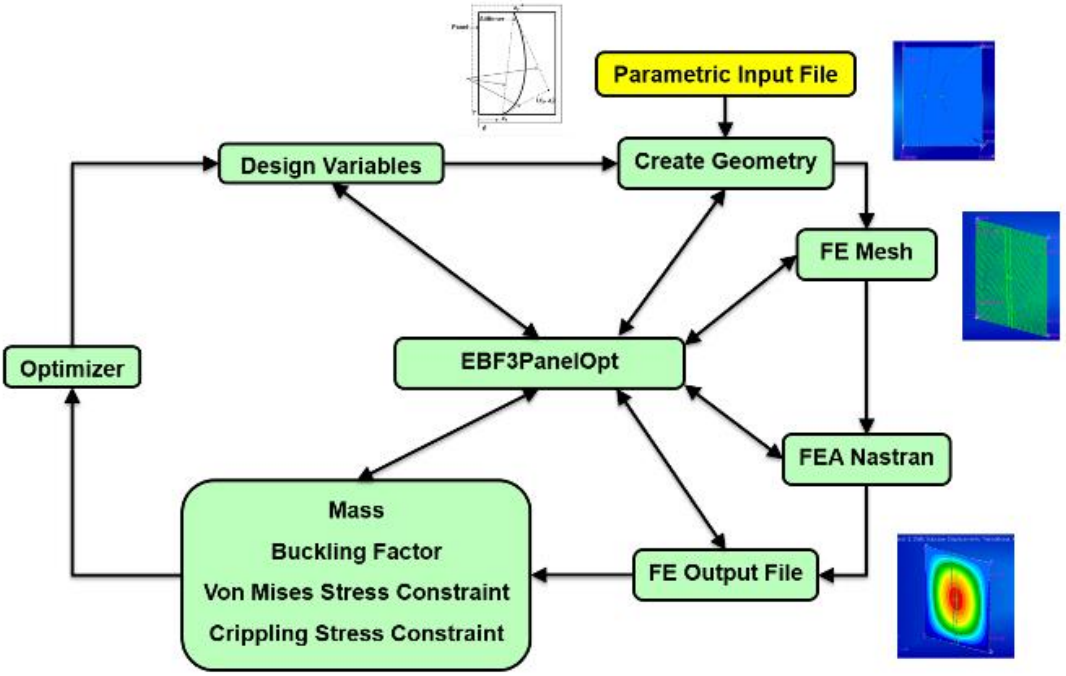
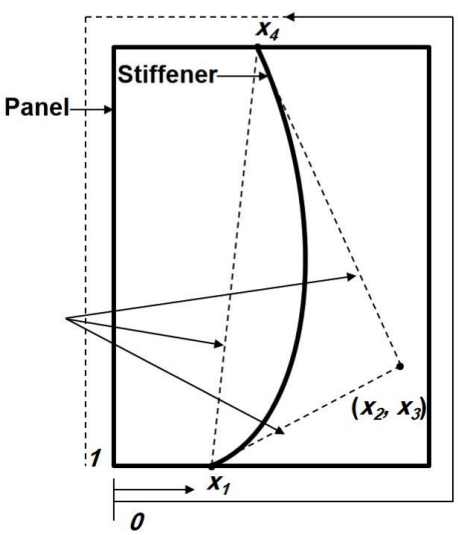


Fig. 1 Flowchart depicting the EBF3PanelOpt framework



	Min	Max	
x_1	0.0	0.23	Stiffener I Geometry
x_2	0.0	1.0	
x_3	0.0	1.0	
x_4	0.5	0.73	
x_5	0.0	0.23	Stiffener II Geometry
x_6	0.0	1.0	
x_7	0.0	1.0	
x_8	0.5	0.73	
x_9	0.01	0.05	Stiffeners Heights
x_{10}	0.01	0.05	
x_{11}	0.001	0.008	Stiffeners Thickness
x_{12}	0.001	0.008	
x_{13}	0.001	0.004	Panel Thickness

Fig. 2 Design variables of the stiffener (left). Definition and range of the design variables for the optimization process (right)

variables, the script then creates the appropriate session file and submits it to MSC Patran to create the geometry and mesh of the stiffened panel, with which MSC Nastran then carries out a finite element analysis producing structural mass, buckling factor, von Mises stress factor, and crippling factor, which finally becomes the input to an optimizer. More details about the framework EBF3PanelOpt can be found in (Mulani *et al*. 2013).

A key feature of EBF3PanelOpt is the ability to specify the geometry of the stiffened panel parametrically. The (parametric) design variables used in this framework determine the stiffeners' shape, position, height, and thickness, and the panel thickness. An example of how the stiffener geometry is defined in terms of geometric parameter lower and upper bounds is shown in Fig. 2 (right). Four design variables are used to calculate the shape and position of every stiffener, as shown in Fig. 2 (left). The stiffener's curve is represented using a third order uniform rational B-spline using two end points (defined by x_1 and x_4) and a control point (x_2, x_3), which guarantees that the stiffener always remains in the panel area. The two stiffener end points lie on the panel's perimeter and hence can each be represented with a single value (x_1 and x_4) that always lies between zero and one. A single perimeter curve is created in Patran and used to localize the end points of the stiffeners using parametric extraction. The values of x_2 and x_3, also between zero and one, determine the control point on the panel surface. Thus a panel with nearly arbitrary geometry, along with the stiffeners, can be represented using this parameterization.

1.2 Implementation of EBF3PanelOpt as a service

In order to achieve truly distributed objective function evaluations, the optimization framework EBF3PanelOpt is implemented as an analysis provider. Analysis providers can be dynamically distributed over a variety of computational resources with the help of SORCER's JavaSpaces technology, which implements a loosely coupled distributed computing system across the network. JavaSpaces not only enables computers on the network to communicate reliably, but also provides

