A Web-Based Database Approach to CFD Post-Processing

Graham Pullan*

Whittle Laboratory, Department of Engineering, University of Cambridge, Cambridge CB3 0DY, UK

An interactive, hierarchical and comparative approach to analysing large ensembles of simulation or experimental data is presented. A database of processed output is stored on a remote server and visualised locally using a web-based client. The structure of the database mirrors the typical questions an engineer might raise when interrogating data, from configuration (geometry, solver, operating condition), to overall performance (lift, drag, efficiency), to detailed performance (lift distributions) and, finally, the flow field (cuts of flow properties). The user is able to view the whole database at a high level and then filter out only the simulations and experimental tests of current interest. Once these "Tasks" are identified, it is then possible to retrieve detailed data and generate One-, Two- and Three-Dimensional plots. These charts can be created, refreshed and removed from the web browser display on demand. The flexibility and utility of the approach is demonstrated using two case studies: the Onera M6 wing and a 3 stage compressor.

I. Introduction

Engineers reliant on Computational Fluid Dynamics (CFD) to guide a design process have an insatiable appetite for compute power. Improvements in hardware (clusters of commodity nodes but also many-core accelerators such as graphics processing units) and software (the widely adopted Message Passing Interface programming model but also tuned maths libraries) have driven advances in high fidelity "capability computing" and design space mapping via medium fidelity "capacity computing." In both of these cases, the challenge to the engineer is the same: how to work efficiently and effectively with the ever-increasing quantity of simulation data.

The NASA CFD Vision 2030 report[1] maps out a path to the CFD capabilities that will be required in 2030. "Knowledge Extraction" is identified as one of the six key technology areas that each require research and development programs to achieve the required technical milestones.[2] The goal for the way CFD will be used is that "a single engineer/scientist must be able to conceive, create, *analyse and interpret the large ensemble of related simulations* in a time-critical period (e.g. 24 hours)." To achieve this goal, the Knowledge Extraction technology roadmap recommends pursuing a three component track: visualization; database management; and variable fidelity data integration. The Vision 2030 report anticipates querying large databases in real-time and that these databases will be enhanced on demand. The present work contributes to this ambition through the development of a web-based interactive visualization platform for the analysis of a database of computational and experimental data.

Other instances of the use of databases in CFD result management and visualization have been reported. NASA's AeroDB[3] was an early example of a database approach to both CFD job initiation and processing of derived data. Tecplot's Chorus software[4] allows engineers to import selected input and output data from many computations and then apply filters to explore the design space and fit reduced order models.

The aim of the present work is to harness the growing eco-system of web-based data visualization and analytics tools, combined with a remote database, to allow engineers to interrogate large numbers of simulations; the software implementation of this approach is called *dbslice*. The paper is organised as follows. Motivation for the work is provided by a hypothetical design review scenario. The conceptual framework for an interactive, hierarchical, comparative post-processing tool is then introduced, followed by the strategy

*Reader in Aerothermal Engineering, AIAA Senior Member

used to implement this framework in *dbslice*. Two test cases are then discussed: the Onera M6 wing and a 3 stage axial compressor. Finally, some comments on the flexibility and utility of the approach are given.

II. Motivation

There are many situations in which engineering decision making and analysis would benefit from immediate and interactive access to multiple data sets. The following scenario is an example: A design team assembles to discuss a promising new compressor blade concept that has been progressed through to a detailed three-dimensional design by a member of the team. A series of charts are presented that show both the high-level performance metrics and the detailed flow field of the candidate blade. The team agrees that the new design is worth pursuing but, naturally, questions arise: what is the behavior of the blade at different operating points; how does the blade compare to other designs that are currently being considered; and how does is stack up against designs that are either in service or were candidates for previous projects? Unable to proceed without further analysis of *existing* data, the meeting breaks up and will reconvene when the required plots have been generated.

The inefficiency of the above process is evident. The interactions between the engineers in the room - the thinking behind each of the questions that was raised - will have to be established again at a future meeting. The data to answer the questions posed exists, but the engineers were unable to access it in a spontaneous, interactive manner. Aerospace engineers face similar challenges when looking at experimental data. In contrast, the same individuals are familiar with accessing and filtering large amounts of data routinely in online applications such as travel booking web sites. Can the growing eco-system of web-based data processing tools be harnessed in an engineering design context?

III. Conceptual framework for an interactive, hierarchical, comparative post-processing tool

A. Vision

An *interactive*, *comparative*, and *hierarchical* approach to post-processing is envisaged. This approach mirrors the structure of the questions that are typically raised in discussing a computational or experimental dataset (or set of data sets): how was the task carried out - what was the set up of the computation or experiment; what was the high level performance compared to similar tasks; what, in the detail of the results, has given rise to these overall changes? These questions have a hierarchy and provide a context for the data through comparison with similar results. The central contribution of the present work is a software framework to enable these questions to be answered interactively.

