ChatGPT

ConStellaration Fusion Challenge: Benchmarks and Solution Strategies

Introduction

The **ConStellaration Fusion Challenge** is a collaborative effort by Proxima Fusion and Hugging Face to apply advanced optimization and machine learning methods to stellarator fusion reactor design [1] [2]. Participants are tasked with solving **three benchmark stellarator design problems** of increasing complexity, each reflecting critical requirements for a future fusion power plant [3] [4]. These benchmarks are: (1) a **Geometric Optimized Stellarator** (minimize plasma shape elongation under fixed geometry constraints), (2) a **"Simple-to-Build" Quasi-Isodynamic (QI) Stellarator** (achieve good plasma confinement via QI symmetry while using smoother, more manufacturable magnetic fields), and (3) a **Multi-Objective MHD-Stable QI Stellarator** (optimize a trade-off between device compactness and coil simplicity while ensuring plasma confinement and stability) [5]. Each problem includes reference implementations, physics-based evaluation scripts, and baseline solutions using classical optimization, which participants are challenged to outperform [6] [7].

Stellarator design is inherently a **high-dimensional, constrained optimization** problem requiring balance of physics performance and engineering feasibility [8] [9]. Traditional approaches involve iteratively adjusting a 3D plasma boundary surface (parameterized by Fourier coefficients) and computing the magnetohydrodynamic (MHD) equilibrium to evaluate metrics like magnetic field quality, stability, and coil complexity [10] [11]. This is computationally intensive and historically required expert-driven tuning. The ConStellaration challenge provides an **open dataset of ~160k QI-like stellarator configurations** [12] and standardized metrics so that data-driven techniques (e.g. surrogate modeling, generative design) can be applied in tandem with physics-based optimization [13] [14]. By leveraging this dataset and modern tools, participants can explore novel stellarator shapes more efficiently and potentially "learn" the design space rather than rely solely on brute-force physics simulations.

In this report, we analyze the **requirements and success criteria** for each benchmark, identify key physical and engineering **constraints and objectives**, and propose solution strategies. We review optimization approaches – from classical **gradient-based** and **gradient-free** algorithms with constraint handling, to advanced **augmented Lagrangian methods** and **generative models (diffusion or flow-matching)** – and discuss the viability of tools like **SIMSOPT**, **VMEC++**, **DESC**, and modern autograd frameworks (**PyTorch**, **JAX**) for implementation. For each benchmark, we outline a detailed plan for producing **high-scoring solutions** that meet all constraints, optimize the target metrics, and exploit the dataset's structure. We also compare the strengths of physics-based vs. ML-driven approaches and provide recommendations for a robust, hybrid design strategy.

Stellarator Design Metrics and Constraints

Successful stellarator optimization requires meeting numerous **physics and engineering metrics** simultaneously [9]. The challenge benchmarks evaluate designs on a common set of metrics (summarized in **Table 1**), which serve as objectives or constraints in each problem. Below we define the key metrics and their significance:

- **Aspect Ratio (A)** – The ratio of the plasma's major radius to its minor radius. Lower aspect ratio means a more compact device (desirable for cost and size) but often makes magnetic shaping more difficult [15]. High aspect ratio tends to ease achieving quasi-symmetry but leads to a larger machine.
- **Elongation (E)** – The maximum elongation of the plasma cross-section, i.e. how "stretched" the plasma is vertically vs. horizontally. Excessive elongation can indicate complex shape features. Minimizing elongation tends to simplify the plasma shape.
- **Triangularity (δ)** – A shape parameter describing how triangular (vs. circular) the plasma cross-section is. It's defined by the indent of the plasma shape. Triangularity is fixed in the Geometric benchmark to maintain a specific shape class [16].
- **Rotational Transform (ι)** – The field line twist per toroidal period, often given as ι (equal to 1/q in tokamak notation). In stellarators, ι is set by 3D shaping. Specifying the **edge rotational transform per field period** (ι<sub>edge/Nfp</sub>) controls the magnetic field's winding and can help avoid low-order rational surfaces. It is constrained in Benchmark 1 [16].
- **Magnetic Mirror Ratio (M)** – The ratio of maximum to minimum magnetic field strength |B| on a flux surface. A large mirror ratio means deep magnetic wells (big |B| variations) in which particles can reflect; QI configurations often naturally have some mirror structure. However, too large M can make QI easier to achieve by allowing strong wells, so Benchmark 2 limits M to avoid trivial solutions [17].
- **Quasi-Isodynamic Residual (Δ<sub>QI</sub>)** – A quantitative measure of how closely the field approaches perfect quasi-isodynamic symmetry. In a perfectly QI field, contours of |B| are poloidally closed and the net bounce-averaged drift of trapped particles is zero [18]. The residual (or "QI deviation") metric, defined per Goodman *et al.* (2023), is essentially zero for a truly QI field and grows as the configuration deviates from QI [19]. Minimizing this residual improves fast particle confinement.
- **Vacuum Magnetic Well (W)** – A criterion for ideal MHD stability. A "magnetic well" exists if the magnetic field pressure increases outward (plasma volume decreases with increasing flux), providing a restoring force against interchange instabilities [20]. We use the vacuum magnetic well depth (or a related proxy) to ensure **ideal-MHD stability**: W > 0 means the configuration is Mercier stable at zero plasma beta [20].
- **Bad-Curvature Flux Compression (C)** – A geometric proxy for turbulent transport, correlating with ion-temperature-gradient (ITG) driven turbulence [21]. It measures how much flux surfaces compress in regions of unfavorable curvature (where field line curvature and pressure gradients drive turbulence) [22]. Lower values of C indicate less drive for turbulence. In Benchmark 3, C must be limited (below a threshold) to mitigate expected turbulent transport [23].
- **Normalized Magnetic Field Gradient Scale Length (L<sub>grad</sub>)** – This metric quantifies **coil complexity / simplicity**. It is the minimum scale length over the plasma surface for variation in the normal component of B (normalized by the major radius) [24] [19]. Intuitively, a large L<sub>grad</sub> means the field doesn't vary too sharply on the boundary, allowing coils to be placed farther away or with gentler curvature [25]. Higher L<sub>grad</sub> corresponds to *simpler, more feasible coil shapes*, whereas a small L<sub>grad</sub> indicates sharp field variations that require tightly wound, high-curvature coils. This "coil simplicity" metric (from Kappel *et al.* 2024) is used as an objective in Benchmarks 2 and 3 [24] [19].

Table 1 below summarizes how these metrics appear in each benchmark either as objectives to optimize or constraints to satisfy. Each benchmark has a specific formulation (single-objective for Problems 1 and 2, biobjective for Problem 3) [26] [27]. The ultimate goal is to **maximize a score** that rewards achieving the objectives while meeting all constraints. For single-objective tasks, the score is a normalized scalar function of the objective value minus any penalties for constraint violations [28]. For the multi-objective task, solutions are evaluated by the **hypervolume** of the dominated portion of objective space, using a reference point, which effectively measures the quality of the Pareto front discovered [29].

Metric / Feature	**Role in Benchmark 1** (Geometric)	**Role in Benchmark 2** (Simple QI)	**Role in Benchmark 3** (MHD-Stable QI)
Aspect Ratio (A)	Fixed (constraint: A = A<sub>target</sub>) [16]	Constrained (A ≤ A<sub>max</sub> to avoid huge device) [30]	**Objective** (minimize A for compactness) [15]
Triangularity (δ)	Fixed (constraint: δ = δ<sub>target</sub>) [16]	Not explicitly constrained (free)	Not explicitly constrained (free)
Rotational Transform (ι<sub>edge</sub>/Nfp)	Fixed (constraint: ι = ι<sub>target</sub>) [16]	Not constrained (free; determined by QI optimization)	Not explicitly constrained (free; QI shaping yields ι)
Max Elongation (E)	**Objective** (minimize E) [16]	Constrained (E ≤ E<sub>max</sub> to prevent extreme shaping) [30]	Not explicitly constrained (implicitly limited by stability)
Mag. Mirror Ratio (M)	Not considered	Constrained (M ≤ M<sub>max</sub> to prevent trivial QI) [30]	Not explicitly constrained (implicitly limited by stability/QI)
Coil Simplicity (L<sub>grad</sub>)	Not considered (coils not in scope)	**Objective component** (maximize L<sub>grad</sub>; i.e. favor simpler coils) [19]	**Objective** (maximize L<sub>grad</sub>; i.e. minimize coil complexity) [15]
QI Residual (Δ<sub>QI</sub>)	Not considered (QI not targeted)	**Objective component** (minimize Δ<sub>QI</sub>; achieve quasi-isodynamic field) [19]	**Constraint** (Δ<sub>QI</sub> ≤ ε to ensure QI confinement; QI stellarator design) [31] [32]
Vacuum Magnetic Well (W)	Not applicable (vacuum stability not targeted)	Not considered (no pressure/stability constraint)	**Constraint** (W ≥ 0; require magnetic well for MHD stability) [20]
Bad-Curvature Compression (C)	Not applicable	Not considered	**Constraint** (C ≤ C<sub>max</sub>; limit turbulent transport drive) [21]

Table 1: Key metrics in each benchmark problem, indicating whether they appear as objectives (to minimize or maximize) or as constraints (with required target or bounds). Problem 1 fixes basic geometry parameters and optimizes a shape metric; Problem 2 optimizes a combined coil-friendly and confinement objective under bounding constraints; Problem 3 treats two objectives (compactness vs. coil simplicity) with multiple physics constraints. In all cases, **feasible solutions** must satisfy the constraints exactly or within tolerance – any violation incurs a heavy score penalty [28] or exclusion from the Pareto set.

