Big-data Efficient and Automated Science Transfer (BEAST): an open-source software architecture for arc jet data management, modeling, and automation

Magnus A. Haw*,
NASA, Moffett Field CA, 94035

Megan E. MacDonald†
Flying Squirrel, Brussels, 1150, Belgium

Sebastian V. Colom‡,
Analytical Mechanics Associates, Moffett Field CA, 94035

Big-data Efficient and Automated Science Transfer (BEAST) is a facility data management application developed for the NASA Ames arc jet facilities. The current decentralized data management practices limit statistical tracking, synchronization between video/time series, search capability, data throughput, and data processing speed/efficiency. Consequently, BEAST was developed to provide a new data infrastructure with streamlined data collection, processing, transfer, and analysis. This new framework also seeks to implement the FAIR principles of data stewardship: Findable, Accessible, Interoperable, and Reusable.

The BEAST framework is based on a combination of the Python Django web framework and the Python data stack to provide a monolithic, open-source platform for data management, automation, and machine learning. This architecture was chosen for maintainability and scalability for a small, in-house development team. This paper will describe the application framework, deployment, and discuss the benefits and future plans for the system.

Contents

*Plasma physicist, Thermophysics Materials Branch, NASA Ames Research Center, MS 234-4, Member AIAA, magnus.haw@nasa.gov
†Aerospace Engineer, Thermophysics Facilities Branch, NASA Ames Research Center, Member AIAA, megan.e.macdonald@nasa.gov
‡Aerospace Engineer, Thermophysics Materials Branch, NASA Ames Research Center, MS 234-4, Member AIAA, sebastian.v.colom@nasa.gov

I. Introduction

The following summarizes the principal objectives for the BEAST project,

- Centralized and searchable data archive for future reference and study
- Automated data processing, transfer, storage, statistics, and search/retrieval
- Faster data delivery to customers/modeling efforts
- Automated facility modeling via machine learning

More broadly, the data infrastructure seeks to automate all manual tasks related to data processing, transfer, storage, and search/retrieval. This system is designed to manage the data products from arc jet facilities and could be adapted with minimal effort for other similar facilities. The principal motivation is to provide a digital archival capability implementing the FAIR principles of data stewardship: Findable, Accessible, Interoperable, and Reusable. This capability will save various stakeholders (e.g., arc jet principal investigators, facility test engineers, customers) significant time in test planning, debugging, data visualization, modeling, and data processing and more broadly improve the quality of the facility data analysis and data products.

At the same time, the Ames arc jet facilities are undergoing a significant upgrade over the next decade through the Arc Jet Modernization project (AJM). This effort will modernize many aspects of the data systems and is supporting BEAST as a pilot project for a future knowledge management system. Consequently, the BEAST system is being developed to maximize compatibility with the new hardware and data formats.

A. Terminology

The following list describes Ames arc jet data terms which are not self-explanatory:

- **test**: a test series of one or more runs for a given customer or objective
- **run**: a single experimental run where the arc is turned on and off once
- **model**: a calorimeter or material sample which is mounted on a sting arm for insertion into the high enthalpy flow
- **sting arm**: a motorized arm for inserting models into the arc jet flow
- **eu file**: engineering unit file with calibrated data

B. Current data archival practices

Historically, comprehensive arc jet facility data has not been internally archived to ensure that proprietary customer data was not being inadvertently saved. Certain important facility measurement data are tabulated over time (e.g., calorimeter measurements) to track facility performance but other configuration data (e.g., number of cathodes, number of constrictor disks, humidity, air temperature) are not. Additionally, auxiliary diagnostic systems (video, laser diagnostics, and vacuum diagnostics) are collected and stored separately. Data processing for these auxiliary systems is

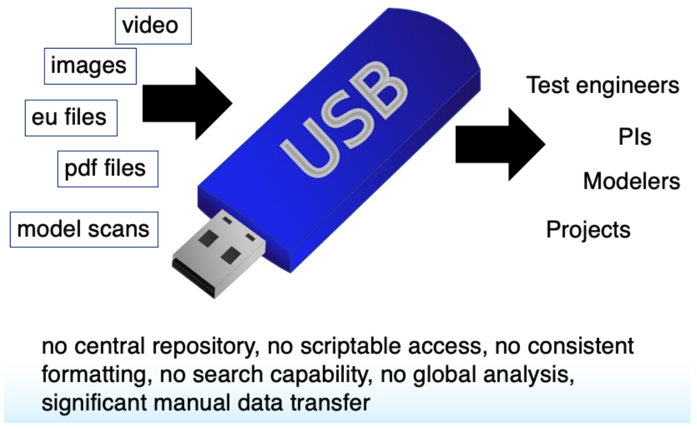

video
images
eu files
pdf files
model scans
USB
Test engineers
PIs
Modelers
Projects
no central repository, no scriptable access, no consistent formatting, no search capability, no global analysis, significant manual data transfer

Fig. 1 Current data archival architecture

done separately and often manually (e.g., video editing) which is typically the time limiting step in data delivery to the customer. Lastly, there is no digital search capability for historical data, settings, or errors. Consequently, the current data infrastructure makes it inconvenient to search historical data, cross-reference certain diagnostics, collect statistics, automate data processing, or quantify uncertainties across multiple tests and/or diagnostic systems.

C. BEAST data archival infrastructure

BEAST seeks to provide a new paradigm for archiving facility data 2. Instead of transferring large excel files on flash drives, BEAST has a central server with automated data upload and processing features (see Sec. II.B). The database then stores facility data and associated metadata in a large set of interlinked tables (see Sec. VII.C for detailed table diagrams). These interlinked tables (also called the database schema) ensure that all uploaded data is saved in a structured, consistent format. Other auxiliary files (video, images, documents) are saved to disk with a unique location and identifier that is saved in a database table.

During or after file uploads, data processing pipelines (see Sec. III) will parse formats, apply calibrations, infer measurements, flag errors, generate automated summary reports/forms, and tabulate statistics. These pipelines use several machine learning (ML) models (see Sec. IV) which enable robust and consistent post-processing. After the data is uploaded and processed, it will be visualized in the browsing interface (Sec. II.A), can be modified via the admin interface (Sec. II.E), and will show up in search results (Sec. II.C).

This infrastructure also implements the FAIR principles of data stewardship:
- Findable: all data can be located and retrieved using the search, API, admin, and browsing interfaces
- Accessible: web interface with SAML authentication (NASA LaunchPad)
- Interoperable: standard input/output formats (CSV, PDF, Excel, JSON)
- Reusable: object-oriented data models with highly cross-linked metadata (sufficient metadata retained for arbitrary future analyses)

The following sections describe each of these features in greater detail.

II. User interfaces

The application contains several interfaces to the database: a data browsing interface, an application programming interface (API), and an admin interface.

A. Data browsing interface

The data browsing interface provides a tree of hyperlinks to browse the application data. It is accessed via the 'Browse' link on the left-side menu bar. This interface allows the user to browse to the appropriate run data via the following tree [Facility → Test → Run]. Once the user reaches the appropriate run page, the interface displays the metadata and graphs of all relevant data as well as links to all related objects/components/configurations (Fig. 3). The automatic data visualization can also be configured to show only the desired diagnostics. This is an important feature as

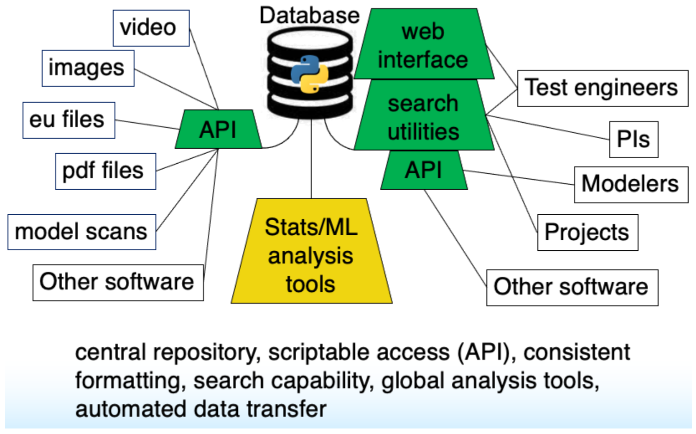


Fig. 2 BEAST data archival architecture

it is significantly more convenient than browsing a static set of graph files or creating many charts in Excel.

