



An Exception Algorithm to Ease Convergence in Multistage Turbomachinery Throughflow Calculations

Mario Di Martino¹

Centre for Propulsion and Thermal Power Engineering,
Gas Turbine Engineering Group,
Cranfield University,
School of Engineering,
Cranfield, Bedfordshire MK43 0AL, UK
e-mail: mario.di-martino@cranfield.ac.uk

Vassilios Pachidis

Centre for Propulsion and Thermal Power Engineering,
Gas Turbine Engineering Group,
Cranfield University,
School of Engineering,
Cranfield, Bedfordshire MK43 0AL, UK
e-mail: v.pachidis@cranfield.ac.uk

Iterative solvers used in turbomachinery, spanning throughflow formulations and higher-order computational fluid dynamics (CFD), depend critically on how they are initialized. The starting state determines not only the number of iterations but often whether a solution is found at all when residuals are noisy or non-smooth. In such regimes, fast gradient updates can stall, oscillate, or violate bounds even when the underlying model is sound. This work introduces a solver-level exception algorithm that detects those failure signatures at run time and temporarily replaces the gradient step with a bounded, derivative-free fallback to reestablish a safe descent under strict feasibility limits. Control is returned to the fast path once a reliable reduction is found, preserving turnaround time in easy regions while adding robustness where the reference update is untrustworthy. The practical motivation is to reduce user workload and eliminate manual operating-point marching while enabling wider off-design coverage and characteristics traverses without a discernible accuracy or speed penalty. Although illustrated within a streamline curvature throughflow solver, the approach is general and can be interfaced with other throughflow methodologies and, more broadly, with higher-order CFD. Validation on single- and multistage axial-flow turbines shows improved robustness without a discernible accuracy or speed penalty within a state-of-the-art throughflow environment.

[DOI: 10.1115/1.4070540]

Keywords: streamline curvature, throughflow method, automation, derivative-free solvers, multistage turbomachinery, computational fluid dynamics (CFD), fan and compressor aerodynamic design, turbine aerodynamic design, turbomachinery blading design

1 Introduction

In turbomachinery, reliable throughflow and higher-order computational fluid dynamics (CFD) simulations require robust initialization and update rules that remain trustworthy under noisy or non-smooth residuals.

This sensitivity is amplified by near-choke operating points, strong diffusion, end-wall gradients, and tight annulus curvature, which create residuals with plateaus, kinks, and rapidly changing slopes. Under these conditions, fast gradient updates that are attractive for turnaround time can make stall, oscillate, or violate feasibility bounds. In practice, “robust initialization” is pursued with pragmatic devices, e.g., restart maps, mean-line or duct-flow seeds, external guidance from experimental or higher-fidelity data, and local mesh coarsening, which all add to pre-processing time. While these measures may stabilize iterations, they rely on prior knowledge of the expected state, require user intervention

(e.g., operating-point marching), and can bias the path taken by the solver, particularly in multistage turbomachines where each blade row re-projects upstream errors.

This work investigates whether solver robustness can be improved by reducing the dependence on *perfect initialization*. The central idea is to make the update rule itself resilient when the starting point is imperfect and the local residual state is unreliable.

Although the approach is illustrated within a streamline curvature (SLC) throughflow solver, chosen because its close-coupling of streamlines’ geometry, spanwise gradients, and thermodynamics yields a stringent, stiff test case, the underlying concept is general and transferable to other throughflow methodologies and to higher-order CFD.

This article develops this idea and evaluates its impact on robustness and user workload, while preserving speed and accuracy appropriate to preliminary design.

2 The Streamline Curvature Throughflow Method and Its Stiffness

The importance of throughflow models in the aerodynamic design of turbomachinery lies in their continued utility, particularly

¹Corresponding author.

Contributed by the International Gas Turbine Institute (IGTI) of ASME for publication in the JOURNAL OF TURBOMACHINERY. Manuscript received May 30, 2025; final manuscript received November 18, 2025; published online December 30, 2025. Assoc. Editor: Emil Göttlich.

in preliminary design to generate initial blade shapes and support iterative design processes and to initialize higher-order CFD.

Developing throughflow methods as effective preliminary design analysis tools for turbomachines requires codes to have an acceptable accuracy, to be computationally economical, but also to demonstrate a high convergence rate.

The SLC method is a throughflow calculation that solves the Eulerian formulation of the Navier–Stokes momentum, continuity, and energy equations on a meridional grid (Fig. 1) whose nodes lie at streamline intersections with blade leading and trailing edges. Assumptions are steady, axisymmetric, and adiabatic inviscid flow; viscous effects enter via loss correlations.

The SLC method is governed by the radial equilibrium equation (REE), which couples streamline geometry, meridional/spanwise velocities, and thermodynamic gradients via Eq. (1)

$$\frac{dV_m^2}{dt} = b_1 V_m^2 + b_2 V_m + b_3 \quad (1)$$

where the coefficients b_1 , b_2 , and b_3 contain geometric details of turbomachinery stacking and the *unknowns*, including the streamlines' geometry, two components of the velocity vector (meridional V_m and spanwise V_w), and intensive thermodynamic properties. A solution to the REE has not been expressed in a closed-form turbomachinery solution other than the canonical case of a swirling inviscid and incompressible flow in a duct, i.e., Greitzer's form [2]; numerical root-finding is required.

As throughflow is a 2D fluid-dynamics problem, in order to obtain the full flow conditions at each of the nodes, two components of the velocity vector $\underline{V} = V(x, r)$ are required, and, for the *state postulate* of thermodynamics, only two independent, intensive, thermodynamic coordinates are needed to calculate all the others. Hence, this is a problem of four independent unknowns, which requires four fundamental equations.

The continuity equation that sets the mass flowrate (W) through the annulus, which, without leakages, is preserved throughout the turbomachine and is provided as a boundary condition at the inlet, hence it is a *known*. This is solved, at a given station, to provide the average magnitude of velocity; noting that velocity and density at a given station x are only a function of the spanwise direction (t) and defining stream-surface thickness (φdt), the elemental area that is to be integrated in the continuity equation can be written as

$$W = \int_{t_{hub}}^{t_{tip}} \rho(t) V_m(t) \varphi dt \quad (2)$$

In addition to continuity, the system is then closed via two additional equations. The velocity triangles allow us to relate turbocomponents' inlet velocity and angles to outlet velocity and angles via trigonometric relationships by knowing the blade speed $U = Nr$ and considering flow deviation.

Finally, the energy equation relates the inlet and outlet total enthalpy through the inlet and outlet kinetic energy and static enthalpy

$$\underbrace{h_1 + \frac{V_1^2}{2}}_{\text{known}} = \underbrace{h_2^{id} + \Delta h_{loss}}_{h_2} + \frac{V_2^2}{2} = \underbrace{c_{p2} t_2}_{h_2} + \frac{V_2^2}{2} \quad (3)$$

The REE, Eq. (1), together with stream-tube continuity Eq. (2) and the energy equation, Eq. (3), yields a tri-fold nonlinear problem which can be advanced in nested loops: the outer loop enforces boundary conditions, while the inner loops solve REE, continuity, and energy in sequence (Fig. 2).

2.1 Stiffness of the Streamline Curvature Method. In practice, nine (①–⑨) recurring issues are observed, hence modern throughflow implementations include convergence aids.

These issues arise from a combination of flow physics and numerical choices; established convergence-aiding techniques, such as restart maps, can relax state updates: they often render ①–④ tractable and can delay or soften ⑤–⑧. But they also impose costs: extra pre-processing, parameter tuning, reduced spanwise fidelity, and a risk of masking physically-meaningful structures. In addition, issue ⑨ is introduced by the choice of the core solver, which is the main objective of this study.

① **Near-choke sensitivity of the continuity loop and non-optimal initialization conditions.** Small mass-flow changes cause large pressure-ratio swings (especially near choking, see Fig. 3). Gradual operating-point stepping from a known solution, and careful relaxation often make the case tractable; nevertheless, the region remains not robust and can still fail. This can make the whole process cumbersome, which will negate two of the objectives of throughflow methods, ease of use and speed.

② **High spanwise gradients from geometric boundaries.** Persistent near hubs and tips and aggravated by tight axial spacing or strong annulus curvature, these induce changes in streamline positions and can trigger nonphysical behavior (e.g., streamlines cross-over, reversal). Careful interpolation/extrapolation, local refinement near geometric kinks, and judicious reduction of nodes in uniform zones usually stabilize the iteration, albeit at added pre-processing cost.

③ **Anisotropic meshes.** Large streamwise spacing relative to spanwise spacing leads to non-uniform error propagation. Mesh design (including controlled coarsening or refinement) can reduce sensitivity and make the problem tractable.