Throughout the challenge, plasma boundaries are represented by a **truncated Fourier series** in cylindrical coordinates (R, Z as functions of poloidal angle θ and toroidal angle φ) with stellarator symmetry enforced [33] [34]. This parametric representation allows a rich space of non-axisymmetric shapes (see **Figure 1** for examples) and serves as the decision variables for optimization. The forward model to evaluate any given boundary is the **VMEC++** ideal MHD equilibrium solver [35], which computes the magnetic field configuration and returns the metrics above. Participants submit the Fourier coefficients of candidate boundaries, and an automated pipeline runs VMEC++ to compute metrics and scores the design against the benchmark criteria [36].

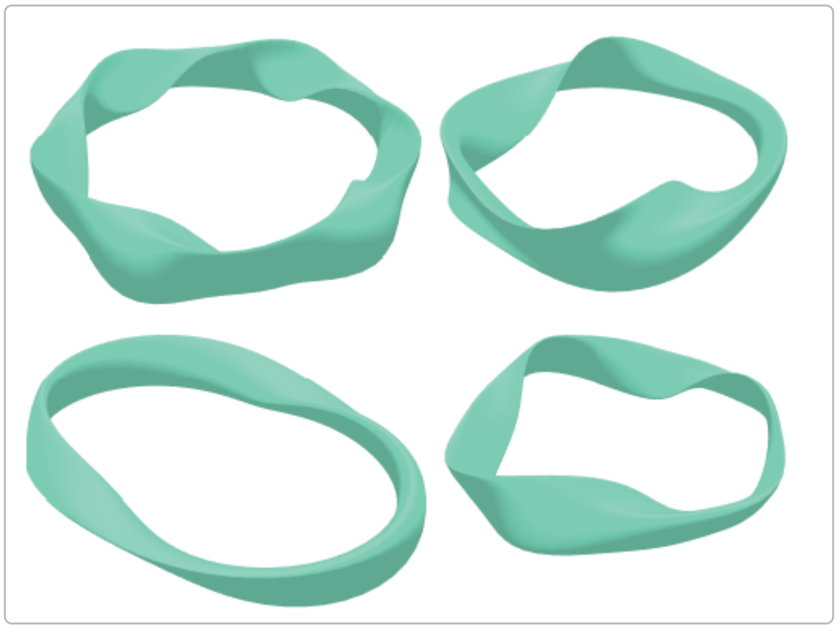

Figure 1: Examples of diverse stellarator plasma boundary shapes (surfaces) from the ConStellaration dataset [37]. These 3D boundaries illustrate the non-axisymmetric, twisted geometries typical of stellarators, which must be optimized for both physics performance (confinement, stability) and engineering feasibility (coil tolerances).

With the context of metrics and representation established, we now examine each benchmark in detail, identifying the specific goals, constraints, and solution strategies.

Benchmark 1: Geometric Optimized Stellarator

Problem Definition: *Design a stellarator plasma shape that minimizes its elongation, subject to fixed aspect ratio, fixed triangularity, and fixed rotational transform (per period). In formal terms,* Problem 1* is a single-objective constrained optimization:

- **Objective:** minimize the **maximum elongation (E)** of the plasma cross-section [16]. Elongation is evaluated at the outboard midplane cross-sections; the goal is to make the plasma cross-section as circular as possible (low E) rather than strongly stretched. This tends to simplify engineering and improves uniformity of confinement.

- **Constraints:** The design must achieve specific target values for: **Aspect Ratio (A)**, **Average Triangularity (δ)**, and **Edge Rotational Transform (ι/Nfp)** [16]. These are treated as essentially equality constraints (or very tight bounds around given targets). For example, one might require *A = 6*, *δ = 0.2*, and *ι/Nfp = 0.45* (hypothetical values), meaning the final plasma shape should have those properties. The triangularity δ is defined as the average of the shape's triangularity in symmetric cross-sections, and ι/Nfp is the rotational transform per field period (so if Nfp=3 field periods, total ι_edge ≈ 1.35 in this example) [38].

- **Key Metrics:** A, δ, and ι are *inputs* (specified targets) for this task, while E is the output to minimize. There are no explicit physics performance metrics like QI or well in this problem – it is purely geometrical. By fixing A and δ, we ensure the size and basic shape character are held constant. By fixing ι, we keep the magnetic field twist constant (this is akin to imposing a certain amount of rotational transform which might influence stability and confinement, but here it's just a constraint to make the problem well-defined). The **success criterion** is *how low an elongation E* one can achieve **without deviating from those fixed constraints**. Designs are ranked by E (lower is better), with any violation of A, δ, or ι targets resulting in a penalty or disqualification [39]. In practice, a small tolerance is allowed (the evaluation script likely treats a submission as feasible if each constrained metric is within a relative tolerance of the target) [40].

- **Evaluation:** This benchmark's score is a monotonic mapping of elongation, normalized such that the best possible E gets a high score (close to 1) and worse elongations score lower [41]. If a submission does not meet the A, δ, ι constraints, the "normalized constraint violation" term heavily reduces the score [40]. Thus, participants must *prioritize constraint satisfaction first*, then minimize E.

Requirements and Constraints: The rationale for this problem is to **"onboard" participants** with a simpler, purely geometric optimization [42]. It tests one's ability to handle basic shape constraints and manipulate the Fourier parameters to achieve a desired shape outcome. The constraints ensure apples-to-apples comparison: everyone is optimizing elongation for the *same class of configurations* (same aspect ratio, etc.) so that improvements come from genuine shape optimization rather than trivial changes in size or twist. One challenge is that the **rotational transform ι** is a physics-determined quantity (coming from the MHD equilibrium); it's not a direct geometric parameter. Achieving a target ι requires iterative adjustment of the Fourier coefficients (since ι depends on global plasma curvature and Shafranov shift). So the optimizer must implicitly solve for shape that yields the desired ι while minimizing elongation. This coupling makes the problem non-trivial even though no explicit physics objective is present.

Optimization Strategy: To solve Problem 1, we need an approach for **constrained shape optimization** in ~10–20 dimensional space (depending on how many Fourier modes are allowed). A straightforward strategy is to use a **gradient-based constrained optimizer** with the help of finite-difference or analytic derivatives for the metrics. For example, one could employ a Sequential Quadratic Programming (SQP) or interior-point algorithm that takes steps to reduce elongation while applying corrections to maintain A, δ, ι. Indeed, the baseline tried a SciPy trust-region interior point method (`trust-constr`) with finite-difference gradients [43]. However, the results showed that approach struggled: the SciPy optimizer did not find a feasible optimum in the allotted time [44]. Likely reasons include the **noisy, non-convex landscape** introduced by the VMEC solver (small changes in shape can cause non-smooth changes in ι or VMEC non-convergence) and the difficulty of maintaining the ι constraint via finite differences.

Given this, a more robust approach is to use an **Augmented Lagrangian Method (ALM)** or similar constraint-handling technique. In the provided baseline, an Augmented Lagrangian with a gradient-free solver (Nevergrad's NGOpt) was the only method that managed to satisfy all constraints for Problem 1 [45] [46]. The ALM approach works by transforming the constrained problem into a sequence of unconstrained sub-problems with penalty terms for constraint violations, updating Lagrange multipliers iteratively [47]. This method is well-suited when gradients are unreliable, as it allows a flexible solver (even random or evolutionary search) to explore and gradually focus in on the feasible region. We recommend using **ALM with a derivative-free optimizer** for this task, as it proved effective: for example, NGOpt (which auto-selects among CMA-ES, PSO, etc.) found a feasible minimum elongation shape where others failed [44]. Starting from an initial guess (e.g. a known stellarator configuration with the desired A, δ, ι – perhaps a "rotating ellipse" analytical shape [48]), the ALM will penalize any drift in A, δ, ι and drive the shape to reduce elongation E.

