



SCIENTIFIC PYTHON & ENGINEERING AUTOMATION

Custom scientific programming and engineering automation that convert repetitive analysis into efficient, traceable and reproducible computational workflows.



AUTOMATE.

Eliminate repetitive steps.
Save time and reduce errors.



ANALYZE.

Transform raw data into
meaningful engineering insights.



INTEGRATE.

Connect tools, data and workflows
into a unified process.



REPRODUCE.

Ensure traceability, validation
and consistent results.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from pathlib import Path
5
6 def process_results(input_dir, output_dir):
7     files = sorted(Path(input_dir).glob("*.csv"))
8     all_data = []
9     for f in files:
10         df = pd.read_csv(f)
11         df['case'] = f.stem
12         all_data.append(df)
13
14     data = pd.concat(all_data, ignore_index=True)
15     summary = data.groupby('case').agg(
16         'disp_max': 'max',
17         'stress_von_mises': 'max'
18     )
19     summary.to_csv(Path(output_dir)/'summary.csv')
20     plot_results(summary, output_dir)
21
22 def plot_results(df, output_dir):
23     fig, ax = plt.subplots(figsize=(8,5))
24     df['disp_max'].plot(kind='bar', ax=ax, color='royalblue')
25     ax.set_xlabel('Displacement (mm)')
26     ax.set_title('Maximum Displacement by Case')
27     plt.tight_layout()
28     plt.savefig(Path(output_dir)/'disp_max.png', dpi=300)
29     plt.close()
```



PYTHON

- Scientific Computing
- Data Processing
- Automation



AUTOMATION

- Batch Execution
- Workflow Orchestration
- Error Handling



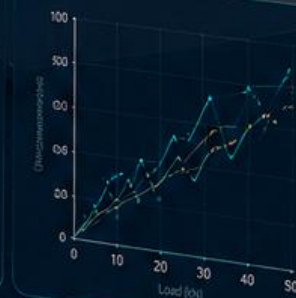
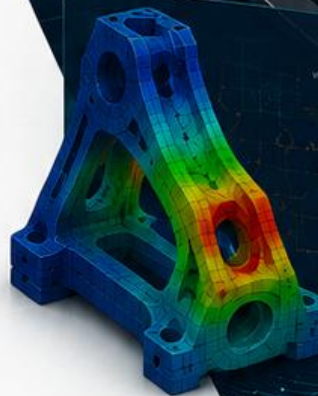
POST-PROCESSING

- Data Extraction
- Visualization
- Reporting



REPRODUCIBILITY

- Version Control
- Logging
- Documentation



Case	Max. Stress (MPa)
C01	312
C02	298
C03	346
C04	289
C05	301



PYTHON



MATLAB



NUMERICAL
METHODS



SIMULATION
AUTOMATION



DATA
PROCESSING



REPORTING &
DOCUMENTATION



SCIENTIFIC PYTHON & ENGINEERING AUTOMATION

Custom scientific programming and engineering automation that convert repetitive analysis into efficient, traceable and reproducible computational workflows.



TYPICAL CLIENT CHALLENGES



Engineering calculations or post-processing are repetitive, manual or error-prone.



Large simulation batches require automated setup, execution, output extraction, comparison or post-processing.



Data move through disconnected spreadsheets, scripts and solver outputs.



A custom scientific script or computational tool is needed for a project-specific workflow.



Existing code needs restructuring, validation, documentation or reproducibility improvements.



SCOPE OF SUPPORT

Capability

Typical support



Scientific Python / MATLAB

Engineering calculations, numerical methods, data transformation, scientific scripting and problem-specific computational analysis.



Simulation automation

Batch setup/execution, parameter sweeps, solver orchestration and repeatable run management where software interfaces and licences permit.



Post-processing