④ **Strongly diffusing passages that invert energy consistency.** Numerically, the outlet static pressure can exceed the total pressure, forcing negative meridional speed in the energy balance. This problem can be exacerbated by non-uniform pressure and temperature profiles (irrotational flow), which can result in a sudden change to the flow gradients. Smoother interpolation and, when necessary, temporary numerical blockage to limit the

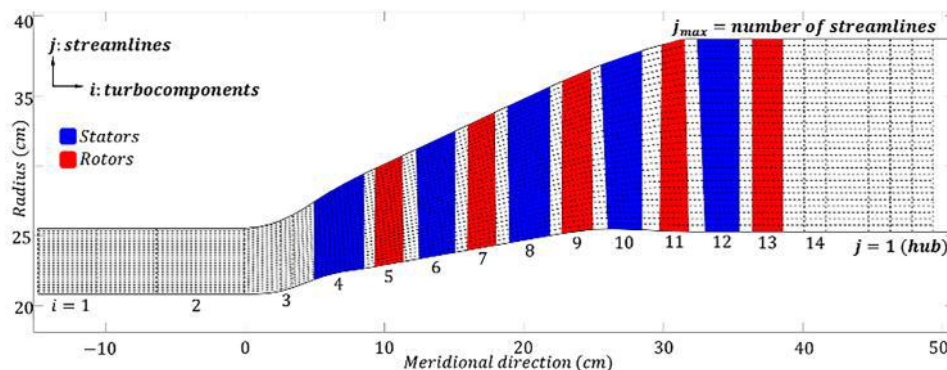


Fig. 1 SLC method meridional grid for the five-stage EEE LPT [1]

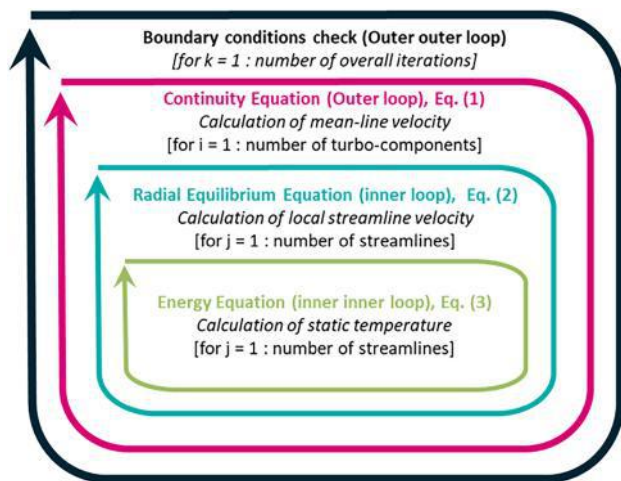


Fig. 2 Nested structure for the SLC method

effective area can carry the solution to a consistent state; these remedies, however, must be tapered to avoid biasing the spanwise load.

⑤ **Introduction of local loss/deviation correlations.** When correlations are introduced against the iterating flow field, the flow field becomes non-smooth. If the correlations are coupled with flow parameters, any mis-estimation injects a biased loss/deviation that is then amplified by the solver, especially when such correlations are activated too early in the run. Choosing entropy as an unknown further exacerbates this due to the exponential pressure–entropy relation. If auxiliary functions are introduced, for example, to model cooling flow via an approximate entropy change calculation, a small entropy error is an exponentially magnified error on the pressure ratio with a significant impact on the radial equilibrium equation. Smoothing and relaxation can delay failure, but do not resolve the underlying non-smooth update logic.

⑥ **Distortions from blockage formulations.** Introducing blockage to emulate viscous displacement thickness can lead to non-physical annulus coordinates. Because the remedy is itself a modeling device, increasing blockage does not fix the root cause and can conceal rather than resolve errors.

⑦ **Nonphysical aliasing from imposed higher-fidelity codes' losses and deviation,** which, e.g., if mapped without thermodynamic consistency (temperature–pressure–entropy triad), force incompatible states and contaminate other variables, hence causing the throughflow solver to chase incompatible residuals. No amount of smoothing or relaxation reliably prevents this if the targets are not dynamically consistent with the throughflow equations.

⑧ **Sensitivity to local flags causing oscillatory behavior or ambiguity,** e.g., across loss correlations or topology switches in mixed hyperbolic/elliptic regions.

⑨ **The update logic of all throughflow implementations is gradient-based:** secant (quasi-Newton) [3] or Newton-type steps [4–7], and Runge–Kutta stepping [4,8–11]. Gradient-based schemes are fast but are also prone to divergence if reliable Jacobian information is not available. As a result, the same failure signatures recur: (i) near-equal residuals, e.g., caused by non-smooth regions, ill-conditioned stencils can cause divisions by near-zero quantities in the radial equilibrium/energy loops and the iteration to be “thrown-out of the region of interest” (Fig. 4) or non-numeric arithmetic (NaNs), upon which the iteration cannot determine a direction; (ii) local minima/maxima of the residual (no root in the vicinity) or plateaus can cause vanishing steps and the update to stall; (iii) physical bound violations, e.g., steps to negative meridional speed, implausible relative Mach Number, or deviation angle overshoots followed by repeated gradient-based projections can trap the method on a boundary with no descent direction; (iv) reliance on local slope information exactly where slopes are oscillatory or undefined. Without a solver-level remedy, relaxation, smoothing, or mesh adjustments, these failures can be postponed, but their root cause will not be removed.

2.2 Why Multistage Turbines Are More Demanding?. All nine issues (①–⑨) are more severe in multistage turbomachines because instabilities do not remain local. Each blade row inherits its inlet state, mass flow, swirl, loss, and temperature profiles from the upstream row. An unstable update or small error in one row is immediately convected downstream and mapped onto neighboring stream-surfaces. The coupled radial equilibrium, continuity, and energy loops then re-project that perturbation into the next row's boundary conditions, where it can be amplified by near-choke sensitivity, strong diffusion, or end-wall gradients. With many rows in series, this streamwise and spanwise feedback creates a cascade in which a single misstep can lock the turbomachine into oscillations, stall global progress, or drive nonphysical states. Consequently, robustness requirements are materially higher than in single-stage cases.

For turbines specifically, issues ①–⑤ are further exacerbated: ① sensitivity of pressure ratio to mass-flow is acute near design because most turbines operate at or near choking; ② combustor exit profiles impose stronger radial temperature non-uniformities than typically found in compressors; ③ turbines often have lower aspect ratio (tighter axial spacing), aggravating anisotropy; ④ turbine passages are inherently diffusing; ⑤ cooling, if and where present, can further sharpen spanwise and streamwise non-uniformities, but it is an amplifier rather than the primary cause.

2.3 Convergence-Aiding Techniques. OEMs rely on proprietary codes and extensive empirical initialization tuned to familiar architectures, which aids convergence but leaves little room for fundamental design change. In the public domain, comparative studies of iterative strategies are scarce. Accordingly, a concise

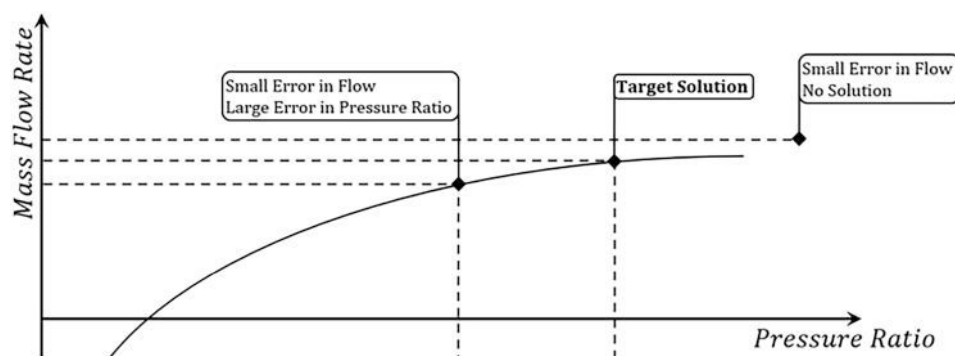


Fig. 3 Mass flow convergence problem in throughflow

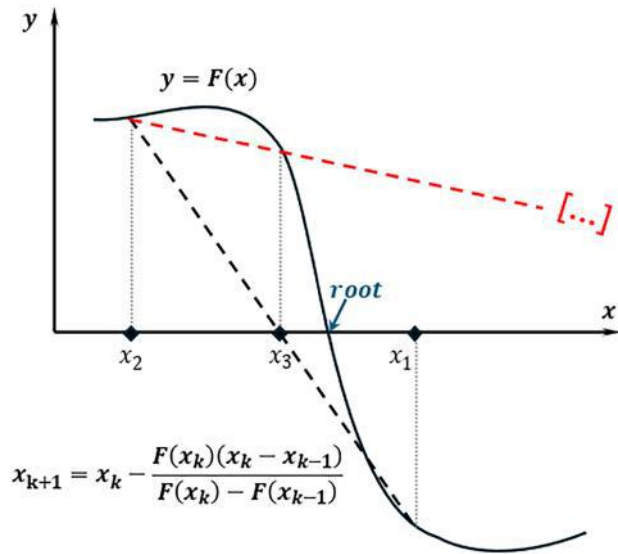


Fig. 4 Illustration of the secant method, in a particular example of divergence: the dashed secant through two nearby points yields an unstable next iterate because the residual slope is near-vertical

review of stability-oriented techniques is required before proposing a new approach.

Cranfield's Synthesis of Correlations for the Robust Assessment of Turbomachinery Engine Systems (SOCRATES) [12] implements the key techniques, notably dynamic numerical damping and spanwise mixing, making it a suitable state-of-the-art platform to test our exception algorithm, allowing the incremental benefit to be isolated and measured.

- (a) **Better initialization.** Pragmatic initialization reduces early instability. Typical approaches include: (i) uniform or duct-flow seeds that satisfy global mass-flow and area constraints [4,8,13]; (ii) mean-line or lower-order predictions to pre-populate meridional and spanwise velocity components [14–16]; (iii) restart maps that reuse converged solutions at neighboring operating points to traverse a speed/flow characteristic [17]; and (iv) selective borrowing of CFD/experimental guidance if available [18,19]. These strategies lower the probability of non-number residuals and shorten the transient to a viable iteration but cannot, alone, recover a calculation once the update direction becomes unreliable. Systematic schedules that climb the speed/flow map from a validated point (rather than cold-starting each condition) are effective but cumbersome for full characteristics, interrupting automation, and adding time.
- (b) **Interpolation/Extrapolation that preserves smoothness.** Because SLC computes derivatives from nodal data on stream-surfaces, robust interpolation and carefully constrained edge extrapolation are essential. Poor stencils amplify spurious curvature/gradient estimates in the radial equilibrium and energy loops and can exacerbate oscillations. Smooth interpolation techniques are widely implemented in throughflow [20].
- (c) **Numerical damping.** Relaxation factors on key state updates are widely used to prevent overshoot in sensitive regions (high curvature, strong diffusion, shock/loss pockets). Fixed damping values < 1.0 are relatively simpler but problem dependent; dynamic schemes modulate the factor with iteration count [6] or residual trend [3]. While numerical damping is an effective guardrail, it only scales the update $V_{k+1} = V_k + \alpha \Delta V_k$, it cannot correct a misdirected step or recover from plateaus/non-smooth residuals where gradients or secant differences are unreliable; it merely

slows divergence or prolongs stalling. By contrast, the exception algorithm activates only when specific failure signatures occur (e.g., equal-residual divisions, iteration plateaus, bound violations) and temporarily replaces the damped secant with a bounded, derivative-free local search that discovers a new descent direction while enforcing hard feasibility limits. Once a viable reduction is found, control returns to the fast secant iteration, so there is no global speed penalty in easy regions, no case-by-case tuning of α , and no need for coarse re-meshing or manual restarts.

