Wing Optimization Pipeline - User Manual

Table of Contents

Quick Start

Minimal Setup (15 minutes)

Step 1: Install OpenVSP and Setup Python Environment

```
# After installing OpenVSP 3.45.0, navigate to python folder
cd "C:\Program Files\OpenVSP\python"

# Run setup script (creates vsppytools conda environment)
.\setup.ps1

# Activate the environment
conda activate vsppytools
```

Step 2: Install Required Python Packages

```
# Make sure vsppytools environment is active
pip install platypus-opt
pip install tqdm
```

Step 3: Prepare Project Files Ensure you have these files in the same directory:

- `structured_vspaero_optimizer.py`
- `optimizer_config.json`
- `ClarkY.dat`

Step 4: Run the Optimization

```
# Make sure vsppytools environment is active
python structured_vspaero_optimizer.py
```

The optimization will start automatically and show a progress bar. Results are saved continuously to CSV files.

Important: Always run the script with the `vsppytools` conda environment activated!

Prerequisites & Installation

Required Software

1. OpenVSP 3.45.0 (or latest version) with Python 3.11

Download and Install OpenVSP:

- Download OpenVSP from: http://openvsp.org/
- Install version 3.45.0 or later

Install OpenVSP Python API:

1. **Navigate to the Python folder** in your OpenVSP installation:

   ```
   cd "C:\Program Files\OpenVSP\python"  # Adjust path as needed
   ```

2. **Run the setup script**:

   ```
   .\setup.ps1
   ```

   This will create a conda environment named `vsppytools`

3. **Activate the environment**:

   ```
   conda activate vsppytools
   ```

4. **Verify the installation**:

   ```
   python
   >>> import openvsp as vsp
   >>> print(vsp.GetVSPVersion())
   ```

   You should see the OpenVSP version number (e.g., "3.45.0")

Configure VS Code (Recommended):

1. Open VS Code
2. Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac)
3. Type "Python: Select Interpreter"
4. Choose the interpreter from the `vsppytools` conda environment
    - Look for path like: `...\anaconda3\envs\vsppytools\python.exe`

2. VSPAERO Executable

- Usually bundled with OpenVSP installation
- The script will attempt to find it automatically
- Location options:
    - Same folder as Python's openvsp module
    - System PATH
    - Manual specification in config file

Required Python Packages

Install in the vsppytools environment:

1. **Open Anaconda terminal**
2. **Activate the environment**:

```
conda activate vsppytools
```

3. **Install Platypus optimization library**:

```
pip install platypus-opt
```

4. **Install TQDM progress bar** (usually included, but verify):

```
pip install tqdm
```

5. **Verify installations**:

```
python
>>> from platypus import Problem, Real, SMPSO
>>> from tqdm import tqdm
>>> print("All packages installed successfully!")
```

Required Files

Your working directory should contain:

```
your_project_folder/
├── structured_vspaero_optimizer.py  # Main script
├── optimizer_config.json            # Configuration file
└── ClarkY.dat                       # Airfoil data
```

System Requirements

- **RAM**: 4 GB minimum, 8 GB recommended
- **Disk Space**: 1 GB for results
- **CPU**: Multi-core recommended (VSPAERO uses 4 cores)
- **Time**: 3-7 hours for default optimization

Understanding the Pipeline

What Does This Pipeline Do?