Several convenience buttons are also available at the top of the interface,

- Edit- link to edit run data via admin interface
- Process- reruns all relevant data pipelines for current experimental run
- Display Mode- page with large font for runtime display of run settings on overhead monitors
- PDF RunSheet- auto-generates a pre-run checklist pdf
- Export to Excel- exports run data and metadata to an Excel spreadsheet
- PDF Summary- generates a post-run summary report

These convenience functions provide access to the typical tools needed for a particular run. Dropdown menus from the various display sections provide additional features for adding/deleting/editing data and copying metadata from the previous run.

There are also a number of other cross-links to component pages (e.g., nozzle, cathode, constrictor disks), heater conditions pages (e.g., sets of measurements and statistics for specific heater configurations), model description pages, and download links for any attached files including video, images, or spectral data. These cross-links appear automatically when data is uploaded to the database and greatly speed up the process of finding relevant data compared to searching large spreadsheets.

B. File Upload

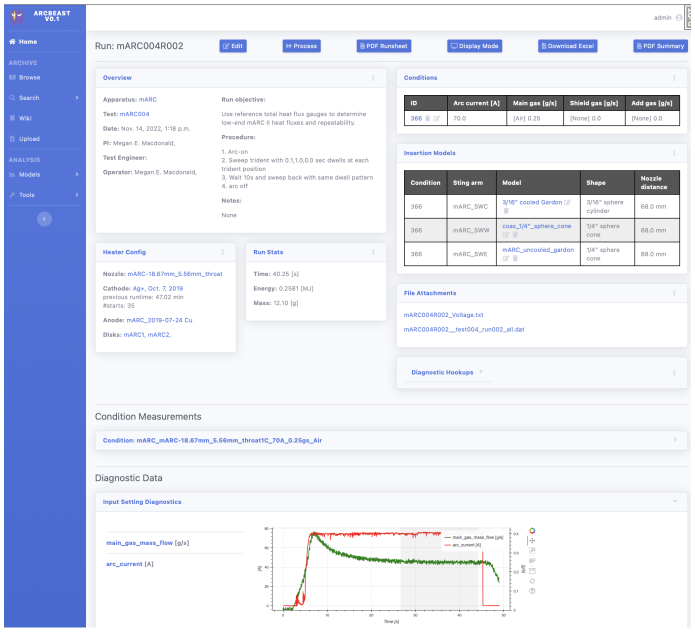
The file upload interface provides a single interface for uploading various data types. The upload requires the following inputs: facility, test number, run number, file, and format type. The currently implemented formats are the internal Ames formats (eu file, hookup file), standard CSV (single header line with column labels), and several metadata formats (PDF III.B, JSON, and plain text). After upload, the interface will return a banner message describing the success or failure to upload. Each file format requires its own data pipeline to parse the file and allocate it into the database.

C. Search interface

The search interface currently has two options (Fig. 5): searching for facility conditions within a certain parameter range and searching for diagnostic time series to compare traces across various runs. These were the two most requested features: (1) being able to find which historical settings/runs had a desired heat flux/stagnation pressure and (2) being able to quickly compare/visualize traces from different runs. Additional search options will likely be added with more user input.

Condition search

The condition search allows filtering using a range of condition parameters and condition measurements (date, stagnation heat flux, stagnation enthalpy, stagnation pressure, apparatus, nozzle, etc.). These fields were suggested by



Run: mARC004R002

Overview

Apparatus: mARC

Test: mARC004

Date: Nov. 14, 2022, 1:18 p.m.

PI: Megan E. Macdonald,

Test Engineer:

Operator: Megan E. Macdonald,

Run objective:

Use reference total heat flux gauges to determine low-end mARC II heat fluxes and repeatability.

Procedure:

1. Arc-on
2. Sweep trident with 0.1,1.0,0.0 sec dwells at each trident position
3. Wait 10s and sweep back with same dwell pattern
4. arc off

Notes:

None

Conditions

ID	Arc current [A]	Main gas [g/s]	Shield gas [g/s]	Add gas [g/s]
366	70.0	[Air] 0.25	[None] 0.0	[None] 0.0

Insertion Models

Condition	Sting arm	Model	Shape	Nozzle distance
366	mARC_SWC	3/16" cooled Gardon	3/16" sphere cylinder	68.0 mm
366	mARC_SWW	coax_1/4"_sphere_cone	1/4" sphere cone	68.0 mm
366	mARC_SWE	mARC_uncooled_gardon	1/4" sphere cone	68.0 mm

Heater Config

Nozzle: mARC-18.67mm_5.56mm_throat

Cathode: Ag+, Oct. 7, 2019
previous runtime: 47.02 min
#starts: 35

Anode: mARC_2019-07-24 Cu

Disks: mARC1, mARC2,

Run Stats

Time: 40.25 [s]

Energy: 0.2581 [MJ]

Mass: 12.10 [g]

File Attachments

mARC004R002_Voltage.txt

mARC004R002__test004_run002_all.dat

Diagnostic Hookups

Condition Measurements

Condition: mARC_mARC-18.67mm_5.56mm_throat1C_70A_0.25gs_Air

Diagnostic Data

Input Setting Diagnostics

main_gas_mass_flow [g/s]

arc_current [A]

Fig. 3 Frontend interface showing a detail page for a mARC II experimental run with only calorimeter insertions. Several cards display metadata for run setup, model configuration, heater settings, run stats, hookup definitions, and run measurements. All diagnostic time series graphs can be browsed directly below the metadata information (only one of many plots is visible in this screenshot). These plotting windows are also interactive with various capabilities (e.g., zoom, pan, export png, reset).

various arc jet principal investigators based on the typical searches they do when preparing for testing. The results are returned below the form similar to the format of google search results with a list of links to relevant condition pages coupled with a short metadata summary.

Diagnostic search/compare

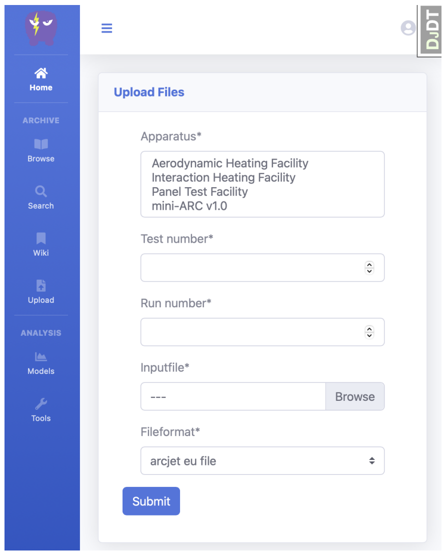
The diagnostic search/compare interface allows the user to select a set of runs and diagnostics and plot the resulting time series. If multiple diagnostics are selected, then each diagnostic is plotted on a different graph. The right side of Figure 5 shows a search for the arc current diagnostic over a set of 5 runs on the mARC facility. This interface enables quick comparisons across many different sets of runs.

D. Application programming interface (API)

The API is created using the Django REST Framework (DRF) library. It contains URL access points for data retrieval of the important models defined in the data schema and provides large scale data upload capability (Fig. 6). The API endpoints use the typical JSON serialization of database objects. Due to security concerns, the API access is restricted to admin users in the existing deployments of the BEAST framework and is mainly used for large scale data uploads.

E. Admin interface

The admin interface provides user view and edit access to the database objects including users, permissions, and permission groups. The admin interface is not designed for public use and is only available to admin users. However, it provides an important capability to access and control the application without requiring direct server access or changes to the source code.



