A fast, memory-efficient panel method for large-scale multidisciplinary design optimization under uncertainty using graph-based modeling

Luca Scotzniovsky*, and John T. Hwang†
University of California, San Diego, La Jolla, California, 92023

Comprehensive large-scale multidisciplinary design optimization for aircraft design has become feasible thanks to recent advances in automatic adjoint-based sensitivity analysis. However, a major limitation of existing methods and previous demonstrations is the use of low-fidelity models to address key disciplines. Aerodynamics, one of these key disciplines, is currently restricted to very low or high fidelity methods. Panel methods bridge this gap in modeling fidelity; however, existing tools are not suited for large-scale MDO because they lack adjoint-based sensitivity analysis needed for efficient gradient computation. In addition, panel methods naturally incur a prohibitive n^2 cost. In this paper, we address these needs by presenting a modern, graph-based panel method with automatic adjoint-based sensitivity analysis. We discuss novel methods focused on reducing the prohibitive quadratic cost with respect to grid size by using a new approach we call partitioned vectorization, and present a method incorporating POD that reduces the memory cost for solving the linear system to $O(n)$. With this, we present results on numerical experiments comparing timing and memory for other standard approaches, showing memory improvements by up to two orders of magnitude with minimal increase in computation time. We then present a large-scale multidisciplinary design optimization under uncertainty (MDOUU) demonstration on the blended wing body aircraft concept using our graph-based panel method. We highlight the importance of robustness to input variation by identifying significant constraint violations of the MDO design in the MDOUU setting.

I. Introduction

Large-scale multidisciplinary design optimization (MDO) is the application of numerical optimization to high-dimensional design problems that span multiple engineering disciplines. In aircraft design, there has been significant recent progress toward *comprehensive* large-scale MDO, i.e., applications considering the entire aircraft and a broad set of nominal and off-nominal conditions. Sarojini et al. demonstrated large-scale MDO of the NASA lift-plus-cruise concept, considering over 100 design variables [1]. Key disciplines, such as aerodynamics, structures, acoustics, and electric propulsion, were considered using physics-based models, and the system-level optimization resulted in an 11.4 % reduction in gross weight. Ruh et al. expanded on this work by increasing the number of design conditions to model a full mission profile [2]. In this work, stability analysis and higher-fidelity tools for acoustics were incorporated. The full-scale MDO problem considered just under 400 design variables and more than 200 constraints across 20 different design conditions, and the gross weight was reduced by 8%. However, a major limitation of this and other existing comprehensive large-scale MDO algorithms is that they use low-fidelity models to address key disciplines—a tradeoff made to efficiently explore a large design space for high-level conceptual design.

In MDO, models for aerodynamics—a critical discipline—form a bimodal distribution along the fidelity axis. Rapid conceptual design often utilizes low-fidelity methods such as the vortex lattice method [1–4] or the lifting line method to model aerodynamics. RANS CFD is commonly used for high-fidelity MDO of aircraft to capture physics accurately [5–7]. This approach is often used to capture coupled phenomena, such as aeroelastic [8, 9], aeroacoustic [10, 11], and aeropropulsive [12] coupling. The choice of fidelity is driven by the tradeoff between accuracy and cost. Panel methods bridge this gap in modeling fidelity. They are mid-fidelity models that solve the linearized potential flow equations to compute the flow around arbitrary three-dimensional bodies. Although the potential flow assumption implies inviscid and irrotational flows, panel methods can capture general flow behavior by computing full pressure distributions around

*PhD Student, Department of Mechanical and Aerospace Engineering, lscotzni@ucsd.edu, AIAA Student Member
†Associate Professor, Department of Mechanical and Aerospace Engineering, jhwang@ucsd.edu, AIAA Member

a three-dimensional geometry. For comprehensive large-scale MDO of aircraft, panel methods are an attractive choice for their reasonable accuracy and computational cost.

CSDL (Computational System Design Language) [13], a graph-based modeling framework, has been a significant enabler for comprehensive, large-scale MDO. Graph-based modeling refers to an approach for constructing and working with models that explicitly exposes the computational graph. Using this graph, CSDL offers automatic reverse-mode differentiation and adjoint-based sensitivity analysis. In addition, CSDL can transform the computational graph to optimize various computations—e.g., graph transformations can be used to accelerate uncertainty propagation [14].

There has been limited recent work in panel methods or in the development of new panel codes, in spite of the fact that they have been workhorse tools for aerodynamic modeling since the late twentieth century [15–17]. Modern formulations have focused on higher-order methods [18, 19] or ways to improve panel method scaling [20]. However, many of the existing tools are not suited for large-scale MDO because they lack the adjoint-based sensitivity analysis needed for efficient gradient-based optimization.

Panel methods perform calculations involving n^2 interactions to solve for the flow field (n being the number of panels). This becomes a computational barrier when modeling large bodies. Several different methods exist to accelerate the n^2 evaluations. Vectorization is a straightforward method to accelerate evaluation time by looping over interactions at the compiled-language level. Although this approach avoids the slowdown of iterating through interactions, the simultaneous data processing incurs significant memory overhead, especially for reverse-mode automatic differentiation (AD) or adjoint sensitivity analysis for large-scale MDO. Full vectorization incurs the upper limit of memory cost, scaling quadratically with the number of panels. As a result, it is only practical with relatively small meshes. The use of graphical processing units (GPUs) is also a simple and efficient approach to manage the time-cost of the n^2 interactions. Although GPUs can improve computation time, the n^2 interactions can still pose a memory bottleneck. GPUs are also expensive to purchase and not always readily accessible. In addition, the changing orders of floating-point arithmetic in GPUs are known to introduce nondeterminism in the output. Fast multipole methods (FMMs) are commonly paired with panel methods to reduce the cost to $O(n\log(n))$; FMMs use multipole expansions to approximate numerous far-field interactions as a single cluster or source [21]. However, the discontinuities that form by switching between various near-field (direct) and far-field (multipole) expansions can introduce noise and loss of smoothness, which is problematic for gradient-based optimization. Reduced order modeling (ROM) is a widely used technique to simplify complex numerical models while still maintaining accuracy, resulting in accelerated model evaluations. Proper orthogonal decomposition (POD) is common in CFD to capture only the dominant modes and reduce overall computational cost [22, 23]. However, data generation for ROMs incurs a non-trivial cost. Many full-order model evaluations must be conducted to generate training data for a reduced-order model; the training data requires a large number of samples to account for variations in the input space, such as flow properties and geometry parameters. ROMs for panel methods have not been explored in the literature, but it is a promising avenue for improving the computational cost of the panel method.

This paper presents the development of a modern panel method within the graph-based modeling paradigm. There are four specific contributions. First, we describe the implementation of a panel code in CSDL, focusing specifically on the graph-based design aspects, and present code-verification results. Second, we present a novel set of numerical methods for solving the dense linear system in the panel method. These methods center around the novel idea of partitioned vectorization, which is time-efficient while significantly reducing the large memory footprint of traditional vectorization. Third, we present findings for numerical experiments that compare the computational efficiency of the partitioned vectorization approach with other acceleration strategies such as POD. Finally, we use this graph-based panel method, leveraging the partitioned vectorization to demonstrate the solution of a large-scale MDO under uncertainty (MDOUU) aircraft design problem of pioneering scale.

The paper is summarized as follows. Section II outlines relevant background theory and methods. Section III summarizes our graph-based implementation of the panel method. Section IV discusses our novel partitioned vectorization-based formulation that is efficient in time and memory. Section V provides comparisons between five approaches for solving the panel method linear system. Section VI presents an MDOUU demonstration using our novel method and graph-based panel code.

II. Background

A. Panel method

Panel methods are a class of numerical methods that solve the linearized potential flow equations around arbitrary bodies. Compressibility can be modeled using a simple correction; however, in this work, we assume incompressible flow.