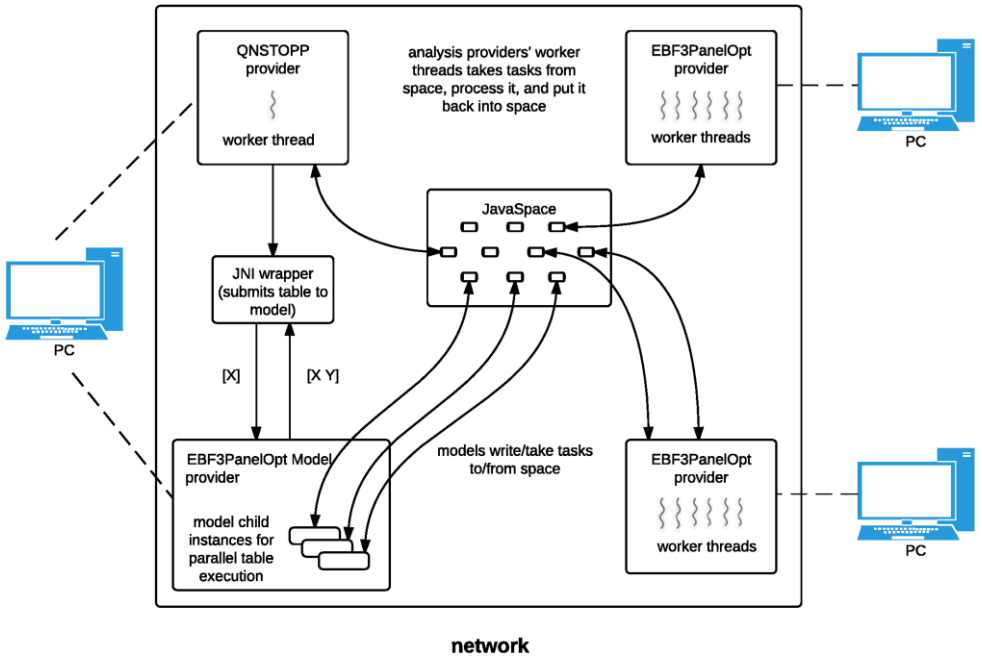


Fig. 3a Use of JavaSpaces technology (a) and Catalog technology (b) in a dynamic distributed computing environment for the panel optimization studies

load balancing capability to cope with dynamic resources. Hence, computing resources can be added during the course of an optimization study, thereby enhancing productivity. The JavaSpaces technology provides a type of shared memory where exertion evaluators can drop tasks they wish to be processed by service providers. When services providers are started on the network, they use Jini discovery mechanisms to find *spaces* on the network. If unprocessed tasks reside in the space, the provider picks up the task from the space and executes the appropriate service. Once execution is completed, the task is returned to the space and marked as processed. Processed tasks are removed from the space by the evaluator.

For the parallel implementation of QNSTOP that constructs a table of runs (a modification of the OpenMP parallel code QNSTOPP), the EBF3PanelOpt provider is configured to have a fixed number of worker threads. The number of worker threads determines the number of tasks a provider can process in parallel. The providers are started on multiple machines to distribute the work. During optimization, the model provider first creates child instances for every row in the table and drops these tasks into the space. Then, based on the number of worker threads, each EBF3PanelOpt provider picks up unprocessed tasks from the space, executes these tasks in parallel, and upon completion, returns them to the space.

Fig. 3a shows the use of JavaSpaces technology in a dynamic distributed computing environment for the panel optimization studies. An alternative to JavaSpaces is Catalog, referred to here as SORCER/Catalog and shown in Fig. 3b, that has advantages in certain contexts, such as multicore or single large parallel distributed memory machines. Here, service providers publish proxies to the catalog. The requestor passes a service request to the catalog, which “matches” the service request with one of the proxies. If there are multiple proxies that match the request, the catalog uses an algorithm such as round robin to select the proxy. This proxy is then passed to the

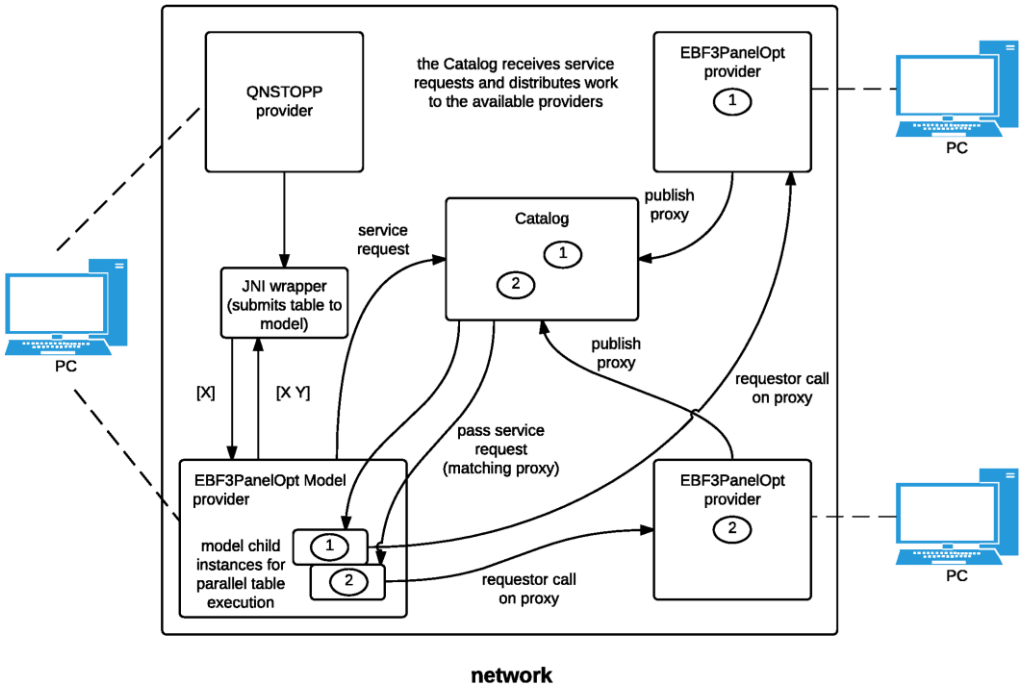


Fig. 3b Use of JavaSpaces technology (a) and Catalog technology (b) in a dynamic distributed computing environment for the panel optimization studies

requestor, who uses the proxy to make the remote call directly to the provider (as in Fig. 1, Part 1). In Fig. 3b the EBF3PanelOpt providers #1 and #2 publish their proxies with the catalog (the QNSTOPP provider and the EBF3PanelOpt Model provider also publish proxies, but this is not shown in the figure). The JNI wrapper submits a table to the EBF3PanelOpt Model provider. The EB3PanelOpt Model provider submits a service request to the catalog, which satisfies that request and passes the proxies for the EBF3PanelOpt providers to the model, which then makes a call on the proxies to access the EBF3PanelOpt providers.

2. Numerical results

This section presents results for optimization of curvilinear blade-stiffened panels using VTDIRECT95 and QNSTOP. The section is further divided into two subsections-Experiment 1 and Experiment 2. Experiment 1 presents optimization results for a stiffened panel of dimensions 0.6096 m×0.7112 m; Experiment 2 presents optimization results for a stiffened panel of dimensions 0.4064 m×0.5080 m.