- (d) **Mesh coarsening.** Reducing spanwise resolution, either by coarsening the number of streamlines [19] or attenuating spanwise derivatives can desensitize the solver to local stiffness and carry a difficult case to completion. Some authors introduce a quasi-mean-line mode [17] for off-design sweeps. This lever however trades spanwise accuracy for robustness.
- (e) **Stream function method fallback.** When an iteration update stalls in a confined region, some implementations [6,9,21] switch locally to finite differences: derivatives along the streamline are split into upwind/downwind terms so the resulting coefficient matrix is banded and solvable with linear algebra. This regularization, akin to a local stream function method [22], requires a priori knowledge of the local flow type to choose the correct stencil. In mixed hyperbolic/elliptic regions, the characteristic direction is ambiguous and changes across the field, adding the cost of additional discretization choices, which add to the pre-processing time and are not robust across operating points and geometries.
- (f) **Numerical blockage factor.** If highly diffusing flows result in the static pressure to temporarily exceed the total pressure in the energy equation (issue 4), the local velocity can temporarily become negative, hence a numerical blockage factor can be used [8,23] to decrease the effective flow area counteracting this effect. This, however, can mask modeling errors on loss and deviation, biasing the spanwise loading; fine-tuning will impact pre-processing time.
- (g) **Numerical spanwise mixing.** A technique attributed to Denton [24], implemented also in time-marching throughflow [7], it redistributes entropy between adjacent streamlines, and it is conceived to suppress stress loss gradients, especially near end-walls, where the passage vortex vorticity is strongest and so secondary flow losses peak.

In throughflow, the increase in entropy along a streamline “j” is dependent on the loss across a blade row, i.e., the outlet entropy at station “i” is related to that of the inlet at station “i–1”

$$s_{ij} = s_{i-1,j} + \underbrace{\Delta s}_{\text{loss model}} \quad (4)$$

With spanwise mixing, this changes as follows:

$$s_{ij} = (1 - a_M) s_{i-1,j} + a_M \underbrace{\left(\frac{s_{i-1,j-1} + s_{i-1,j+1}}{2} \right)}_{\text{contribution from mixing}} + \Delta s \quad (5)$$

where a_M gauges the relative importance of diffusion versus convection (i.e., it is akin to the reciprocal of a spanwise Péclet number). Because this is framed as a generic scalar-transport device, the same form can be applied to total enthalpy and total pressure-loss scalars. Physically, one may justify the mixing via turbulent diffusion; numerically, it is an artifice. As Hirsch and Denton noted, a choice of $a_M \approx 0.5$ “cures any problems of entropy buildup and has little effect on single-stage calculations” [24]. This also

implies limited validation for multistage turbomachinery, and subsequent implementations have not reported robust evidence in that regime. Moreover, while multistage failures often arise from streamwise error propagation, this technique primarily smooths in the spanwise direction, so its benefit may be limited precisely where cross-row robustness is needed.

3 The Exception Algorithm

3.1 Motivations Behind a New Solver. Crucially, the remedies reported in Sec. 2.3 act on the iterate or the flow fields, not on the decision rule that selects the next state. That is why the same failure signatures persist.

Most importantly, the restart/initialization strategy itself is user-intensive. To navigate difficult regimes, the analyst typically scripts operating-point marches, curates seed states, and intervenes after non-convergence, restarting from neighboring conditions, re-tuning relaxation, or locally modifying the discretization. This hands-on workflow treats the symptoms at the run level, not the cause inside the update rule. Consequently, the familiar failure signatures persist in difficult regions. This motivates a solver-level change with two key requirements:

- **Maintain speed.** Gradient methods set the performance baseline and should remain the default where the residual is smooth and well-conditioned, i.e., without sacrificing the turnaround speed expected of throughflow.
- **Replace manual restarts** with an automated, solver-level decision when the evidence says the gradient step is untrustworthy.

The exception algorithm does exactly this. It (i) detects the signatures linked to ❶–❸ at run time, (ii) temporarily switches from the gradient update to a bounded solver, and (iii) returns control to the fast path as soon as a safe decrease is found. In effect, what used to be a user choice, stop, reseed, and restart from a safer point, becomes a built-in, local, and automatic action of the solver.

3.2 Derivative-Free Optimization. Significant research on derivative-free methods has been conducted in the past decade, as the key drivers to speed up a calculation scheme are being shifted to the number of increasingly complex function evaluations rather than linear algebra or other contributors in the computational method.

The underlying principle is to find the global minimum (or maximum) of an objective function without requiring the derivatives of the function to be evaluated.

The basic aim is to find the $\underline{x}_0$ in a region A such that [25]

$$f(\underline{x}_0) \leq f(\underline{x}), \quad \forall \underline{x} \in A \quad (6)$$

where $f(\underline{x})$ is the *objective function* at the core of the optimization problem, which can be known or unknown, analytical or discrete, and the underlying set of parameters $\underline{x}$ within region A are not required to be continuous or smooth, which is a clear advantage over gradient-based methods.

This type of formulation has applications in hyperparameter tuning, where a group of parameters $\underline{x}$ is optimized such to maximize or minimize a specific key performance indicator. The common trait amongst all derivative-free methods is *grid search*, which is to probe the underlying objective function against a large set of points within a *trust region*. In a grid search, for each iteration, the trust region can be dynamically updated to be centered onto the point where the objective function is minimum. Applying further criteria to how the center or the radius of the *trust region* is updated and choosing a specific technique for search of the optimum is what discriminates amongst different derivative-free methods.

Powell's methodology update interpolation models (as such is not entirely derivative-free) to infer search directions, however, if the model becomes unreliable, the search degenerates to broader sampling: the *trust region* is chosen such to ensure the *well-posedness of the problem* [26], i.e., the interpolation of the objective function yields a unique and stable solution for two or three points of the set for a linear (Powell's COBYLA [27]) or parabolic interpolation (Powell's NEWUOA [28]), respectively. Logically, if this condition cannot be guaranteed, many more function evaluations are required to achieve convergence through grid search, increasing the overall computational cost.

In Powell's methods, the search direction is updated every other iteration, giving rise to a bidirectional search. This reduces the risk of finding a local minimum rather than a global minimum.

3.3 Reformulation of Optimization Methods as Solvers. Derivative-free optimization methods are particularly valuable for problems where the objective function is noisy and discontinuous, which makes throughflow a perfect candidate.

Denoting $y(x_k)$ the residual against the guess x_k at iteration k , the objective is to find

$$\min y(x_k), \quad x_k \in A \quad (7)$$

In other words, to recursively calculates the residual until a local minimum, that meets a pre-defined error tolerance ϵ , is found.

The set A , containing the solution, can be chosen arbitrarily large but it must be ensured that nonphysical solutions (e.g., negative velocity or highly supersonic solutions in the REE) are ruled out to avoid singularities. This is easily implemented by enforcing global maximum and minimum, which are problem-dependent.

This formulation enables the direct application of Powell-type algorithms to solve the REE, continuity, and energy equations as the search direction is based on the minimization of the residuals. Dealing with the residuals yields the property that the *sufficient decrease* [25] condition is automatically met.

Powell's linear pattern search takes the form

$$x_k = x_{k-1} + (y_k - y_{k-1}) \left. \frac{dx}{dy} \right|_{x_{k-2}} \quad (8)$$

Powell's quadratic pattern search is

$$x_k = x_{k-1} + (y_k - y_{k-1}) \left. \frac{dx}{dy} \right|_{x_{k-2}} + \frac{1}{2} (y_k - y_{k-1})^2 \left. \frac{d^2x}{dy^2} \right|_{x_{k-2}} \quad (9)$$

An illustration of how this works in practice can be presented in Fig. 5 for a linear pattern search. Once the solution space has been pre-sampled, the directions of search are alternatively changed through the iterations.

As a quadratic pattern tends to a super-linear convergence rate, it has been preferred by the authors over the linear pattern search.

Fully replacing the baseline gradient-based update with a Powell-type method would have been ill-suited. First, model-based optimization typically needs many more function evaluations to meet tight tolerances; in our cases, Powell's methods give a >10× clock-time with respect to that of Newton–Raphson, secant, or Runge–Kutta's methods due to trust-region sampling. Second, without an informed initialization, Powell-type schemes center the initial trial set at the trust-region center (effectively a bracketed search), which can waste iterations or breach feasibility when the starting state is poorly located.

Instead, we retained the fast gradient path as the default and introduced an exception-driven switch to a bounded local Powell's NEWUOA method only when failure signatures are detected. The NEWUOA step is warm-started from recent iterates, constrained by hard feasibility limits, and managed with adaptive trust-region updates; control returns to the gradient update as soon as a sufficient decrease is secured. This selective interfacing preserves turnaround time in well-behaved regions while providing the stability needed in stiff regimes.