B. Framework specification

The workflow outlined above suggests a client-server framework. A database of processed results is store on a server. The engineers connect to, and interact with, this database using a graphical client-side application:

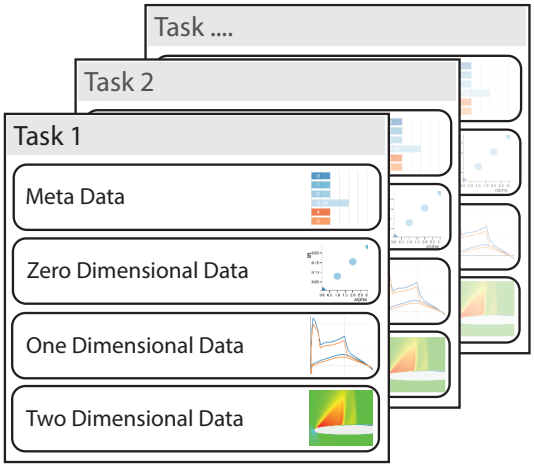
1. Database structure

A *Task* is a collection of data that has been obtained from a particularly computation or build of an experiment, see Fig. 1(a). For example, test results from an airfoil at a certain Mach number, Reynolds number and incidence would be expected to be from the same Task; if the incidence were changed, a new Task would be used. The configuration of a Task is described by its *Meta Data* (airfoil shape, test conditions, tunnel used, date, etc).

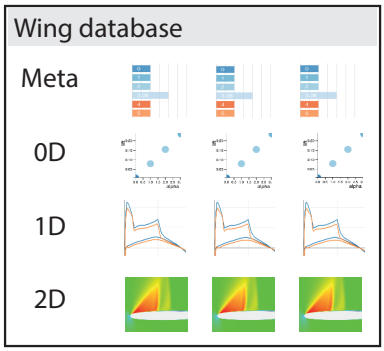
The data from each Task are stored at a number of *Slices*. The data at a Slice can be a single value (Zero-Dimensional), a line (One-Dimensional) or a surface (Two-Dimensional). To retrieve a Property from a particular Task and Slice, the database is queried with a Task number, a Slice name, and a Property name.

2. Data visualization client

The role of the client is to display the Meta Data for the entire database, shown schematically in Fig. 1(b). The user can then apply filters to narrow down the available Tasks into those of current interest. Once



(a) Server: database stores a hierarchy of data for each Task (computation or experiment)



(b) Client: a graphical user interface retrieves and plots data. Plots are added, refreshed and cut on demand

Figure 1. Client-sever strategy for the interactive visualisation of large numbers of computational and experimental datasets

individual Tasks are selected, the data at particular Slices can be retrieved from the database, plotted and compared. It is advantageous if some of the Slice data can also be used to apply filters to the database. In *dbslice*, the user can filter based on the Zero-Dimensional data, as well as on the Meta Data.

IV. Implementation

A. Introduction

The conceptual framework described in Section III is considered robust in that it has been found to accommodate a broad range of requirements for data types that aerospace engineers wish to store, recall, visualize and compare. The underlying software implementation required to support this framework is likely, however, to change over time. This is particularly true on the client side where the life-spans of web browser programming approaches and libraries are notoriously short. Details of the current implementation strategy are presented here so that, (a) users can appreciate the opportunities and limitations of *dbslice*, and (b) developers are introduced to the role of each library.

B. Server side

1. *Database*

MongoDB[5] was chosen as the database. *MongoDB* is a "schema-less" database which means, in contrast to a traditional database, the structure does not need to specified *a priori*. Data in a *MongoDB* database is stored in a "documents" with each document containing a series of field:data pairs. The structure of each document can be different. In *dbslice*, the following document types are defined:

1. type=header. There is one document of this type per database. The document contains lists of Property and Slice names.

2. type=data_0d. Each Task has a document of this type. The document contains all Meta-Data and Zero-Dimensional data for the Task.

3. type=data_1d. Each instance of One-Dimensional data is stored in a document of this type. A separate document is used for each Task, Slice and Property.

4. type=data_2d. As for the type=data_1d documents, but for the storage of Two-Dimensional array data.

2. Construction of the database

The database is built on the server. It is assumed that the database is connected to the same file system as the raw computational or experimental data. The data can then be processed using scripts on the server, and assembled into the appropriate document types, before being added to the database. *Python* is a suitable language for this task (*PyMongo* provides interoperability with *MongoDB*).