On the other hand, because this problem is relatively low-dimensional and doesn't require evaluating exotic physics metrics, **gradient-based methods might still be viable** if gradients can be obtained accurately. A promising avenue is using the **DESC code** (Differentiable Equilibrium Solver). DESC can solve free-boundary equilibria and compute sensitivities of outputs to shape parameters via automatic differentiation [49]. By formulating the ι constraint and elongation objective within DESC, one could directly obtain $\partial E/\partial$(coefficients) and $\partial \iota/\partial$(coefficients) and use a constrained optimizer or even a custom Jacobian-based root-finding for ι while minimizing E. This physics-informed adjoint approach could drastically reduce iterations. However, implementing such a **physics-differentiable optimization** requires expertise and ensuring DESC's equilibrium matches VMEC's outputs. Participants without deep fusion background might opt for data-driven surrogates instead.

Data-Driven Assistance: The open dataset of ~160k stellarators [14] can be leveraged here as well. We can filter or interpolate within the dataset for configurations at the target A, δ, ι and examine their elongations. The dataset likely contains many QI-like shapes, which may not exactly fix ι or δ, but some subset will incidentally have similar values. Training a **regression model (surrogate)** for elongation and other metrics as functions of the Fourier coefficients is possible. For instance, a small neural network or Gaussian Process could predict E, A, δ, ι for any given shape vector. One could then run an off-the-shelf constrained optimizer on this *surrogate* (which is cheap to evaluate and differentiable via PyTorch/JAX) to find a candidate optimum. This candidate's coefficients can be fine-tuned using the actual VMEC evaluation. The risk is the surrogate must be accurate near the optimum and respect the constraints – if not, it might mislead. Still, given the simplicity of objectives here, a surrogate-assisted approach could speed up the search dramatically by avoiding thousands of VMEC calls.

Another ML-based idea is to use a **generative model** to sample shapes that inherently satisfy the constraints. For example, one could train a conditional generative model (like a diffusion model or normalizing flow) on the dataset, conditioning on aspect ratio ≈ A<sub>target</sub> and rotational transform ≈ ι<sub>target</sub>. This model would learn the distribution of plausible shapes at those parameters. By sampling many candidates and evaluating elongation, one might quickly find some low-E configurations to then locally optimize. This leverages the *prior knowledge* embedded in the dataset – the generative model implicitly learned how to make realistic stellarator shapes with given global properties. A diffusion model, in particular, can gradually refine random samples into realistic shapes; by incorporating a score or classifier that favors lower elongation during the sampling (so-called "guided diffusion"), one could bias the generation toward solutions with small E. While this is an advanced approach, it exemplifies how **physics-based constraints can be "baked into" the design generation** via ML, reducing the burden on the optimizer.

Implementation Details: For Problem 1, the implementation can be relatively straightforward. We recommend using the provided **GitHub codebase** for evaluation (which wraps VMEC++ and computes A, E, etc. for a given set of Fourier coefficients) [50] [36]. One could implement an ALM loop: in Python, for instance, use Nevergrad's optimization algorithms or SciPy's COBYLA, evaluating a custom objective = E + Σλ_i * (constraint_i residual)^2. Because each VMEC evaluation might take a few seconds and gradients are not analytical, one should use batch or parallel evaluations if possible. For example, one can run multiple candidate shapes on different CPU cores simultaneously (the problem is embarrassingly parallel in evaluation). Given that Proxima's baseline used 96 vCPU for hours [51], it's clear that **computation must be managed** – e.g., limit the number of Fourier modes or start with coarse VMEC resolution to scout the landscape, then refine at higher fidelity once near optimum [52].

If using a surrogate model, one must generate training data. This could be done by randomly sampling shape coefficients near the initial guess or from the dataset, then running VMEC to get metrics. Since the dataset already contains many VMEC solutions, one might directly use those entries for training; however, they might not exactly satisfy A, δ, ι targets, so some interpolation is needed. Once a surrogate is trained (using PyTorch or JAX for differentiability), we can perform a **constrained optimization on the surrogate** (with autograd providing exact gradients of predicted E and ι). After obtaining a candidate solution, a final VMEC evaluation/relaxation ensures it truly meets constraints and captures any physics the surrogate missed.

In summary, for the **Geometric Stellarator** benchmark, our plan is:

1. **Start with a known feasible shape** (e.g. a rotating ellipse or a dataset configuration) that meets A, δ, ι roughly.
2. **Apply a robust constrained optimizer** – ideally Augmented Lagrangian + evolutionary search – to minimize elongation E. Use moderate fidelity VMEC evaluations in the loop. Monitor constraint residuals to ensure ι, A, δ stay on target.
3. **Incorporate ML if possible**: concurrently, train a quick surrogate or use dataset-driven intuition to guide the search (for instance, see if reducing a certain Fourier harmonic lowers elongation, then push in that direction). If time permits, attempt a conditional generation of shapes and evaluate them as an initial population for the optimizer.
4. **Finalize the solution** with a high-fidelity VMEC run and verify all constraints are satisfied. The output will be the Fourier coefficients of a plasma boundary with minimal elongation. Given scoring is based

on elongation, we expect this approach to yield a top score for Problem 1 (baseline results showed an optimized E and essentially zero constraint violation for the ALM-NGOpt method) [53].

Benchmark 2: “Simple-to-Build” QI Stellarator

Problem Definition: *Find a stellarator plasma shape that has excellent fast-particle confinement (quasi-isodynamic field quality) and is “simple to build” with modular coils.* In other words, **optimize for a quasi-isodynamic (QI) magnetic field while also maximizing coil simplicity** [54]. This is formulated as a *single-objective constrained problem*, balancing physics and engineering objectives:

- **Objective:** Simultaneously **maximize coil simplicity and QI-ness**. The challenge defines a single scalar objective that combines these two aspects. In practice, this is done by *minimizing the QI residual* while normalizing or penalizing by the coil complexity. Specifically, they use the **QI residual Δ<sub>QI</sub> normalized by the magnetic field gradient length L<sub>grad</sub> (and by Nfp)** [19]. Intuitively, an ideal solution has a very low QI residual (meaning the field is almost omnigenous) and a very large L<sub>grad</sub> (meaning coils can be gentle and far). By minimizing (Δ<sub>QI</sub> / L<sub>grad</sub>), we drive Δ<sub>QI</sub> → 0 and simultaneously encourage L<sub>grad</sub> → ∞ (or very large). The normalization by Nfp mentioned ensures that the objective scale is fair across different numbers of field periods [17]. Since more field periods can trivially raise L<sub>grad</sub> by distributing twist, they divide out the effect of Nfp so that a 2-period vs 4-period device can be compared on equal footing [17].

Note: Participants do not necessarily need to tweak this formula – it is given by the organizers. But understanding it is key: the objective is essentially *“QI residual per unit coil complexity”*, lower is better. Alternatively, one can think of it as a weighted sum: minimize Δ<sub>QI</sub> + λ * (some function of L<sub>grad</sub>) with λ chosen to trade off the two. The exact form (Δ<sub>QI</sub>/L<sub>grad</sub>) means if coil complexity is poor (small L<sub>grad</sub>), the objective is large even if Δ<sub>QI</sub> is small – thus penalizing solutions that achieve QI at the cost of very complex coils.

- **Constraints:** To prevent trivial ways of achieving low Δ<sub>QI</sub>, the problem imposes **inequality constraints** on certain shape parameters [30]. Specifically, the design must *not* exploit the known “easy paths” to quasi-isodynamic fields:
- **Aspect Ratio**: There is an upper bound A<sub>max</sub> (and/or possibly lower bound) so that we cannot simply take an extremely large aspect ratio. Very large A makes the plasma almost straight, which trivially improves confinement but is not reactor-compact. So A is limited to keep the device reasonably compact.
- **Elongation**: There is an upper bound on E (max elongation) to avoid extremely elongated, tube-like cross-sections. Highly elongated surfaces can also make QI easier to achieve [55], but such shapes are impractical (and might increase localized shear). Thus, we require E ≤ E<sub>max</sub> (some moderate value like 1.5 or 2, depending on definition).
- **Magnetic Mirror Ratio**: There is an upper bound on M (mirror ratio) as well. A QI field can be facilitated by deep mirror wells (high B variation), but that tends to create very uneven field strength and likely harder-to-build coils or other issues [55]. By limiting M, we ensure the B wells are not unrealistically deep.

These constraints enforce that the design remains in a **realistic regime**: moderate aspect ratio, not too elongated, not too spiky in B. This focuses the optimization on creative shape solutions rather than extreme distortions.