This system optimizes a wing design for a small aircraft (UAV) to maximize flight endurance. It:

1. **Generates** wing geometries using OpenVSP
2. **Analyzes** aerodynamics using VSPAERO (CFD simulation)
3. **Optimizes** design variables using particle swarm optimization
4. **Enforces** constraints (stability, lift requirements)
5. **Outputs** the best wing design

Design Variables Being Optimized

The optimizer adjusts 6 parameters:

Variable	Range	Description
Semi-Span	0.75 - 1.25 m	Half the total wingspan
Semi-Area	Auto	Half the wing planform area
Taper Ratio	0.2 - 1.0	Tip chord / Root chord
Sweep Angle	0 - 5°	Leading edge sweep
Twist	-5 to +5°	Geometric twist (washout)
Root Incidence	-5 to +5°	Root airfoil angle

Constraints Applied

The optimizer ensures:

1. **Stability**: Pitching moment coefficient (Cm) ≤ 0
2. **Minimum Lift**: Lift coefficient (Cl) ≥ 0.2

3. **Maximum Lift**: Lift coefficient (Cl) ≤ 0.25

Objective

Maximize Endurance = Maximize (Cl^1.5 / Cd)

This metric represents how long the aircraft can stay airborne.

Configuration Guide

The Configuration File: `optimizer_config.json`

This JSON file controls all aspects of the optimization. It's divided into 5 sections:

1. Environment Settings

```
"environment": {
    "vsp_path": "AUTO",
    "airfoil_file": "ClarkY.dat",
    "aero_results_file": "aero_results.csv",
    "opt_results_file": "wingopt_results.csv",
    "debug_vsp_file": "modified_wing.vsp3",
    "polar_file": "modified_wing.polar"
}
```

What you can change:

- `vsp_path`: Set to `"AUTO"` for automatic discovery, or provide full path to VSPAERO folder
- `airfoil_file`: Path to your airfoil coordinate file
- Output filenames (if you want different names)

2. Aircraft Parameters

```
"aircraft": {
    "weight_kg": 7.0,
    "wing_loading_min": 4.8,
    "wing_loading_max": 6.2
}
```

What you can change:

- `weight_kg`: Your aircraft's total weight
- `wing_loading_min`: Minimum wing loading (kg/m²)
- `wing_loading_max`: Maximum wing loading (kg/m²)

Important: Wing loading defines the allowable wing area:

- Minimum area = weight / (2 × max_loading)

- Maximum area = weight / (2 × min_loading)

3. Simulation Parameters

```
"simulation": {
    "rho_kg_m3": 1.2256,
    "velocity_m_s": 20.0,
    "viscosity": 1.784e-05,
    "mach": 0.06,
    "aoa_deg": 0.0,
    "beta_deg": 0.0
}
```

What you can change:

Parameter	Description	Typical Values
`rho_kg_m3`	Air density	1.225 (sea level), 1.007 (2000m)
`velocity_m_s`	Cruise speed	15-30 m/s for small UAVs
`viscosity`	Air viscosity	1.784e-05 (standard)
`mach`	Mach number	velocity / 340.3
`aoa_deg`	Angle of attack	0° for cruise analysis
`beta_deg`	Sideslip angle	0° for symmetric flight

4. Design Variables (Optimization Bounds)

```
"design_variables": {
    "semi_span": {
        "min": 0.75,
        "max": 1.25
    },
    "semi_area": {
        "min": "AUTO",
        "max": "AUTO"
    },
    "taper": {
        "min": 0.2,
        "max": 1.0
    },
    "sweep": {
        "min": 0.0,
        "max": 5.0
    },
    "twist": {
        "min": -5.0,
        "max": 5.0
```

```
    },
    "root_incidence": {
        "min": -5.0,
        "max": 5.0
    }
}
```

How to modify:

1. **Wider Search Space**: Increase the range

   ```
   "semi_span": {"min": 0.5, "max": 2.0}
   ```

2. **Narrower Search Space**: Decrease the range (faster optimization)

   ```
   "semi_span": {"min": 0.9, "max": 1.1}
   ```

3. **Keep semi_area as "AUTO"**: This automatically calculates bounds from wing loading

Guidelines:

- Full wingspan = 2 × semi_span
- Full wing area = 2 × semi_area
- Taper = 1.0 means rectangular wing
- Taper = 0.2 means highly tapered
- Negative twist = "washout" (improves stall)