3. Serving the database

In the present implementation the server-side code is written in *Python* using the *Flask*[6] framework. *Flask* provides a straightforward way to parse and respond to queries, via the addition of GET requests to a standard URL address, and then serve the required data from the database. It is also possible to add to the database using this type of process but the data flow in the current implementation is only from server to client.

C. Client side

1. Database filtering, Meta-Data and Zero-Dimensional plots

The client side code runs in a web browser and is written in Javascript. Once the database is selected, all of the Meta Data and Zero-Dimensional data is requested and stored in the browser. To save memory and reduce the time needed for data transfer, no One-Dimensional or Two-Dimensional data are requested at this stage.

The Meta-Data and One-Dimensional data are assembled into a single "flat table" structure. In this form, the data are easily filtered and plotted using the *crossfilter*[7] and *dc*[8] javascript libraries. *dc* is used to display bar charts, histograms and scatter plots, The bar charts and histograms are also interfaces for the filters (performed in *crossfilter*) that are used to select the Tasks that are of interest. For example, from the database of all wing computations, the user is able to click on the appropriate bars to select only a certain geometry, solved with a particular code, at a given inlet condition. Equally, using a selected range on a histogram, the user could view all wings in a particular incidence and Mach number range.

By default, all Meta-Data is plotted, in bar chart form, when *dbslice* is started. The user can then select additional plots (scatter plot or histogram) of Zero-Dimensional data at selected Slices. However many Zero-Dimensional plots are added, they remain part of the *crossfilter* and so can be used to down-select the Tasks. By hovering over each data point in a scatter plot, the user can add a an individual Task to a list of Tasks that is available for One-, Two- or Three-Dimensional plotting.

2. Plotting of lines and surfaces

One-Dimensional (line plot), Two-Dimensional (an individual surface) or Three-Dimensional (multiple surfaces) plots can be added to those displayed in the web browser. When each line or surface is added, the data is fetched from the database on the server. The line plots are generated using the *plotly*[9] library, and the surface plots using *threejs*.[10] The latter is able to tack advantage of hardware-accelerated rendering.

V. Example operation of *dbslice*

To provide an illustration of *dbslice*, a test database is generated as follows. The database comprises 100 Tasks. Each Task is a three-dimensional block of a Property, *f*. The Property is defined such that,

$$f(x,y,z) = f_{\text{bg}}(x) + f_{\text{shape}}(y,z) \qquad 0 \leq (x,y,z) \leq 1 \tag{1}$$

$$f_{\text{bg}}(x) = f_\alpha + x f_\beta \tag{2}$$

$$f_{\text{shape}}(y,z) = f_\gamma \exp\left(\frac{-10}{(y-0.5)^2 + (z-0.5)^2}\right) \tag{3}$$

where f_α, f_β and f_γ are constants between 0 and 1 and take a random value for each Task.

The database stores Slices at planes of constant x spaced at regular intervals of $\Delta x = 0.1$. At each Slice, the Properties listed in Table 1 are stored in the database. As Meta-Data, each Task is arbitrarily assigned

(a) All 100 Tasks

(b) "Solver 2" and "Geom 5" filters are applied - 7 tasks remain. Task 52 is selected for further plotting.

Figure 2. Meta Data and Zero-Dimensional data from the Test database (*dbslice* screenshots).

a notional "solver" (4 options), "mesh" (8 options) and "geom" (5 options) - these are user-defined fields that can be used for filtering.

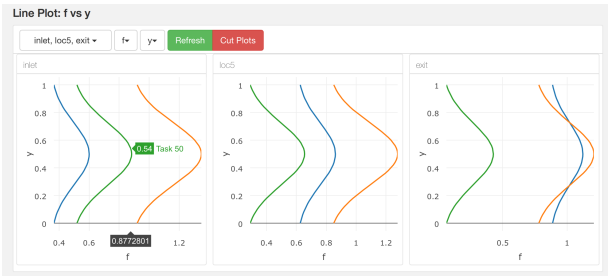
Type	Name	Definition
data_0d	f_{ave}	Area-average of f at the Slice
data_0d	f_{peak}	Maximum value of f at the Slice
data_1d	$f_{\text{bar}}(y)$	Averaged value of f, in the z direction, at fixed y, at the Slice
data_2d	$f(y, z)$	The values of f at the Slice