All experiments presented here are conducted on Intel machines each with 16GB of memory and a single quad-core processor, in which each core supports hyper-threading. The experiments are conducted using GNU Fortran 4.9.1, GNU C 4.9.1, Python 2.6.6, Open MPI 1.8.1, and Java 1.8.0_25 on x86_64 RHEL running CentOS 6.6. The framework EBF3PanelOpt is configured to use Nastran 2014 and Patran 2014. For all experiments, the mesh size parameter in MD-Patran is set to 0.01 during mesh generation for the stiffened panels. This implies that the average elemental

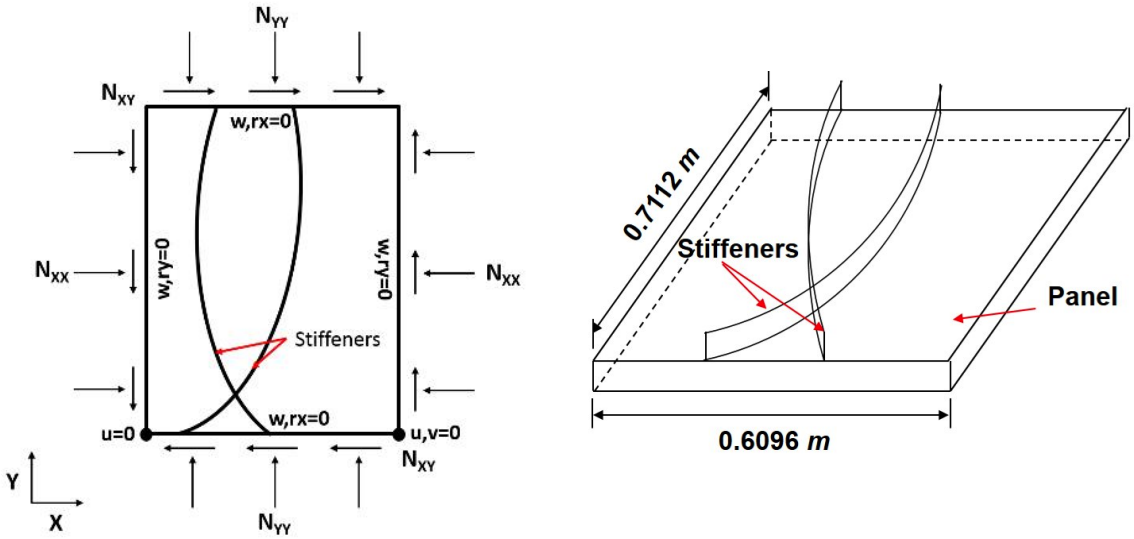


Fig. 4 Curvilinear stiffened panels under combined shear and normal loads

Table 1 Pylon wing panel, 0.6096 m×0.7112 m, optimization results, four stiffeners

	PSO + GBO	pVTdirect
Mass (kg)	3.2337	3.1269
Buckling factor	1.0015	0.9890
KSC	0.2109	0.2074
Crippling Criterion	0.9137	0.8290
Stiffener 1 height (m)	4.8865E-02	4.1333E-02
Stiffener 2 height (m)	4.8893E-02	4.1333E-02
Stiffener 3 height (m)	1.3403E-02	1.1666E-02
Stiffener 4 height (m)	2.9993E-02	2.9000E-02
Stiffener 1 thickness (m)	1.0003E-03	1.1666E-03
Stiffener 2 thickness (m)	1.1290E-03	1.1000E-03
Stiffener 3 thickness (m)	2.5810E-03	1.8333E-03
Stiffener 4 thickness (m)	1.0002E-03	1.1666E-03
Plate thickness (m)	2.4502E-03	2.4000E-03

edge length would be approximately 0.01. The (nonlinear, nonconvex) objective function (a penalty function based on mass and other panel design criteria) to be minimized is calculated by the (black box) program EBF3PanelOpt, subject to bound constraints on the design variables.

2.1 Experiment 1

The framework EBF3PanelOpt is applied to a panel with four curvilinear stiffeners and subjected to biaxial normal and shear loads (Fig. 4). The panel is fixed such that rotations and transverse displacements are not allowed, but in-plane displacements are allowed. The panel

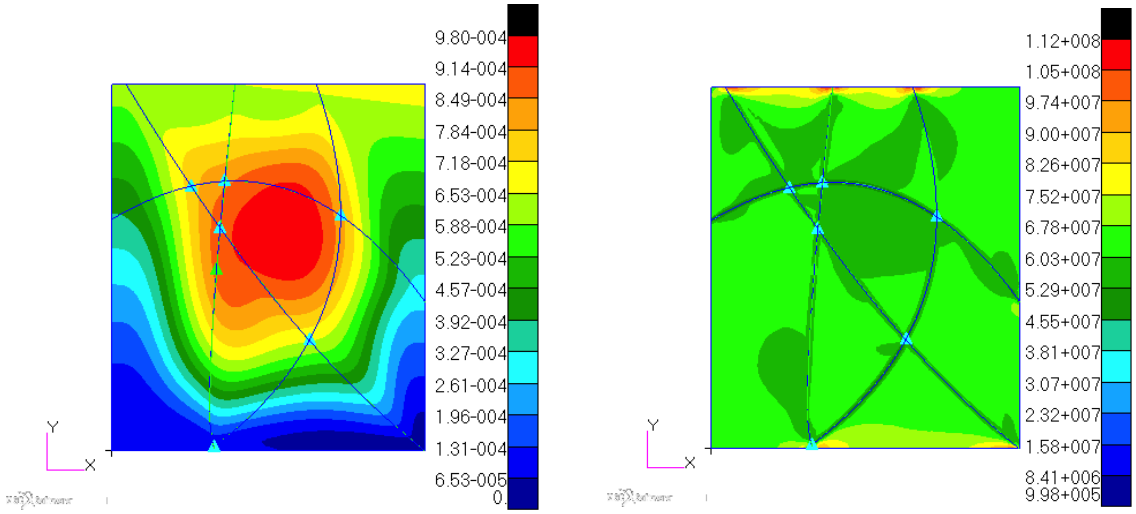


Fig. 5 Pylon wing panel, 0.6096 m×0.7112 m, four stiffeners, optimization using pVTdirect: (a) displacement (m), and (b) von Mises stress distribution (Pa)

dimensions are 0.6096 m×0.7112 m, and the applied loads are N_{XX}=15,761 N/m, N_{YY}=140.280 N/m, N_{XY}=44,307 N/m. In (Mulani *et al*. 2013), the framework EBF3PanelOpt was used with the heuristic particle swarm optimization technique followed by the gradient based method of feasible directions to approximate a locally optimal design for this same at rectangular panel. The material properties, load cases, sizing design variables' constraints, and other details can be found in (Mulani *et al*. 2013).

Using the inputs given in (Mulani *et al*. 2013), optimization is carried out with pVTdirect. For optimization with pVTdirect, a feasible box inside the sizing design variables' constraints was chosen. The values (mass, buckling factor, Kreisselmeier-Steinhauser criterion, and crippling criterion) and designs reported in (Mulani *et al*. 2013) were approximately reproduced (cf. Table 1). However, these results and the algorithmic efficiency cannot be compared to those in (Mulani *et al*. 2013), because (Mulani *et al*. 2013) did not report the number of function evaluations used by their heuristic algorithm.

The von Mises stress distribution and displacement for results obtained by pVTdirect are shown in Fig. 5.

2.2 Experiment 2

Fig. 6 shows a simply supported at rectangular 0.4064 m×0.5080 m panel representing a large wing engine pylon rib, which is optimized for minimum mass using VTDIRECT95 and QNSTOP. The material used for these panels is the aluminum alloy *Al 2139*, whose properties are given in Table 2. The panel is subjected to combined compression and shear in-plane loads (N_{YY}=462,200 N/m and N_{XY}=106,100 N/m). The problem of a simply supported panel with two curvilinear stiffeners, subjected to the same combined compression and shear in-plane loads, and a crack with nonprescribed location is examined in (Jrad 2015).