Given our assumptions, the continuity equation is

$$\nabla^2 \Phi = 0, \tag{1}$$

where Φ is the velocity potential. The velocity potential is related to the velocity as

$$\nabla \Phi = \nabla \Phi^* + \nabla \Phi_\infty = \vec{U}, \tag{2}$$

which is satisfied everywhere in space. $\vec{U}$ represents the velocity vector and Φ represents the velocity potential, decomposed into the freestream influence Φ_∞ and the small perturbation from the lifting body Φ^*. Applying Green's theorem to Equation 1, we formulate a boundary value problem of the form

$$\Phi(x, y, z) = \frac{1}{4\pi} \int_\Omega \mu \mathbf{n} \cdot \nabla \left(\frac{1}{r}\right) dS - \frac{1}{4\pi} \int_\Omega \sigma \left(\frac{1}{r}\right) dS + \Phi_\infty = \Phi^* + \Phi_\infty, \tag{3}$$

where the doublet strength μ and source strength σ represent an incremental change in velocity and potential across the surface boundary, respectively. This form of the boundary value problem is known as the Dirichlet formulation, where the potential is prescribed along the surface and in the body. An equivalent formulation called the Neumann problem explicitly enforces zero flow penetration at the surface. The underlying goal is to determine a unique set of sources and doublets that satisfy the boundary conditions on a discrete surface. By discretizing a lifting surface with n panels, we need to satisfy n boundary conditions. The boundary condition is applied at the collocation point of each panel. This reduces Equation 3 to a linear system of the form

$$A\vec{\mu} + B\vec{\sigma} = 0, \tag{4}$$

where the vectors $\vec{\mu}, \vec{\sigma} \in \mathbb{R}^n$ represent the doublet and source strengths, respectively, across the panels. The matrices $A, B \in \mathbb{R}^{(n,n)}$ represent the n^2 induced potential interactions between panels. We assume a low-order, constant-strength distribution of singularities, so the singularity terms are decoupled from the integrals in Equation 3. To uniquely define the flow field, we set the source strengths to

$$\sigma = -\vec{U}_\infty \cdot \vec{n}, \tag{5}$$

where $\vec{U}_\infty$ defines the free-stream velocity. This is equivalent to setting a zero-perturbation-potential boundary condition along and inside the surface: $\Phi^*|_\Omega = 0$. In addition, the Kutta condition is used to set the wake strengths. The Kutta condition enforces that the circulation at the trailing edge, where the upper surface, lower surface, and wake meet, must be zero. From this, we deduce that the wake doublet strength can be directly found from the surface doublet strengths by

$$\mu_w = \mu_u - \mu_l, \tag{6}$$

where the subscripts u, l represent the upper and lower surface of the trailing edge, respectively. With these two conditions, the only unknown in Equation 4 is the doublet strength vector.

Once the doublet strengths are found by solving the linear system in Equation 4, the singularity strengths are used to compute the total velocity at the collocation points, represented as

$$\vec{Q} = \vec{U}_\infty + \nabla \mu + \sigma \mathbf{n}, \tag{7}$$

where $\mathbf{n}$ represents the outward unit normal vector at the collocation point. The doublet gradient is tangent to the surface. The second and third terms correspond to the local change in velocity as a result of the presence of the lifting body. The pressure coefficient at each panel is computed using

$$C_p = 1 - \frac{||Q||^2}{||U_\infty||^2}, \tag{8}$$

derived from the Navier-Stokes momentum equation under the assumptions of steady, linearized potential flow. The pressure coefficients are integrated over each panel to compute the force as

$$F_k = -\frac{1}{2}(C_p)_k \rho ||U_\infty||^2 S_k \mathbf{n}_k, \tag{9}$$

where k represents the panel index and S represents the panel area. Lift and drag are computed by decomposing the forces into components parallel to and normal to the free stream flow.

B. Proper orthogonal decomposition

Proper orthogonal decomposition is a projection-based model reduction technique that extracts dominant system information from data to compute a reduced basis. We use the method of snapshots and singular value decomposition (SVD) to generate a reduced basis. Given a data set $\mathbf{X} \in \mathbb{R}^{(n,s)}$, we apply SVD to decompose our data set into

$$\mathbf{X} = U\mathbf{\Sigma}V^T, \tag{10}$$

where $U \in \mathbb{R}^{(n,n)}$ is the set of POD modes, or basis vectors. However, U contains the same n^2 elements as the linear system within the panel method. To reduce this, we use the concept of relative information content (RIC), which can be interpreted as the relative energy contained in the first r modes. RIC is defined as

$$\text{RIC} = \frac{\Sigma_i^r \sigma_i}{\Sigma_i^n \sigma_i}, \tag{11}$$

where σ_i is the i^{th} singular value of $\mathbf{\Sigma}$. The reduced basis size r is selected such that RIC captures a certain percentage of relative energy; the desired threshold varies based on application and desired accuracy of the ROM. We then define our reduced basis as $\phi = U[:, :r]$. Applying the reduced basis to the panel method is an intrusive process. We rearrange the panel method linear system in Equation 4 to a residual form:

$$R(\mu) = A\mu - b = 0, \tag{12}$$

where the vector notation has been removed from the doublets for simplicity. We relate the full and reduced space solutions as

$$\mu = \phi\mu_r, \tag{13}$$

where μ_r represents the solution to the linear system in the reduced space. We project the reduced basis onto the linear system in Equation 12 to formulate a new residual of the form

$$\phi^T R(\phi\mu_r) = A_r\mu_r - b_r = 0, \tag{14}$$

where $A_r = \phi^T A\phi$ and $b_r = \phi^T b$ represent the residual terms in reduced form. Equation 14 requires solving an r-dimensional linear system, where $r << n$. In this form, we see the computational benefits of using POD with the panel method. For very large systems, applying POD dramatically reduces the number of elements needed for the linear system and significantly accelerates the linear system solve time. Once the reduced-space solution μ_r is found, the full-space solution is recovered with Equation 13.

C. Optimization under uncertainty

There are three different approaches for design optimization under uncertainty: robust design, which focuses on statistical moments; reliability-based design, which focuses on modeling probability distributions of outputs; and risk-based design, which considers the extent of constraint violation. In this work, we focus on robust design considering input variability; model uncertainty is not considered. The general structure of the OUU workflow is compared to the deterministic formulation in Fig 1.

Objectives and constraints in the OUU problem are represented using mean and standard deviation. Higher-order moments, such as skewness and kurtosis, are ignored; these statistics are indicators of how to model input distributions and are more significant when explicitly modeling output probability distributions. In addition, these parameters are highly nonlinear and require many evaluations to estimate accurately. Exploring a large random input space within the optimization loop incurs scaling issues similar to those in forward model evaluation and the direct method for derivative

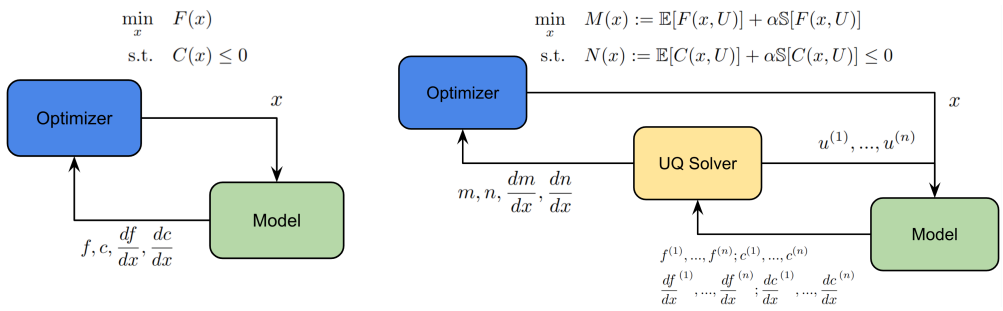


Fig. 1 This graphic shows the general optimization problem formulation and workflow structure for deterministic (left) and OUU (right) problems. In the OUU formulation, the internal loop dictates additional model evaluations corresponding to the random input variables to compute statistical moments.

computation. Traditional sampling methods like Monte Carlo rely on the law of large numbers to accurately capture randomness. First-order perturbation methods are limited in accuracy. Direct quadrature methods are highly accurate, but incur exponential scaling with the size of the random input space.

In this work, we apply univariate dimension reduction (UDR) [24] to model statistical outputs. UDR approximates functions as a sum of univariate functions:

$$\begin{aligned} f(u) = \hat{f}(u) &\approx \Sigma_i^d f_i(u_i) - (d-1) f(\bar{u}), \\ f_i(u_i) &= f(\bar{u}_1, ..., \bar{u}_{i-1}, u_i, \bar{u}_{i+1}, ..., \bar{u}_d), \end{aligned} \tag{15}$$

where $\bar{u}$ represents the input mean [25]. The mean and standard deviation from UDR can be estimated using numerical quadrature:

$$\begin{aligned} \mu_f &= \Sigma_i^d f_i(u_i)\rho_i(u_i)du_i - (d-1) f(\bar{u}), \\ \sigma_f^2 &= \Sigma_i^d (f_i(u_i) - f(\bar{u}))^2 \rho_i(u_i) du_i \end{aligned} \tag{16}$$

where ρ represents the corresponding probability distribution function of the random variable. Note that μ_f and σ_f in Equation 16 are unrelated to the doublet and source strengths described in Section II.A. The UDR approximation exhibits linear scaling with the number of random variables. The total model evaluation count of UDR can be represented as

$$\text{model evaluations} = 1 + \Sigma_i^d k_i, \tag{17}$$

where d is the number of random input variables and k is the number of quadrature evaluation points for each random input. If k is the same for each input, the model evaluation cost becomes $1 + kd$, which we see is linear with the number of dimensions. UDR is also third-order accurate in estimating the mean [25].