Upload Files

Apparatus*
Aerodynamic Heating Facility
Interaction Heating Facility
Panel Test Facility
mini-ARC v1.0

Test number*

Run number*

Inputfile*

Fileformat*
arcjet eu file

Fig. 4 Screenshot of upload interface: image shows the necessary inputs for file upload. This image also shows how the HTML interface (Bootstrap4) dynamically adapts to a smaller screen/window)

III. Data pipelines

The BEAST application contains a large number of data pipelines for various processing tasks: calibration, file parsing, report generation, measurement extraction, metadata extraction, etc. The following sections provide an overview of a few of the most important pipelines.

A. Engineering Unit (eu) file parsing

Engineering unit (eu) files are the standard output file format from the arc jet facilities. The database has a simple pipeline to parse these files and input the relevant data into the database. The files are in tab-delimited format making parsing simple. The data ingestion of the files consists of parsing the header information (names, dates, times, models, diagnostics, units), cleaning the data (casting names to lower case, fuzzy matching of cleaned column headers to database equivalents), and then inputting the cleaned/matched data into the database schema.

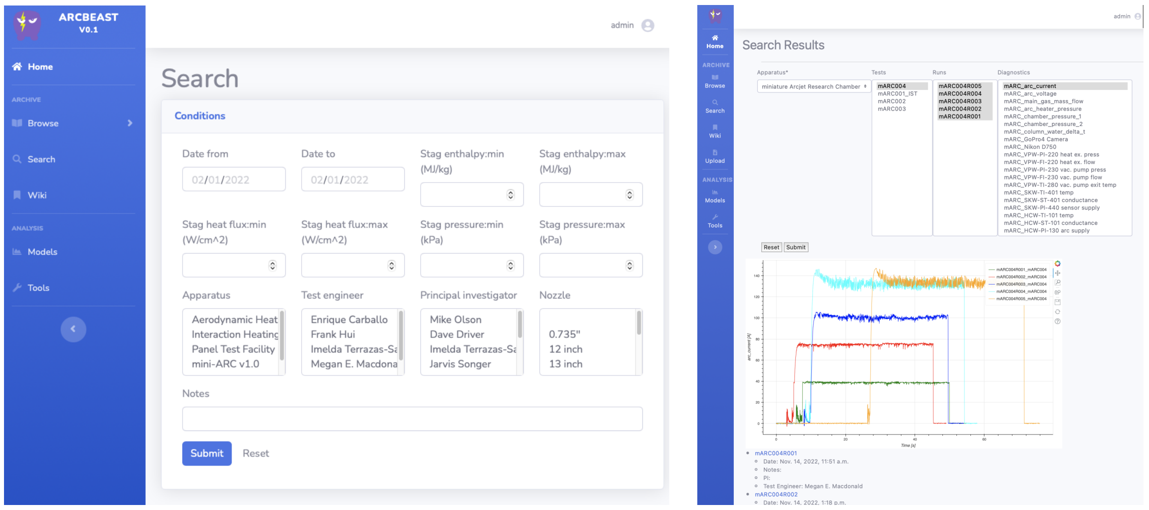
B. PDF parsing

A large percentage of metadata (condition settings, personnel, measurements, objectives, notes, etc.) for historical arc jet tests is only available in PDF summary files. Many of these files are scanned images embedded as PDF pages so no digital text is available. Consequently, to recover and collate this data, two capabilities are needed: (1) extraction of digital text in PDF pages and (2) extraction of text from images embedded in PDF pages.

Several python libraries were leveraged to provide this capability (pytesseract, pyPDF4). The resulting parsing capability allows for PDF text extraction from documents with basic text and/or mixed images. This simplifies metadata extraction greatly since only one tool is needed to digitize a mixed PDF document with both scans and text pages.

The extraction operates by first separating scanned image pages and regular text pages in the PDF. Then, the image pages are parsed using the pytesseract library (Google's optical character recognition package) and text is extracted from the regular PDF pages using the pyPDF4 library. The total text from all pages is then combined and parsed into a Python dictionary object using a set of regular expressions which identify relevant metadata. The dictionary is then directly ingested into the database.

This PDF parsing capability coupled with the API is significantly more efficient than manual data entry and is broadly compatible with a wide variety of current and legacy PDF records. To date, this pipeline has been used to automate mining of historical metadata from old arc jet tests without comprehensive digital text records.



Search

Conditions

Date from	Date to	Stag enthalpy:min (MJ/kg)	Stag enthalpy:max (MJ/kg)
02/01/2022	02/01/2022		
Stag heat flux:min (W/cm^2)	Stag heat flux:max (W/cm^2)	Stag pressure:min (kPa)	Stag pressure:max (kPa)
Apparatus	Test engineer	Principal investigator	Nozzle
Aerodynamic Heat Interaction Heating Panel Test Facility mini-ARC v1.0	Enrique Carballo Frank Hui Imelda Terrazas-Sa Megan E. Macdona	Mike Olson Dave Driver Imelda Terrazas-Sa Jarvis Songer	0.735" 12 inch 13 inch

Notes

Submit Reset

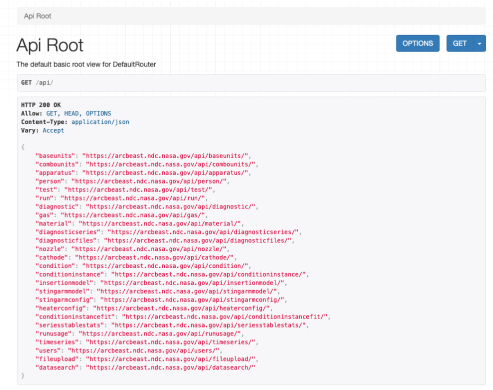
Search Results

Apparatus*	Tests	Runs	Diagnostics
miniature Arcjet Research Chamber	mARC004 mARC001_IST mARC002 mARC003	mARC004R005 mARC004R004 mARC004R003 mARC004R002 mARC004R001	mARC_arc_current mARC_arc_voltage mARC_main_gas_mass_flow mARC_arc_heater_pressure mARC_chamber_pressure_1 mARC_chamber_pressure_2 mARC_column_water_delta_t mARC_GoPro4 Camera mARC_Nikon D750 mARC_VPW-PI-220 heat ex. press mARC_VPW-FI-220 heat ex. flow mARC_VPW-PI-230 vac. pump press mARC_VPW-FI-230 vac. pump flow mARC_VPW-TI-280 vac. pump exit temp mARC_SKW-TI-401 temp mARC_SKW-ST-401 conductance mARC_SKW-PI-440 sensor supply mARC_HCW-TI-101 temp mARC_HCW-ST-101 conductance mARC_HCW-PI-130 arc supply

Reset Submit

- mARC004R001
  - Date: Nov. 14, 2022, 11:51 a.m.
  - Notes:
  - PI:
  - Test Engineer: Megan E. Macdonald
- mARC004R002
  - Date: Nov. 14, 2022, 1:18 p.m.

Fig. 5 (Left) Condition search interface form. None of the fields are required, allowing for flexible filtering of facility conditions. (Right) Diagnostic search form with resulting plot of a single diagnostic (arc current) across several runs.