Table 1. Properties stored in the Test database

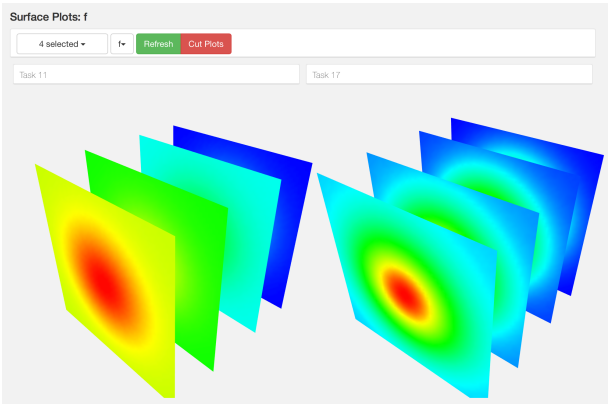
On opening *dbslice*, bar charts of the Meta Data are displayed, as shown in the upper half of Fig. 2(a). The user can then add, dynamically, scatter plots of Zero-Dimensional data such as the plots of f_{peak} as a function of f_{ave} shown, at three Slices ("inlet", "loc5" and "exit"), in the lower half of Fig. 2(a).

The bar charts and scatter plots are connected by the same *crossfilter* data structure. To illustrate this, in Fig. 2(b) the user has selected to view only those Tasks that are tagged with "Solver 2" and "Geom 5"; the number of Tasks displayed has reduced from 100 to 7. Figure 2(b) also illustrates how individual Tasks can be selected from the scatter plot and added to one or more lists. These lists contain the Tasks that are to be plotted when One-Dimensional (lines), Two-Dimensional (single surface) or Three-Dimensional (multiple surfaces) are requested.

Figure 3 shows plots of the Tasks that are members of the One-Dimensional and Three-Dimensional lists. In Fig. 3(a), line plots have been added to show the variation of f_{ave} with y for three Tasks at three Slices, ("inlet", "loc5" and "exit"). Figure 3(b) shows a view of four Slices of two Tasks. The plots in Fig. 3(b) can be rotated and zoomed by the user. Once a row of plots (One-, Two- or Three-Dimensional) has been made, it is straightforward to add additional data, from either other Slices or from separate Tasks, and redraw using the "Refresh" button. The "web page" that has been constructed by the user, containing a reconfigurable package of data from Meta to Three-Dimensional, remains displayed at all times so that the user can scroll to different parts of the data hierarchy for analysis.



(a) One-Dimensional data at 3 Slices from 3 Tasks



(b) A three-dimensional plot of Two-Dimensional data: 4 Slices from 2 Tasks

Figure 3. Line and surface data from the Test database (*dbslice* screenshots).

VI. Case studies

A. Introduction

Two test cases are presented to demonstrate the use of *dbslice* in aerospace applications. The first case is the Onera M6 wing; the second is a 3 stage compressor. In both cases, the utility of a hierarchical recall of data (Meta Data, One-, Two- and Three-Dimensional) and the dynamic generation of comparative charts, between and within the Tasks is emphasised. These case studies can be accessed at http://www.dbslice.org/demos.

B. Onera M6 wing

Computations of the Onera M6 wing[11] were performed, using the Turbostream[12] GPU-accelerated RANS code, at a series of Mach numbers and incidences. At one inlet condition, experimental data are also stored in the database. The nominal values of Mach number, incidence, and also whether the Task is computational or experimental, are stored as Meta Data. In Fig. 4, the inlet Mach number 0.84 is selected and the lift coefficient versus incidence charts, at 3 cuts (at different spanwise locations) are plotted.

Using the circles in the incidence-lift coefficient scatter plots, the Tasks corresponding to the computational and the experimental data at an incidence of 3.06 degrees are selected and added to the list of Tasks for line and surface plotting. Distributions of pressure coefficient (One-Dimensional data) are requested and added to the data displayed in the browser, at 5 Slices, as shown in Fig. 5(a). Having selected line charts of pressure coefficient at these Slices, it is easy to add other Tasks to the list and, by refreshing the One-Dimensional plots, add the associated pressure distributions to the charts; in Fig. 5(b), incidences of 2 and 4 degrees have been added.

The database for the Onera M6 wing computations also contains Two-Dimensional data: Mach number on the same Slices used for the lift distributions of Fig. 5, and isentropic Mach number on the wing surface. Figure 6(a) shows surface lift distributions, at an inlet Mach number of 0.84, for an incidence of 2 and 4 degrees. The lift distributions are taken at three Slices. To better understand the flow responsible for these distributions, the user adds the Mach number contor plots, for "cut3", at incidences of 2 and 4 degrees, Fig. 6(a). By default, the user's request for Two-Dimensional charts generates a separate contour plots for each Task in the 2D List at each required Slice. Is is then possible to zoom in and out of the view in the browser.