5. Optimization Settings

```
"optimization": {
    "swarm_size": 50,
    "max_evaluations": 50,
    "step_size": 1
}
```

What you can change:

Parameter	Description	Effect	Recommended
`swarm_size`	Number of particles	More = better search, slower	30-100
`max_evaluations`	Total iterations	More = better results, longer	50-200
`step_size`	Logging frequency	1 = log every step	1-10

Computational Cost:

- Total simulations ≈ swarm_size × max_evaluations
- Each simulation ≈ 5-10 seconds
- Example: 50 × 50 = 2500 runs ≈ 3-5 hours

Running the Optimization

Basic Execution

```
python structured_vspaero_optimizer.py
```

What You'll See

1. **Initialization Messages:**

```
Using VSPAERO Path: /path/to/vspaero
Starting Wing Optimization...
Design Variables: ['semi_span', 'semi_area', 'taper', 'sweep', 'twist',
'root_inc']
Running for 50 evaluations with swarm size 50...
```

2. **Progress Bar:**

```
VSPAERO Runs: 1234/2500 [========>    ] 49% [00:15:23<00:16:12, 1.2s/run]
```

3. **Final Summary:**

```
Optimization Complete.
=== Best Feasible Solution Found ===
Objective (Negative Endurance): -8.234567
------------------------------
Semi-Span      : 1.1234 m (Full Span: 2.2468 m)
Semi-Area      : 0.6789 m^2 (Full Area: 1.3578 m^2)
Taper Ratio    : 0.4567
Sweep          : 2.3456 deg
Twist          : -1.2345 deg
Root Incidence : -0.9876 deg
```

Monitoring Progress

While running, check the output files:

1. Real-time Progress: `aero_results.csv`

```
Iteration,Cl,Cd,Cm,L/D
1,0.2134,0.0312,-0.0145,6.84
2,0.2289,0.0298,-0.0132,7.68
...
```

- Every simulation is logged here
- Watch Cl, Cd values to see if designs are reasonable

2. Best Solutions: `wingopt_results.csv`

```
iteration,objective,semi_span,semi_area,taper,sweep,twist,Root_inc
1,-7.234,1.124,0.678,0.456,2.345,1.234,-0.987
10,-7.456,1.156,0.689,0.478,2.123,0.987,-1.123
...
```

- Only feasible solutions
- Track the best design over time

Stopping the Optimization

Graceful Stop: Press `Ctrl+C`

- The optimization will stop cleanly
- All results up to that point are saved
- Progress bar closes properly

Note: You can resume by:

1. Analyzing the results so far
2. Adjusting `max_evaluations` if needed
3. Running again with refined bounds

Customizing Parameters

Example 1: Different Aircraft Weight

Scenario: You have a 10 kg aircraft instead of 7 kg.

Change in `optimizer_config.json`:

```
"aircraft": {
    "weight_kg": 10.0,
    "wing_loading_min": 5.0,
    "wing_loading_max": 7.0
}
```

Result: Wing area bounds automatically adjust to maintain wing loading.

Example 2: Higher Speed Optimization

Scenario: Optimize for 25 m/s cruise speed instead of 20 m/s.

Change in `optimizer_config.json`:

```
"simulation": {
    "rho_kg_m3": 1.2256,
    "velocity_m_s": 25.0,
    "viscosity": 1.784e-05,
    "mach": 0.073,
    "aoa_deg": 0.0,
    "beta_deg": 0.0
}
```

Note: Update `mach = velocity / 340.3`

Example 3: Faster Optimization (Less Accuracy)

Scenario: Quick exploration, don't need perfect results.

Change in `optimizer_config.json`:

```
"optimization": {
    "swarm_size": 30,
    "max_evaluations": 30,
    "step_size": 5
}
```

Result: 30 × 30 = 900 runs ≈ 1-2 hours instead of 5 hours.

Example 4: Thorough Optimization (High Accuracy)

Scenario: Final design, need best possible results.