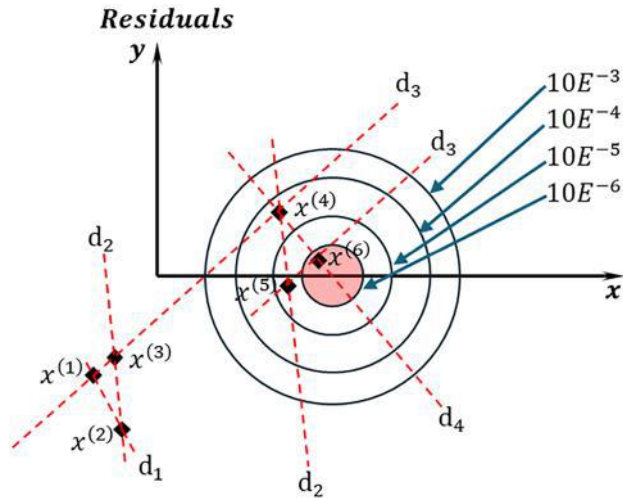


Fig. 5 Illustration of the derivative-free bidirectional method developed against the residuals of a nonlinear equation

3.4 Interfaces of the Exception Algorithm. We describe below how the SOCRATES core solver was modified to employ NEWUOA in place of a gradient step.

A module was written by the authors in FORTRAN, which exposes two iterative solvers that share the same call structure:

- SOCRATES_Solver_Secant_Iteration (f, i, j), baseline
- SOCRATES_Solver_Powell_Iteration (f, i, j), fallback

Powell's derivative-free solvers (including NEWUOA) are available in the open-source PDFO package [29], which we use here as the back-end for the fallback path. The call specifies the type of closure via an integer flag $f \in [1 \dots 6]$ that maps to six shared objective functions: inlet or outlet annulus mass-flow, inlet or outlet meridional velocity on a streamline, and inlet or outlet static temperature on a streamline. Internally, the module holds the indices (i, j, k) identifying the active row, streamline, and global iteration, and it manipulates the shared flow fields (velocities, thermodynamics, areas, etc.) through established SOCRATES boundary-condition subroutines. In practical terms, the interface is "set the guess $x \in \mathbb{R}^1$, rebuild the local physics consistently, measure a scalar mismatch $y(x) \in \mathbb{R}^1$, and drive $y \rightarrow 0$ subject to physical bounds." The default driver is a safeguarded secant step; however, the solver is designed so that any of a small number of signatures of unreliability (*Exceptions*) in the nonlinear update causes a temporary hand-off to NEWUOA.

The objective function passed to both solvers is of the form

$$\min_{x \in \mathcal{S}} y(x) = \Phi(x) - \Phi^* \quad (10)$$

where $\mathcal{S}$ is an admissible set, e.g., $\mathcal{S} = \{V_m \in [5, 400] \text{ m/s}\}$ or $\mathcal{S} = \{t \in [250, 2000] \text{ K}\}$, $\Phi(x)$ is the model-predicted quantity after consistently updating the state with the trial x , and Φ^* is its target.

For continuity ($f = 1, 2$), the unknown is the mean-line meridional speed $x \equiv V_m$ at the inlet or outlet. The residual is calculated by setting $\Phi(x)$ equal to the calculated mass flow from the spanwise contributions, Φ^* is the known mass flow, set as an overall boundary condition, e.g., for a turbo-component's inlet

$$y(x) = \left| \sum_j W_{in}(j) - W_{in} \right| \quad (11)$$

Once $V_m(i, j = j_{\text{meanline}})$ is set, the code recomputes the inlet/outlet triangles and boundary conditions on the mean-line and then populates all spanwise values by solving the REE.

In other words, through $f = 3, 4$ the code evaluates the radial equilibrium representation of the spanwise geometry through

precomputed coefficient sets and computes the streamlines' geometric distance from the mean-line to match a target geometric distance already stored (frozen within this loop). This is unchanged from the existing SOCRATES' solver structure.

Finally, for the *temperature closures* ($f = 5, 6$), the unknown is the static temperature on a line, $x \equiv t_s(i, j)$, with temperature-dependent $c_p(t_s)$, $\gamma(t_s)$, etc., recomputed at each trial. This is also unchanged. In all three classes, the trust region is terminated when either $y \leq \epsilon$ (the corresponding tolerance), when an *Exception* is triggered (in the case of the secant iteration), or when a cap on function evaluations is reached; in the case of NEWUOA "box" violations return a very large y so that NEWUOA stays within.

For NEWUOA calls, as each closure is 1D, we pass a single unknown $x \in \mathbb{R}^1$ to NEWUOA together with the objective subroutine $y(x) = |\Phi(x) - \Phi^*|$. The solver is called with a trust-region radius that shrinks from a coarse setting to a problem-level tolerance. The initial guess x_1 is seeded from the current iteration.

Box-type bounds are enforced by returning a large penalty value when $x \notin \mathcal{S}$, which keeps NEWUOA inside the admissible set.

3.5 Taxonomy and Detection Logic. The secant method remains the fast path and is attempted first with problem-specific initial guesses, *dynamic bounds* derived from the current field extrema (a multiplicative band), and *fixed physical bounds*. Its update $x_{k+1} = x_k - y_k(x_k - x_{k-1})/(y_k - y_{k-1})$ is monitored for a compact set of failure signatures; these constitute the *exception taxonomy* and are the gatekeepers for when the optimizer path is invoked.

There are seven exceptions, and they are as follows:

- Equal starting guesses (①) or NaN starting guesses (②) indicate an ill-posed slope estimate;
- NaN initial residuals indicate that inserting either starting value leads to an invalid physical state (③), which would propagate if not actioned;
- reaching the iteration budget without satisfying $|y| \leq \epsilon$ (④) or encountering equal residuals ($y_k = y_{k-1} > 0$, ⑤) signals a flat or inconsistent local mapping;
- an NaN next iterate (⑥) highlights catastrophic non-smoothness in the secant update;
- violations of *dynamically defined bounds* (⑦) reveal divergence relative to previously converged field magnitudes. Even when the secant residual is decreasing, violating these dynamic bounds flags a runaway step (e.g., from a spurious local slope or non-smooth residual) and triggers the exception path. This extra guard is particularly valuable in multistage configurations, where an outlier in one row can destabilize downstream solutions. Let $\mathcal{S}$ be the set of valid values of the same quantity from the current iteration and let $c_m > 1$ be a multiplier (e.g., 1.5). A candidate update x_k is admissible only if it remains within a relative band around *all* accepted values

$$\forall x \in \mathcal{S}, \min(x) \leq c_m x_k \leq c_m \max(x) \quad (12)$$

- violations of *fixed bounds* (⑧) are hard physics breaches.
- *adjacent-consistency* violations (⑨) detect jumps that are incompatible with the immediately upstream or neighboring streamline solution.

Each event triggers a specific remediation: either a bounded *repair* e.g., a Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) extrapolation along the radius, or an assignment from the adjacent mean-line value in ducts), or a controlled *curtailment* for temperature estimates when acceptable, or, most commonly, a *delegation* to NEWUOA for that closure only.

The numbered circles in Fig. 6 denote which exceptions trigger which branch: the left branch is the global fallback to NEWUOA, and the right branch is an alternative solution, e.g., follow the local repair/curtail rules described above or force the parent iteration

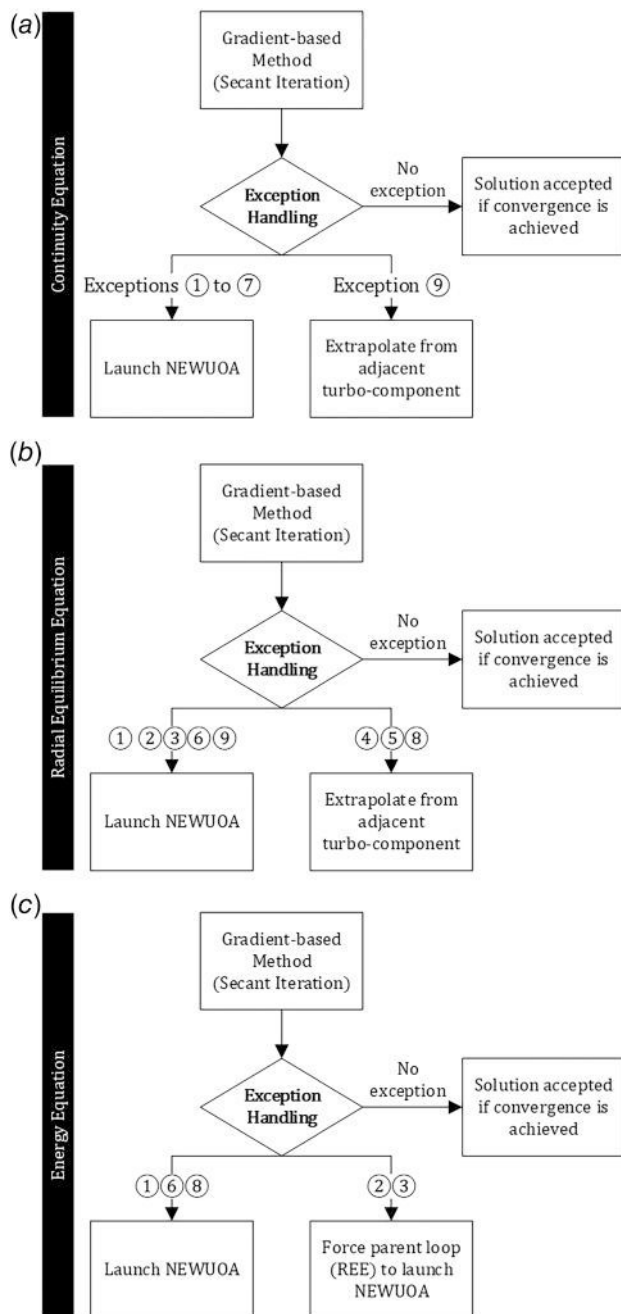


Fig. 6 Exception algorithm detection logic

loop to fall back to NEWUOA. Exception triggering propagates, and the scope of flag propagation (local only versus including parent routines) is itself configurable, e.g., keeping exceptions local prevents a single troublesome streamline from poisoning the global mass-flow loop, while enabling parent resets can be useful to remove persistent NaNs at the annulus level.