III. Graph-based implementation of the panel method

In this section, we discuss the implementation of our panel method and provide verification and validation results.

A. Implementation details

We have developed a low-order, unstructured panel code called VortexAD* using CSDL. With CSDL, we have access to the explicit graph, as shown in Figure 2. Our panel code utilizes vectorization to minimize model evaluation time; the vectorization also reduces the graph node count, improving both compile time and runtime. The model is compiled through JAX using Just-In-Time (JIT) compilation.

Our panel code computes potential and velocity influence coefficients based on the VSAERO formulation [15]; it is derived from the original Hess and Smith method [26] but computes potential and velocity influences using a global reference frame rather than local coordinate frames centered at each panel. We use numerical differentiation to compute the doublet-induced velocity term in Equation 7. We approximate the doublet gradient with a Taylor series expansion at point j relative to point i:

*https://github.com/LSDOlab/VortexAD

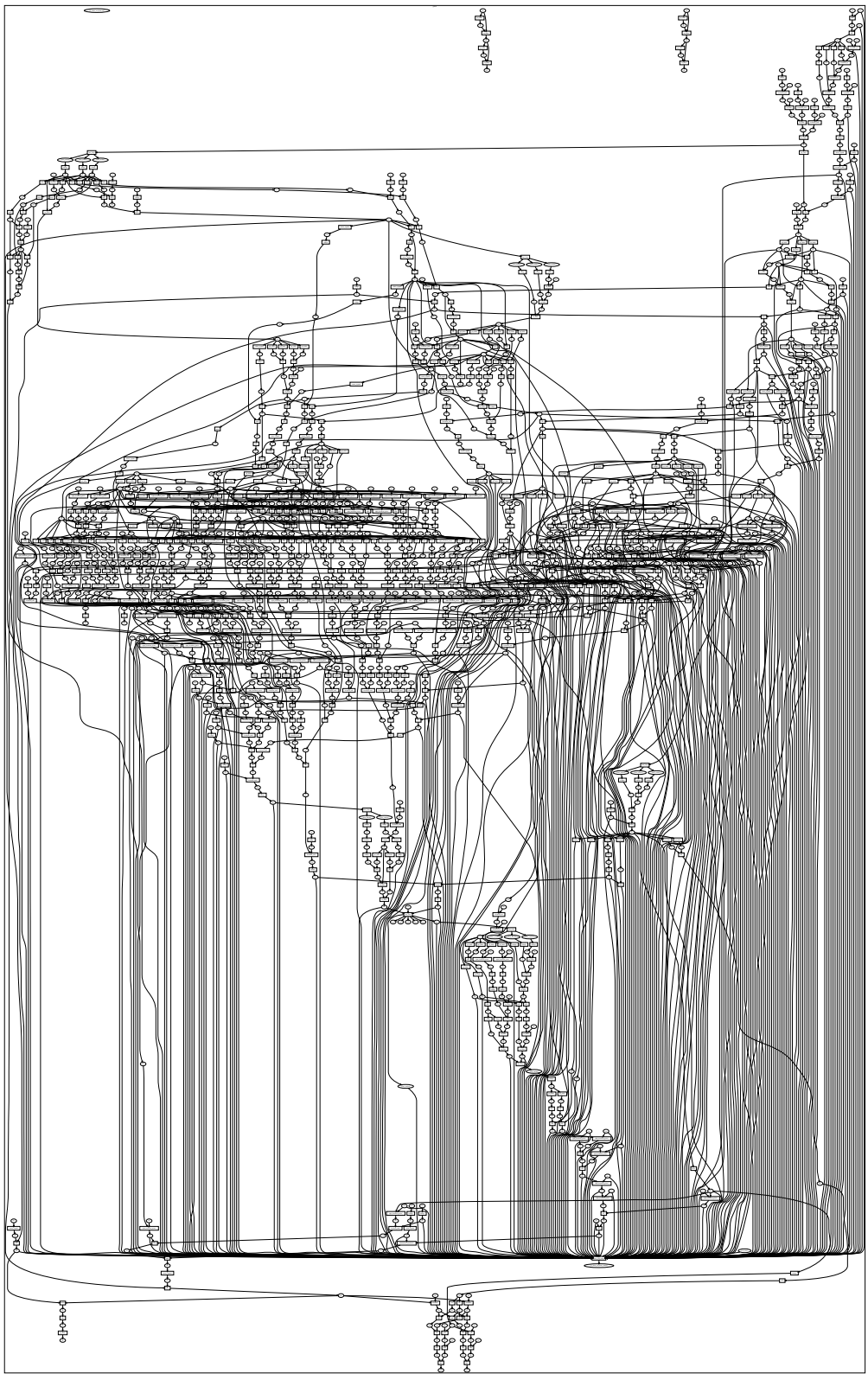


Fig. 2 CSDL panel method graph

$$\mu_j = \mu_i + \frac{\partial \mu}{\partial l}\Delta l + \frac{\partial \mu}{\partial m}\Delta m, \tag{18}$$

where l, m represent two orthogonal in-plane vectors of panel i. The partial derivative terms can be combined to simplify the expression to

$$(\nabla \mu_i) \cdot r_{ij} = \mu_j - \mu_i, \tag{19}$$

where r_{ij} is a vector of displacements in the l and m directions. For a panel i, this condition must be enforced with each neighboring panel. We reformulate this as a least-squares problem of the form

$$\begin{bmatrix} \Delta l_{i1} & \Delta m_{i1} \\ \Delta l_{i2} & \Delta m_{i2} \\ \dots & \dots \\ \Delta l_{iN} & \Delta m_{iN} \end{bmatrix} \begin{bmatrix} \frac{\partial \mu}{\partial l} \\ \frac{\partial \mu}{\partial m} \end{bmatrix}_i = \begin{bmatrix} \mu_1 - \mu_i \\ \mu_2 - \mu_i \\ \dots \\ \mu_N - \mu_i \end{bmatrix}, \tag{20}$$

where N is the number of neighbors of panel i.

Our panel code incorporates additional pre- and post-processing tools to facilitate usage and visualization. Modeling lifting behavior in potential flows requires trailing edge information; however, collecting this data manually is tedious and difficult. Our tool contains an automated trailing-edge detection method to facilitate this process. The method identifies elements, nodes, and edges at the trailing edge by searching for two criteria regarding the normal vectors of two adjacent panels. The first criterion is that the outward-normal vectors of adjacent panels must both point downstream. The second criterion assesses whether the two panels meet at a sharp edge by comparing the difference in normal vectors to a threshold angle θ_{TE}. Mathematically, this is represented as

$$\vec{n}_1 \cdot \vec{n}_2 < \cos(\theta_{TE}), \tag{21}$$

where an edge and corresponding panels are marked as trailing edge elements if this condition is satisfied.

Our panel code also supports general off-body analysis to compute flow properties in space, such as pressure and velocity. Given a field of doublets and sources, we apply superposition to compute the total velocity induced by the body at a given point in space. We model doublets as vortex rings to compute induced velocity. This is illustrated in Figure 3, which demonstrates a three-dimensional pressure coefficient field and corresponding streamlines over a generic passenger transport aircraft.