The results for the optimization of curvilinear blade-stiffened panels subjected to the combined compression and shear in-plane loads mentioned above, containing two stiffeners (Case 1) and

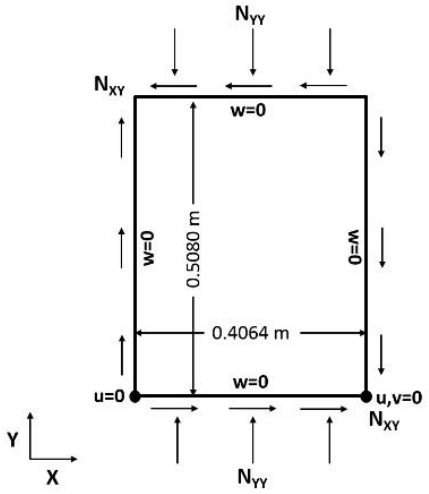


Fig. 6 Conventional aircraft wing panel geometry along with the loads

Table 2 Material properties for Al 2139

Young's modulus (GPa)	73.085
Poisson's ratio	0.33
Yield Stress (MPa)	427.47
Density (kg/m^3)	2795.00

Table 3 The sizing design variables' constraints

	Lower bound (m)	Upper bound (m)
Stiffeners 1 & 2 Geometry (x_1, x_5)	0.0	0.23
Stiffeners 1 & 2 Geometry (x_2, x_6)	0.0	1.0
Stiffeners 1 & 2 Geometry (x_3, x_7)	0.0	1.0
Stiffeners 1 & 2 Geometry (x_4, x_8)	0.5	0.73
Stiffener Height	0.01	0.05
Stiffener Thickness	0.001	0.007
Plate Thickness	0.001	0.007

four stiffeners (Case 2), using the subroutines VTdirect, pVTdirect, QNSTOPS, and QNSTOPP are presented below.

2.2.1 Case 1: optimization of curvilinear blade-stiffened panels containing two stiffeners

The (parametric) design variables that represent the geometry of the panel, stiffeners' position, height, and thickness, and the panel thickness are given in Table 3. For optimization with VTdirect, the design variables' constraints in Table 3 are the lower and upper bounds on X. The stopping condition is a limit of 1000 on the number of objective function evaluations. Additionally, for pVTdirect, the number of processes is set to four. Since pVTdirect is incompatible with

SORCER/JavaSpace/table model query, results for this case are omitted.

For optimization with (the serial subroutine) QNSTOPS, the sizing design variables' constraints in Table 3 are the lower and upper bounds on the design vector *X*, which is an ellipsoid center in the QNSTOP algorithm. The initial ellipsoid center *X* is the mean of the lower and upper bounds. It is generally desirable to run QNSTOP from multiple start points (Amos *et al*. 2014b). The optional argument NSTART, set to 5 here, is the number of start points to be automatically generated when the optional argument SWITCH=3. These start points, of which the initial input *X* is the first, are derived by Latin hypercube sampling in the feasible box. The size *N* of the experimental design used in each quasi-Newton iteration is set to 20. QNSTOPS is run in mode '*G*' (deterministic mode) and the factor by which TAU (the initial radius for the ellipsoidal design region) is decayed is specified by the optional argument GAIN, which is set to 1.0. The stopping rule is a limit of 1000 on the number of objective function evaluations (200 evaluations per start point).

Additionally, for (the parallel subroutine) QNSTOPP, the optional argument OMP is set to 2, which corresponds to parallelizing just the loop over the sampling point objective function evaluations. The environment variable OMP_NUM_THREADS is set to 4 to match the hardware capability. These input values apply to the JNI runs without SORCER; recall that QNSTOPP has to be modified for SORCER as described in Section 3.5.2, Part 1. QNSTOPP for SORCER does not use OpenMP to achieve parallelism, but rather instead interacts with the model provider via a table model query, which in turn spawns a child instance for each row in the table for parallel execution of the table row evaluations. Hence, for all QNSTOPP runs with SORCER, OMP should always be set to 0 (no OpenMP parallelization) and the environment variable OMP_NUM_THREADS should be set to 1. It is also important that the number of rows in the table be compatible with the machine's underlying hardware. For all QNSTOPP runs with SORCER presented in this paper, two identical machines are used. The program carrying out optimization as a service and the model provider are started on one machine, and the EBF3PanelOpt provider on the other. For all QNSTOPP runs, the table is configured with four rows, resulting in four concurrent objective function evaluations.

The results for the two-stiffener panel optimization using the subroutines VTdirect, pVTdirect, QNSTOPS, and QNSTOPP are shown in Table 4. The execution times for pVTdirect, VTdirect, QNSTOPS, and QNSTOPP, with and without SORCER, are listed in Table 5. The last column in

Table 4 Pylon wing panel, 0.4064 m×0.5080 m, optimization results, two stiffeners

	VTdirect	QNSTOPS	QNSTOPP
Mass (*kg*)	2.5274	2.5009	2.4189
Buckling factor	0.9825	0.9506	0.9879
KSC	0.2956	0.3253	0.3233
Crippling Criterion	0.7206	0.3489	0.3212
No. of function evaluations	1131	950	950
Stiffener 1 height (m)	3.0000E-02	3.5613E-02	2.2977E-02
Stiffener 2 height (m)	2.5555E-02	3.0199E-02	3.1799E-02
Stiffener 1 thickness (m)	4.0000E-03	4.3402E-03	3.5181E-03
Stiffener 2 thickness (m)	1.3333E-03	3.6315E-03	3.8770E-03
Plate thickness (m)	4.0000E-03	3.6784E-03	3.6860E-03

Table 5 Execution time (s) for pylon wing panel optimization (two stiffeners)

	VTdirect	pVTdirect	QNSTOPS	QNSTOPP	E_p
With SORCER and script robustness	13,009	N/A	11,388	3,545	0.80
With SORCER, without script robustness	8,957	N/A	7,994	2,542	0.79
With SORCER/Catalog, without script robustness	8,487	N/A	7,597	2,458	0.77
Without SORCER and script robustness	8,460	2,924	7,560	2,309	0.82