In effect, the solver behaves as a single integrated procedure: “try a fast secant step under dynamic and fixed safeguards; if and only if the observed update belongs to a small, interpretable set of failure modes, switch to the optimizer on the very same objective until a safe descent is reestablished, then return control to the fast path.”

The core loop proceeds as: (i) attempt a fast secant step under dynamic, fixed, and adjacent bounds; (ii) if the step exhibits a recognized failure signature, temporarily switch the *same* scalar closure to NEWUOA to reestablish a safe decrease; (iii) once feasibility and descent are recovered, return control to the secant path.

This arrangement keeps the high throughput of the gradient-based route in smooth regions, while the exception-guarded, trust-region fallback supplies robustness precisely where the nonlinear map is stiff, noisy, or non-smooth. The result is a single, integrated procedure whose behavior is predictable and traceable via the compact exception taxonomy.

4 Results and Discussion

This section verifies that the exception algorithm in ascending complexity: (i) reproduces legacy results where the baseline converges, (ii) damps transients and recovers from poor initialization on a controlled single-stage test, and (iii) extends robustness to multistage configurations without manual reseeding. We report accuracy, stability, and time performance behavior, and we distinguish between “relaxed” and “conservative” exception settings when relevant.

4.1 Reproducing Legacy Solver Results (Cases That Always Converge). We first verify that the NEWUOA-guarded solver reproduces the baseline secant method on configurations where the legacy solver converges. Back-to-back runs on the single-stage AGARD-AR-175 Hannover turbine [30] show that the exception algorithm closely overlays the legacy solution across the quasi-non-dimensional characteristics (Fig. 7) and spanwise distributions (Fig. 8). Deltas on the characteristic map remain small, and global isentropic-efficiency trends versus flow are preserved.

For timing (refer to Fig. 9), both the baseline and exception algorithm reach single-point convergence in ~ 12 – 16 s on a standard laptop (Intel i7, 32 GB RAM). A stand-alone Powell/NEWUOA solver (without exception gating) takes >1 h per point, prohibitive for full maps. Selective, exception-driven hand-offs retain throughput while adding robustness only when needed. We then compare two exception algorithm settings. Conservative settings use tight multiplicative bands for the dynamic-bound check and escalate directly to NEWUOA (our preliminary variant). Relaxed settings widen the bands and attempt inexpensive, problem-aware repairs before invoking NEWUOA (as described in Sec. 3.5).

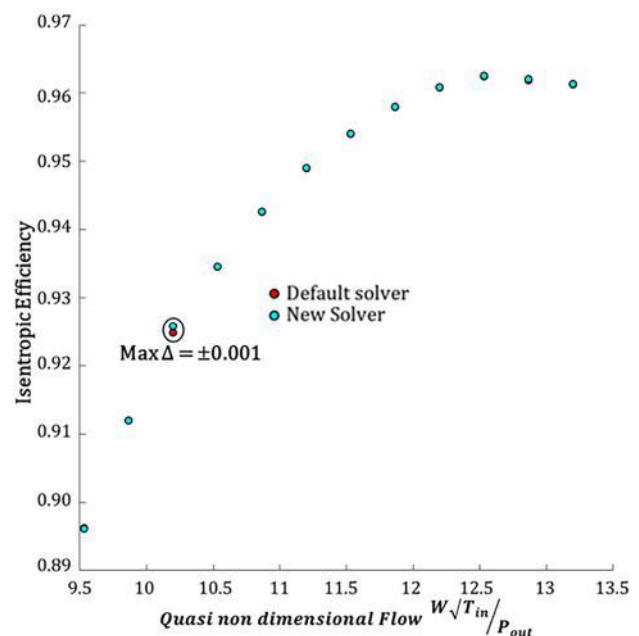


Fig. 7 AGARD-AR-175 Hannover turbine isentropic efficiency versus flow, units in SI

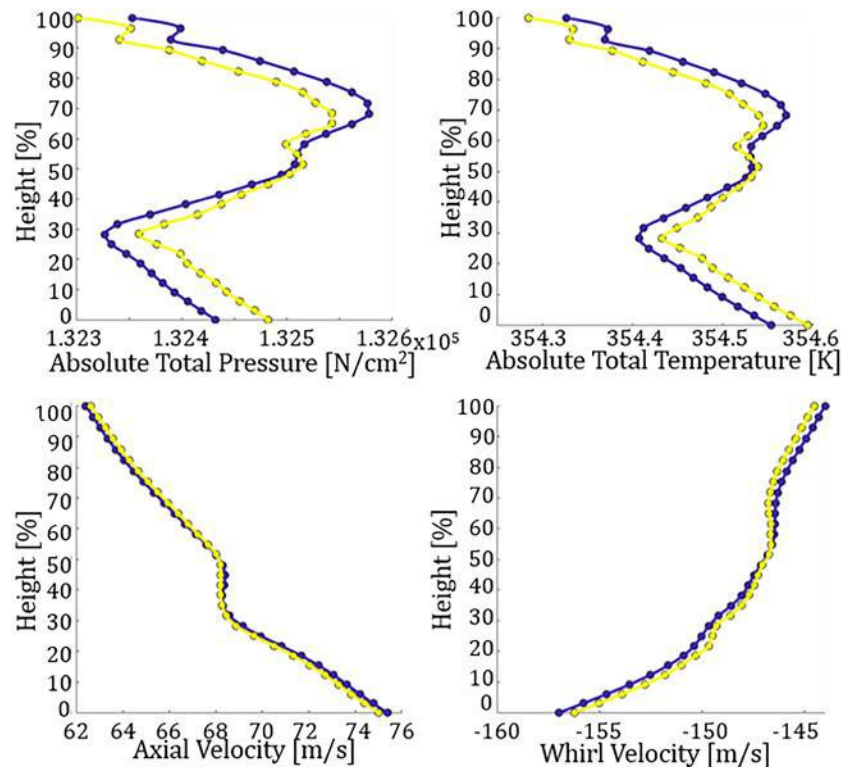


Fig. 8 AGARD-AR-175 Hannover turbine spanwise comparison for the baseline solver (dark color) versus exception algorithm (light color) at the rotor outlet, $N \bar{T} = 196.7$, $W \bar{T} P = 4.2$ (off-design), units in SI.

The exception algorithm with relaxed bounds achieves comparable time to the legacy secant solver and is typically the fastest because it minimizes unnecessary optimizer hand-offs; conservative settings yield a consistently slower simulation, as tighter multiplicative bands trigger more frequent exception routes and optimizer hand-offs; accuracy remains indistinguishable, but overhead increases. These data justify the use of relaxed settings as the default and conservative settings as a *safe mode* when exploring very stiff maps.

As an additional case, for the single-stage AGARD-AR-275 [31] run at off-design, both solvers reproduce experimental trends and improve agreement relative to a published streamline-function solution [32] (Fig. 10). We also include AGARD-AR-275 to anchor SOCRATES's performance as a viscous throughflow tool for turbines. Follow-on work by the authors, which will be presented separately, will address accuracy in depth; these studies are out of scope here, as this article focuses on stability and

establishes the prerequisite for a substantial accuracy campaign on multistage turbines.

Together, these single-stage cases confirm that where the baseline secant loop converges, the exception-driven solver incurs no loss in accuracy.

4.2 Damping Behavior of the Exception Algorithm. We evaluate and damp behavior on a reduced-complexity single

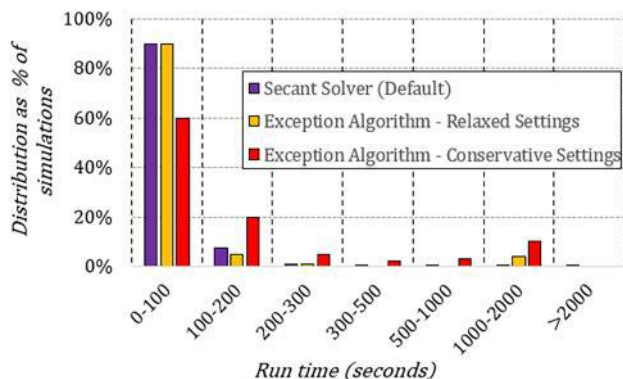


Fig. 9 AGARD-AR-175 Hannover turbine time performance in seconds of the baseline versus the exception algorithm

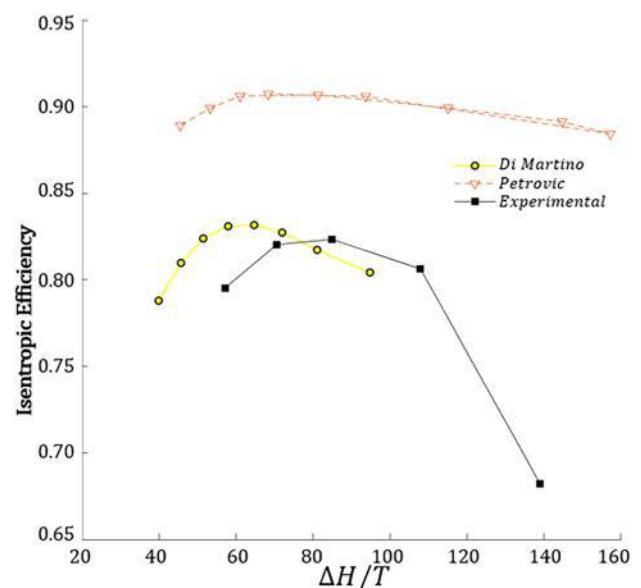


Fig. 10 AGARD-AR-275, absolute performance comparison of SOCRATES (all solvers) versus experimental data/other authors