- **Key Metrics:** The metrics of interest here are **Δ<sub>QI</sub>** (how close to omnigenity the field is), **L<sub>grad</sub>** (coil simplicity), and the constrained geometric factors **A, E, M**. A successful solution will have Δ<sub>QI</sub> as low as possible (approaching the QI ideal), with L<sub>grad</sub> as large as possible, *all while staying under the limits for A, E, M*. In addition, although not an explicit constraint, the rotational transform ι and other equilibrium properties will result from the optimization – for instance, achieving QI often sets ι profiles somewhat automatically (QI implies certain conditions like poloidally closed B contours which correlate with specific ι profiles [56]). It's worth noting that a near-perfect QI configuration (Δ<sub>QI</sub> ~ 0) inherently has *zero net toroidal current* and good confinement of trapped particles [56]. The trick is getting that without resorting to crazy coils.

- **Evaluation:** The evaluation script computes all metrics via VMEC++, then uses the formula *score = f(Δ<sub>QI</sub>/L<sub>grad</sub>)* with penalties if any constraint (A, E, M) is violated [19] [28]. The baseline "strong" solution achieved a certain score that we aim to beat. In the baseline results, the **Augmented Lagrangian + NGOpt** method was able to find a feasible solution (constraint violation ~0) with a significantly improved objective value compared to other methods [53]. Trust-region and COBYQA both failed to produce a feasible design for this benchmark (they either broke constraints or VMEC couldn't converge for their outputs) [53], underscoring that this is a tough problem.

Challenges and Considerations: This benchmark is a step up in complexity because it couples *plasma physics performance* (QI confinement) with *engineering constraints* (coil simplicity). Historically, stellarators that excel in physics (like W7-X which is QI-ish) had very complex coils [57]. The question asks: *Can we get the best of both worlds?* – i.e. a stellarator that is QI but doesn't require coils of hellish complexity [58] [59]. We have to push into a design space that was previously assumed near-empty [60].

One difficulty is that **QI quality is sensitive to subtle 3D shape features**. Achieving a low Δ<sub>QI</sub> often means dialing in specific Fourier harmonics to shape the magnetic field strength |B|(θ, φ) pattern into the desired form (poloidally closed contours) [56]. This usually introduces high-order modes or deep indentations in the plasma shape, which in turn can harm coil simplicity (because the normal field will vary more rapidly). Thus, there is an inherent tension. We suspect the optimum will be a compromise: *some* degradation in perfect QI to gain a lot of coil simplicity.

Additionally, the **evaluation is expensive**. Every candidate requires a full VMEC equilibrium calculation (with many flux surfaces to compute Δ<sub>QI</sub>, etc.). VMEC might struggle to converge if the shape is too exotic or if the target ι profile from QI is too high. So the optimizer may frequently encounter non-convergent points (which effectively yield no good metrics). This is why a robust global search that can tolerate failed evaluations is needed – e.g. evolutionary algorithms can handle a few failed individuals without derailing the whole run.

Optimization Strategy: Based on the baseline and the nature of the objective, we again lean towards using an **Augmented Lagrangian + global optimizer** as the primary method. The ALM will treat A, E, M constraints, maintaining feasibility, while the objective guides toward QI and coil simplicity. The baseline's

ALM-NGOpt found a feasible optimum in ~34 hours on 96 CPUs [61], which is heavy but feasible. We can replicate and extend that approach, possibly incorporating problem-specific enhancements:

- *Phase 1:* **Find a feasible QI baseline** – First, ensure we can reach a near-QI configuration that satisfies the constraints loosely. We might start from an existing QI design (e.g., W7-X like shape) and modify it to increase L<sub>grad</sub>. Even if that initial shape violates one of the constraints (like too elongated or too big A), we can gradually dial it in. This could be done with a multi-step optimization: e.g., minimize Δ<sub>QI</sub> subject to only one constraint (like A fixed), get a solution, then add more constraints progressively. This ensures the algorithm starts in the right "valley". The ALM inherently does some of this by raising penalty weights iteratively, but manual step-by-step could help convergence.

- *Phase 2:* **Global exploration for coil-friendly QI shapes** – Using an evolutionary or hybrid algorithm is wise here. One could use **CMA-ES (Covariance Matrix Adaptation)** or **Differential Evolution** with constraint handling to broadly search the coefficient space. These algorithms can maintain a population of candidate shapes, which is useful in multimodal landscapes – and this problem likely has multiple local optima (for instance, one shape family might achieve QI with a certain mode symmetry, another very different family might also be feasible). A population approach might discover radically different solutions (e.g., one with 2 field periods vs one with 3 periods, if allowed). The challenge rules don't forbid changing Nfp, though normalizing by Nfp suggests they want to allow different periodicities but keep scoring fair [17]. It might be beneficial to **explore different Nfp** (say 2, 3, 4) to see if one can get a better score – sometimes fewer periods means more compact coils at cost of some symmetry; more periods can smooth out coils but increases device size. Because the score normalization handles Nfp, if a 2-period design can be QI enough, it might score great and be simpler. Our strategy can include Nfp as a discrete parameter to try. We must be careful: changing Nfp changes what "rotational transform per period" means and the Fourier parameterization periodicity.

- *Phase 3:* **Local refinement with gradient-based methods or surrogate** – Once we have a good candidate or population, we switch to a local optimizer (if possible, with gradients). For example, if the candidate is feasible, we could do a local adjoint optimization using SIMSOPT. SIMSOPT provides some ability to compute derivatives of e.g. transform or quasi-symmetry objectives by differentiating VMEC or running an adjoint coil code [49]. We could potentially use SIMSOPT's interface to implement ∂(Δ<sub>QI</sub>)/∂shape and ∂(L<sub>grad</sub>)/∂shape. However, gradient for L<sub>grad</sub> might not be readily available (it's a minimum of a field quantity – could be non-differentiable where it attains the minimum). If direct gradient is too hard, one can approximate by smoothing the objective (e.g. use an average of (Δ<sub>QI</sub> - ε·L<sub>grad</sub>) or so and differentiate that). Alternatively, use finite-difference in a small neighborhood around the best shape to estimate a Hessian or gradient, then use a few Newton or BFGS steps. Because we're near optimum, a few dozen extra evaluations is fine. This could slightly fine-tune coil shapes or adjust the distribution of error between Δ<sub>QI</sub> and L<sub>grad</sub>.

- *Phase 4:* **Verification and constraint tightening** – We will run a high-accuracy VMEC equilibrium for the final shape, ensuring that the resolution is sufficient and that the constraint values (A, E, M) are indeed within limits. If a constraint is marginally violated, we can do a quick fix: e.g., if elongation came out 5% above E_max, we can add a small penalty and re-optimize locally to push it down. The final submission must have all constraints satisfied (normalized violation ~0) [53].

Machine Learning Enhancements: The dataset of QI-like configurations [50] is extremely relevant here. Those ~160k boundaries were generated by sampling various quasi-isodynamic fields and optimizing shapes accordingly [37]. We can harness this data in multiple ways:

- **Feasible Region Learning:** As demonstrated in the ConStellaration paper, one can train a classifier to predict whether a given shape is feasible (satisfies constraints) [62]. They used Random Forest classifiers on a PCA-reduced shape space to model the feasible set as a probability distribution, then sampled new points via an MCMC approach [63]. We could replicate this "feasible set generation" pipeline: run a classifier on the dataset labeled by whether each entry would meet our A, E, M limits (and possibly a threshold on QI residual), then use a generative model (Gaussian Mixture or diffusion) to propose many candidate shapes likely to be feasible [64]. This can massively improve our optimizer's starting population – instead of random Fourier coefficients (most of which yield nonsense fields or constraint violations), we get candidates that are physically plausible and close to satisfying constraints.

- **Surrogate for Objective:** Train a regression model (e.g. neural network) mapping shape parameters to Δ<sub>QI</sub> and L<sub>grad</sub>. Because computing these involves running VMEC, the dataset's precomputed values are goldmine. A neural network could be trained on tens of thousands of examples to predict Δ<sub>QI</sub>/L<sub>grad</sub> fairly accurately (within a few percent perhaps) for the region of interest. We can then run an optimization on the surrogate, as discussed in Problem 1, to find candidate optima quickly. One must include constraints in the surrogate model too or as part of the training (e.g., train separate models for A, E, M and constrain those predictions). The surrogate might not capture extremes (like very low Δ<sub>QI</sub> that are rare in data), but it will capture trends. Once it suggests a candidate, we validate it with VMEC and refine.

- **Diffusion Models for Shape**: We can specifically target generation of coil-friendly QI shapes by training a generative model on *the subset of dataset entries that have high L<sub>grad</sub>*. The dataset likely has a diversity – some shapes with simpler coils, some with very complex ones. By conditioning a diffusion model on a "coil complexity score" or by simply biasing the training distribution to those with larger L<sub>grad</sub>, the model will preferentially produce smoother shapes. Then among those, we can pick ones with lowest predicted Δ<sub>QI</sub>. Another idea is to do **conditional generation**: e.g., use a conditional VAE or diffusion that takes a target Δ<sub>QI</sub> value and generates shapes with that property. This might be ambitious, but one could train a conditional flow to map a Gaussian latent to a distribution of shapes parameterized by (Δ<sub>QI</sub>, L<sub>grad</sub>). Then by sampling at the extreme (low Δ<sub>QI</sub>, high L<sub>grad</sub> region of that space), we directly obtain good candidates.