When Three-Dimensional charts are chosen, all selected Slices for a particular Task are displayed in one plot. The spanwise variation of the shock structure is illustrated in Fig. 6(b). Isentropic surface Mach number on the wing, and the "cut3" Slice, are shown at the same two incidences of Fig. 6(a). The user is able to rotate and zoom the view. Additional Slices, and Tasks, can be selected, and then added to the plot, using the Refresh button.

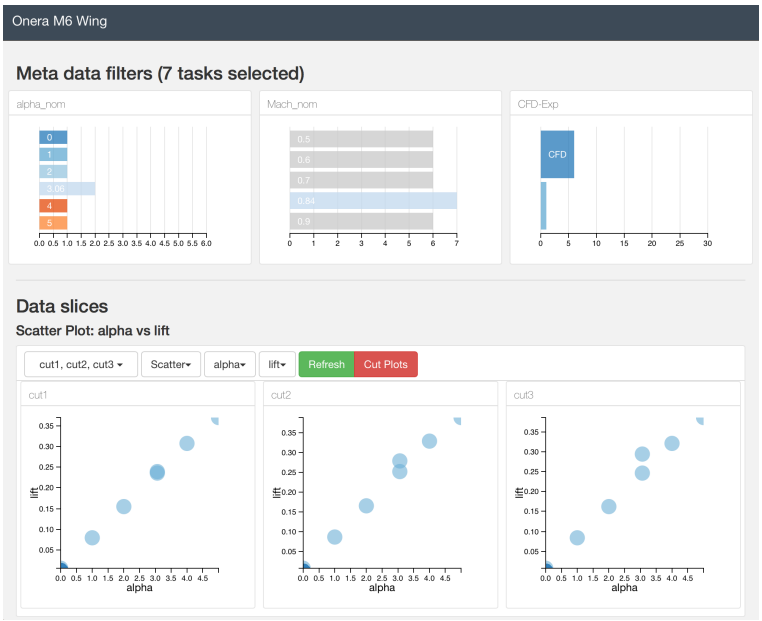
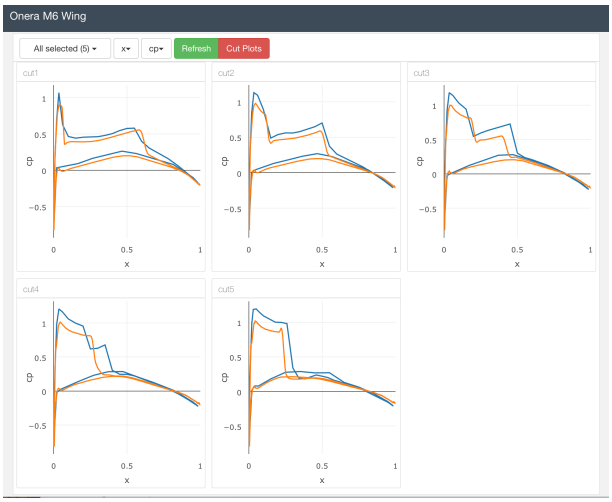
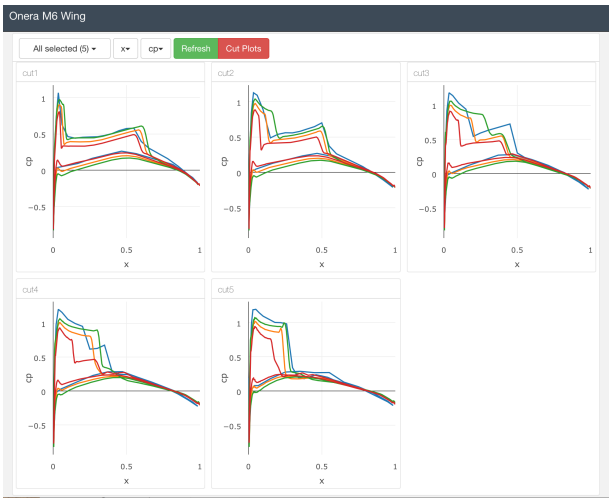


Figure 4. Meta Data and Zero-Dimensional data from the Onera M6 wing database. 7 Tasks (6 computations and 1 experiment) remain after filtering at one incidence and Mach number (*dbslice* screenshot).