Api Root

The default basic root view for DefaultRouter

```
GET /api/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "baseunits": "https://arcbeast.ndc.nasa.gov/api/baseunits/",
    "combounits": "https://arcbeast.ndc.nasa.gov/api/combounits/",
    "apparatus": "https://arcbeast.ndc.nasa.gov/api/apparatus/",
    "person": "https://arcbeast.ndc.nasa.gov/api/person/",
    "test": "https://arcbeast.ndc.nasa.gov/api/test/",
    "run": "https://arcbeast.ndc.nasa.gov/api/run/",
    "diagnostic": "https://arcbeast.ndc.nasa.gov/api/diagnostic/",
    "gas": "https://arcbeast.ndc.nasa.gov/api/gas/",
    "material": "https://arcbeast.ndc.nasa.gov/api/material/",
    "diagnosticseries": "https://arcbeast.ndc.nasa.gov/api/diagnosticseries/",
    "diagnosticfiles": "https://arcbeast.ndc.nasa.gov/api/diagnosticfiles/",
    "nozzle": "https://arcbeast.ndc.nasa.gov/api/nozzle/",
    "cathode": "https://arcbeast.ndc.nasa.gov/api/cathode/",
    "condition": "https://arcbeast.ndc.nasa.gov/api/condition/",
    "conditioninstance": "https://arcbeast.ndc.nasa.gov/api/conditioninstance/",
    "insertionmodel": "https://arcbeast.ndc.nasa.gov/api/insertionmodel/",
    "stingarmmodel": "https://arcbeast.ndc.nasa.gov/api/stingarmmodel/",
    "stingarmconfig": "https://arcbeast.ndc.nasa.gov/api/stingarmconfig/",
    "heaterconfig": "https://arcbeast.ndc.nasa.gov/api/heaterconfig/",
    "conditioninstancefit": "https://arcbeast.ndc.nasa.gov/api/conditioninstancefit/",
    "seriesstablestats": "https://arcbeast.ndc.nasa.gov/api/seriesstablestats/",
    "runusage": "https://arcbeast.ndc.nasa.gov/api/runusage/",
    "timeseries": "https://arcbeast.ndc.nasa.gov/api/timeseries/",
    "users": "https://arcbeast.ndc.nasa.gov/api/users/",
    "fileupload": "https://arcbeast.ndc.nasa.gov/api/fileupload/",
    "datasearch": "https://arcbeast.ndc.nasa.gov/api/datasearch/"
}
```

Fig. 6 Visualization of BEAST API URL endpoints

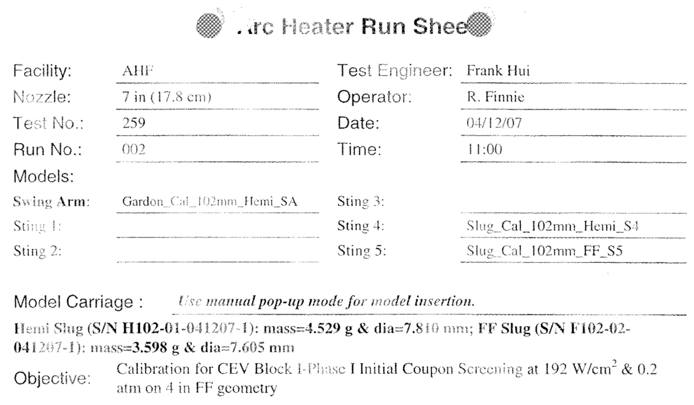
C. Condition statistics

Arc jet steady state conditions are typically defined by an arc current, nozzle diameter, and mass flows (main gas, add gas, shield gas). Determining the start and stop times of these conditions requires parsing the steady-state intervals of each input setting, finding the intersections of those intervals to determine the set of unique settings, and then matching those settings to a particular database condition entry. For runs with more than two conditions, this can be manually onerous. Consequently, it is of interest to automate this process.

This pipeline automates acquisition of these statistics by segmenting steady-state intervals for each input setting using the 1D convolutional neural net (CNN) model described in Section IV.B. Then the intersections of these steady-state intervals are taken to get the unique condition intervals (Figure 8). Once these intervals are determined, it is simple to calculate simple statistics for all diagnostics during an individual condition instance (e.g., mean, max, min, stdev). Tabulating these statistics is important as it quantifies the variation present in a given steady-state condition.

Historically, this profusion of unique conditions (> 300 per facility) made manual statistical tracking too arduous. However, this condition parsing pipeline makes it possible to record every instance of every condition and calculate associated metrics. This provides the first statistical uncertainty quantification for each condition.

Currently this pipeline provides automated statistics for system level diagnostics (arc current, arc voltage, arc pressure, box pressure etc.) and omits any insertion diagnostics (model TCs, calorimeters, etc.).



.rc Heater Run Shee

Facility:	AHF	Test Engineer:	Frank Hui
Nozzle:	7 in (17.8 cm)	Operator:	R. Finnie
Test No.:	259	Date:	04/12/07
Run No.:	002	Time:	11:00
Models:			
Swing Arm:	Gardon_Cal_102mm_Hemi_SA	Sting 3:	
Sting 1:		Sting 4:	Slug_Cal_102mm_Hemi_S4
Sting 2:		Sting 5:	Slug_Cal_102mm_FF_S5

Model Carriage : *Use manual pop-up mode for model insertion.*

Hemi Slug (S/N H102-01-041207-1): mass=4.529 g & dia=7.810 mm; FF Slug (S/N F102-02-041207-1): mass=3.598 g & dia=7.605 mm

Objective: Calibration for CEV Block I-Phase I Initial Coupon Screening at 192 W/cm^2 & 0.2 atm on 4 in FF geometry

Fig. 7 Typical scanned runsheet from historical arc jet test series. The quality of the scans can be quite poor but the optical text recognition software (pytesseract) will still recover enough text to extract a large fraction of the metadata of interest.

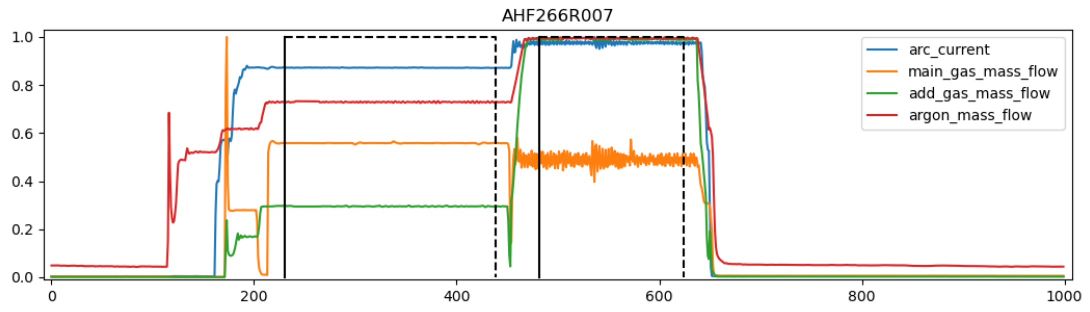


Fig. 8 Automated condition segmentation of normalized input time series. Black dotted traces indicate segmented steady-state condition intervals. This trace was chosen to demonstrate the large variation in noise even between conditions within a single run.

D. Measurement pipelines

A number of pipelines are devoted to processing diagnostic time series data to extract measurements. Currently, the following measurement pipelines have been implemented:

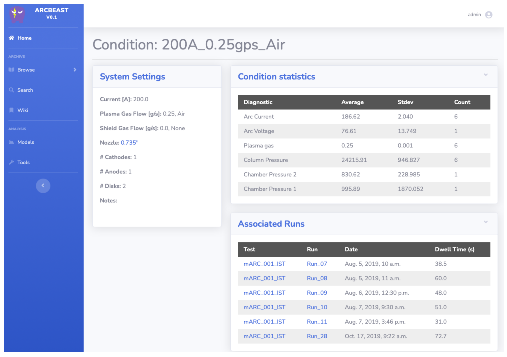
- Bulk enthalpy measurement from water cooling (EB2)
- Sonic enthalpy pipeline
- Slug Calorimeter pipeline
- Gardon gauge pipeline
- Coax TC Calorimeter pipeline

Running these pipelines will generate typed measurement objects which are attached to the relevant arc jet condition as well as plots of the raw diagnostic data during the measurement time interval. These measurement values and plots then become available via the search interface, any associated object pages (e.g., run page, condition page, insertion model page), and will be included in the report outputs. These measurement pipelines have been ported from existing Igor scripts used by the arc jet facilities to process their data.