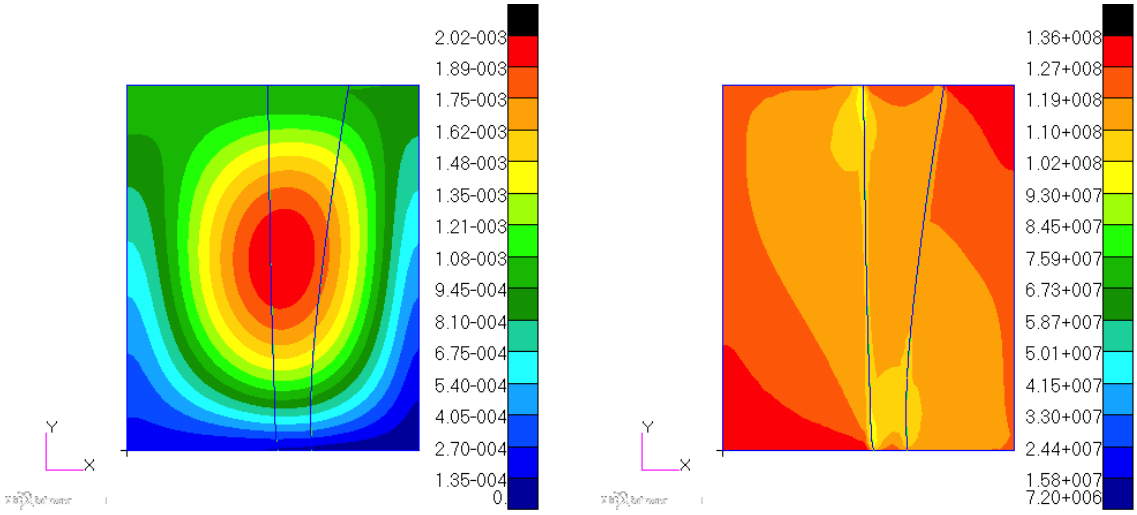


Fig. 7 Pylon wing panel, 0.4064 m×0.5080 m, two stiffeners, optimization using VTdirect: (a) displacement (m), and (b) von Mises stress distribution (Pa)

the table represents the parallel efficiency with QNSTOPP. For the runs that do not involve SORCER, the parallel efficiency is

$$E_p = \frac{((serial\ QNSTOPS\ time)/(parallel\ QNSTOPP\ time))}{(total\ number\ of\ OMP\ threads)}.$$

For the runs of (serial) QNSTOPS with SORCER and (modified parallel) QNSTOPP SORCER, the parallel efficiency is

$$E_p = \frac{((QNSTOPS\ time)/(parallel\ QNSTOPP\ time))}{(number\ of\ function\ evaluation\ threads)}.$$

In this experiment, for QNSTOPP runs without SORCER, OMP_NUM_THREADS=4, and for QNSTOPP runs with SORCER, the number of rows in the model query table is four. In Tables 5, 8, and 9 "script robustness" refers to a Java utility `GenericUtil` separate from SORCER, recommended for use of scripts in production distributed computing, that increases the robustness of scripts and communication links across different operating systems. "SORCER/Catalog" is an

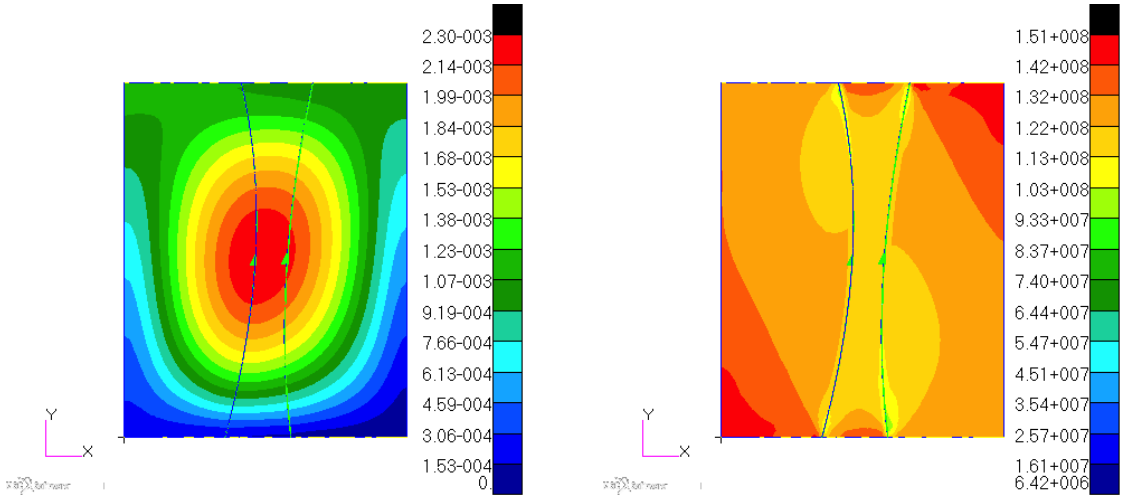


Fig. 8 Pylon wing panel, 0.4064 m×0.5080 m, two stiffeners, optimization using QNSTOPS: (a) displacement (m), and (b) von Mises stress distribution (Pa)

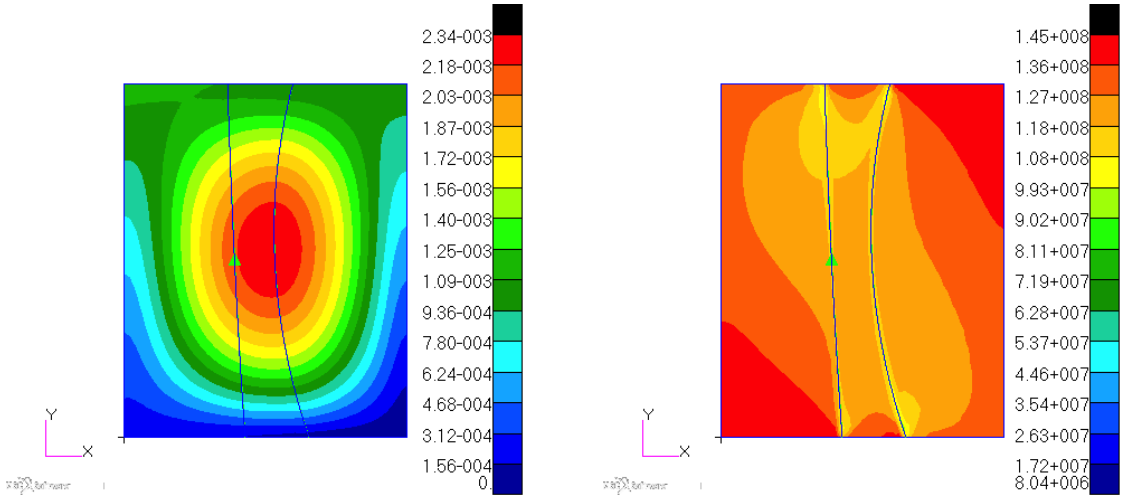


Fig. 9 Pylon wing panel, 0.4064 m×0.5080 m, two stiffeners, optimization using QNSTOPP: (a) displacement (m), and (b) von Mises stress distribution (Pa)

alternative to SORCER with JavaSpaces that is more appropriate for a multicore machine. The Catalog concept from a precursor of SORCER is essentially the service registry shown in Fig. 1, Part 1. The parallel efficiencies E_p for QNSTOPP runs without and with SORCER directly measure the overhead cost incurred with SORCER: using SORCER without script robustness rather than the OpenMP/MPI parallel directives reduces the parallel efficiency slightly. Using script robustness with SORCER (most general and reliable production mode) for all the codes-VTdirect, QNSTOPS, QNSTOPP-increases the execution time significantly but changes the parallel efficiency only slightly. Note that the cost of script robustness and reliable distributed execution is roughly independent of the problem dimension and the cost of the function evaluations, so with more expensive function evaluations the parallel efficiency would improve.