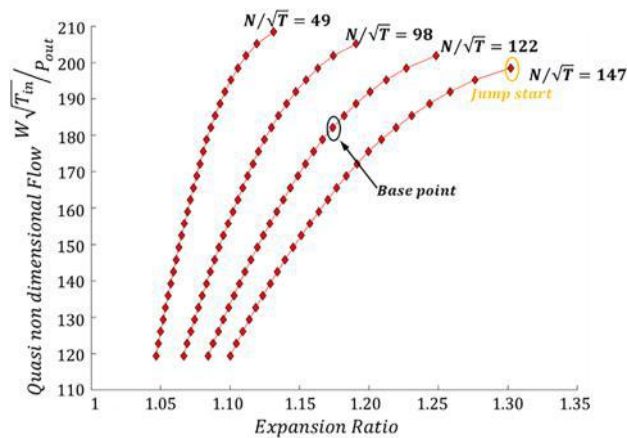


Fig. 11 First-stage characteristic of the EEE LPT $[W \sqrt{T} P] = \text{kg sK}^{1/2} (\text{N cm}^2)$, $[N \sqrt{T}] = \text{K}^{1/2} (\text{min})$

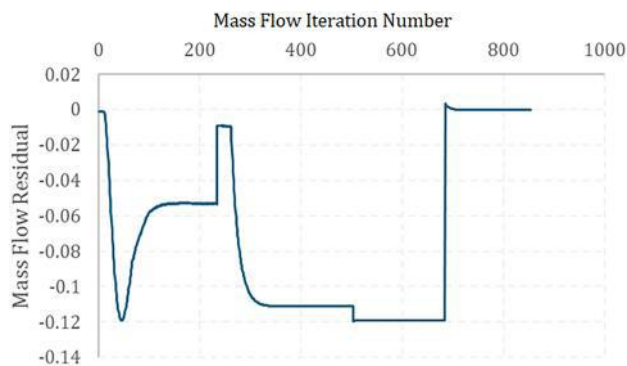


Fig. 12 EEE LPT first stage—residuals history for the secant method (base-point prediction)

stage of an energy-efficient engine low-pressure turbine (EEE LPT) at a well-conditioned “base point” and then at a deliberately ill-initialized “jump-start” point (Fig. 11), to probe the convergence envelope and recovery from poor initialization.

With identical mean-line initialization, both the legacy (Fig. 12) and exception-enabled (Fig. 13) solvers converge quickly; however, residual histories reveal that the exception logic damps early oscillations (Fig. 14) via dynamic bounds (⑦) and adjacent-consistency (⑨), curbing transient overshoots in continuity and energy residuals. In this regime, NEWUOA hand-offs are rare; gains come from guarding the secant step rather than replacing it completely.

We then stress the solver by reusing the base-point fields at a different operating point (no warm-start from a neighboring solution).

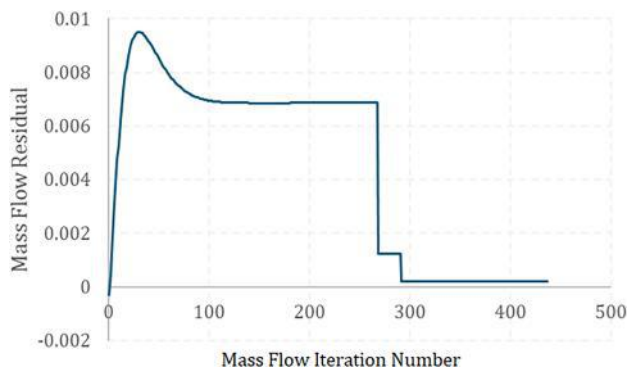


Fig. 13 EEE LPT first stage—residuals history for the exception algorithm (base-point prediction)

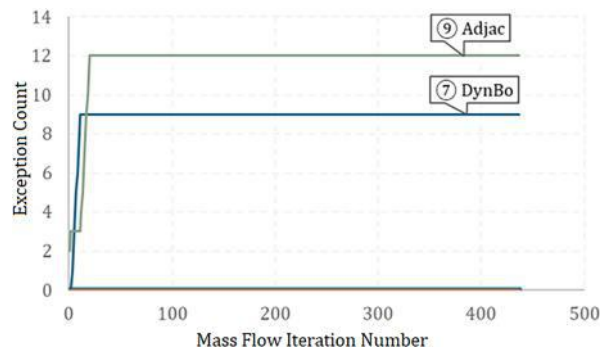


Fig. 14 Overall number of exceptions triggered in a base-point calculation—AGARD-AR-175 single-stage turbine

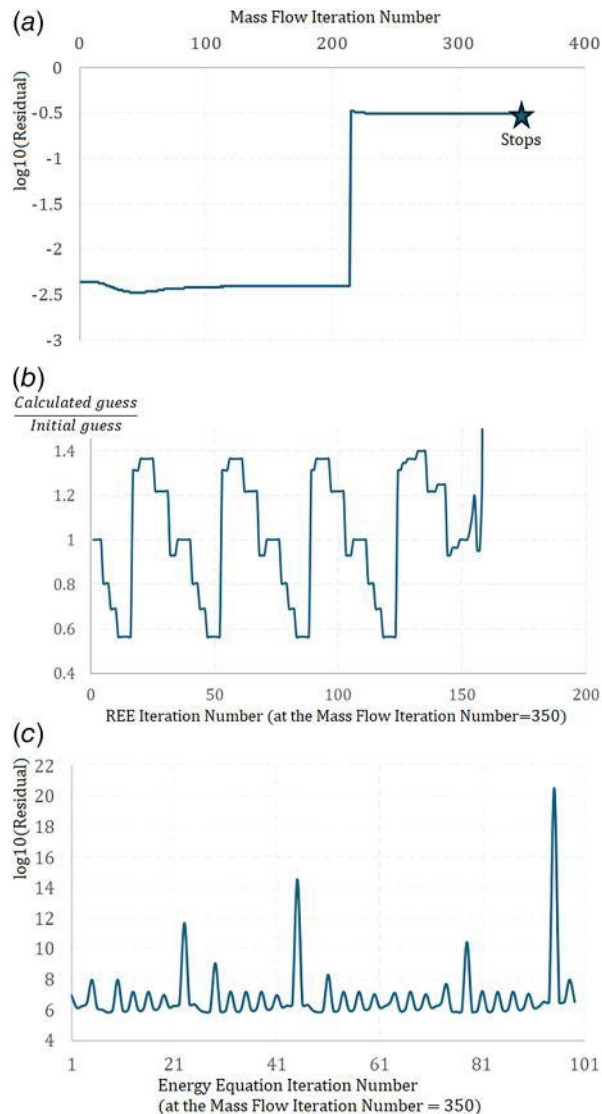


Fig. 15 EEE LPT first-stage “jump start” with the baseline solver. Secant evolutions for (a) continuity and of the (b) inner REE, and (c) energy loops at the mass-flow iteration number 350 (where the simulation stops).

The legacy secant typically exhausts its iteration budget as the REE/energy coupling becomes ill-posed, leading to growing oscillations and eventual overshoot (Fig. 15). The exception algorithm (Fig. 16) intercepts the runaway via a dynamic-bounds (⑦) breach; if the secant slope becomes uninformative (⑥) or yields NaNs