IV. Machine Learning Models

A. N-dim Linear Models

The database has an automated capacity to fit N-dimensional linear models to facility metrics. This provides a simple, interpretable, and data-driven model for all output parameters from an arbitrary number of input variables. The



Condition: 200A_0.25gps_Air

System Settings

Current [A]: 200.0

Plasma Gas Flow [g/s]: 0.25, Air

Shield Gas Flow [g/s]: 0.0, None

Nozzle: 0.735"

# Cathodes: 1

# Anodes: 1

# Disks: 2

Notes:

Condition statistics

Diagnostic	Average	Stdev	Count
Arc Current	186.62	2.040	6
Arc Voltage	76.61	13.749	1
Plasma gas	0.25	0.001	6
Column Pressure	24215.91	946.827	6
Chamber Pressure 2	830.62	228.985	1
Chamber Pressure 1	995.89	1870.052	1

Associated Runs

Test	Run	Date	Dwell Time (s)
mARC_001_IST	Run_07	Aug. 5, 2019, 10 a.m.	38.5
mARC_001_IST	Run_08	Aug. 5, 2019, 11 a.m.	60.0
mARC_001_IST	Run_09	Aug. 6, 2019, 12:30 p.m.	48.0
mARC_001_IST	Run_10	Aug. 7, 2019, 9:30 a.m.	51.0
mARC_001_IST	Run_11	Aug. 7, 2019, 3:46 p.m.	31.0
mARC_001_IST	Run_28	Oct. 17, 2019, 9:22 a.m.	72.7

Fig. 9 Frontend interface displaying a mARC facility condition. Shows condition statistics for primary diagnostics and all associated runs at this condition.

current set of input variables contains six numerical variables (arc current, main gas flow, add gas flow, shield gas flow, model stagnation distance, nozzle diameter) and three categorical variables (main gas, add gas, model shape). The categorical variables are one-hot encoded (e.g., a dimension is added for each category option) so the total number of input dimensions is quite large (N-dim = 6 + 24 = 30). However, this large number of dimensions provides a potentially better fit than a simple 1-2 dimension fit by considering all input variables simultaneously.

Figure 10 shows the N-dim model residuals for the bulk enthalpy from water cooling (EB2, $R^2 = 0.77$) for the IHF (Integrated Heating Facility). For comparison, the best analytic correlations for IHF bulk enthalpy from water cooling (EB2) currently have $R^2 <= 0.61$ [1]. These N-dim models are also more comprehensive than typical engineering correlations since they can predict values across the entire input parameter space.

The N-dim models are automatically updated when new data is added and are used to flag outlier measurements. Currently, these models are only created for a subset of output variables (e.g., arc pressure, arc voltage, stagnation heat flux, stagnation enthalpy, stagnation pressure, sonic enthalpy, and bulk enthalpy from water cooling).

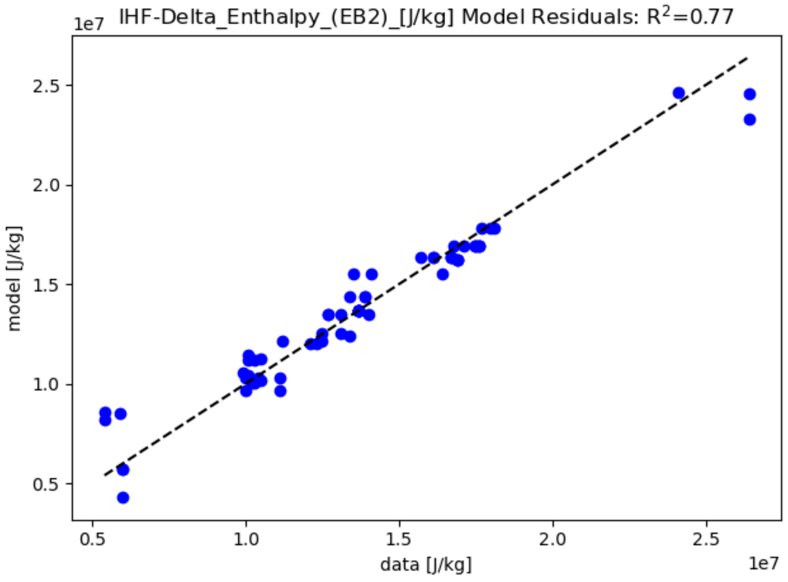


Fig. 10 Plots of N-dim model residuals for the bulk enthalpy from water cooling (EB2, $R^2 = 0.77$) for the IHF (Integrated Heating Facility).

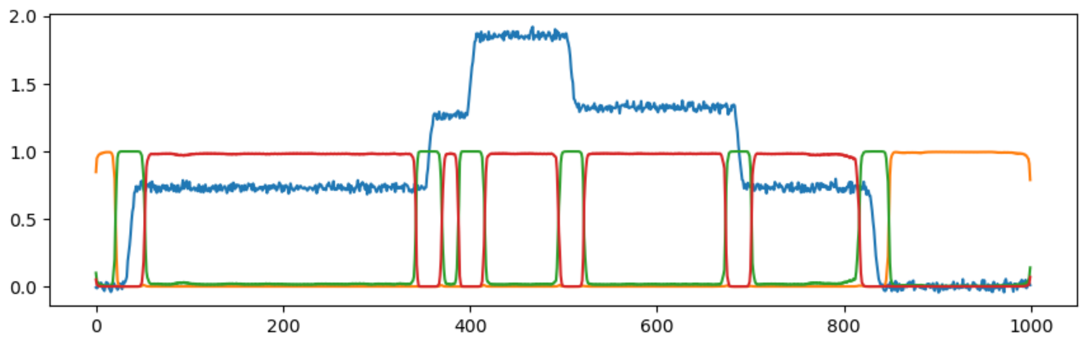


Fig. 11 Plot of normalized current trace (blue) and classifications by 1D CNN model (red: on-condition, green: transition, orange: off-condition

B. 1-dim Convolutional Neural Net (CNN)

Manual time segmentation is complex for arc jet data as there are at least four input variables (current, primary gas flow, add gas flow, shield gas flow, etc.) which change over the course of a run. A facility condition is only reached when all input settings are held constant for a period of time. Consequently, time segmentation for facility conditions must identify all time intervals where all input conditions are constant. This is a time-consuming and subjective process if completed manually.

A new machine learning (ML) model was developed to automate this time-segmentation process. A ML model was chosen since a variety of simple thresholding algorithms (value-based, derivative-based, and integral-based) were insufficiently robust. Since the arcjet data is a different length every run, the model had to be able to process arbitrary length sequences and tolerate a wide variety of noise levels. With these constraints, a new 1D convolution neural net (CNN) architecture was developed for semantic segmentation of time series. Existing ML time series segmentation methods typically classify sub-sequences of a fixed length with a single label. This network instead provides semantic classification (e.g., classifies each point in a series rather than giving a single label to the full sequence) and, due to its fully convolutional character, can be applied to arbitrary length sequences. The precise architecture will be omitted here since this work is currently unpublished and a draft paper is currently being written.

The resulting classifier model performs much better than previous thresholding algorithms on difficult cases. Figure 11 shows a typical example of a multi-condition time series trace and the resulting classification by the CNN. The CNN is trained to identify 3 classes: off-condition, on-condition, and transitions. This new robust classifier makes the condition statistics data pipeline described in Sec. III.C possible.

V. Automated Report Generation

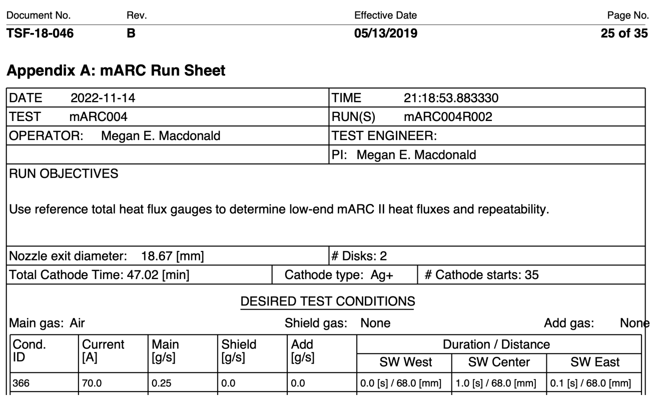
One of the advantages of a centralized digital database is the ability to quickly synthesize data summaries as all the measurements, images, plots, and metadata are cross-linked and accessible. The following sections describe several automated report generation pipelines that are currently implemented in BEAST.