Automated extraction, transformation, plotting, reporting and comparison of simulation/experimental outputs.



Custom computational tools

Project-specific scripts, modules and lightweight engineering utilities.



Workflow reproducibility

Structured inputs/outputs, configuration, logging, validation checks, dependency control, documentation and repeatable execution.



AUTOMATE

Eliminate repetitive tasks and reduce human error.



INTEGRATE

Connect data, tools and workflows seamlessly.



ANALYZE

Extract meaningful insights and accelerate decisions.



REPRODUCE

Ensure traceability, consistency and quality.

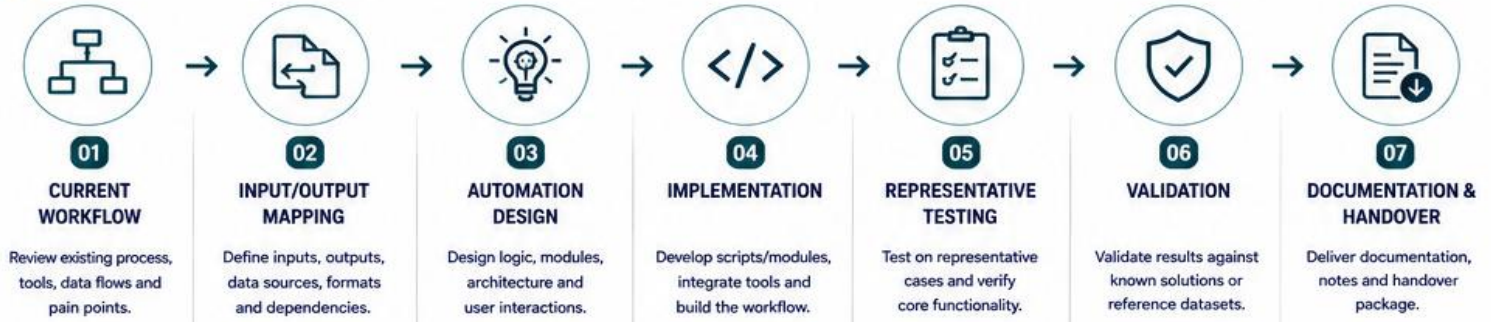


CUSTOMIZE

Solutions tailored to your engineering needs.



TYPICAL TECHNICAL WORKFLOW



TYPICAL DELIVERABLES

-  Python/MATLAB scripts or agreed computational tool components.
-  Defined input/output structure and configuration files where applicable.
-  Automated processing, plotting or reporting workflow.
-  Representative tests and validation evidence against known results, manual calculations or reference cases where feasible.
-  README/user notes, dependency information, configuration requirements and execution instructions appropriate to scope.
-  Source files and handover package as defined in the proposal.



TECHNICAL QUALITY FOCUS

-  Numerical correctness and engineering consistency of implemented calculations.
-  Clear handling of units, data types, missing/invalid inputs and failure conditions where relevant.
-  Reproducible execution and traceable inputs/outputs.
-  Testing on representative cases rather than demonstration-only code.
-  Maintainability and documentation proportional to whether the deliverable is a prototype, project script or reusable tool.

BEST SUITED FOR



TYPICAL CLIENT INPUTS

-  Current workflow description
-  Representative input/output files
-  Formulas and technical rules
-  Existing scripts where relevant
-  Solver / software environment
-  Expected batch volume
-  Required output format
-  User / deployment constraints
-  Representative acceptance cases

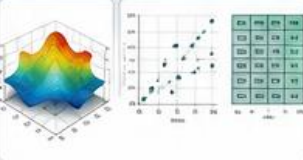
SCOPE BOUNDARIES

-  Prototype scripts, project utilities and production-ready software are different delivery classes, GUI, packaging, deployment, security hardening, support and long-term maintenance must be explicitly scoped.
-  External software/API integration depends on available interfaces, licences, access permissions and the client environment.
-  New functionality, changed file formats, additional integrations or expanded user requirements after scope approval are treated as scope changes.

REPRESENTATIVE TECHNICAL FOUNDATION

```

import numpy as np
import pandas as pd
from scipy import optimize
from matplotlib import pyplot as plt
  
```



AriyaTech combines engineering-domain knowledge with Python/MATLAB scientific computing, numerical methods, simulation automation, data processing and custom computational workflow development, with emphasis on engineering correctness and reproducibility.

START A TECHNICAL CONVERSATION