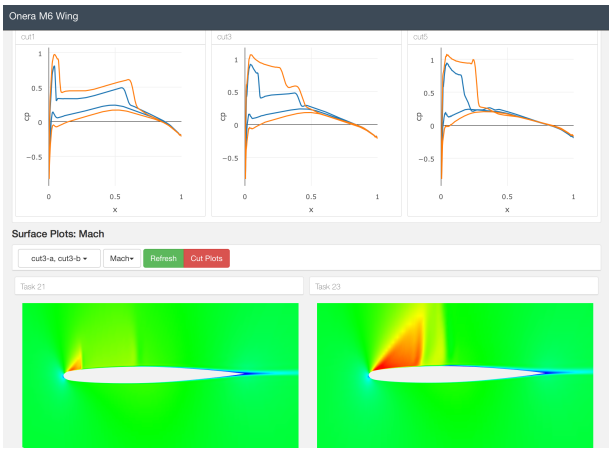


(a) Computation (orange) and experiment (blue) at $M = 0.84$ and $\alpha = 3.06$ deg, at 5 Slices.

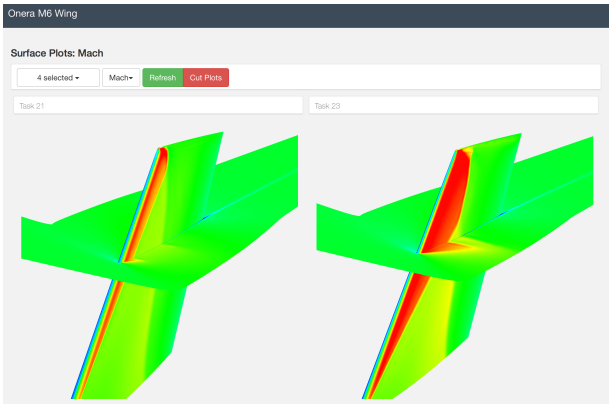


(b) lift distributions at $\alpha = 2$ deg and $\alpha = 4$ deg are added

Figure 5. One-Dimensional data from the Onera M6 wing database (*dbslice* screenshots).



(a) Lift distributions at 3 Slices, Mach contours at the "cut3" Slice.



(b) Wing surface isentropic Mach number and the "cut3" Slice. Three-dimensional plots can be rotated by the user.

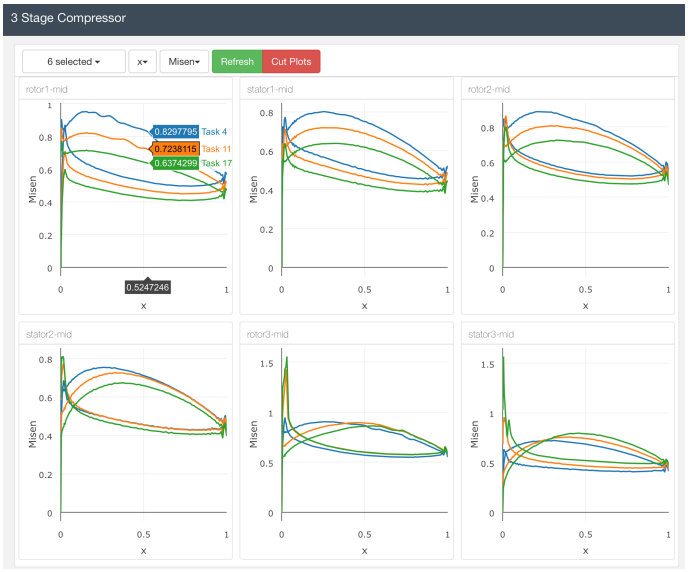
Figure 6. One- and Two-Dimensional data from the Onera M6 wing database at $M = 0.84$, and $\alpha = 2$ deg and $\alpha = 4$ deg (*dbslice* screenshots).

Figure 7. Meta Data and Zero-Dimensional data from the 3 stage compressor database. Pressure-ratio vs massflow characteristics are shown for 3 speed-lines, at 3 Slices ("overall", "stage2" and "stage3"), (*dbslice* screenshot with the notional working line - dashed red - added).

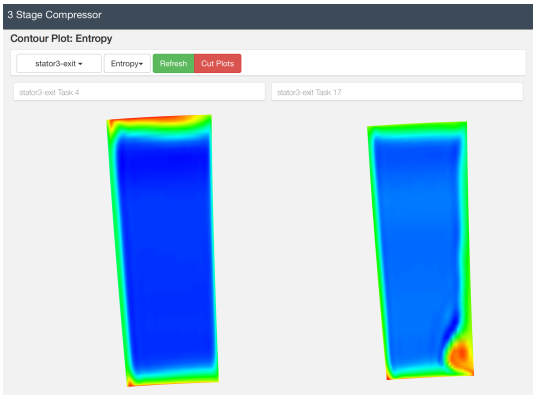
C. Multi-stage compressor

The analysis of multi-stage compressors is challenging because of the requirement to quantify the behaviour of the machine at a range of rotational speeds and operating points. As the compressor is operated away from its nominal design point, the matching of the blade rows is altered so that the profiles no longer sit at their optimum conditions (for example, of incidence or inlet Mach number). A typical results analysis procedure follows the hierarchical approach encouraged by the *dbslice* framework: single value data for overall, stage and row performance; line plots (blade surface pressure distributions or circumferentially averaged radial profiles); contour plots of surface information on blades, annulus walls or cut planes in the flow. *dbslice* promotes this approach and enhances it by enabling interactive comparisons with other operating points, computations or experiments.