Change in `optimizer_config.json`:

```
"optimization": {
    "swarm_size": 100,
    "max_evaluations": 200,
    "step_size": 10
}
```

Result: 100 × 200 = 20,000 runs ≈ 24-48 hours. Better convergence.

Example 5: Larger Wings

Scenario: You want to explore larger wing sizes.

Change in `optimizer_config.json`:

```
"design_variables": {
    "semi_span": {
        "min": 1.0,
        "max": 2.0
    }
}
```

Result: Full wingspan range becomes 2.0 - 4.0 meters.

Example 6: Different Airfoil

Scenario: Use NACA 2412 instead of ClarkY.

Steps:

1. Get NACA2412.dat file (same format as ClarkY.dat)
2. Place in same folder
3. Change config:

```
"environment": {
    "airfoil_file": "NACA2412.dat"
}
```

Understanding Results

Output Files Generated

1. `aero_results.csv` - Detailed log of every simulation
2. `wingopt_results.csv` - Best designs at each step
3. `modified_wing.vsp3` - Last evaluated geometry (OpenVSP file)
4. `modified_wing.polar` - Last simulation results (VSPAERO output)

Reading the Results

File 1: `wingopt_results.csv`

What to look for:

- **Objective column**: More negative = better endurance
- **Last row**: Usually contains the best design
- **Trend**: Should get more negative over iterations

Example:

```
iteration,objective,semi_span,semi_area,taper,sweep,twist,Root_inc
10,-7.234,1.124,0.678,0.456,2.345,1.234,-0.987
20,-7.856,1.156,0.689,0.478,2.123,0.987,-1.123
30,-8.234,1.189,0.695,0.512,1.987,0.876,-1.234
```

Best design is at iteration 30 (objective = -8.234, most negative)

File 2: `aero_results.csv`

What to look for:

- **Cl values**: Should be between 0.2-0.25 (constraint satisfied)
- **Cm values**: Should be negative (stable)
- **L/D values**: Higher is better (efficiency)

Good design indicators:

- Cl ≈ 0.22-0.23 (middle of allowed range)
- Cd ≈ 0.025-0.035 (low drag)
- Cm < 0 (stable)
- L/D > 7.0 (efficient)

Interpreting the Optimal Design

Terminal output shows:

```
Semi-Span      : 1.1234 m (Full Span: 2.2468 m)
Semi-Area      : 0.6789 m^2 (Full Area: 1.3578 m^2)
Taper Ratio    : 0.4567
Sweep          : 2.3456 deg
Twist          : -1.2345 deg
Root Incidence : -0.9876 deg
```

What this means:

Parameter	Value	Interpretation
Full Span	2.25 m	Total wingspan tip-to-tip
Full Area	1.36 m^2	Total wing surface area
Taper	0.46	Moderate taper (not rectangular)
Sweep	2.3°	Slight forward sweep
Twist	-1.2°	Slight washout (tip twisted down)
Root Inc.	-0.98°	Root slightly nose-down

Typical optimal characteristics:

- High aspect ratio (long, narrow wings)
- Moderate taper (0.4-0.6)
- Small sweep (1-3°)
- Negative twist (washout)
- Small negative root incidence

Calculating Performance Metrics

From the optimal design's Cl and Cd:

```
Endurance Parameter = Cl^1.5 / Cd
Example: 0.22^1.5 / 0.028 = 3.69

Aspect Ratio = Span² / Area
Example: 2.25² / 1.36 = 3.72

Wing Loading = Weight / Area
Example: 7 kg / 1.36 m² = 5.15 kg/m²
```

Troubleshooting

Problem: "Could not import 'openvsp'"

Cause: OpenVSP Python bindings not installed correctly or wrong Python environment

Solution:

1. **Verify OpenVSP Python API installation**:

   ```
   # Navigate to OpenVSP python folder
   cd "C:\Program Files\OpenVSP\python"

   # Run setup script
   .\setup.ps1
   ```