A. Run sheets

Preparing for an arc jet run typically requires a run sheet describing the specific run, target conditions, and planned model insertions. This is currently filled in by hand or keyboard by the test engineer. However, if the appropriate metadata has been previously uploaded to the BEAST system, the run sheet can be automatically generated by BEAST and exported to PDF. This save a great deal of personnel time in filling out/scanning forms, especially for test series with a large number of runs.

B. Run summary PDF

Generation of quick-look summary reports is a critical feature to ensure quick data delivery to customers after a experimental run. The current implementation of this in BEAST produces a PDF file with several text sections (run objectives, notes, associated personnel, heater configurations, model configurations, heater conditions, and calorimeter


Document No. TSF-18-046 | Rev. B | Effective Date 05/13/2019 | Page No. 25 of 35

Appendix A: mARC Run Sheet

DATE 2022-11-14	TIME 21:18:53.883330
TEST mARC004	RUN(S) mARC004R002
OPERATOR: Megan E. Macdonald	TEST ENGINEER:
	PI: Megan E. Macdonald

RUN OBJECTIVES

Use reference total heat flux gauges to determine low-end mARC II heat fluxes and repeatability.

Nozzle exit diameter: 18.67 [mm] | # Disks: 2

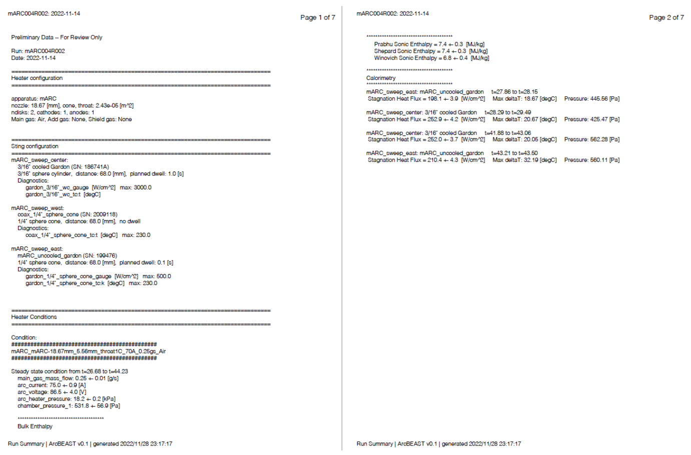
Total Cathode Time: 47.02 [min] | Cathode type: Ag+ | # Cathode starts: 35

DESIRED TEST CONDITIONS

Main gas: Air Shield gas: None Add gas: None

Cond. ID	Current [A]	Main [g/s]	Shield [g/s]	Add [g/s]	Duration / Distance		
					SW West	SW Center	SW East
366	70.0	0.25	0.0	0.0	0.0 [s] / 68.0 [mm]	1.0 [s] / 68.0 [mm]	0.1 [s] / 68.0 [mm]

Fig. 12 Screenshot of auto-generated runsheet for mARC facility


```
mARC004R002: 2022-11-14                                Page 1 of 7

Preliminary Data -- For Review Only

Run: mARC004R002
Date: 2022-11-14

==========================================================================
Heater configuration
==========================================================================

apparatus: mARC
nozzle: 18.67 [mm], cone, throat: 2.43e-05 [m^2]
ndisks: 2, cathodes: 1, anodes: 1
Main gas: Air, Add gas: None, Shield gas: None


==========================================================================
Sting configuration
==========================================================================
mARC_sweep_center:
   3/16" cooled Gardon (SN: 186741A)
   3/16" sphere cylinder,  distance: 68.0 [mm],  planned dwell: 1.0 [s]
   Diagnostics:
       gardon_3/16"_wc_gauge  [W/cm^2]  max: 3000.0
       gardon_3/16"_wc_tc:t  [degC]

mARC_sweep_west:
   coax_1/4"_sphere_cone (SN: 2009118)
   1/4" sphere cone,  distance: 68.0 [mm],  no dwell
   Diagnostics:
       coax_1/4"_sphere_cone_tc:t  [degC]  max: 230.0

mARC_sweep_east:
   mARC_uncooled_gardon (SN: 199476)
   1/4" sphere cone,  distance: 68.0 [mm],  planned dwell: 0.1 [s]
   Diagnostics:
       gardon_1/4"_sphere_cone_gauge  [W/cm^2]  max: 500.0
       gardon_1/4"_sphere_cone_tc:k  [degC]  max: 230.0


==========================================================================
Heater Conditions
==========================================================================

Condition:
##############################################
mARC_mARC-18.67mm_5.56mm_throat1C_70A_0.25gs_Air
##############################################

Steady state condition from t=26.68 to t=44.23
   main_gas_mass_flow: 0.25 +- 0.01 [g/s]
   arc_current: 75.0 +- 0.9 [A]
   arc_voltage: 86.5 +- 4.0 [V]
   arc_heater_pressure: 18.2 +- 0.2 [kPa]
   chamber_pressure_1: 531.8 +- 56.9 [Pa]

   ****************************************
   Bulk Enthalpy

Run Summary | ArcBEAST v0.1 | generated 2022/11/28 23:17:17
```

```
mARC004R002: 2022-11-14                                Page 2 of 7

   ****************************************
     Prabhu Sonic Enthalpy = 7.4 +- 0.3  [MJ/kg]
     Shepard Sonic Enthalpy = 7.4 +- 0.3  [MJ/kg]
     Winovich Sonic Enthalpy = 6.8 +- 0.4  [MJ/kg]

   ****************************************
   Calorimetry
   ****************************************
   mARC_sweep_east: mARC_uncooled_gardon    t=27.86 to t=28.15
    Stagnation Heat Flux = 198.1 +- 3.9  [W/cm^2]    Max deltaT: 18.67 [degC]    Pressure: 445.56 [Pa]

   mARC_sweep_center: 3/16" cooled Gardon    t=28.29 to t=29.49
    Stagnation Heat Flux = 252.9 +- 4.2  [W/cm^2]    Max deltaT: 20.67 [degC]    Pressure: 425.47 [Pa]

   mARC_sweep_center: 3/16" cooled Gardon    t=41.88 to t=43.06
    Stagnation Heat Flux = 252.0 +- 3.7  [W/cm^2]    Max deltaT: 20.05 [degC]    Pressure: 562.28 [Pa]

   mARC_sweep_east: mARC_uncooled_gardon    t=43.21 to t=43.50
    Stagnation Heat Flux = 210.4 +- 4.3  [W/cm^2]    Max deltaT: 32.19 [degC]    Pressure: 560.11 [Pa]

Run Summary | ArcBEAST v0.1 | generated 2022/11/28 23:17:17
```

Fig. 13 First two pages of auto-generated run summary for mARC run mARC004R002. The run objectives, associated personnel, configurations, conditions, and calorimeter measurements are all output in human readable format into a PDF document. The following pages contain annotated plots of relevant system diagnostics and calorimeter measurements.

measurements, see Fig. 13), several pages of plots showing system diagnostics with labeled condition intervals, and lastly a set of annotated plots of model insertion diagnostics.

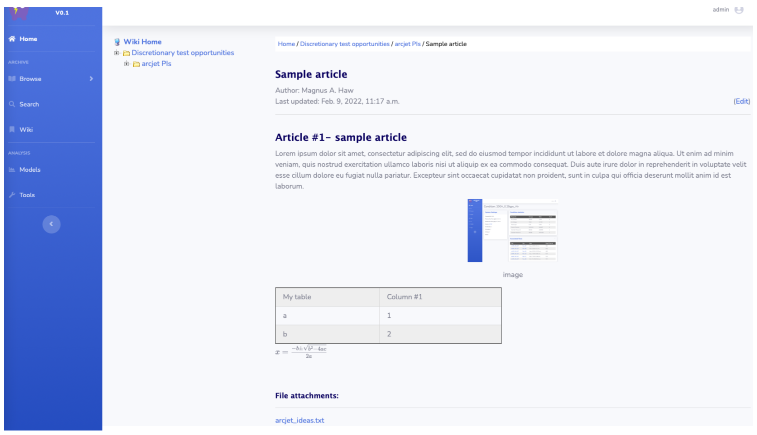
C. Excel output

The data and summary text for a given run can also be exported to an Excel file. The first tab will contain the same summary text as the PDF summary. Time series data from a given data acquisition system is output into its own separate tab in standard columnar format (header row with column names, first column is time, etc.). Given the complexity of many arc jet runs, an Excel file was simpler to export than a set of CSV files.