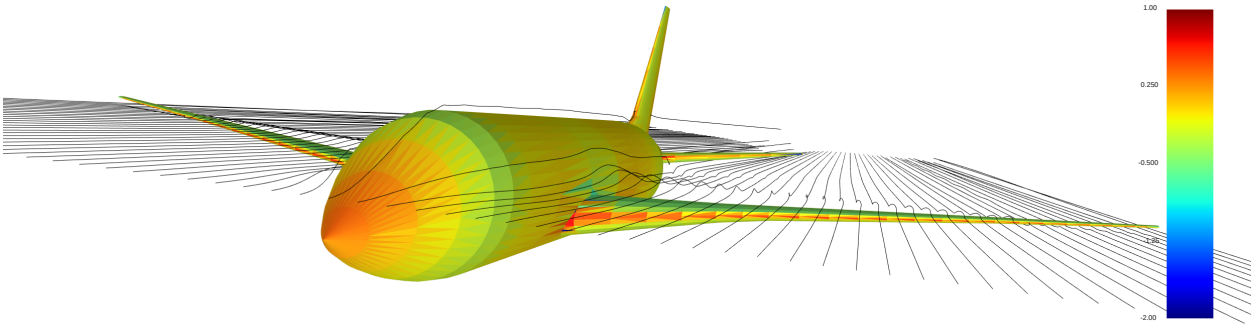
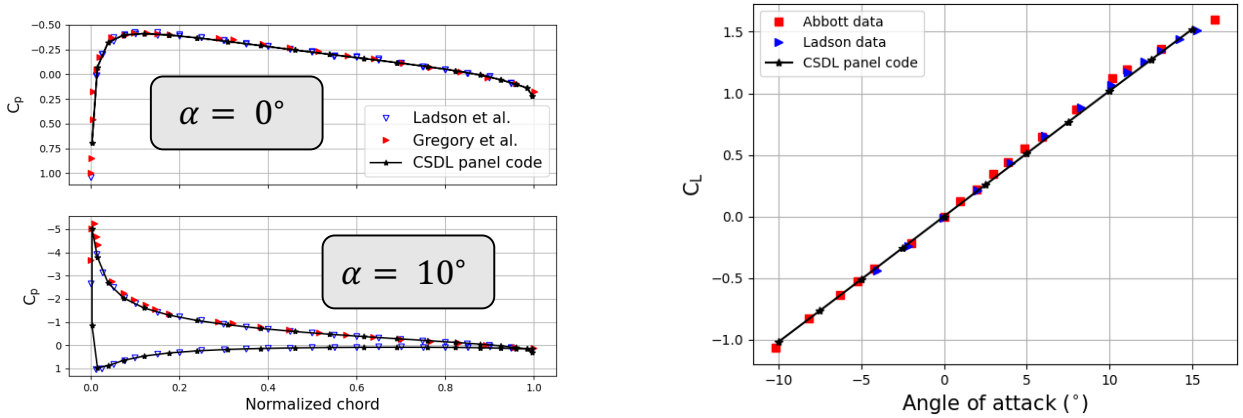


Fig. 3 Post-processing analysis of a general airliner configuration showing a pressure coefficient distribution and streamlines.

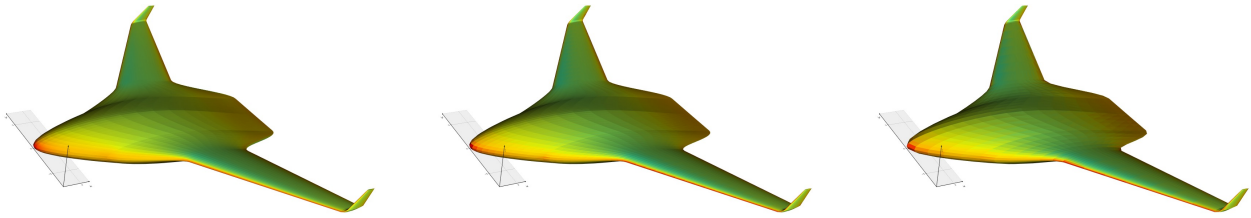
B. Verification and validation

We have validated our panel code with experimental data [27, 28] for a rectangular wing with a NACA0012 airfoil. The NACA0012 wing was modeled in VortexAD with a 10-meter wingspan and an aspect ratio of 10. As shown in Figure 4, we see good agreement between experimental data and the results from VortexAD for the pressure coefficient



(a) Pressure coefficient comparison **(b) Lift vs. angle of attack**

Fig. 4 Comparison of NACA0012 pressure coefficient distributions and lift coefficient with experimental data.



(a) Flightstream simulation [29] **(b) VortexAD (fine mesh)** **(c) VortexAD (coarse mesh)**

Fig. 5 Comparison of pressure coefficient distributions of the BWB between FlightStream [29] and VortexAD. The VortexAD fine mesh contains 9392 elements, and the VortexAD coarse mesh contains 3984 elements with refinement at the leading and trailing edges.

distributions at both 0 and 10 degrees angle of attack at the center of the NACA0012 wing. In addition, the lift coefficients also show good agreement, with a maximum error of around 5%.

We have also performed code verification on the Blended Wing Body (BWB) configuration against FlightStream [29], a surface-vorticity solver developed by Research in Flight[†]. As shown in Figure 5, we see good qualitative agreement between the pressure distributions from Flightstream and VortexAD for a BWB at zero angle of attack. We see similar regions of local flow acceleration corresponding to low pressure, along with a stagnation pressure line forming at the leading edge. We also compare high-level outputs like the lift and induced drag coefficients, along with the drag polar plot. We see relatively small errors in both C_L and C_{Di}. These distributions also demonstrate grid independence; as the grid size increases, the error of our outputs is reduced.

IV. Fast, memory-efficient panel methods using partitioned vectorization and POD

In this section, we present novel graph-based methods to reduce the n^2 cost of the panel method equations.

A. Partitioned vectorization

In this section, we present a novel partitioned vectorization approach for solving the panel method equations. As discussed in Section II.A, the panel method solves a linear system to compute the doublet strengths. This means that we have to form some representation of the matrix A in Equation 4. In this work, we focus on directly solving our linear systems by explicitly assembling A. Generally, vectorization is the fastest approach; by avoiding any iterative procedures, sequences of computations occur simultaneously, as viewed from the level of the computational graph. However, vectorization poses a major memory bottleneck due to the n^2 interactions present in the panel method. As a

[†]https://researchinflight.com/

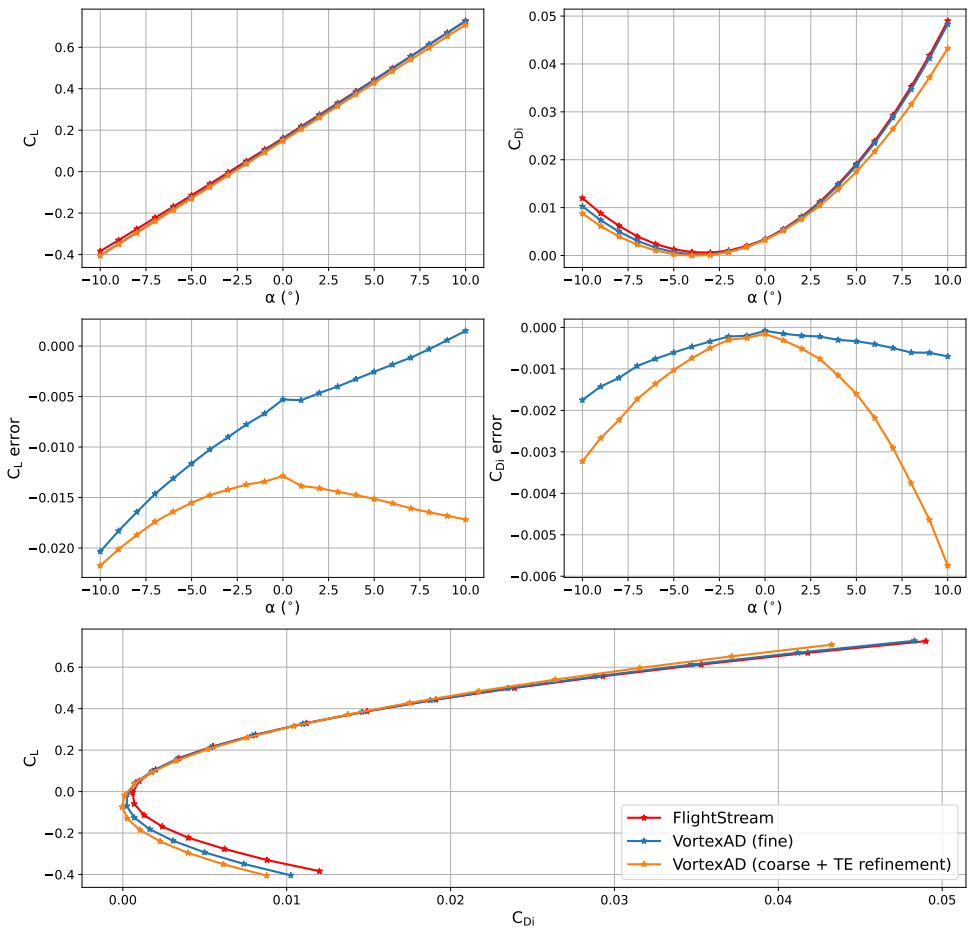


Fig. 6 Comparison of C_L, C_{Di} and the drag polar from FlightStream [29] and VortexAD for the BWB configuration.

result, vectorization is not viable for large problems.

We propose a partitioning method to alleviate the n^2 memory cost. This partitioning approach can be used to generally separate computations in the linear system matrix to balance memory cost and run time. Figure 7 illustrates this concept. For general matrix assembly, there are four types of partitioning: row, column, row-column, and column-row, shown from left to right in Figure 7; a and b represent the row and column size, respectively. The two partitioned methods on the right indicate a nested column partition or a nested row partition. The indices in the top left corner of each box represent the partition order determined by the partition type.