The compressor simulated here is a 3 stage axial machine used as an example design by Denton.[13] 19 computations, at three rotational speeds, were performed using the Multall[13] RANS code with mixing planes between the blade rows. The pressure ratio characteristics for the three speed lines are shown in Fig. 7: one for the compressor overall and one each for stages 2 and 3. The upper point on each speed line is the highest pressure ratio for which a converged solution was obtained. A notional working line (added as a by-hand annotation on top of the *dbslice* screenshot) indicates the three Tasks that have been selected, one from each speed line, for further analysis.



(a) Mid-span surface Mach number distributions for each blade row for 3 Tasks.



(b) Slice at the exit of the final blade row showing high entropy loss core due to pressure-surface separation at 80% speed (bottom-right of Task 17 plot).

Figure 8. One and Two-Dimensional data from the 3 stage compressor database. Each Task shown is on the notional working line: Task 4 (100% speed), Task 11 (90% speed) and Task 17 (80% speed) (*dbslice* screenshots).

The isentropic mid-span Mach number distributions for the three Tasks on the compressor working line are shown in Fig. 8(a). The data for the 80% speed line (green curves) show reversed loading over the front quarter of the chord in the final stator due to negative incidence. Having spotted this, the engineer can assess the consequences by calling up the entropy field at the stator 3 exit Slice for the 100% speed (Task 4) and 80% speed (Task 17) cases, Fig. 8(b). The negative incidence at the hub of the 80% speed computation results in a pressure-surface separation and the consequent high loss zone in the lower-right corner of the plot in Fig. 8(b).

The database for the 3 stage compressor contains blade surface and hub, mid, tip and exit Slices for each blade row. There are circumstances where it is desirable to display many of these surfaces in a single plot. In Fig. 9, 24 Slices (hub, mid and blade surfaces) have been selected and relative Mach number (isentropic Mach number on the blade surfaces) contours shown for each. Once again, Tasks 4 and 17 are compared and

the pressure-surface separation on the final blade row is visible as a quarter pitch zone of low Mach number flow adjacent to the last blade on the hub Slice of Task 17.

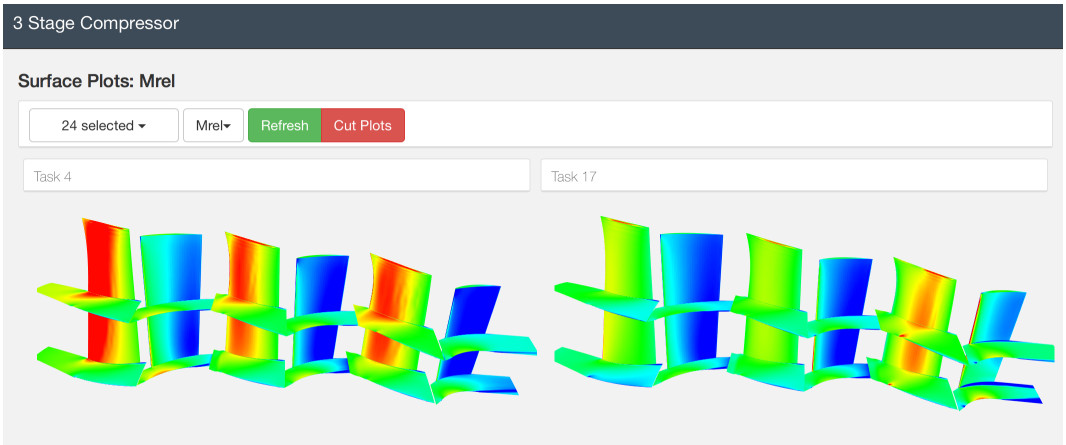


Figure 9. Three dimensional views of multiple Two-Dimensional data (relative Mach number) from the 3 stage compressor database. Task 4 is on the working line at 100% speed, Task 17 is on the working line at 80% speed (*dbslice* screenshot).

VII. Discussion

The test cases presented in Section VI demonstrate that a hierarchical database approach to storing computational and experimental data provides a natural framework for engineers to analyse results. The whole database is visible, at a high level, using the Meta and One-Dimensional data. From this perspective, the engineer is able to select specific Tasks to interrogate in greater detail using One-, Two-, or Three-Dimensional charts. The flexibility with which these plots can be added and removed, as well as the Tasks and Slices that are requested, allows engineers to work with large databases and answer questions as they arise.