Table 6 The sizing design variables' constraints

	Lower bound (m)	Upper bound (m)
Stiffeners 1 & 2 Geometry (x_1,x_5)	0.0	0.23
Stiffeners 1 & 2 Geometry (x_2,x_6)	0.0	1.0
Stiffeners 1 & 2 Geometry (x_3,x_7)	0.0	1.0
Stiffeners 1 & 2 Geometry (x_4,x_8)	0.5	0.73
Stiffener 3 Geometry (x_9)	0.23	0.5
Stiffener 3 Geometry (x_{10})	0.0	1.0
Stiffener 3 Geometry (x_{11})	0.0	1.0
Stiffener 3 Geometry (x_{12})	0.73	1.0
Stiffener 4 Geometry (x_{13})	0.15	0.5
Stiffener 4 Geometry (x_{14})	0.0	1.0
Stiffener 4 Geometry (x_{15})	0.0	1.0
Stiffener 4 Geometry (x_{16})	0.5	1.0
Stiffener Height	0.01	0.05
Stiffener Thickness	0.001	0.007
Plate Thickness	0.001	0.007

The von Mises stress distribution and displacement for results obtained by VTdirect, QNSTOPS, and QNSTOPP are shown in Figs. 7, 8, and 9, respectively. For all three cases, the buckling constraint value is close to 1, indicating that the corresponding masses are near optimal. The KSC criterion value is slightly higher for the QNSTOPS and QNSTOPP results indicating that the panel is more stressed for better designs. The crippling criterion value for the best mass obtained with VTdirect is moderately close to saturation compared to those obtained with QNSTOPS and QNSTOPP. This indicates that the stiffener is more prone to crippling when one or more flanges buckle in a local buckling mode with wavelength unrelated to the length of the beam.

2.2.2 Case 2: optimization of curvilinear blade-stiffened panels containing four stiffeners

The design variables that represent the geometry of the panel, stiffeners' position, height, and thickness, and the panel thickness are given in Table 6. For optimization with VTdirect, the design variables' constraints in Table 6 are the lower and upper bounds on *X*. Similar to Case 1, the stopping condition is a limit of 1000 on the number of objective function evaluations. For pVTdirect, the number of processes is set to 4. Since pVTdirect is incompatible with SORCER/JavaSpace/table model query, results for this case are omitted.

For optimization with (the serial subroutine) QNSTOPS, the sizing design variables' constraints in Table 6 are the lower and upper bounds on the design vector *X*, which is an ellipsoid center in the QNSTOP algorithm. The initial ellipsoid center *X* is the mean of the lower and upper bounds. As in the previous case, the optional argument NSTART (number of start points) is set to 5 when the optional argument SWITCH=3. The size *N* of the experimental design used in each quasi-Newton iteration is set to 40. QNSTOPS is run in mode 'G' (deterministic mode) and the factor by which TAU (the initial real radius for the ellipsoidal design region) is decayed is specified by the optional argument GAIN, which is set to 1.0. The stopping rule is a limit of 1000 on the number of

Table 7 Pylon wing panel, 0.4064 m×0.5080 m, optimization results, four stiffeners

	VTdirect	QNSTOPS	QNSTOPP
Mass (kg)	2.5472	2.7024	2.8863
Buckling factor	0.9901	0.9667	0.91172
KSC	0.2901	0.3277	0.3378
Crippling Criterion	0.5364	0.3676	0.2567
No. of function evaluations	1165	825	825
Stiffener 1 height (m)	3.0000E-02	2.8342E-02	3.2533E-02
Stiffener 2 height (m)	1.6666E-02	2.4909E-02	2.5158E-02
Stiffener 3 height (m)	3.0000E-02	3.3505E-02	2.9751E-02
Stiffener 4 height (m)	1.6666E-02	2.9939E-02	3.0364E-02
Stiffener 1 thickness (m)	2.0000E-03	3.9661E-03	4.6432E-03
Stiffener 2 thickness (m)	2.0000E-03	3.1711E-03	4.3679E-03
Stiffener 3 thickness (m)	2.0000E-03	3.7376E-03	4.0771E-03
Stiffener 4 thickness (m)	2.0000E-03	3.8998E-03	3.5480E-03
Plate thickness (m)	4.0000E-03	3.6572E-03	3.8516E-03

Table 8 Execution time (s) for pylon wing panel optimization (four stiffeners)

	VTdirect	pVTdirect	QNSTOPS	QNSTOPP	E_p
With SORCER and script robustness	14,450	N/A	10,370	3,676	0.71
With SORCER, without script robustness	10,384	N/A	7,451	2,697	0.69
With SORCER/Catalog, without script robustness	9,815	N/A	7,088	2,615	0.68
Without SORCER and script robustness	9,786	3,789	7,052	2,408	0.73

objective function evaluations (200 evaluations per start point).

Additionally, for (the parallel subroutine) QNSTOPP, the optional argument OMP is set to 2. The environment variable OMP_NUM_THREADS is set to 4 to match the hardware capability. For all QNSTOPP runs with SORCER, OMP is set to 0 and the environment variable OMP_NUM_THREADS is set to 1. The table that is passed from the JNI wrapper to the model provider is configured to have four rows, resulting in four concurrent objective function evaluations.

The results for the four-stiffener panel optimization using the subroutines VTdirect, QNSTOPS, and QNSTOPP are shown in Table 7. The execution times for pVTdirect, VTdirect, QNSTOPS, and QNSTOPP, with and without SORCER and/or script robustness (described earlier), are listed in Table 8. The last column in the table represents the parallel efficiency with QNSTOPP.

In this experiment, note that the numbers of function evaluations (in Table 7) for VTdirect and QNSTOP* are different than those for Case 1 (in Table 4), since these numbers depend on the problem dimension and the sample size (per start point for QNSTOP*). The trends and