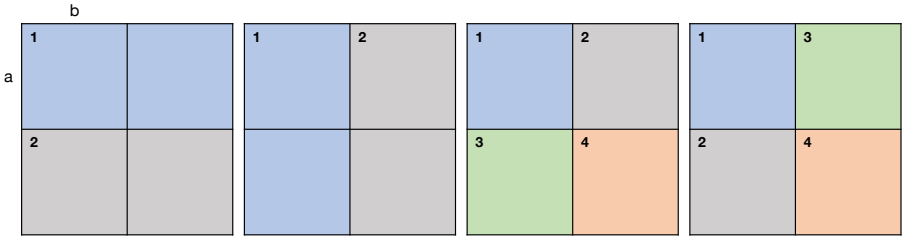


Fig. 7 A comparison of different partitioning methods for explicit matrix assembly

We can determine the scaling cost of this approach by analyzing the operation and element count, and the stored data sizes. We first quantify the cost of the vectorized approach. Computing a single element of the A matrix requires

upwards of 100 sequential operations; we identify this cost as C, which is also the minimum allowable partition cost. In the vectorized approach, we store Cn^2 values. With partitioning, considering row and column partition sizes a, b, the element count stored in memory becomes Cab; accounting for the full n^2 interactions in the final explicit matrix generation, the total data cost can be expressed as $n^2 + Cab$. We see that generalized partitioning has the potential to reduce memory cost by a significant factor. For small a, b, the scaling due to sequential operation count becomes negligible; however, partitioning alone is still too prohibitive for large problems as the dominating n^2 cost remains.

B. Partitioned vectorization with POD

In this section, we present a novel method for solving the panel method equations by combining partitioned operations with POD. When applying POD to the panel method, we solve the reduced linear system in Equation 14; the reduced linear system matrix is computed with a series of matrix-matrix products. We can utilize partitioning within the matrix-matrix products to reduce the stored data beyond partitioning alone.

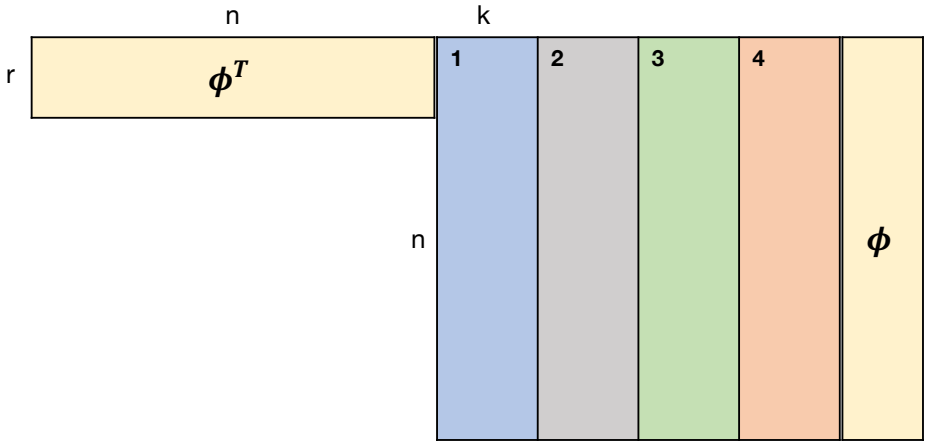


Fig. 8 A demonstration of the partitioning operation applied to matrix-matrix products with POD. Each partition of width k is stored alone in memory, so the dynamic cost of applying partitioning to matrix-matrix products with POD scales with k.

Figure 8 illustrates the sequential matrix-matrix products when using POD with the panel method; the yellow boxes represent the reduced basis vectors, and the center square represents the linear system of the full order model A. The reduced basis vectors represent a fixed cost, as these are generated fully during SVD; this fixed cost is $2nr$; because $r << n$, the reduced basis vectors introduce a linear cost with n. This method is motivated by the idea that the full order model matrix A is never fully computed; we take advantage of this using matrix-matrix products within each partitioning iteration.

We will first outline the simplest partitioned POD approach, which is pure column partitioning of A. With this approach, we take direct matrix-matrix products between ϕ^T and $A[:, ki : k(i+1)]$, where k represents the column partition size, $i \in [0, s-1]$ represents the current partition and s represents the number of partitions. The only additional elements stored in memory are equal to the size of the partitioned matrix, which is nk. With column partitioning, the total cost is $2nr + nk = n(2r + k)$. For small values of k, we see that the partitioned POD matrix-matrix product can avoid the n^2 interaction cost.

More complex partitioning schemes can be implemented to further reduce memory cost. For example, simultaneous row partitioning along A and column partitioning along ϕ^T of size l reduces the n-dominant scaling; however, this would likely penalize the computation time, as the partitioning has to iterate n^2 times. We can formulate a theoretical lower limit to the expected memory scaling: as k and l both approach 1, this cost approaches $2nr$, which equals the memory cost of storing the reduced basis matrices.

V. Numerical experiments assessing the panel method computational cost

In this section, we discuss the results of numerical experiments conducted on the memory cost and run time of the panel method. The studies were run using the BWB configuration with 3856 panels. We compare four different

methods, each shown in Table 1 with its corresponding predicted memory scaling cost. We use the JAX memory profiler to estimate the memory cost. The memory-time data is plotted in Figure 9.

Vectorization	Vectorization + POD	Partitioned Vectorization	Partitioned POD
Cn^2	$Cn^2 + 2nr(n+r+1)$	$n^2 + Cn$	$n(2r+k)$

Table 1 Memory scaling for the various solution methods

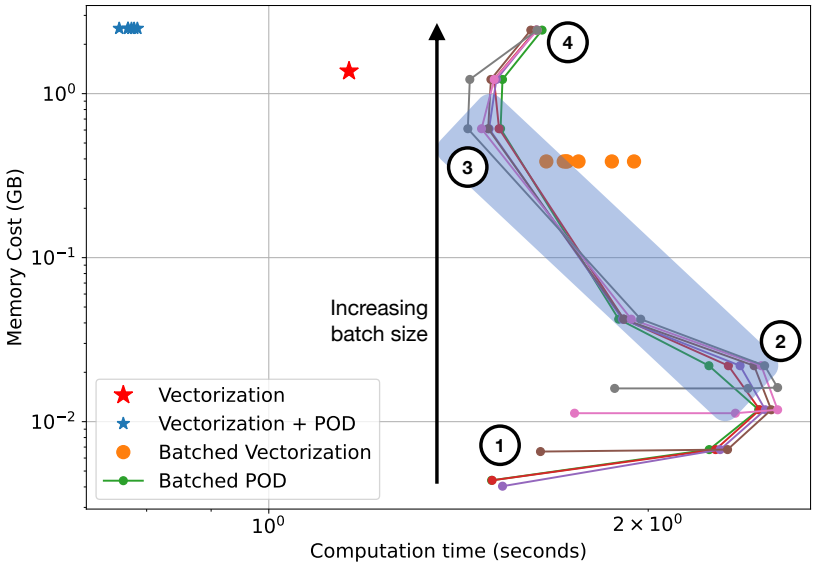


Fig. 9 Memory and time relationship for the different matrix assembly methods.

A priori, we expect memory and time to be inversely correlated. Methods that are more memory efficient usually handle less data at a given time; likewise, fast model evaluations typically arise from simultaneous or parallel operations (like vectorization). The vectorized methods scale as expected. The vectorized POD runs faster than full vectorization because the linear system size is reduced; however, it significantly increases memory cost, as the dense matrix-matrix products introduce an additional n^2 scaling. Additionally, the partitioned vectorization matrix assembly also exhibits the expected trend. The partitioned assembly requires iterative assembly of the linear system matrix; fewer elements of the system matrix are stored at a given moment, but looping compromises the run time.

The partitioned POD trends are not as clear as the other methods. Each curve represents a different reduced basis size, and each point along the curve represents a different partition size. For a given partition size, memory cost does not vary significantly with basis size; at this range of partition and basis sizes, the cost is dominated by n, the number of panels. The memory scaling at large partition sizes behaves as expected. Around the upper knee point of Figure 9 between regions 3 and 4, each subsequent point along a line represents twice the partition size, which requires roughly twice the memory, which is consistent with the expected scaling. Within the blue shaded region, we see the expected memory-time tradeoff. However, the positive correlations between memory and timing between regions 1 and 2 and regions 3 and 4 do not agree with intuition. These regions are influenced by how JAX utilizes different subroutines based on the data structure. JAX calls different kernels with certain data structures to optimize performance; however, the positive correlation indicates that the data structure is not optimal to activate more efficient linear algebra subroutines, or where temporary memory storage is activated due to high memory pressure. As a result, computations become slow and inefficient as the partition size increases.