2. **Activate the vsppytools environment**:

   ```
   conda activate vsppytools
   ```

3. **Verify you're using the correct Python interpreter**:

```
which python  # Linux/Mac
where python  # Windows
# Should point to: .../anaconda3/envs/vsppytools/python.exe
```

4. **Test the import**:

```
python
>>> import openvsp as vsp
>>> print(vsp.GetVSPVersion())
```

5. **If using VS Code**, ensure correct interpreter:
   - Press `Ctrl+Shift+P`
   - Select "Python: Select Interpreter"
   - Choose the `vsppytools` environment

Problem: "VSPAERO path not found"

Cause: Script can't locate VSPAERO executable

Solution:

1. **Find VSPAERO manually**:
   - Windows: Search for `vspaero.exe`
   - Linux/Mac: `which vspaero`
2. **Set path explicitly** in config:

```
"environment": {
    "vsp_path": "C:/Program Files/OpenVSP/vspaero"
}
```

Problem: No feasible solutions found

Cause: Constraints too restrictive or bounds unrealistic

Solution:

1. **Check wing loading**: Ensure it allows sufficient area for lift

```
Required Cl ≥ 0.2
At 20 m/s, 7 kg needs minimum area
```

2. **Relax constraints**: Temporarily adjust in code

```
# Line in code:
c2 = -cl + 0.15  # Changed from 0.2 to 0.15
```

3. **Widen bounds**: Allow more design space

Problem: Optimization stuck (not improving)

Cause: Converged to local optimum or insufficient evaluations

Solution:

1. **Run longer**:

```
"max_evaluations": 100
```

2. **Increase swarm size**:

```
"swarm_size": 100
```

3. **Restart with tighter bounds** around current best

Problem: VSPAERO simulations failing

Symptoms: Many zeros in `aero_results.csv`

Solution:

1. **Check geometry**: Open `modified_wing.vsp3` in OpenVSP GUI
2. **Verify airfoil file**: Ensure `ClarkY.dat` is properly formatted
3. **Check bounds**: Extreme values may create invalid geometry

Problem: Very slow execution

Cause: High resolution or many evaluations

Solution:

1. **Reduce mesh density** (in code):

```
vsp.SetParmVal(wing_id, "SectTess_U", "XSec_1", 8)  # Was 10
vsp.SetParmVal(wing_id, "Tess_W", "Shape", 15)      # Was 20
```

2. **Reduce swarm size**:

```
"swarm_size": 30
```

3. **Use step_size for less frequent logging**:

```
"step_size": 10
```

Advanced Usage

Modifying the Objective Function

Location: Line ~330 in `structured_vspaero_optimizer.py`

Current objective (maximize endurance):

```
endurance = -(cl**1.5) / cd
```

Alternative objectives:

1. Maximize Range:

```
range_param = -(cl / cd)  # L/D ratio
return [range_param], [c1, c2, c3]
```

2. Maximize L/D with penalty for Cm:

```
ld_ratio = -(cl / cd)
cm_penalty = 100 * abs(cm)  # Prefer Cm closer to zero
objective = ld_ratio + cm_penalty
return [objective], [c1, c2, c3]
```

3. Minimize Drag at fixed Cl:

```
# Change constraint to equality: Cl = 0.22
drag_objective = cd  # Minimize (don't negate)
return [drag_objective], [c1, c2, c3]
```

Adding New Constraints

Location: Line ~340 in function `weighted_sum_wing_analysis`

Example: Add maximum span constraint

```
# After existing constraints
c4 = semi_span - 1.5  # Semi-span must be ≤ 1.5m

# Update return statement
return [endurance], [c1, c2, c3, c4]

# Update problem definition (main function)
problem = Problem(6, 1, 4)  # Changed 3 to 4 constraints
problem.constraints[:] = ["<=0", "<=0", "<=0", "<=0"]
```

Using a Different Airfoil

Steps:

1. **Obtain airfoil coordinates** in Selig format:
   - Download from airfoiltools.com
   - Or generate NACA using tools
2. **Format requirements**:
   - Text file, two columns: x/c, y/c
   - Points from trailing edge → leading edge → trailing edge
   - Normalized (x and y between 0 and 1)
3. **Update config**:

```
"environment": {
    "airfoil_file": "your_airfoil.dat"
}
```

Exporting Optimal Geometry

Option 1: Use OpenVSP GUI

1. Open `modified_wing.vsp3`
2. File → Export → STL/STEP/IGES
3. Use in CAD or manufacturing

Option 2: Recreate from parameters

```
# Using values from wingopt_results.csv
import openvsp as vsp

vsp.ClearVSPModel()
wing_id = vsp.AddGeom("WING", "")
```

```
# Set optimal parameters
vsp.SetParmVal(wing_id, "Span", "XSec_1", 1.1234)
vsp.SetParmVal(wing_id, "Area", "XSec_1", 0.6789)
# ... etc

vsp.Update()
vsp.WriteVSPFile("optimal_wing.vsp3")
```

Visualizing Results

Plot convergence:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('wingopt_results.csv')

plt.figure(figsize=(10, 6))
plt.plot(df['iteration'], -df['objective'])
plt.xlabel('Iteration')
plt.ylabel('Endurance Parameter')
plt.title('Optimization Convergence')
plt.grid(True)
plt.savefig('convergence.png')
plt.show()
```

Plot drag polar:

```
aero = pd.read_csv('aero_results.csv')

plt.figure(figsize=(8, 6))
plt.scatter(aero['Cd'], aero['Cl'], alpha=0.3)
plt.xlabel('Drag Coefficient (Cd)')
plt.ylabel('Lift Coefficient (Cl)')
plt.title('Drag Polar - All Evaluated Designs')
plt.grid(True)
plt.savefig('drag_polar.png')
plt.show()
```

Quick Reference

Command Cheat Sheet

```
# Activate the conda environment (ALWAYS DO THIS FIRST!)
conda activate vsppytools
```

```
# Run optimization
python structured_vspaero_optimizer.py

# Stop gracefully
Ctrl+C

# Check OpenVSP installation
python -c "import openvsp as vsp; print(vsp.GetVSPVersion())"

# Check environment
conda info --envs

# View results while running
tail -f aero_results.csv  # Linux/Mac
Get-Content aero_results.csv -Wait  # Windows PowerShell
```

File Checklist

- ☐ `structured_vspaero_optimizer.py`
- ☐ `optimizer_config.json`
- ☐ `ClarkY.dat` (or your airfoil file)
- ☐ OpenVSP installed
- ☐ Python packages installed

Config Quick Changes

What you want	Where to change	Example
Different weight	`aircraft.weight_kg`	`10.0`
Faster/slower speed	`simulation.velocity_m_s`	`25.0`
Longer wings	`design_variables.semi_span.max`	`2.0`
Quicker test	`optimization.swarm_size`	`20`
Better results	`optimization.max_evaluations`	`200`

Expected Results Ranges

Metric	Typical Range	Good Value
Cl	0.20 - 0.25	0.22
Cd	0.025 - 0.040	0.028
Cm	-0.02 - 0.00	-0.01
L/D	6.0 - 9.0	7.5+
Endurance	-6.0 - -9.0	-8.0+

Contact & Support

For questions about this pipeline:

- Check the comprehensive documentation in `Documentation.pdf`
- Review code comments in `structured_vspaero_optimizer.py`
- OpenVSP forum: https://groups.google.com/g/openvsp

Project Information:

- Organization: Indiflo Pvt. Ltd.
- Department: Mechanical sub-division
- Created by: Shubham Tamboli
- LinkedIn: https://www.linkedin.com/in/shubham-t-816785201/

Good luck with your wing optimization! 🚁✈