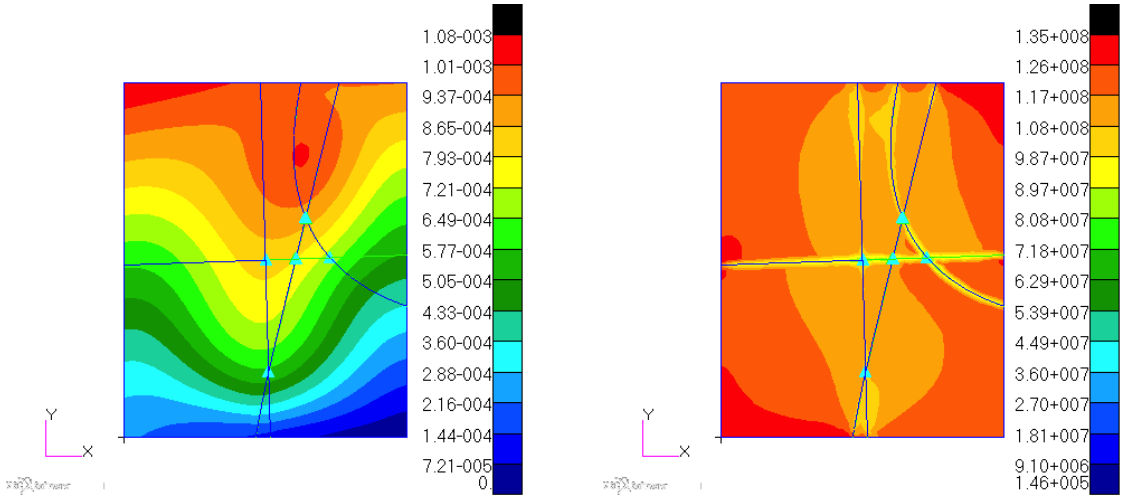


Fig. 10 Pylon wing panel, 0.4064 m×0.5080 m, four stiffeners, optimization using VTdirect: (a) displacement (m), and (b) von Mises stress distribution (Pa)

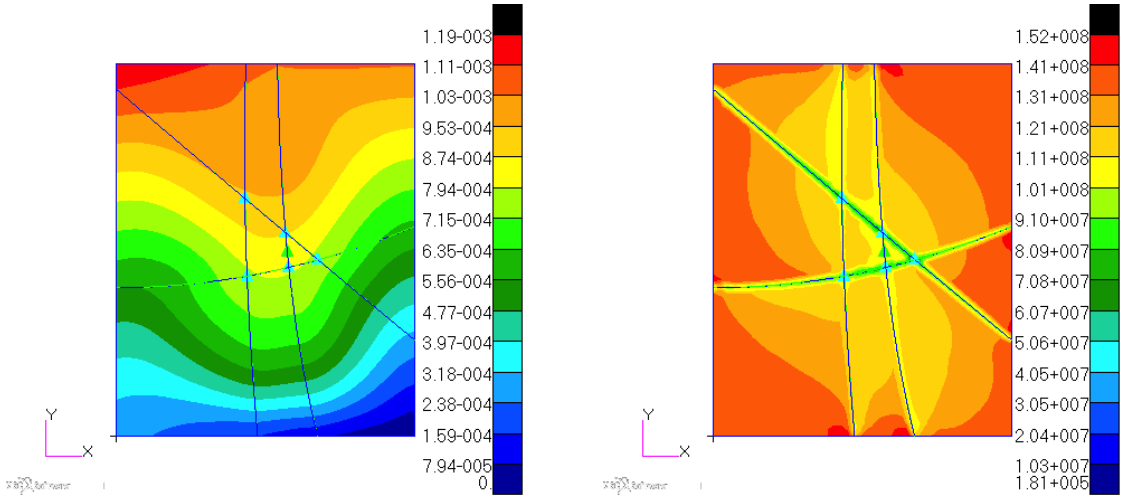


Fig. 11 Pylon wing panel, 0.4064 m×0.5080 m, four stiffeners, optimization using QNSTOPS: (a) displacement (m), and (b) von Mises stress distribution (Pa)

implications of Table 8 are similar to those of Table 5. Even though the problem size doubled (from 13 to 25), the parallel efficiencies decreased because the number of function evaluations by QNSTOPP was less for Case 2 than for Case 1.

The von Mises stress distribution and displacement for results obtained by VTdirect, QNSTOPS, and QNSTOPP are shown in Figs. 10, 11, and 12, respectively. For all three cases, the buckling constraint is close to 1, indicating that the corresponding masses are nearly optimal. The design of Fig. 12 is interesting from a stability point of view, since the two close stiffeners in the middle of the panel behave like supports to clamp the panel in the middle and eliminate, therefore, the low order buckling modes. Moreover, the other two stiffeners divide the panel into smaller subpanels and help increase its buckling load.

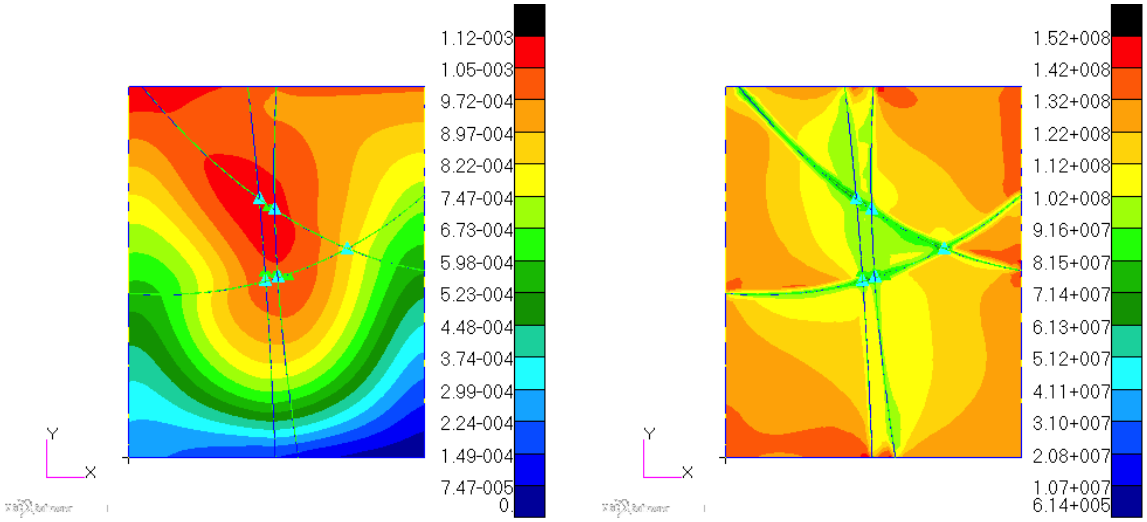


Fig. 12 Pylon wing panel, 0.4064 m×0.5080 m, four stiffeners, optimization using QNSTOPP: (a) displacement (m), and (b) von Mises stress distribution (Pa)

Table 9 Objective function evaluation time (s) for pylon wing panel (two and four stiffeners)

	n=13	*n*=25
With SORCER and script robustness	11.13	12.90
With SORCER, without script robustness	7.36	9.14
Without SORCER and script robustness	7.32	9.10

The computational expense of each objective function evaluation is listed in Table 9, where *n* is the problem dimension. To estimate the average computational expense for an objective function evaluation, runs were made with VTdirect for a total of 100 function evaluations, recording the time between when the arguments are passed to the evaluation code, and the time when the function value comes back. With SORCER with and without script robustness, runs were made with VTdirect for a total of 100 function evaluations, recording the time between when the call to *execute* on the analysis provider that provides EBF3PanelOpt as a service is made and when the function value is sent to the model client. The SORCER with script robustness overhead per function evaluation ($\gg 4s$) is about the same for both problem sizes, and without script robustness the SORCER overhead is negligible. Robustness and portability do not come cheap.