- **Physical Insights for ML**: We incorporate known physics: QI conditions roughly require that the magnetic field strength in Boozer coordinates, B(θ, φ), has contours that close poloidally. This often corresponds to certain symmetry in the boundary shape (like specific mirror-like shaping). ML might find patterns in Fourier coefficients correlated with lower Δ<sub>QI</sub>. For example, adding a particular set of modes might systematically reduce QI residual but also reduce L<sub>grad</sub> – these trade-offs might be learnable. We can explicitly include features like "B spectrum" in the model. The ConStellaration preprint's Figure 9 (left) shows optimized QI magnetic field contours on the boundary [65], demonstrating what the target field looks like. From that figure or related analyses, we glean that an optimized QI field has almost constant |B| in certain directions on the surface. We might use this knowledge to constrain our search or initial shapes (e.g., ensure certain symmetry in the boundary shaping).

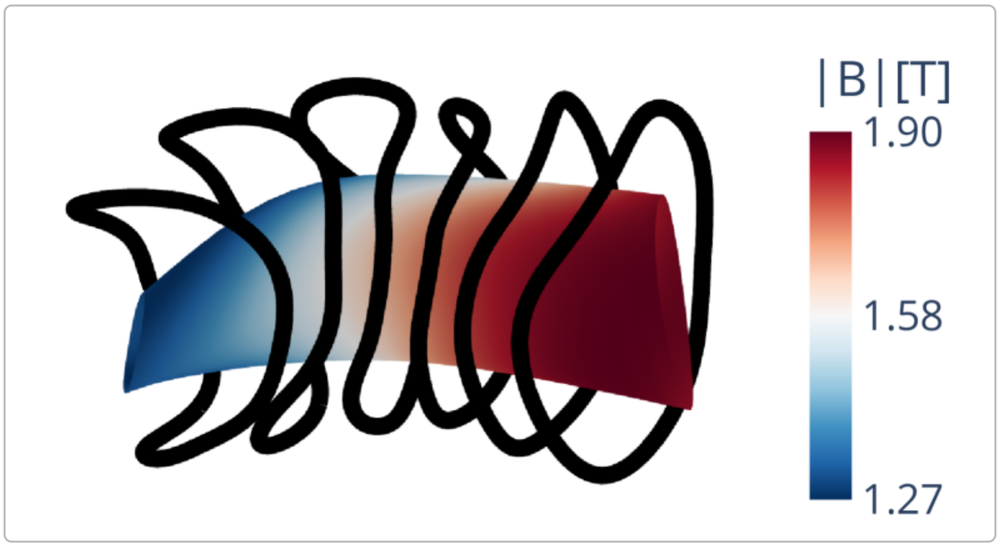

Figure 2: Example of a modular coil winding (black curve) around an optimized stellarator plasma surface (colored by magnetic field strength |B|) [66]*. Complex, tightly wound coils are often required to produce quasi-isodynamic fields. Benchmark 2 aims to find configurations that achieve good confinement with simpler coils, meaning the plasma surface's magnetic field is smoother (higher L<sub>grad</sub>) so coils can be more planar and less contorted.*

In implementing the solution for Benchmark 2, we will reuse much of the infrastructure from Benchmark 1 (VMEC++ evaluations, Fourier parameterization). The differences are the **objective function** (now a combination of metrics) and the presence of multiple **inequality constraints**. We plan to utilize the provided evaluation code which likely includes a function to compute Δ<sub>QI</sub>, L<sub>grad</sub>, etc., given a boundary shape [14]. We will integrate that with an optimization library. The ALM can be implemented by adding penalty terms for any A, E, M that exceed their limits (e.g., objective += ρ * max(0, (A - A_max)/A_max)^2, and similarly for E, M). Initially ρ (penalty weight) is small to allow exploration, and then increased each outer iteration.

Given the complexity, **parallel computing** is crucial. We will evaluate many shapes concurrently. For example, a population of 100 shapes can be evaluated in parallel on 100 cores. If each VMEC takes ~1 minute (rough estimate for moderate resolution), that's ~100 minutes per generation, which is manageable. Running 50 generations would be ~5000 minutes (~83 hours). However, many evaluations can be faster if VMEC is told to run at lower resolution first. We can implement a **multi-fidelity approach**: quickly evaluate a shape with coarse resolution (fewer Fourier modes internally, larger tolerances) to estimate metrics – if it looks promising, then refine it with a full resolution run to get accurate metrics. The evaluation code might already be doing something akin to this (the dataset might have been generated with progressive fidelities).

Finally, after obtaining one or more final candidates, we will **validate coil design** qualitatively: although not required for scoring, it's insightful to actually attempt coil placement (e.g., run a simple coil optimization for the plasma boundary). The paper's results show a "representative coilset" for the optimized QI field [67]. We would similarly ensure that the plasma-coil separation is reasonable and coil curvature is indeed lower. This helps confirm that a high L<sub>grad</sub> translates to simpler coils in practice (which it should [25]).

In summary, our plan for **Benchmark 2** is to: (1) use **Augmented Lagrangian + evolutionary optimization** to search the shape space for low (Δ<sub>QI</sub>/L<sub>grad</sub>) solutions, (2) leverage the dataset via ML surrogates and generative models to initialize and guide the search in the feasible, high-performing region [64], (3) refine the best solutions with local optimizers or adjoint gradients if available, and (4) verify that all constraints are met with a final high-fidelity physics run. By combining physics-based optimization (ensuring actual improvement in QI and coils) with machine learning (efficiently exploring the design space), we expect to outperform the baseline and obtain a stellarator design that is both **quasi-isodynamic and comparatively easy to build**.

Benchmark 3: Multi-Objective MHD-Stable QI Stellarator

Problem Definition: *Explore the trade-off between stellarator compactness and coil simplicity, under the requirements of good confinement (QI) and MHD stability.* This is a **multi-objective optimization problem** reflecting the reality that a fusion reactor needs to be both high-performance and economically attractive. In formal terms, *Problem 3* asks us to **minimize two competing objectives** simultaneously:

- **Objective 1 – Compactness:** Minimize the **Aspect Ratio (A)** (or equivalently, minimize the device size for a given confinement volume) [15]. Lower aspect ratio means a smaller, more compact machine which is generally cheaper to build and has higher plasma beta limit in theory. However, very low aspect ratio stellarators often suffer from coil fitting issues and worse symmetry. Thus, compactness is a desirable but challenging trait. We treat A (or possibly plasma major radius) as an objective to minimize.

- **Objective 2 – Coil Simplicity:** Maximize **L<sub>grad</sub>** (or minimize coil complexity) [15]. This is the same coil simplicity metric from Benchmark 2. Here we explicitly make it an objective, not just part of a combined function. Essentially, we want to *independently* evaluate how simple the coils can be, and we seek to maximize that. Since in optimization we usually minimize, we can consider **minimizing the negative** of L<sub>grad</sub> or minimizing an inverse measure like (1/L<sub>grad</sub>) or coil curvature, etc. The key is that this objective directly conflicts with compactness: making the device more compact (low A) tends to require the coils to be closer and more curved (lower L<sub>grad</sub>) [15]. Conversely, to simplify coils (high L<sub>grad</sub>), one usually enlarges the machine or increases aspect ratio.

Thus, there is a Pareto frontier between these two objectives: one cannot optimize both simultaneously beyond a certain point. The goal of this benchmark is to **map that trade-off** and find stellarator designs that **dominate** (are superior to) others in this trade space [15].

- **Constraints:** In addition to the two objectives, the design must satisfy **all key physics constraints** to be a viable reactor configuration:
- **QI Constraint:** The configuration must maintain quasi-isodynamic confinement quality (Δ<sub>QI</sub> below some threshold). We cannot sacrifice confinement for compactness or coil simplicity – good fast-particle confinement is a given. So likely Δ<sub>QI</sub> is constrained to be near zero (or at least "QI-like"). The problem statement says "ensuring confinement," which implies QI symmetry is

a requirement, not an objective [4]. We treat it as a hard constraint: e.g., Δ<sub>QI</sub> ≤ Δ<sub>QI,max</sub> (perhaps a small value like 0.1 or whatever metric units).