VI. Additional Features

A. Lessons learned wiki

A simple lessons learned wiki has been implemented. This system is organized into three primary tiers (Category → Topic → Subtopic) and each tier can have attached articles and files. Each article is similar to a blog post and can contain text, images, and LaTeX equations (Figure 14). Two navigation options are visualized: an expandable outline on the left hand side, and linked breadcrumb navigation at the top (e.g., Home / MyCategory / MyTopic / Myarticle). Providing an internal wiki is convenient for cross-referencing and linking data to the articles while writing them.



Wiki Home
Discretionary test opportunities
arcjet PIs

Home / Discretionary test opportunities / arcjet PIs / Sample article

Sample article

Author: Magnus A. Haw
Last updated: Feb. 9, 2022, 11:17 a.m. (Edit)

Article #1– sample article

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

image

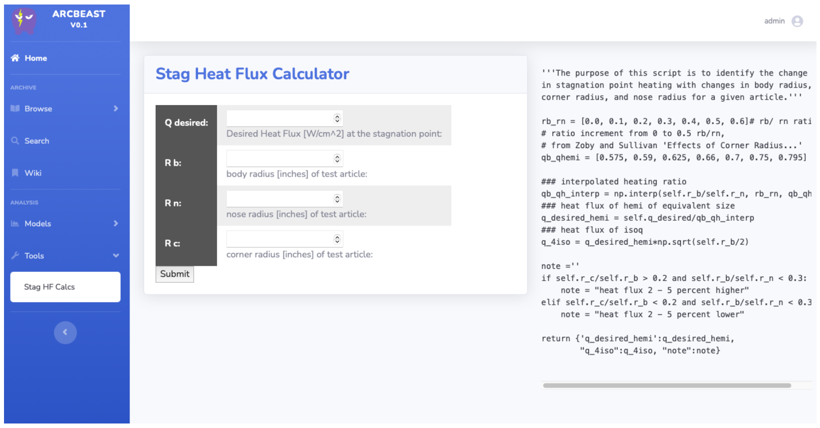
My table	Column #1
a	1
b	2

$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

File attachments:

arcjet_ideas.txt

Fig. 14 Wiki sample article showing image, table, text, link, file attachment, and equation capabilities.



Stag Heat Flux Calculator

Q desired:	Desired Heat Flux [W/cm^2] at the stagnation point:
R b:	body radius [inches] of test article:
R n:	nose radius [inches] of test article:
R c:	corner radius [inches] of test article:

```
'''The purpose of this script is to identify the change
in stagnation point heating with changes in body radius,
corner radius, and nose radius for a given article.'''

rb_rn = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]# rb/ rn rati
# ratio increment from 0 to 0.5 rb/rn,
# from Zoby and Sullivan 'Effects of Corner Radius...'
qb_qhemi = [0.575, 0.59, 0.625, 0.66, 0.7, 0.75, 0.795]

### interpolated heating ratio
qb_qh_interp = np.interp(self.r_b/self.r_n, rb_rn, qb_qh
### heat flux of hemi of equivalent size
q_desired_hemi = self.q_desired/qb_qh_interp
### heat flux of isoq
q_4iso = q_desired_hemi*np.sqrt(self.r_b/2)

note =''
if self.r_c/self.r_b > 0.2 and self.r_b/self.r_n < 0.3:
    note = "heat flux 2 - 5 percent higher"
elif self.r_c/self.r_b < 0.2 and self.r_b/self.r_n < 0.3
    note = "heat flux 2 - 5 percent lower"

return {'q_desired_hemi':q_desired_hemi,
        "q_4iso":q_4iso, "note":note}
```

Fig. 15 Heat flux calculator tool: calculates equivalent heat flux for iso-q and hemi insertion model geometries.

B. Calculation tools

Arc jet PIs requested that various calculation tools be integrated into the BEAST interface. This makes the tools accessible, consistent, and organized whereas previously they only existed as scripts or spreadsheets on an individual's computer. The first such tool implemented is a simple calculator for heat flux equivalence between the iso-q and hemi model geometries (Fig. 15).

VII. Software Architecture

The software architecture uses the Django web-application framework [2] and the Python [3] data stack coupled to an enterprise database. This architecture was chosen to minimize complexity and ensure long-term maintainability (single language, object-relational model, built-in security, monolithic application, database agnostic, internal data science packages). Python will continue to be the language of choice for data science and the Django framework will likely be supported for decades to come. In-house Python programmers will also be easy to obtain as compared with any other language. Other key packages include the Django REST Framework for creating the API, numpy/pandas/sci-kit-learn/Tensorflow for data processing and Bokeh for plotting/data visualization.

The Django web-framework is chosen instead of a desktop application because the database needs to securely ingest and export data to various sources. This does not require the application to be connected to the internet but provides a simple way to standardize the interface on all devices and provide encrypted data transfer (i.e., all users will have access to and be familiar with a browser application).

The visual interface is implemented using Bootstrap4 templates in combination with the Django template engine. This allows for minimal coding and a uniform styling across the whole application. The inclusion of Bootstrap4 also makes the layout dynamic and suitable for desktop, tablet, and mobile use.

The application is also designed to be database agnostic and can be deployed using any common enterprise database (MySQL, SQLite, Postgres, MongoDB, etc.).

Fig. 16 Software architecture makes use of the Python language and the Django web-application framework

A. Django application structure

The application follows the Django convention of organizing different tasks into ’apps’ each with their own folder. These apps are modular and can be reused in other Django projects. Each app folder has a set of python files which define the app’s data schema (models.py), interface (views.py, templates/*.html), admin interface (admin.py), and url paths (urls.py). conforming to the Model-Template-View paradigm .The BEAST application has the following folder structure:

- **data/**: primary app which contains run, test, apparatus, diagnostic and calibration related models.
- **system/**: app for tracking component configurations and histories (i.e., disks, nozzles, cathodes, etc.)
- **stats/**: app for tracking statistics, measurements, and ML models.
- **units/**: app defining unit schema and conversion methods
- **pdfs/**: app defining pdf parsing and conversion to text methods
- **wiki/**: app for wiki pages
- **myapi/**: app for application programming interface (API)
- **static/**: directory housing static file resources (css, js, logos)
- **media/**: directory organizing file uploads (images, videos, pdfs, etc.)
- **project_settings/**: app containing global application settings.

B. Python environment

The primary dependencies are Django, Bokeh, Numpy, Scikit-learn, and Keras/Tensorflow. Since the application is pure python, all the packages/dependencies are cross-platform and can be freely downloaded. A full list of packages (~230) can be found in the top directory (environment.yml). However, the latest versions of all packages are not necessarily available on all operating systems (Tensorflow does not yet work with Apple M1 chips). Consequently, specific python environments are recommended/supplied for each operating system.

C. Database Schema

The database is defined using the Django object relational model (ORM) system where data tables and relationships are defined by Python classes and the Django framework translates all interactions to and from SQL. This has the advantage of only needing to know/use Python and being able to easily modify the database schema using the Django migration tool. The data is partitioned into three top-level categories: data, system, and stats. Although these categories have significant overlap, it provides a useful division to encapsulate different types of information. These categories are implemented as separate Django app folders in the software.

Data

The data schema includes the configuration tables for experimental test series, experimental runs, diagnostics, diagnostic calibrations, and diagnostic time series. The schema is graphed in Figure 17.