3. Conclusions and future work

From Table 9, the SORCER with script robustness overhead per function evaluation is about four seconds (essentially all due to script and communication link robustness across different operating systems), and similar overhead would be expected had the VTdirect, QNSTOPS, and QNSTOPP runs without SORCER been done with script robustness. With 1000 function evaluations on the order of 10 seconds each, this SORCER with script robustness overhead is

significant as reflected in Table 8. The parallel efficiency E_p is a fair measure of the extent to which parallel hardware resources are being utilized efficiently, and $E_p>0.5$ is considered acceptable. If 1000 function evaluations taking one hour each were done instead, the SORCER with script robustness overhead would be relatively negligible and $E_p\approx1.0$. Apart from the script robustness overhead for function evaluations, Tables 5 and 8 show that the other SORCER overhead is not significant for expensive function evaluations in a distributed computing environment. MDO may involve function evaluations ranging in cost over several orders of magnitude (<10 *s* to >1000 *s*), so the conclusion is that the right SORCER paradigms (JavaSpaces, though the most general approach, is but one of several approaches) must be used in the right contexts, and no single SORCER paradigm is a general purpose solution to parallel and distributed computing for MDO. On the continuum of distributed computing technology (MPI, Globus, Legion, Condor, SORCER), SORCER is at the heavyweight end.

The use of JNI to wrap the corresponding native code provided a clean and elegant interface between the optimization algorithm and the Java block that evaluates the objective for a design point. While JNI is associated with high development overhead, JNA (Java Native Access) is a library that provides easy access to native shared libraries without boiler plate/glue code. However, JNA has more execution overhead and is slower than JNI. Some of the disadvantages of JNI include: need for boiler plate code in C, poor performance, and weak security. While JNI is the current standard programming interface for interfacing native methods with Java, the Java FFI (Foreign Function Implementation) has been proposed to address the difficulties of JNI. The Java FFI is a metadata system that will allow access to native functions with native memory management at the Java level. This feature would not require the developer to deploy boiler plate code or have expertise in JVM internals. As Java FFI matures, it would be interesting to replace the existing JNI code with Java FFI and carry out trade-off studies between performance and the cost of development.

Two different optimization algorithms widely used in multidisciplinary engineering design-VTDIRECT95 and QNSTOP-were implemented as services on a SORCER grid. Source code for the service wrappers is in (Raghunath 2015), and source code for VTDIRECT95 and QNSTOP is in the published references. The EBF3PanelOpt framework was successfully integrated with SORCER, hence facilitating the optimization of curvilinearly stiffened panels in a truly distributed manner. Further, the implementation of the EBF3PanelOpt framework as an analysis provider within SORCER provided flexibility; the objective function evaluations could be moved to less-burdened machines as and when required. With the JavaSpaces technology, computers could be added on the fly, providing flexibility, and enabling distributed parallelism and load balancing across computational resources in a dynamically scalable environment.

It should also be mentioned that the installation of SORCER is far from routine-SORCER is currently a research code with limited documentation and requires considerable knowledge of network computing to install and utilize. Future work includes modifying the parallel subroutine pVTdirect of VTDIRECT95 to accommodate chunking of function evaluations at a synchronization point, which would then be compatible with a master-slave paradigm in which design points are readily accessible by a master. This implementation could be integrated with the SORCER/JavaSpace/table model query paradigm, thus providing both VTDIRECT95 and QNSTOP as robust distributed SORCER services.

Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8650-09-2-3938. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. The EBF3PanelOpt code was developed under a research contract from NASA Fundamental Aeronautics Program to Virginia Polytechnic Institute and State University with Karen M. B. Taminger as the Program Manager. The authors thank Scott A. Burton for help with the SORCER installation, service implementation, and experiments.

References

Amos, B.D., Easterling, D.R., Watson, L.T., Thacker, W.I., Castle, B.S. and Trosset, M.W. (2014b), "Algorithm XXX: QNSTOP: Quasi-Newton algorithm for stochastic optimization", Computer Science Technical Report 2014-07, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Bisagni, C. and Lanzi, L. (2002), "A post-buckling optimisation of composite stiffened panels using neural networks", *Compos. Struct.*, **58**(2), 237-247.

Boni, L., Fanteria, D. and Lanciotti, A. (2012), "A post-buckling behavior of flat stiffened composite panels: experiments vs. analysis", *Compos. Struct.*, **94**(12), 3421-3433.

Jrad, M. (2015), "Multidisciplinary optimization and damage tolerance of stiffened structures", Ph.D. Thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute & State University, Blacksburg, VA.

Jrad, M., Khan, A.I. and Kapania, R.K. (2014), "Buckling Analysis of Curvilinearly Stiffened Composite Panels with Cracks", *Proceedings of the AIAA 2014-0165, 55th Structures, Structural Dynamics, and Materials Conference (SDM)*, Maryland, National Harbor.

Lanzi, L. and Giavotta, V. (2006), "A post-buckling optimization of composite stiffened panels: computations and experiments", *Compos. Struct.*, **73**(2), 208-220.

Liu, Q., Jrad, M., Mulani, S.B. and Kapania, R.K. (2015), "Integrated Global Wing and Local Panel Optimization of Aircraft Wing", *Proceedings of the AIAA 2015-0137, 56th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Kissimmee, Florida.

Locatelli, D., Liu, Q., Tamijani, A.Y., Mulani, S.B. and Kapania, R.K. (2013), "Multidisciplinary optimization of supersonic wing structures using curvilinear Spars and Ribs (SpaRibs)", *Proceedings of the AIAA 2013-1931, 54th AIAA/ASME/ASCE/AHS/ASC Structures Conference*, Boston, Massachusetts.

Montemurro, M., Vincenti, A. and Vannucci, P. (2012), "A two-level procedure for the global optimum design of composite modular structures-application to the design of an aircraft wing. Part 1: theoretical formulation", *J. Optim. Theor. Appl.*, **155**(1), 1-23.

Montemurro, M., Vincenti, A. and Vannucci, P. (2012), "A two-level procedure for the global optimum design of composite modular structures-application to the design of an aircraft wing. Part 2: numerical aspects and examples", *J. Optim. Theor. Appl.*, **155**(1), 24-53.

Mulani, S.B., Locatelli, D. and Kapania, R.K. (2010), "Algorithm development for optimization of arbitrary geometry panels using curvilinear stiffeners", *Proceedings of the AIAA 2010-2674, 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Orlando, Florida.

Mulani, S.B., Slemp, W.C.H. and Kapania, R.K. (2013), "EBF3PanelOpt: an optimization framework for curvilinear blade-stiffened panels", *Thin Wall. Struct.*, **63**, 13-26.

Raghunath, C. (2015), *Service Oriented Computing Environment (SORCER) for Deterministic Global and Stochastic Optimization, M.S. thesis, Department of Computer Science*, Virginia Polytechnic Institute & State University, Blacksburg, VA.

Taminger, K.M.B. and Hafley, R.A. (2003), "Electron beam freeform fabrication: a rapid metal deposition process", *Proceedings of the Third Annual Automotive Composites Conference*, Troy, Michigan.

EC