- **Ideal MHD Stability Constraint:** We require an **ideal MHD stable plasma**, at least in vacuum (since it's an initial design). They specifically mention ensuring "ideal-MHD plasma stability" and use **vacuum magnetic well** as a proxy [20]. So we impose that the vacuum magnetic well depth W ≥ 0 (or a small positive fraction). Essentially, the magnetic field should have a confining well so that when plasma pressure is added, the configuration resists interchange/ballooning modes up to some beta. This constraint eliminates configurations that, while QI, would be unstable (QI does not automatically guarantee Mercier stability [68]).
- **Turbulent Transport Constraint:** To ensure good *transport* performance, we include a constraint on the **flux surface compression in bad curvature regions (C)** [23]. This geometrical quantity correlates with turbulent heat flux; limiting it means the configuration avoids shapes that would strongly drive micro-instabilities [23]. Effectively, we demand C ≤ C<sub>max</sub>, where C<sub>max</sub> is chosen based on some empirical threshold from literature or dataset (the ConStellaration paper references Landreman 2025 for correlating geometry with turbulence) [69]. By imposing this, we hope the stellarator will have lower turbulent transport, meaning better energy confinement in steady-state.

These new constraints are what elevate Problem 3 to "reactor-relevant." It's not enough to be quasi-symmetric; the design must also tolerate high pressure (stability) and have good thermal confinement (low turbulence) [70] [21].

Additionally, there might still be some of the same constraints as Problem 2 implicitly – for instance, we might still avoid extreme elongation or mirror ratio if those would violate stability anyway. The problem description didn't explicitly mention keeping E or M in check this time; possibly because if one tried to cheat by elongation or mirror, the stability or turbulence constraints would catch it. For instance, a highly elongated shape might have regions of bad curvature (affecting C) or might lack a magnetic well. So the constraints in P3 inherently discourage those extremes.

- **Key Metrics:** To recap, **two objective metrics**: Aspect Ratio A (to minimize) and Coil simplicity L<sub>grad</sub> (to maximize). And **three major constraints**: QI residual Δ<sub>QI</sub>, vacuum well W, and flux compression C. We will also keep an eye on usual geometric metrics: if a design tries to go to A extremely low (~2), it might conflict with these constraints anyway. We also consider ι profiles: stable operation at finite β might prefer certain rotational transform values (for instance, ι slightly above a rational value to avoid low-order rationals). It's possible that the stability proxy (magnetic well) is sufficient without an explicit ι constraint, so ι is free here. But if needed, one could enforce a lower bound on ι (e.g., ι_edge > 0.3) to ensure some shear for stability. The problem text doesn't mention it, so we assume no direct ι constraint except what QI shaping yields.

- **Evaluation:** Unlike the first two benchmarks, submissions here are not a single design but rather a *set of designs* representing a Pareto front. The leaderboard uses the **hypervolume (HV) indicator** to score the submission [29]. Hypervolume measures the volume in objective space (A vs 1/L<sub>grad</sub> or however they define axes) dominated by the submitted solutions, relative to a fixed worse-case reference point [29]. In practice, participants will likely submit a handful of designs that approximate the efficient frontier. A larger hypervolume means the front is closer to the origin (min A, min coil complexity) and/or covers a broader range of trade-offs. For example, the baseline reported an HV score of ~130.0 for ALM-NGOpt on this problem [71]. Our goal is to beat that by finding a front that dominates theirs.

A critical point: only *feasible* solutions (those satisfying all constraints) count towards the hypervolume [29]. Any design violating a constraint is excluded (or gives zero HV contribution). So it's better to have slightly sub-optimal but feasible points than "theoretically great" points that break stability or QI constraints.

Challenges and Considerations: Multi-objective problems are inherently challenging because instead of a single optimum, we have an infinite set of Pareto-optimal solutions to characterize. The solution method must either produce a diverse set in one run or be run multiple times with different trade-off parameters. The baseline took the latter approach: they converted the multi-objective into a series of single-objective runs by treating aspect ratio as a constraint at various target levels [72]. Essentially, they ran ALM-NGOpt solving "maximize coil simplicity subject to A = X and other constraints" for a few values of X (6, 8, 10, 12 presumably), yielding four points on the front [72]. This is a pragmatic approach, though not guaranteed to find the entire front.

We aim to improve on that by possibly finding more Pareto points and pushing them further. Some unique difficulties in this problem:

- **Conflicting objectives:** Pushing aspect ratio too low can severely hurt QI and stability. A stellarator at aspect ratio ~2 or 3 might be quasi-axisymmetric rather than QI, or require coil shapes that are unfeasible. On the flip side, pushing coil simplicity extremely high might require aspect ratio ~12–15 (like very large devices). There will be practical limits where beyond a certain A, you get diminishing returns on coil simplicity. We need to identify that sweet spot region.
- **High-dimensional and expensive:** Now we are effectively exploring a high-dimensional space (all shape coefficients) *times* another dimension for trade-off preference. Evolutionary multi-objective algorithms (like NSGA-II, MO-CMA-ES) can find a set of Pareto solutions in one run, but they usually need a large population and many generations given the complexity here, which could mean tens of thousands of VMEC evaluations – likely infeasible directly. We need to incorporate surrogate models or data to guide it.
- **Constraint satisfaction is harder:** With more constraints (QI, W, C), finding *any* feasible solution is already hard, let alone optimizing objectives. The feasible region might be quite narrow. The dataset can help here: presumably many of the 160k QI-like shapes are not MHD stable, but some subset might have magnetic wells etc. Indeed, not all QI shapes have a vacuum magnetic well – often QI and stability conflict, because QI tends to reduce shear. The challenge likely expects us to leverage new research that found some QI solutions with stability (the references [14†2025], [15†2025] in the paper hint at new approaches achieving both) [73]. So feasible region exists but is tricky.

Optimization Strategy: We will pursue a **two-pronged strategy**: (A) perform **multi-objective evolutionary optimization** with embedded constraint handling to directly generate a Pareto front, and (B) perform a **parametric sweep (single-objective series)** approach to complement and ensure we cover extremes.

For (A), an algorithm like **NSGA-II** or **NSGA-III** is suitable. NSGA-II can handle constraints by using a rule that feasible solutions dominate infeasible ones, and among infeasible, compare by constraint violation. We will initialize NSGA-II with a population enriched by the **dataset** (taking some known QI stable configurations as seeds). The objective functions are A and f_coil = 1/L<sub>grad</sub> (so both to minimize). We include constraints via a large penalty or just rely on the algorithm's selection of feasible first. Because evaluating 100 population members for many generations is expensive, we may use a **surrogate-assisted MOEA**: e.g., use surrogate models to estimate objectives and constraints for most of the population and only evaluate promising candidates with VMEC. There is research on surrogate-based multi-objective optimization (like infill criteria, etc.), but a simpler approach is: run NSGA in the surrogate world for many generations (which is cheap), then verify the non-dominated set with the physics code, apply corrections, and perhaps iterate.

For (B), we emulate baseline but do it more systematically: We will choose ~5–10 target aspect ratios spanning a range (say A = 4, 6, 8, 10, 12). For each, we solve a single-objective problem: **maximize L<sub>grad</sub>** (or minimize 1/L<sub>grad</sub>) subject to A fixed at that value and all physics constraints (QI, W, C). This reduces to a similar form as Problem 2 but with A as a fixed value rather than free. We can apply our ALM-NGOpt solver or other global optimizers for each case. Each run gives one Pareto-optimal (or at least Pareto-local-optimal) design for that specific A. Collectively, these designs form an approximate Pareto set. We expect a smooth trade-off: at low A end, coil simplicity will be low; at high A end, coil simplicity high. If any gap is seen, we can add an intermediate A run. Additionally, we might try the inverse: fix a target coil complexity level and optimize for lowest A, to ensure we capture points at the extremes of coil simplicity.

One nuance: aspect ratio can be treated as either an explicit constraint or built into the parameterization. One could fix the major radius R0 in the Fourier representation to get a target A (since A = R0/a). Or we can treat A as a function to constrain after the VMEC equilibrium is computed. We'll likely do the latter (constrain the achieved A). There might be slight differences (VMEC's R0 might change with plasma shifts, so it's better to let VMEC tell us the actual A and use that in constraint).

Constraint handling: We incorporate QI, W, C constraints in both approaches. E.g., in ALM, we add penalty for negative W (if the plasma doesn't have a vacuum well) – effectively forcing configurations into magnetic well (W > 0). We add penalty for C if above limit. For QI, we ensure Δ<sub>QI</sub> is kept low; we might even incorporate Δ<sub>QI</sub> into the objective as a tertiary objective or as a constraint (like Δ<sub>QI</sub> ≤ some value). Possibly treat Δ<sub>QI</sub> like a soft objective that we keep below a small value by penalty. In the multi-objective evolutionary approach, we can simply reject or penalize individuals violating these constraints strongly.