Finally, it is important to note the relative magnitudes of changes in both timing and memory. We see that partitioning influences memory relatively more than timing. The timings tend to vary by around a second, which is the same order of magnitude as the run times. On the other hand, the memory cost changes by more than two orders of magnitude between

different methods. With these novel methods for solving the panel method equations, we have found a significant computational cost reduction. We expect similar trends in memory changes for larger grids ($n > 10000$) based on the scaling rules in Table 1. We expect run times to increase as well, but we leave those studies for future work.

VI. Demonstration of MDOUU using partitioned vectorization

In this section, we present a demonstration of MDOUU using our graph-based panel method with partitioned vectorization.

A. Problem formulation

We solve an aerostructural design optimization problem using the BWB concept. We formulate a weighted composite objective between induced drag and wing mass to penalize a major output from both disciplines. The objective is defined as

$$f = w\hat{D} + (1 - w)\hat{m}_{wing}, \tag{22}$$

where w is a weight set to $w = 0.5$. The $\hat{}$ symbol denotes normalized quantities; the induced drag and wing mass are normalized by initial reference quantities such that the ratio is of order unity. Equation 22 represents the MDO objective; for the MDOUU problem, our objective is the mean of the MDO objective f. The MDO and MDOUU formulations are shown in Table 2.

Table 2 MDO and MDOUU problem formulations

	MDO	**MDOUU**	Description	Quantity
minimize	f	$\mathbb{E}[f]$	objective	**1**
with respect to	α	α	pitch angle	1
	b	b	sectional span	3
	c	c	wing chord distribution	3
	θ	θ	wing twist distribution	3
	t_{cap}	t_{cap}	spar cap thickness	10
	t_{web}	t_{web}	spar web thickness	10
Total design variables:				**30**
subject to	$u_{max} \leq 1.5m$	$\mathbb{E}[u_{max}] + 3\mathbb{S}[u_{max}] \leq 1.5$m	max displacement	1
	$L - W = 0$	$\mathbb{E}[F_z] - 3\mathbb{S}[F_z] > 0$	minimum level flight	1
Total constraints:				**2**
with uncertainty in	N/A	ρ, M	flow properties	2
	N/A	$m_{payload}$	payload mass	1
	N/A	ϵ_t	thickness error	1
Total random variables:				**4**

We incorporate an Euler-Bernoulli beam model to analyze structural behavior and capture changes to the overall weight through wing mass. Load transfer between the panel method and the beam model is handled using an inverse-distance weighting (IDW) force map; we do not consider aeroelastic coupling, so there is no feedback. Figure 10 shows the panel mesh and beam mesh with cross sections. The total weight of the aircraft is broken down into three components: payload weight, wing weight, and fixed weight. The wing weight is determined by the spar thickness distribution that satisfies the maximum displacement constraint, while the payload weight is an uncertain input.

The BWB geometry is shown in Figure 11. We use free-form deformation (FFD) to parameterize the geometry; we set up independent FFD blocks around the distinct span sections to provide more robust local control of the geometry. We consider 3 high-level shape variables, parameterized at different sections of the geometry. The sectional span controls the widths of the center section, the transition region, and the wing half-span. The chord and twist distributions are applied to the wing root, halfway point, and tip. These optimization problems did not consider a center volume or

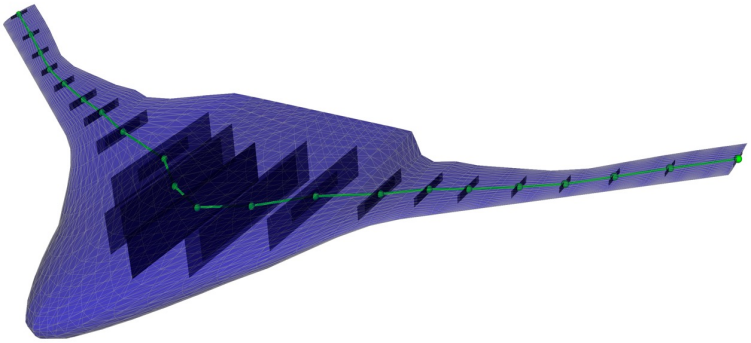
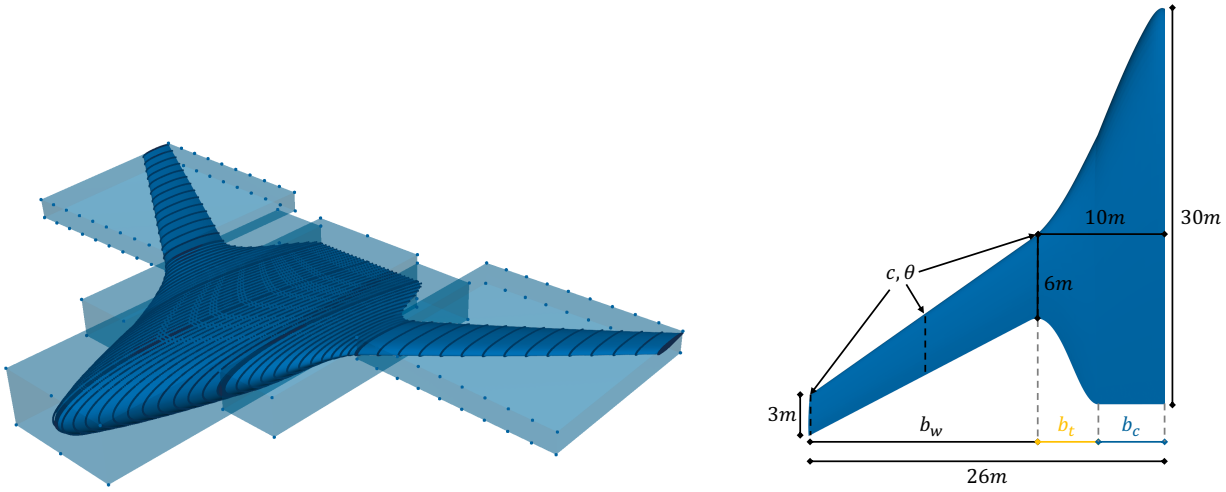


Fig. 10 BWB configuration showing the panel mesh and beam mesh together. Forces at all of the panels are transferred to the green beam nodes as point loads using an IDW map.

packing constraint; as a result, no other shape variables were applied to the center or transition region. With too much freedom, the optimizer is likely to reduce the center-line chord beyond that of a realistic passenger/cargo aircraft design.



(a) FFD blocks around the BWB **(b) BWB geometry**

Fig. 11 The geometry parameterization and initial geometry for the BWB are shown here. FFD blocks are placed around the 5 main sections of the configuration to provide local control in each region.

For the MDOUU problem, we enforce our constraints at the mean value with a margin of 3σ. This threshold is selected to ensure the statistical likelihood of constraint violation is very low (99.7% assuming a normal distribution). We enforce two constraints for the optimization problems. The first is a constraint on maximum wing displacement. The second constraint is level flight trim (lift equals weight). We take a simultaneous analysis and design (SAND) approach and provide the trim residual to the optimizer. For the MDOUU trim constraint, we enforce level flight at the -3σ point, rather than enforcing that trim must be satisfied everywhere in the distribution. This reduces the number of adjoint evaluations during optimization and still captures the impact of the constraint. The MDOUU constraint is applied to ensure that the aircraft can produce enough lift in the most adverse point of the random input space. Table 3 outlines each uncertain variable and the corresponding statistical parameters. The uncertain variables are assumed to follow normal distributions, represented with a Gauss-Hermite probability density function. We use univariate dimension reduction (UDR) to propagate uncertainties through our model. To estimate statistical moments, we use three quadrature points for each random input. According to Equation 17, we require 13 model evaluations.

Random variable	μ	3σ
Density (kg/m^3)	0.459	0.023
Mach number	0.7	0.05
Payload mass (lbs)	75000	10000
Spar thickness error (mm)	0	0.1

Table 3 The random variables are assumed to be distributed normally. The 3σ threshold is imposed to represent 99.7% of the input distribution.

B. Results

In this section, we present results of the aerostructural MDOUU problem. Table 4 summarizes high-level outputs. We see that the MDOUU case converges to a less optimal design to satisfy the constraints under input variability. The maximum displacement in the nominal MDOUU case satisfies the constraint with a margin of 8%. This margin accounts for the $+3\sigma$ variation of the output. We also see that drag increased significantly in the MDOUU case. This significant increase in drag is the penalty to ensure we can satisfy the level flight constraint under random input variability.