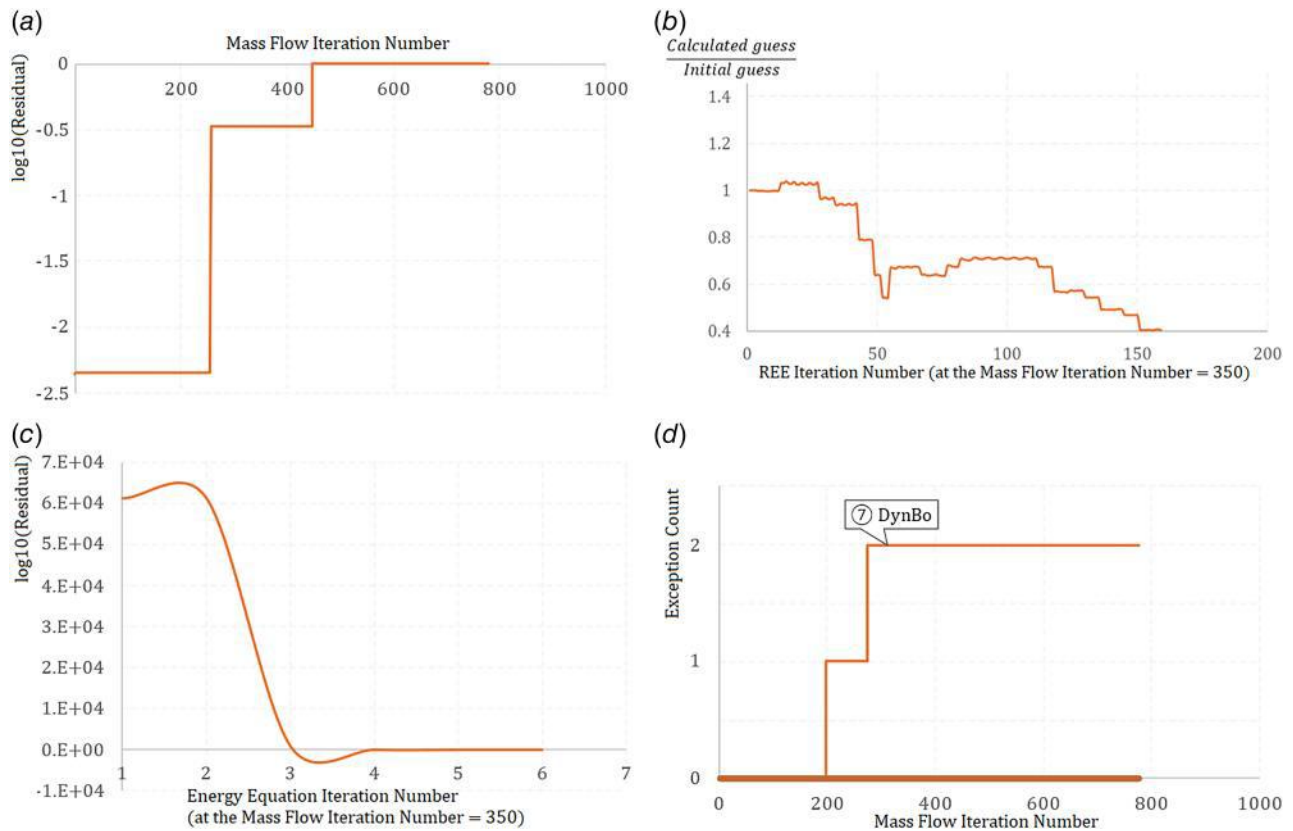


Fig. 16 EEE LPT first-stage “jump start” for the exception algorithm. Evolutions for the (a) continuity equation, and of the (b) inner REE and (c) energy loops at the mass-flow iteration number 350 (where the secant method diverges), and (d) exception count.

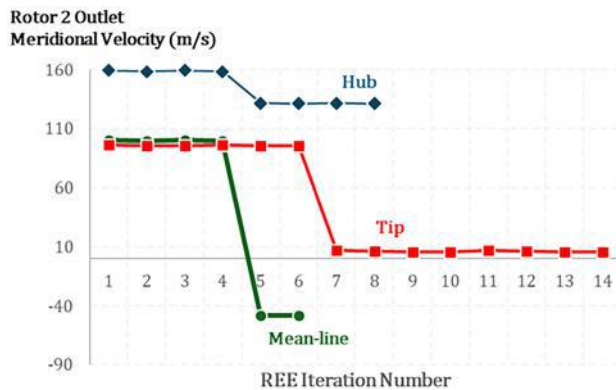


Fig. 17 EEE LPT five-stage turbine. Meridional velocity divergence for the secant method: the absence of bounds causes the guess to diverge to a negative speed and the simulation to halt.

(©), the specific closure is temporarily delegated to NEWUOA under box compliance, after which control returns to the fast path. The run completes at a modest extra cost rather than failing outright. This recovery is valuable to eliminate operating-point marching and manual reseeding.

4.3 Multistage Capability. The EEE LPT serves as validation case to prove the capabilities of the new solver. Interestingly, there is no reported evidence in the literature of the EEE five-stage turbine ever successfully run in throughflow. Attempts have been made to model this in lower-order 1D mean-line [33,34], even in a simplified three-stage configuration, or zero-dimensional simulations [35]. On

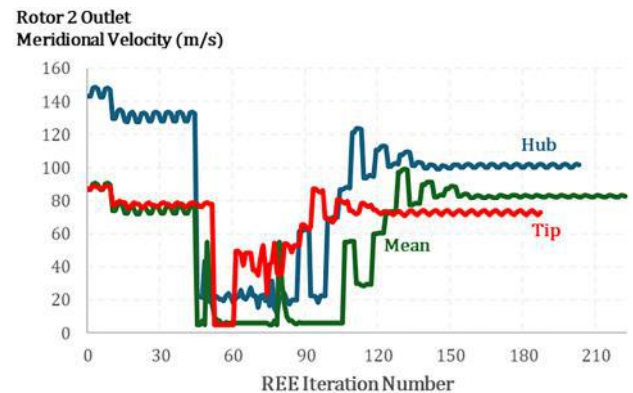


Fig. 18 EEE LPT five-stage turbine. Meridional velocity convergence for the exception algorithm.

the five-stage EEE configuration, local missteps convect downstream and contaminate neighboring stream-surfaces. The baseline secant diverges from Rotor 2 onwards due to the negative speed (Fig. 17).

The exception-enabled solver confines outliers via fixed bounds (©), adjacent-consistency (©), and localized NEWUOA only locally, so most of the calculation proceeds on the fast path. In the specific instance of Rotor 2, it can be visibly shown how the calculation recovers (Fig. 18). We compute full characteristics at design speed without re-initialization or user tuning.

To demonstrate robustness to modeling choices, we repeat with four loss/deviation prescriptions, local calculation of secondary flow-loss with Aungier’s and Traupel’s correlations, plus variants with parabolic secondary-loss distributions. Flow trends are

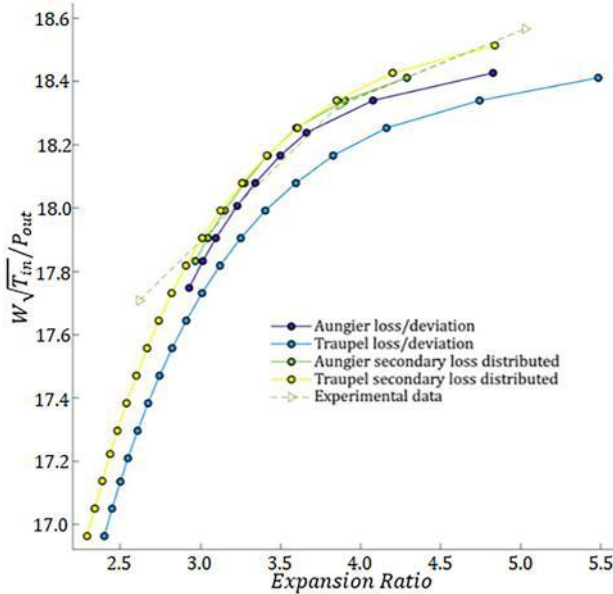


Fig. 19 Five-stage EEE LPT turbine flow characteristic, design speed

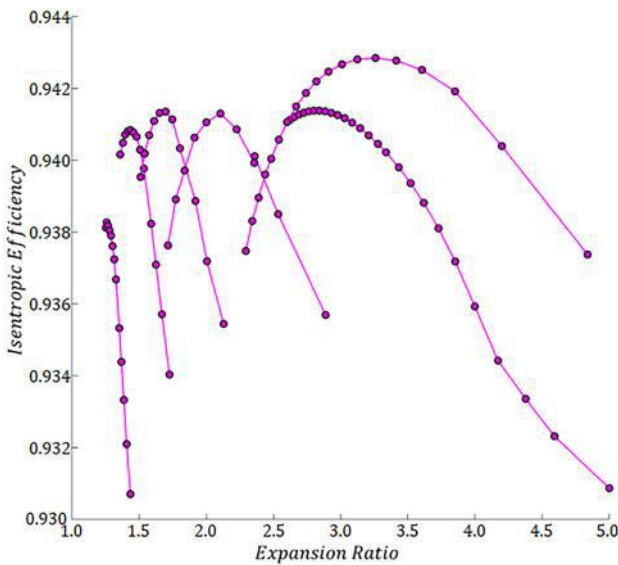


Fig. 20 Exception algorithm—isentropic efficiency versus expansion ratio for the five-stage EEE LPT using Traupel's correlations

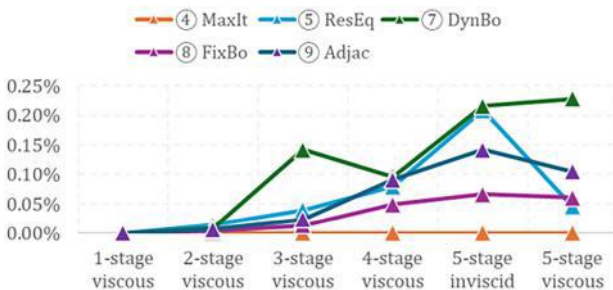


Fig. 21 REE exception count (%) versus total number of secant iteration calls for the EEE LPT

captured and, for the Traupel with parabolic secondary loss, efficiency lies within $\pm 1\%$ of experiment, acceptable within reported scatter (Fig. 19). Without a need to reinitialize the calculation or to fine-tune the initialization conditions, the entire characteristic of the five-stage turbine can be run in less than 30 min starting from very simple duct-flow (blade-less) initialization conditions (Fig. 20).

We complete our analysis with a sensitivity study against stage count to show that the number of exceptions is always a small fraction of the total number of secant iteration calls, confirming that the exception algorithm only intervenes when strictly necessary (Fig. 21).

5 Conclusions

This work introduced an exception-driven update strategy that augments the baseline gradient path in throughflow calculations with a bounded, optimizer fallback only when specific, interpretable failure signatures arise. By monitoring a compact taxonomy of exceptions (equal/NaN residuals, plateauing gradients, dynamic and fixed bound breaches, adjacent-consistency violations), the solver temporarily delegates the offending scalar closure to a trust-region NEWUOA step, then returns control to the fast secant iteration as soon as a safe decrease is restored. In smooth regions, the method behaves identically to the legacy solver; in stiff, non-smooth regimes, it supplies robustness without case-by-case tuning or manual reseeding. Across single- and multistage axial-turbine tests (AGARD-AR-175/275 and the five-stage EEE LPT), the approach reproduced legacy solutions where these exist, damped transients, and completed full characteristics for configurations where the baseline solver does not converge, all without discernible accuracy penalty and with minimal throughput overhead when using relaxed exception settings.