The principal limitation of the approach outlined in the paper is that only data that has already been processed, and added to the database, can be accessed; the engineer can work dynamically with large ensembles of computations or experiments, but the ability to view all possible properties, or create new ones in real time, has been sacrificed. Although any property can be added to the database, it has been a conscious design choice to increase the time taken for this process so that the speed of working with many datasets of "standardised" properties can be increased. If more detail on a particular computational Task is required (for example additional cut planes, streamline tracing, or iso-surfaces) the engineer should load the full solution for that case into visualization software dedicated to analysing a single simulation output.

The observation from working with *dbslice* is that the appreciation of the flow field is enhanced: new features of the flow are noticed and tracked, opportunities for reduced order modelling are seen and the limits of the mapped design space are defined. These outcomes can be produced by a single engineer who uses *dbslice* to become immersed in the data that has been stored. An additional benefit, however, is the ability to rapidly respond to questions arising from team discussions, providing answers within the flow of the conservation, and so promote opportunities for innovation.

VIII. Conclusions

The following conclusions are drawn:

1. A conceptual framework for the interactive, hierarchical and comparative analysis of large ensembles of computational or experimental data is described. The framework leads to a client-server model where pre-processed data is stored on a remote server database and the user retrieves and plots the data using a local client.

2. *dbslice* is an implementation of the framework where the client is a web browser application that is able to take advantage of the growing eco-system of Javascript data analytics and plotting libraries. The user is able to filter Tasks (computational solutions or experimental data from a particular test) by either high level Meta Data (e.g. geometry and solver information) or single number Zero-Dimensional data (e.g. lift coefficient or efficiency metrics). The user can then interactively request One-Dimensional (line), Two-Dimensional (single surface) or Three-Dimensional (multiple surfaces) plots from locations within one Task or from many Tasks.

3. Two test cases are presented to illustrate the operation of *dbslice* in representative aerospace applications. In both the Onera M6 wing and multi-stage compressor cases, the hierarchical storage of the data is shown to reflect the process of solution interrogation that is typically followed by the engineer: task definition, overall performance, line plots and finally surface plots. The flexibility to quickly add plots from different planes (Slices) and solutions (Tasks) is demonstrated.

4. The observation from working with *dbslice* is that the ability to work interactively with large ensembles provides an opportunity for new insight on levels ranging from details of off-design performance to opportunities for reduced order modelling. Above all, software frameworks of this type allow existing data to be interrogated to provide answers to questions *as they are raised* and hence enhance the innovation process.

IX. Acknowledgements

The author is grateful to Dr Chris Clark for the results for the Onera M6 wing test case, and to Professor John Denton for the geometry of the multi-stage compressor test case. The author would also like to thank all those who have given comments, suggestions and encouragement during the development of the software, particularly Professor David Darmofal of MIT.

References

[1]Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," *NASA/CR-2014-218178*, 2014.

[2]Duque, E., Imlay, S., Ahern, S., Chen, G., and Kao, D., "NASA CFD Vision 2030 Visualization and Knowledge Extraction," *AIAA 2016-1927*, 2016.

[3]Chaderjian, N., Rogers, S., Aftosmis, M., Pandya, S., Ahmad, J., and Tejnil, E., "Automated Euler and Navier-Stokes Database Generation for a Glide-Back Booster," *Third International Conference on Computational Fluid Dynamics*, 2004.

[4]"Tecplot Chorus," *http://www.tecplot.com/products/tecplot-chorus/*.

[5]"mongodb - a document oriented database," *https://www.mongodb.com*.

[6]"flask - a Python web server," *http://flask.pocoo.org*.

[7]"Crossfilter - Fast Multidimensional Filteirng for Coordinated Views," *http://square.github.io/crossfilter*.

[8]"dc.js - Dimensional Charting Javascript Library," *https://dc-js.github.io/dc.js*.

[9]"plotly.js - Javascript Graphic Library," *https://plot.ly/javascript*.

[10]"three.js - Javascript 3D Library," *https://threejs.org*.

[11]Schmitt, V. and Charpin, F., "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," *AGARD AR138*, 1979.

[12]Brandvik, T. and Pullan, G., "An Accelerated 3D Navier-Stokes Solver for Flows in Turbomachines," *ASME J. Turbomach.*, 2009.

[13]Denton, J., "Multall - An Open Source, CFD Based, Turbomachinery Design System," *ASME GT2017-63993*, 2017 (in preparation).