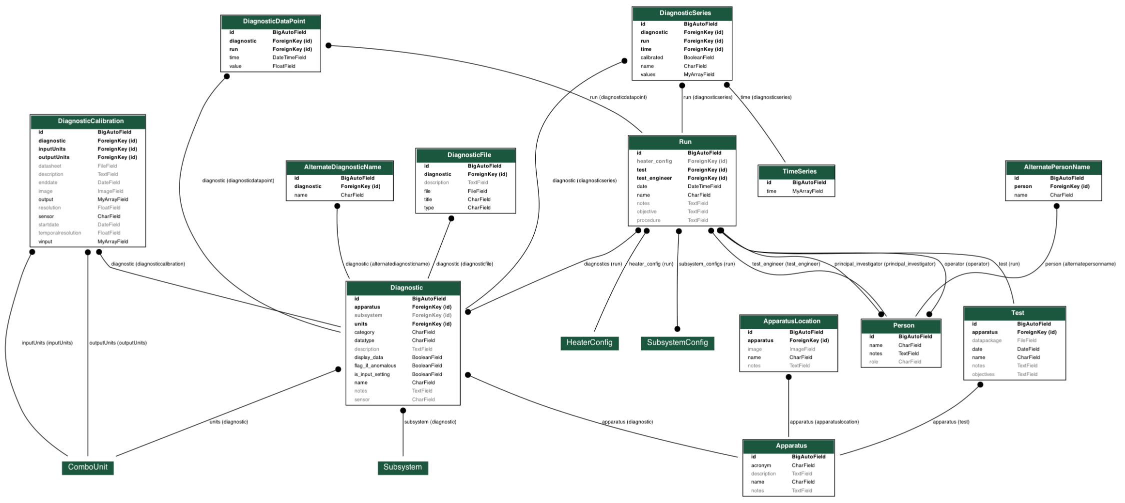


Fig. 17 Data model database schema: each rectangle represents a different database table and the lines represent relationships/references between tables. The central tables are the Diagnostic (center-left) and Run (center-right) tables which each connect to several sub-tables.

System

The system schema comprises all of the system components including nozzles, disks, sting arms, insertion models, and various configuration aggregation tables (e.g., heater config, condition config, subsystem config, camera config, etc.).

Stats

The stats schema covers all the measured or calculated values associated with the data. This was intentionally separated from the raw data schema to ensure that the original data would remain unchanged and that calculated values would not be misinterpreted as a primary source. These schema includes tables for averages, standard deviations, flags, and machine learning models. The schema is graphed in Figure 18.

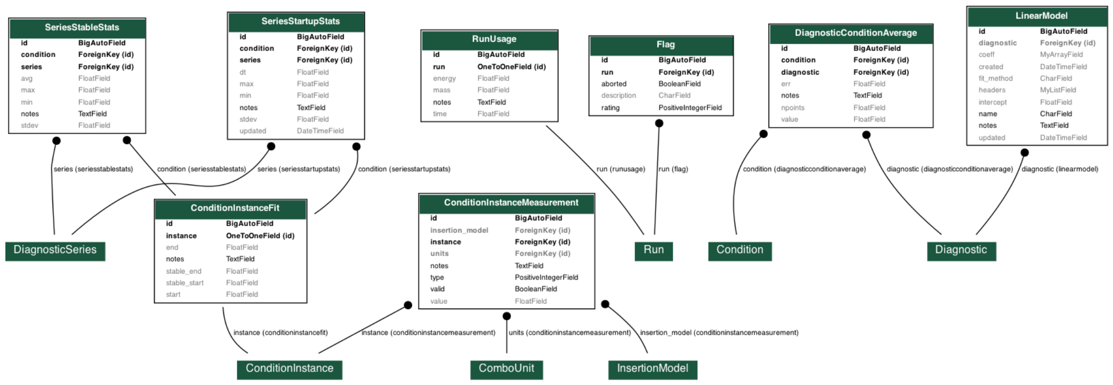


Fig. 18 Stats database schema: each rectangle represents a different database table and the lines represent relationships/references between tables.

Units

The application also contains a unit conversion system which is contained within its own app folder. This stores units as a combinations of the seven fundamental quantities (mass, time, length, temperature, luminosity, mole, and current). This is necessary given the mixed use of metric and English units in the arc jet facilities. The unit schema is shown in Figure 17. Unit checking and global conversion methods are built into each object class to simplify data aggregations and comparisons.

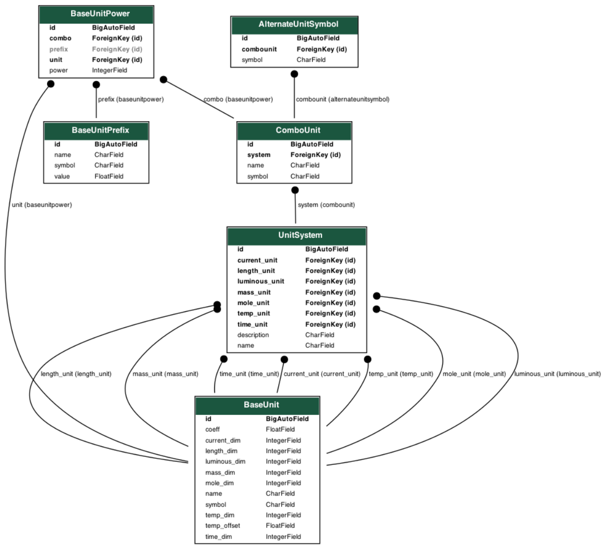


Fig. 19 Database schema for units: the primary class "ComboUnit" is constructed from a combination of multiple "BaseUnits" and "BaseUnitPower" objects and can be associated with a "UnitSystem" such as metric or English.

Wiki

The wiki schema consists of four models: Categories, Topics, Subtopics, and HtmlBlocks. The first three models implement a tiered organization (topics belong to a category, subtopics belong to a topic) of blog posts. HtmlBlocks contain static html code blocks for easy editing of the wiki home page.

D. Deployment

The application can be deployed as a standalone system on a single machine or on a shared server. Both are currently in use: a standalone instance is operating as the data system for the mARC facility and another instance is deployed to a shared server. Deployment of the application requires several steps:

1) Setup the Python environment: install all required packages and ensure sufficient permissions are available for the media folder. Ensure no access credentials are present in code or in web accessible folders.
2) Configure webserver (Apache+WSGI/NGINX+Gunicorn/etc.): configure webserver to point appropriate requests to the Django application and direct webserver to directly serve media and static folders with appropriate urls. Ensure maximum file-upload/download sizes are set appropriately.
3) Setup database (Postgres/MySQL/SQLite etc.): setup a database with appropriate password, location, and users. Modify the settings.py file to reflect database credentials. SQLite is the default and requires no setup.
4) Run database migrations: run 'python manage.py migrate' to create the database tables.
5) Memory requirements: ensure your hardware has sufficient storage capacity to store the entire database.
6) Performance requirements: test speed of pipelines/upload/download to ensure sufficiently fast performance.
7) Setup desired authentication: The Django application can use a variety of authentication methods (password, token, SAML, OAuth) depending on the security requirements. The deployment for NASA arc jets will use the NASA SAML authentication provider (NASA Launchpad) which requires the user to have a NASA badge and be granted access to the database via NAMS requests.

VIII. Discussion

The BEAST application provides a tailored data management system for arc jet facilities. The architecture provides four critical new capabilities (1) a central repository for all data types with global search capability, (2) integrated statistical tracking for uncertainty quantification, (3) integration with advanced tools and ML models, and (4) a single-language, open-source application suitable for continuous development by a small team. This will enable more comprehensive arc jet data stewardship, better quantify existing facility uncertainties, and save significant personnel time via automation.

Future work will focus on releasing the software as an open-source project and further developing the user interface based on feedback from stakeholders.

References

[1] Thompson, C., Prabhu, D., Terrazas-Salinas, I., and Mach, J., *Bulk Enthalpy Calculations in the Arc Jet Facility at NASA ARC*, ???? https://doi.org/10.2514/6.2011-3475, URL https://arc.aiaa.org/doi/abs/10.2514/6.2011-3475.

[2] Django Software Foundation, “Django,” , ???? URL https://djangoproject.com.

[3] Python Software Foundation, “Python3,” , ???? URL https://www.python.org.