Robustness is achieved by changing *how* the next iterate is chosen, not by further damping, smoothing, or mesh compromises; the fallback is *selective* and *local* (per closure), so the global algorithm remains lightweight; in multistage configurations, dynamic and adjacent bounds prevent the propagation of local missteps across rows, enabling stable characteristics without user-curated stepping.

Although demonstrated within an SLC throughflow framework, the underlying principles are readily transferable to higher-order CFD, where nonlinear sub-solvers exhibit analogous failure modes (residual stalling, NaNs from unphysical states, overshoots at high Courant–Friedrichs–Lewy (CFL), etc.). With appropriate adaptations, adjacency-consistency extends to face-neighbor cells and multigrid transfers, and trust-region repairs can be localized to small spatial blocks of the domain.

Local, bounded fallbacks under strict feasibility, minimal disruption to the fast path, and transparent diagnostics are all principles that align with industrial needs for automation, speed, and predictability across full operating maps.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Nomenclature

Latin Symbols

h = static enthalpy

i = turbo-component number

j = streamline number
 k = iteration number
 m = meridional direction
 p = static pressure
 r = radial direction and coordinate, radius
 s = entropy
 t = static temperature
 H = total enthalpy
 N = shaft speed
 P = total pressure
 T = total (stagnation) temperature
 V = flow velocity
 W = mass flowrate
 a_D = damping factor
 a_M = mixing factor
 $b_{1,2,3}$ = REE coefficients
 c_a = adjacent bounds factor
 c_m = dynamic bounds factor
 $c_{p,v}$ = gas specific heat under constant pressure/volume
 V_n = normal (to main flow) velocity component
 V_x = axial velocity component
 V_w = whirl (tangential) velocity component

Greek Symbols

ε = error tolerance
 ρ = density
 φ = stream-surface thickness
 Φ = objective function

Other Symbols

r = spanwise direction

Abbreviations

CFD = computational fluid dynamics
 ODE = ordinary differential equation
 OEM = original equipment manufacturer
 REE = radial equilibrium equation

References

- [1] Bridgeman, M., Cherry, D., and Petersen, J., 1983, NASA/GE Energy Efficient Engine Low Pressure Turbine Scaled Test Vehicle Performance Report, NASA CR-168290.
- [2] Greitzer, E., Tan, C., and Graf, M., 2004, *Internal Flow*, 1st ed., Cambridge University Press, Cambridge, United Kingdom.
- [3] Pachidis, V., Templalex, I., Pilidis, P., and Kotsiopoulos, P., 2010, "A Dynamic Convergence Control Scheme for the Solution of the Radial," *Proc. Inst. Mech. Eng.*
- [4] Katsanis, T., and McNally, W., 1977, Revised FORTRAN Program for Calculating Velocities and Streamlines on the Hub-Shroud Mid-Channel Stream Surface of an Axial-, Radial-, or Mixed-Flow Turbomachine or Annular Duct, NASA TN D-8430.
- [5] Tang, Q., Wu, H., and Lou, H., 2023, "Time-Marching Throughflow Model for Predicting Axial Turbine Performances," *Braz. Soc. Mech. Sci. Eng.*, 4(5).
- [6] Bashi, H., Esfahanian, V., Tavakoli, A., Ansari, M., Izadi, M., and Kordi, M., 2024, "Improved Streamline Curvature Method for Prediction of Gas Turbines Performance," *Arab. J. Sci. Eng.*, 2.
- [7] Simon, J., 2007, "Contribution to Throughflow Modelling for Axial Flow Turbomachines," Ph.D. thesis, Université de Liège, Liège, Belgium.
- [8] Barbosa, J., 1987, "A Streamline Curvature Computational Programme for Axial Compressor Performance Prediction," Ph.D. thesis, Cranfield University, Cranfield, UK.
- [9] Gong, W., Wu, R., and Zhang, B., 2016, "A New Finite Difference Method to Solve the Velocity Gradient Equation in Streamline Curvature Method," *Adv. Mech. Eng.*, 8(9).
- [10] Barsi, D., Canepa, R., and Satta, A., 2011, "A Semi-Empirical Procedure for the Evaluation of Multi-Stage Turbine Performances," *RE&PQJ*, 1(9).
- [11] Bindon, J., 1973, "Stability and Convergence of Streamline Curvature Flow Analysis Procedures," *Int. J. Numer. Methods Eng.*, 7(1), pp. 69–83.
- [12] Templalex, I., Pilidis, P., Pachidis, V., and Kotsiopoulos, P., 2011, "Development of a Two-Dimensional Streamline Curvature Code," *ASME J. Turbomach.*, 133(1), p. 011003.
- [13] Frost, D., 1972, A Streamline Curvature Through-Flow Computer Program for Analysing the Flow Through Axial-Flow Turbomachines, National Gas Turbine Establishment, Reports and Memoranda No. 3687.
- [14] Barger, R., 1973, Streamline Curvature Design Procedure for Subsonic and Transonic Ducts, N. T. D-7368, Langley, BC.
- [15] Letcher, T., 2008, A Matlab Implementation of the Streamline Curvature Method for Axial Turbomachinery, S. D. University, Vermillion, SD.
- [16] Jennions, I., and Stow, P., 1985, "The Quasi-Three-Dimensional Turbomachinery Blade Design System, Part I: Throughflow Analysis, Part II: Computerized System," *ASME J. Eng. Gas Turbines Power*, 107(2), pp. 308–314.
- [17] Casey, M., and Robinson, C., 2008, A New Streamline Curvature Throughflow Method for Radial Turbomachinery, A. GT2008-50187, Berlin, Germany.
- [18] Denton, J. D., 1978, "Through-Flow Calculations for Transonic Axial Flow Turbines," *J. Eng. Power*, 100(2), pp. 212–218.
- [19] Bohn, D., and Kim, T., 1998, A Comparative Throughflow Analysis of Axial Flow Turbines, P. A. J. o. P. a. E. V. 2. I. 2. Proceedings of IMechE, Aachen, Germany.
- [20] Wilkinson, D., 1969, "Stability, Convergence, and Accuracy of Two-Dimensional Streamline Curvature Methods Using Quasi-Orthogonals," *Inst. Mech. Eng. Proc.*, 1(7).
- [21] Tiwari, P., Stein, A., and Lin, Y.-L., 2013, "Dual-Solution and Choked Flow Treatment in a Streamline Curvature Throughflow Solver," *ASME J. Turbomach.*, 135(4), p. 041004.
- [22] Marsh, H., 1968, A Digital Computer Program for the Through-Flow Fluid Mechanics in an Arbitrary Turbomachine Using a Matrix Method, A. R. C. R. N. 3509, London, United Kingdom.
- [23] Pachidis, V., 2006, "Gas Turbine Advanced Performance Simulation," Ph.D. thesis, Cranfield University, School of Engineering, Cranfield, UK.
- [24] Hirsch, C., and Denton, J., 1981, Through Flow Calculations in Axial Turbomachines, Neuilly Sur Seine, France: AGARD Advisory Report No. 175.
- [25] Conn, A., Scheinberg, K., and Vicente, L., 2009, *Introduction to Derivative-Free Optimization*, SIAM, New York.
- [26] Ragonneau, T., and Zhang, Z., 2024, An Optimal Interpolation Set for Model-Based Derivative-Free Optimization Methods, Cornell University, Ithaca, NY, .
- [27] Powell, M., 1994, "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," *Advances in Optimization and Numerical Analysis*, S. Gomez, and J.-P. Hennart, eds., Kluwer Academic, Dordrecht, The Netherlands.
- [28] Powell, M., 2008, "Developments of NEWUOA for Minimization Without Derivatives," *IMA J. Numer. Anal.*, 2(4).
- [29] Ragonneau, T., and Zhang, Z., 2024, PDFO: A Cross-Platform Package for Powell's Derivative-Free Optimization Solvers, D. a. o. a., Hong Kong <https://arxiv.org/pdf/2302.13246.pdf>
- [30] Groschup, G., 1977, "Strömungstechnische Untersuchung Einer Turbinenstufe im Vergleich zum Verhalten der Ebenen Gitter Ihrer Beschauung," Dissertation, Hannover, Germany.
- [31] Förster, W., and Kruse, H., 1990, *Test Case E/TU-3, Single Stage Subsonic Turbine*, AGARD-AR-275, Aachen, Germany.
- [32] Petrovic, M., 2000, "Maximizing Multistage Turbine Efficiency by Optimizing Hub and Shroud Shapes and Inlet and Exit Conditions of Each Blade Row," *Int. J. Turbo. Jet Engines*, 17(4), pp. 267–278.
- [33] Turner, M., Merchant, A., and Bruna, D., 2006, "A Turbomachinery Design Tool for Teaching Design Concepts for Axial-Flow Fans, Compressors, and Turbines," *ASME-GT2006-90105*, Cincinnati, OH.
- [34] Turner, M., Merchant, A., and Bruna, D., 2007, Applications of a Turbomachinery Design Tool for Compressors and Turbines, A. 2007-5152, Cincinnati, OH.
- [35] Claus, W., Beach, T., Turner, M., Siddappaji, K., and Hendricks, E., 2015, Geometry and Simulation Results for a Gas Turbine Representative of the Energy Efficient Engine (EEE), NASA-TM-2015-218408, NASA Glenn, Cleveland, OH.



DSPACE

<https://dspace.org/>

An exception algorithm to ease convergence in multistage turbomachinery throughflow calculations

Di Martino, Mario; Pachidis, Vassilios

2026-06

American Society of Mechanical Engineers (ASME)

<https://doi.org/10.1115/1.4070540>

Di Martino M, Pachidis V. (2026) An exception algorithm to ease convergence in multistage turbomachinery throughflow calculations. Journal of Turbomachinery, Volume 148, Issue 6, June 2026, Article number 061016