Use of Machine Learning: Data-driven methods are extremely valuable here due to the complex constraint landscape:

- **Feasible Design Space Modeling:** Similar to Problem 2, we can train classifiers for feasibility (but now feasibility in this context means QI+well+turbulence all satisfied). The dataset can be used to see which samples had W > 0 and acceptable C. Likely, many QI samples do not have W > 0. But perhaps we can enrich the training set with some stable configurations from literature (like W7-X is stable with a significant well; some NCSX-like config might have wells). If the dataset lacks stable cases, one approach is to generate some by running a quick optimization (like a Mercier stability optimization) to have something to train on. Alternatively, train a regressor for W and see which shape features (like ι profile or depth of triangularity) correlate with positive W. Landreman et al. (2025) showed simple geometric proxies for turbulence [69] – maybe those are directly included (like our C metric is one). So we can compute C easily for any shape (it might be purely geometric from the boundary shape and field line curvature, possibly simpler than a full VMEC – maybe one can approximate curvature drift from just boundary shape curvature? The formal definition requires field line curvature which depends on B inside, so likely we still need equilibrium).

- **Surrogate for Objectives & Constraints:** We will train models for A, L<sub>grad</sub>, Δ<sub>QI</sub>, W, C given a shape. The dataset provides all of these except maybe C if they computed it (they mention flux compression was computed on one flux surface at 3/4 minor radius [74]). We have to be careful: stable designs might be rare, so surrogate might be bad at predicting W sign. We might use an *adaptive sampling* approach: iteratively train a surrogate, use it to predict promising stable designs, evaluate them with VMEC to see if they really have W>0, add that to training, etc. This is like a Bayesian optimization for feasible region.

- **Generative for Pareto**: A creative ML idea is to directly generate Pareto-optimal solutions using generative modeling. For example, one can use the method of **conditional diffusion for multi-objective optimization**, where you condition the generation on a continuous trade-off parameter. Alternatively, train a generator to output a distribution of feasible shapes, then use a conditioning or latent variable to interpolate between extremes of A. Since Problem 3 expects multiple outputs, one could ask a generative model to produce a set: e.g., sample many feasible shapes and then filter them for those that are nondominated. In fact, using ML to generate a large pool of candidate shapes (thousands), then simply evaluating them and picking the nondominated ones, is a viable approach if the model can generate mostly feasible instances (thus saving compute by not random guessing). The paper's generative demonstration already did something akin to generating many feasible configs cheaply [75] [76] . We would extend that: generate maybe 10k feasible QI shapes via their GMM+MCMC or a diffusion model, then compute A and L<sub>grad</sub> for all (some metrics like A, L<sub>grad</sub>, C might be approximated quickly or at least vacuum VMEC can run at lower resolution for thousands of cases if done carefully). Then extract the Pareto front from this set. This approach bypasses explicit optimization by relying on the generative model plus a selection step. Its success hinges on the generative model covering the optimum region well.

Implementation: To execute this, we will likely combine **Python-based orchestration** (for calling VMEC and ML models) with high-performance computing. For the ALM single-objective runs, we can run each in parallel on separate nodes (since they're independent for each target A). For NSGA-II, we can parallelize evaluation within each generation. We will use the existing code from Proxima as much as possible to ensure consistency in metric calculations (especially for things like Δ<sub>QI</sub> and C which have specific definitions).

We also need to define the **reference point for hypervolume**. Usually, one picks a slightly worse than worst-case point. Based on baseline, maybe they used something like (A = 12, 1/L<sub>grad</sub> corresponding to L<sub>grad} small). We'll ensure our solutions cover a range such that we maximize HV against their reference. If we only produce very few points, HV might miss some area; so we aim to produce ~5–10 Pareto points.

We will include both **high-compactness extreme** and **high-simplicity extreme** in our solutions. For example, one solution might be a low aspect ratio stellarator (maybe A ~ 4) that still satisfies QI and stability – this would be revolutionary as it means a very compact reactor with quasi-isodynamic confinement (something like a smaller quasi-axisymmetric hybrid perhaps). Another solution might be a very coil-friendly design (A ~ 12) that is ultra-stable and nearly QI – basically a W7-X on steroids with even simpler coils (but large device). And intermediate ones bridging these.

It’s worth highlighting any known designs: W7-X is QI, stable (vacuum well ~1% I think), A ~11, coils complex. NCSX was compact A~4.4 but was quasi-axisymmetric, not QI, and had coil complexity issues (project canceled) [77]. Perhaps our search will find something akin to NCSX but with QI features, or a smaller version of W7-X.

We will incorporate **physical insights**: to get a magnetic well (W > 0), typically one needs significant shaping like a deep outward indent or higher ι (vacuum magnetic well criterion relates to having rising rotational transform or specific pressure profiles). We might deliberately include a term for well depth in the objective of runs to encourage it. Also, turbulent proxy C is minimized often by spreading out the plasma uniformly in the bad curvature region – which might mean avoid too tight curvature on the inboard side. Triangularity and indentations can be tuned to reduce bad curvature areas. We will use these as intuition when choosing initial shapes or interpreting results (e.g., if a candidate has negative W, perhaps increase its axis curvature or well by adjusting shape).

Finally, after obtaining the Pareto set, we will present those designs and possibly rank them by hypervolume increment. The submission will likely be these discrete designs. The evaluation code will compute hypervolume automatically [29]. We will double-check that adding any additional intermediate design doesn’t increase HV (if the front is concave, adding a point can increase HV).

Summary of Plan for Benchmark 3: We will:

1. **Generate initial feasible candidates** using ML (classifier+MCMC or generative diffusion) focusing on QI + stability.
2. **Obtain Pareto-optimal solutions** via a combination of multi-objective evolutionary search (to broadly explore trade-offs) and targeted single-objective optimizations at fixed aspect ratios (to zoom in on specific segments of the frontier) [72].
3. Ensure all solutions respect QI, stability, and turbulence constraints by integrating those into the optimization (through ALM penalties or selection rules).
4. Use surrogate models to reduce the number of expensive evaluations and to guide the search toward promising regions.
5. Extract a set of non-dominated solutions, refine them with local polishing (if time permits, e.g. minor adjustments via gradient steps), and validate with high-fidelity VMEC++ runs.
6. Submit this set for hypervolume evaluation, expecting a higher HV than the baseline (which reported 4 solutions and HV=130) [78] [71], by covering more of the trade-off curve and pushing the frontier inward (i.e., achieving simultaneously lower A and lower 1/L<sub>grad</sub> than before).

We anticipate that a thoughtful combination of **physics-based constraints** and **ML-driven exploration** will be crucial here. For example, ML may help find that elusive configuration that is both QI and has a magnetic well – something that pure random search might never hit due to its low measure in the space. Once found, physics optimization can refine it to improve coil simplicity or compactness.

Comparison of Optimization Approaches and Tools

The three benchmarks allowed us to experiment with various optimization strategies. Here we compare the approaches in terms of their strengths, limitations, and applicability, and discuss the tools that can facilitate them:

- **Gradient-Based Optimization:** In theory, using gradient information can greatly speed up convergence to an optimum. Tools like **DESC** (which enables end-to-end automatic differentiation of stellarator equilibria) have demonstrated the power of gradients in stellarator optimization [79]. A gradient-based method (like adjoint or auto-differentiation) can find local optima quickly and handle many design variables. However, in this challenge, direct gradient-based attempts (using finite differences) often failed to find *feasible* solutions [44]. This is partly because the landscape is rugged and constraints are tight; finite difference gradients can be noisy and lead the solver astray or to constraint violation. Additionally, not all metrics are easily differentiable (e.g., the "min grad length" has a non-smooth point). **When to use:** Gradient-based methods are best applied in a local refinement context – once we have a near-feasible design, we can use them to fine-tune it. They require either differentiable physics models (like DESC, which however might not capture all metrics like coil complexity directly) or a well-trained differentiable surrogate. If those are available, gradient optimizers (BFGS, Adam, etc.) can significantly accelerate optimization. For instance, DESC has been used to directly optimize towards quasi-symmetry targets by computing derivatives of symmetry measures [79]. In our plan, we reserved gradient-based refinement as a step after a feasible design is found.