Case	Objective	Drag (kN)	Wing mass (kg)	Max displacement (m)
MDO	2.81	27.04	2918.01	1.50
MDOUU (nominal)	4.50	54.00	3617.6	1.38

Table 4 Optimization results

Figure 12 shows the optimized planform for both the MDO and MDOUU cases; the gray geometry is the MDO design and the yellow geometry is the MDOUU design. We see that the two results converge to roughly the same overall span, but trade off the wing span and the center section span. In the MDOUU case, the center body lift is traded with the wing lift to satisfy the minimum-level flight constraint. This trade-off also increases the aspect ratio, which controls the induced drag. However, weight is negatively impacted because the thickness of the wing spar must increase by the root to limit bending, as shown in Figure 13.

In the MDOUU case, the spar thickness is roughly double the MDO distribution towards the transition section and wing root. By increasing the beam cap thickness significantly, the optimizer penalizes the wing mass to satisfy the max displacement constraint at three standard deviations, considering input variability. Towards the center body (beam element indices 0 and 1), the thickness converges to the minimum gauge for both cases. This is expected as well, given the high moment of inertia provided by the center body due to a large chord length.

Figure 14 depicts the overall wing displacement applied to the geometry. During optimization, displacement transfer was not applied from the structural model and the panel method; this displacement was only applied as a post-processing visualization. It is important to note that the initial geometry contains a wing dihedral angle of 4°. We see that the relative displacement of the wing is higher for the MDO case than the MDOUU case.

In addition to quantifying the optimal solution, we also aim to assess the performance of UDR as a method to propagate uncertainty. Table 5 shows the estimated mean and standard deviation of our objective and two constraints. We quantify the error of UDR with statistics computed using 250 Monte Carlo samples. UDR captures the mean very accurately with relatively few model evaluations; recall that we use 13 total model evaluations to compute statistical outputs. However, UDR is not as accurate for computing higher-order moments, where nonlinearities are present. This error can be reduced by using more quadrature evaluation points, which only incurs a linear scaling when using UDR.

QOI	UDR Mean	Error (%)	UDR Std. Dev.	Error (%)
Objective	4.51	0.01	0.08	17.67
Trim constraint (N)	109849.24	0.23	36617.93	16.93
Tip displacement (m)	1.38	0.18	0.04	14.65

Table 5 Error analysis of statistical outputs

In addition, we can quantify how the MDO design fares in the MDOUU problem setting. Table 6 outlines the

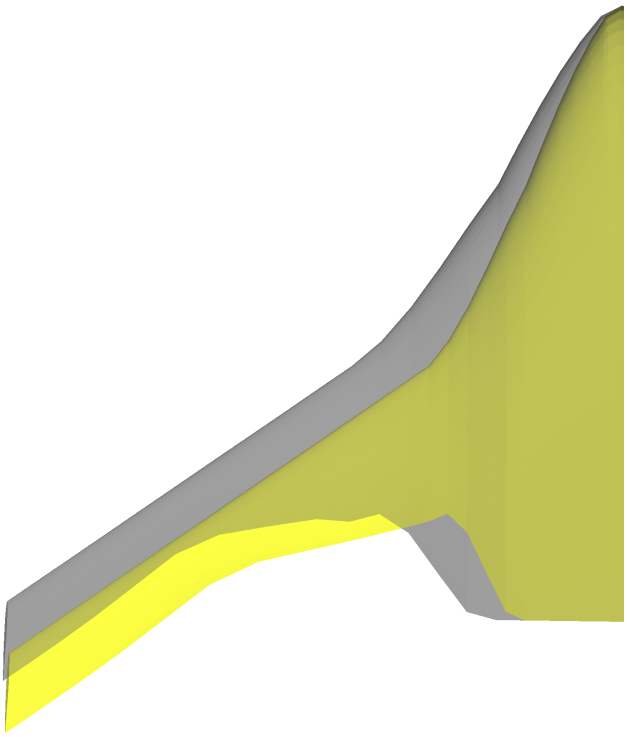


Fig. 12 Comparison of optimized BWB planform geometry. The gray is the MDO result and the yellow is the MDOUU result.

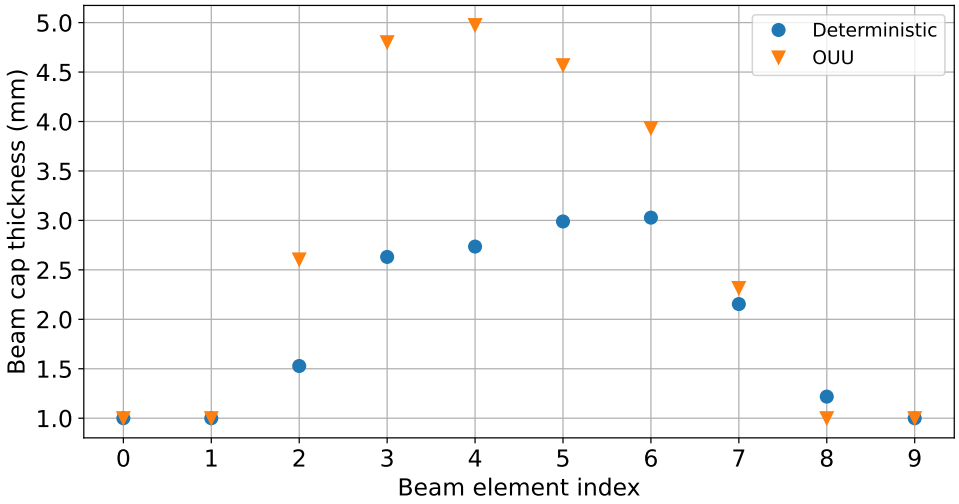


Fig. 13 MDO and MDOUU optimized beam thickness distribution

key outputs of the MDO and MDOUU designs when assessed against the MDOUU objective and constraints. We see that the MDO design satisfies the maximum displacement constraint by a significant factor, but is unable to satisfy the minimum level flight constraint. Although the MDO design converges to a more optimal solution, it is unable to generate the lift necessary to sustain minimum level flight when assessed in the random input space. The reduced amount of lift generated by the MDO design explains the trends behind a more optimal design and a lower maximum wing displacement. A reduction in lift leads to a reduction in drag; the wing mass will also drop because the forces

Fig. 14 Front view of the MDO and MDOUU optimal designs with displacement transfer of the structural solver applied to the geometry

Optimization case	Objective	Min. level flight (kN)	Max displacement (m)	Wing mass (kg)	Lift (kN)
MDO	2.98	-135.04435	0.97885304	2853.45	1066.30
MDOUU	4.51	-0.0046	1.5	3617.6	1221.98

Table 6 Error analysis of statistical outputs

applied to the beam are reduced. The wing can then trim mass while still satisfying the MDO maximum displacement constraint.

Finally, we aim to quantify the computational cost of performing MDOUU at this scale. The overall runtime of the MDOUU case was roughly 11 times that of the MDO case; however, there are similar memory profiles despite significantly more model evaluations and additional inputs. Like the partitioned vectorization approach for the panel method, the MDOUU case utilized a similar partitioning approach for model evaluations at the different quadrature points. Because UDR decomposes a function into a series of univariate, decoupled functions, we can take advantage of partitioning each evaluation to save memory; this is identical to incremental assembly of the linear system matrix using partitioned evaluations.

Optimization run	Run time (minutes)	Memory use (GB)	Model evaluations	Major iterations
MDO	42.5	20.6	186	102
MDOUU	477.40	21.6	144	61

Table 7 Optimization results

VII. Conclusion

In this paper, we outlined the development of a modern panel method formulated within a graph-based modeling paradigm. We presented the formulation and results of verification and validation studies. We then presented novel numerical methods for solving the dense linear system using partitioned vectorization and POD; we determined simple scaling relationships that relate expected memory cost to the number of panels. Then, we presented the results of numerical experiments comparing the computational efficiency of our novel methods with standard approaches such as vectorization and standard POD. With our novel partitioned vectorization, we found close to an order of magnitude reduction in memory. When including POD, we managed to increase this reduction by another order of magnitude and recover a linear scaling cost with respect to the number of panels.

Using these methods, we were able to demonstrate large-scale multidisciplinary design optimization under uncertainty (MDOUU) of the blended wing body concept. We considered up to thirty design variables and four random variables. Our MDOUU-optimal design saw about a 25% increase in wing mass and twice the drag compared to the MDO model to satisfy the given constraints under input variability. We found that the univariate dimension reduction (UDR) method can estimate the mean with under 1% error for relatively few evaluations, compared to Monte Carlo sampling. In addition, the partitioned vectorization applied to UDR incurred a minimal memory increase of about 5% or 1 GB between the MDO and MDOUU problems, despite requiring 13 times the number of model evaluations per forward

evaluation of the optimizer.

Acknowledgments

The material presented in this paper is, in part, based upon work supported by NASA under award No. 80NSSC23M0217. The authors would also like to acknowledge the Multidisciplinary Science and Technology Center of the Aerospace Systems Directorate, Air Force Research Laboratory for funding and supporting this effort through the Collaborative Center for Design and Research of Interdisciplinary Systems program. The authors would also like to express their gratitude to Nicholas Orndorff for his invaluable effort in generating the blended wing body geometry.

References

[1] Sarojini, D., Ruh, M. L., Joshy, A. J., Yan, J., Ivanov, A. K., Scotzniovsky, L., Fletcher, A. H., Orndorff, N. C., Sperry, M., Gandarillas, V. E., Asher, I., Chambers, J. T., Gill, H., Lee, S., Cheng, Z., Rodriguez, G., Zhao, S., Mi, C., Nascenzi, T., Cuatt, T., Winter, T. F., Guibert, A. T., Cronk, A., Kim, H. A., Meng, S., and Hwang, J. T., *Large-Scale Multidisciplinary Design Optimization of an eVTOL Aircraft using Comprehensive Analysis*, 2023. https://doi.org/10.2514/6.2023-0146, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-0146.

[2] Ruh, M. L., Fletcher, A., Sarojini, D., Sperry, M., Yan, J., Scotzniovsky, L., van Schie, S. P., Warner, M., Orndorff, N. C., Xiang, R., Joshy, A. J., Zhao, H., Krokowski, J., Gill, H., Lee, S., Cheng, Z., Cao, Z., Mi, C., Silva, C., Wolfe, L., Chen, J.-S., and Hwang, J. T., *Large-scale multidisciplinary design optimization of a NASA air taxi concept using a comprehensive physics-based system model*, 2024. https://doi.org/10.2514/6.2024-0771, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0771.

[3] Adler, E. J., and Martins, J. R. R. A., "Efficient Aerostructural Wing Optimization Considering Mission Analysis," *Journal of Aircraft*, 2022. https://doi.org/10.2514/1.c037096.

[4] Jasa, J. P., Hwang, J. T., and Martins, J. R., "Open-source coupled aerostructural optimization using Python," *Structural and Multidisciplinary Optimization*, Vol. 57, 2018, pp. 1815–1827.

[5] Chauhan, S. S., and Martins, J. R. R. A., "RANS-Based Aerodynamic Shape Optimization of a Wing Considering Propeller–Wing Interaction," *Journal of Aircraft*, Vol. 58, No. 3, 2021, pp. 497–513. https://doi.org/10.2514/1.C035991, URL https://doi.org/10.2514/1.C035991.

[6] Amano, R., and Malloy, R., "CFD analysis on aerodynamic design optimization of wind turbine rotor blades," *World Academy of Science, Engineering and Technology*, Vol. 60, 2009, pp. 71–75.

[7] Farrokhfal, H., and Pishevar, A., "Aerodynamic shape optimization of hovering rotor blades using a coupled free wake–CFD and adjoint method," *Aerospace Science and Technology*, Vol. 28, No. 1, 2013, pp. 21–30. https://doi.org/https://doi.org/10.1016/j.ast.2012.09.004, URL https://www.sciencedirect.com/science/article/pii/S1270963812001381.

[8] Gray, A. C., Riso, C., Jonsson, E., Martins, J. R. R. A., and Cesnik, C. E. S., "High-Fidelity Aerostructural Optimization with a Geometrically Nonlinear Flutter Constraint," *AIAA Journal*, Vol. 61, No. 6, 2023, pp. 2430–2443. https://doi.org/10.2514/1.J062127, URL https://doi.org/10.2514/1.J062127.

[9] McDonnell, T. G., "Gradient-Based Optimization of Highly Flexible Aeroelastic Structures," Ph.D. thesis, 2023. URL https://www.proquest.com/dissertations-theses/gradient-based-optimization-highly-flexible/docview/2866082369/se-2.

[10] Economon, T., Palacios, F., and Alonso, J., "A coupled-adjoint method for aerodynamic and aeroacoustic optimization," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) conference and 14th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2012, p. 5598.

[11] Fabiano, E., and Mavriplis, D., "Adjoint-based aeroacoustic design-optimization of flexible rotors in forward flight," *Journal of the American Helicopter Society*, Vol. 62, No. 4, 2017, pp. 1–17.

[12] Abdul-Kaiyoom, M. A. S., Yildirim, A., and Martins, J. R. R. A., "Coupled Aeropropulsive Design Optimization of an Over-wing Nacelle Configuration," *Journal of Aircraft*, Vol. 0, No. 0, 0, pp. 1–23. https://doi.org/10.2514/1.C037678, URL https://doi.org/10.2514/1.C037678.

[13] Gandarillas, V., Joshy, A. J., Sperry, M. Z., Ivanov, A. K., and Hwang, J. T., "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis," *Structural and Multidisciplinary Optimization*, Vol. 67, No. 5, 2024, p. 76.

[14] Wang, B., Sperry, M., Gandarillas, V. E., and Hwang, J. T., "Accelerating model evaluations in uncertainty propagation on tensor grids using computational graph transformations," *Aerospace Science and Technology*, Vol. 145, 2024, p. 108843. https://doi.org/https://doi.org/10.1016/j.ast.2023.108843, URL https://www.sciencedirect.com/science/article/pii/S1270963823007393.

[15] Maskew, B., "Program VSAERO theory Document: a computer program for calculating nonlinear aerodynamic characteristics of arbitrary configurations," Tech. rep., NASA, 1987.

[16] Ashby, D. L., "Potential flow theory and operation guide for the panel code PMARC," Tech. rep., NASA, 1999.

[17] Carmichael, R., and Erickson, L., "PAN AIR-A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations," *14th Fluid and Plasma Dynamics Conference*, 1981, p. 1255.

[18] Davis, J., and Marshall, D., "A Higher-Order Method Implemented in an Unstructured Panel Code to Model Linearized Supersonic Flows," *AIAA Scitech 2020 Forum*, 2020, p. 1748.

[19] Goates, C. D., "Development, Implementation, and Optimization of a Modern, Subsonic/Supersonic Panel Method," 2023.

[20] Anderson, R., and Ning, A., "Solving Unsteady Potential Flow Problems in O (n) Time," 2024.

[21] Greengard, L., and Rokhlin, V., "A fast algorithm for particle simulations," *Journal of Computational Physics*, Vol. 73, No. 2, 1987, pp. 325–348. https://doi.org/https://doi.org/10.1016/0021-9991(87)90140-9, URL https://www.sciencedirect.com/science/article/pii/0021999187901409.

[22] Zimmermann, R., and Görtz, S., "Improved extrapolation of steady turbulent aerodynamics using a non-linear POD-based reduced order model," *The Aeronautical Journal*, Vol. 116, No. 1184, 2012, p. 1079–1100. https://doi.org/10.1017/S0001924000007491.

[23] Li, J., and Cai, J., "Massively multipoint aerodynamic shape design via surrogate-assisted gradient-based optimization," *AIAA Journal*, Vol. 58, No. 5, 2020, pp. 1949–1963.

[24] Rahman, S., and Xu, H., "A univariate dimension-reduction method for multi-dimensional integration in stochastic mechanics," *Probabilistic Engineering Mechanics*, Vol. 19, No. 4, 2004, pp. 393–408.

[25] Wang, B., Orndorff, N. C., Sperry, M., and Hwang, J. T., "A gradient-enhanced univariate dimension reduction method for uncertainty propagation," *Aerospace Science and Technology*, Vol. 155, 2024, p. 109602. https://doi.org/https://doi.org/10.1016/j.ast.2024.109602, URL https://www.sciencedirect.com/science/article/pii/S1270963824007314.

[26] Hess, J. L., and Smith, A. O., "Calculation of potential flow about arbitrary bodies," *Progress in Aerospace Sciences*, Vol. 8, 1967, pp. 1–138.

[27] Ladson, C. L., Hill, A. S., and Johnson Jr, W. G., "Pressure distributions from high Reynolds number transonic tests of an NACA 0012 airfoil in the Langley 0.3-meter transonic cryogenic tunnel," Tech. rep., NASA, 1987.

[28] Gregory, N., and O'reilly, C., "Low-speed aerodynamic characteristics of NACA 0012 aerofoil section, including the effects of upper-surface roughness simulating hoar frost," 1970.

[29] Chakraborty, I., Ahuja, V., Comer, A., and Mulekar, O., "Development of a modeling, flight simulation, and control analysis capability for novel vehicle configurations," *AIAA Aviation 2019 Forum*, 2019, p. 3112.