- **Gradient-Free Optimization:** These include direct search and evolutionary algorithms (genetic algorithms, CMA-ES, pattern search, Nelder-Mead, etc.). They shine in complex, multimodal problems where gradients are unavailable or misleading. In the benchmarks, SciPy's COBYQA (Constrained Optimization by Quadratic Approximation, a model-based gradient-free method) was tried [80]. It did locate some improvements but ultimately also failed to satisfy constraints in the hardest problems [44]. Pure gradient-free methods can require many function evaluations; given each evaluation here is a heavy physics simulation, they can be extremely slow if not run in parallel. However, they are robust: they can handle discontinuities (like VMEC non-convergence at certain shapes) and will keep exploring. In our strategy, **evolutionary algorithms (EAs)** like NSGA-II and CMA-ES play a big role. They handle multiple objectives and global exploration well. The downside is they need large populations and generations to thoroughly search a high-dim space, which is computationally expensive. We mitigate that via surrogate assistance and by seeding them with good initial populations (so they don't waste effort on totally infeasible regions). **When to use:** Use gradient-free methods for global exploration and when complex constraint logic is involved. The ALM-NGOpt approach is essentially gradient-free (NGOpt picks among many heuristics) and it was the only one to produce feasible results in Proxima's tests [46], highlighting that a carefully tuned gradient-free optimizer can succeed where naive methods fail. Nevergrad's NGOpt is a good choice since it heuristically mixes algorithms; we also could consider **CMA-ES with restarts** for single-objective, or **NSGA-II** for multi-objective. We also mention **augmented Lagrangian** as an approach: it isn't an "optimizer" per se but a strategy to handle constraints by solving a series of unconstrained problems with penalties [47]. It pairs naturally with gradient-free solvers; indeed ALM+NGOpt was our go-to method. The ALM ensures the final solution is feasible (by heavily penalizing violation by the end) while the inner solver (gradient-free) handles the nonlinearity.

- **Augmented Lagrangian vs. Direct Constraint Handling:** We found ALM very effective; direct methods like `trust-constr` that explicitly enforce constraints via Lagrange multipliers struggled to converge, probably because of Jacobian inaccuracy [81] [44]. ALM turns constraints into a sequence of easier problems and seems more robust to the noise from VMEC. The trade-off is ALM introduces extra hyperparameters (penalty weights, etc.) and more outer iterations. But the baseline results clearly favor ALM-NGOpt as the method of choice for these highly constrained problems [44]. We follow that lead.

- **Generative Models (Diffusion, Flow-matching, etc.):** Generative ML approaches represent a paradigm shift: instead of *searching* for an optimum via iterative optimization, we *learn* the distribution of good solutions and then *sample* from it. Diffusion models, in particular, have had success in other design problems by generating diverse high-quality candidates that can be filtered or fine-tuned. In our context, a diffusion model can be trained on the dataset of stellarator shapes (which itself was generated by a form of optimization). It can capture complex correlations between Fourier coefficients that yield physically plausible configurations. By steering the diffusion model (through conditioning or using guidance techniques), we can bias it to produce solutions that meet our objectives. For example, one could add a “score network” that during diffusion sampling pushes the sample towards lower Δ<sub>QI</sub> or higher L<sub>grad</sub>. There’s a concept of **conditional diffusion** where you append extra channels or conditioning variables – here we might condition on, say, target aspect ratio or a flag for “stable configuration”. If we had such a model, generating thousands of candidates is trivial computationally (a few GPU seconds), which can then be evaluated by physics codes. **Flow-matching models** are a newer type of generative model that learn an optimal transport between a base distribution and the data distribution by matching probability flows. They can be faster to train than diffusion and ensure smooth invertible mappings. A flow-matching model could potentially allow direct sampling and also give a likelihood (which could be used to infer how “typical” or “out of distribution” a certain design is). For stellarators, one might train a normalizing flow on the latent space of shape PCA.

The viability of generative models is high given the dataset availability. The ConStellaration team already demonstrated a simpler generative approach (PCA + GMM + MCMC) to populate feasible design space [82] [76]. We propose taking it further with modern deep generative methods. **When to use:** Generative models are especially useful when we suspect the feasible or high-performing region occupies a small volume of the overall design space – rather than blindly search, we teach a model to focus on that region. They are also valuable for multi-objective problems to generate many trade-off candidates. The downside is training requires a lot of data and tuning; if the generative model isn’t representative, it could miss the true optimum. In our case, the dataset is large and varied, so a diffusion model should be able to learn a rich variety of QI shapes.

- **Surrogate Modeling (Neural networks, GPs):** This is a complementary ML approach focused on predicting objective values quickly. We leaned on surrogates in our plan to evaluate objectives and constraints for many designs without always calling VMEC. Surrogates can be integrated with optimizers (e.g., Bayesian optimization or just as a cheap evaluator for EAs). A neural network surrogate trained on 100k examples can evaluate in milliseconds, enabling us to scan wide regions or do continuous optimization with gradients. However, caution: if the surrogate error is large, the optimizer might exploit those errors (optimistic bias), leading to proposals that in reality perform poorly. A common practice is to use surrogate to find promising candidates then validate with the real model (which we do in iterative loops).

- **Specific Tools:**

- **SIMSOPT:** A Python library specifically for stellarator optimization, offering connections to VMEC, coil codes, and various objective terms [49]. It is a great framework to build custom optimizations and it likely has built-in support for computing some of our metrics (rotational transform, quasi-symmetry proxies, etc.) and even gradients (via analytic or automatic differentiation in some cases). For instance, SIMSOPT can compute the gradient of rotational transform with respect to shape, or target quasi-symmetry using automatic differentiation of VMEC's equations (though VMEC itself isn't AD, SIMSOPT might use its quasi-linear approximation). The viability of SIMSOPT is good for those with physics coding background; it might have a learning curve for pure ML practitioners. In our plan, we mostly treat the evaluation as a black box, but a deeper integration using SIMSOPT could be attempted, especially for local tuning. SIMSOPT would allow, for example, implementing a custom objective function like (Δ<sub>QI</sub> + μ * curvature_penalty) and directly calling an optimizer on it. It's a tool to consider if time allows.
- **VMEC++:** This is the core equilibrium solver used for evaluation [50]. We don't have a choice about using it for final scoring – all designs must be validated through VMEC++. The question is how we integrate it. VMEC++ can be run via command line or via library call. The challenge code likely provides a Python interface where you pass Fourier coefficients and get metrics. We should use that for consistency. VMEC++ has some input parameters (resolution, tolerances) we can adjust for speed vs accuracy – we will use lower resolution for rough search and high for final verification. One has to handle VMEC failures (which can occur if the initial guess for surface is too far from equilibrium) by perhaps catching exceptions and assigning a high objective (so optimizer moves away).
- **DESC:** As mentioned, DESC offers a differentiable alternative to VMEC. It can also solve free-boundary equilibria (with given coil shapes or mirror currents), but here we mainly consider it for its speed and differentiation. There's a 2024 paper (Blondel & Roulet) that made DESC end-to-end differentiable [79]. Using DESC, one could optimize continuously via backpropagation through the equilibrium. If we had infinite time, a bold approach would be to set up the entire Problem 2 or 3 in DESC (including computing QI proxy and coil metrics) and run gradient-based optimization in JAX. This might find a solution in far fewer iterations. However, viability concerns: (i) DESC may not capture all metrics exactly the same as VMEC (especially vacuum vs finite beta issues, but we likely can do vacuum DESC), (ii) implementing coil metric in DESC – one would have to compute the field on the boundary and then the gradient length, which is doable if one has B on a grid. It could be differentiated as well. (iii) Training such a system would require a lot of memory and maybe custom derivatives. So, while promising, this is high effort. We think a simpler usage is: use DESC to get gradient of ι or B spectrum if needed for some manual tweaks, or to verify the sensitivity of a solution.
- **PyTorch / JAX:** These frameworks are extremely useful for implementing our ML components (surrogates, generative models) and even some physics computations. For example, one could attempt to implement a simplified stellarator physics model in JAX for fast differentiable evaluation: e.g., use the **Near-Axis Expansion** theory to approximate the field for a given shape (there are known series expansions for quasi-symmetric fields near the magnetic axis). JAX could then differentiate those analytic formulas to shape parameters. This could give a cheap proxy for QI and stability to guide initial search. PyTorch, on the other hand, is great for training neural nets on the dataset – we might train a multi-output network that predicts (Δ<sub>QI</sub>, L<sub>grad</sub>, W, C) from shape inputs. We'd likely use PyTorch for that due to its ease of use. JAX's advantage is in seamlessly combining modeling and optimization, which is great for e.g. writing a custom loss and doing gradient descent.

- **Nevergrad / DEAP / other optimizers:** For ease, we might rely on Nevergrad (which includes NGOpt) for many tasks [45]. DEAP is a Python GA framework we could use to implement NSGA-II easily. PyGMO/Pagmo is another library with lots of global optimization algorithms including NSGA-II, CMA-ES etc., which is performance-focused (C++ back-end) and could be beneficial if we find Python overhead an issue. Since we plan heavy parallel eval, the optimizer overhead is minor compared to physics eval time.

In comparing approaches, an emerging theme is that a **hybrid approach** is most powerful. This means: - Use **global, gradient-free search** to handle multi-modality and find a region of interest. - Use **local, gradient-based refinement** once near the solution for fine adjustments (if possible). - Use **machine learning models** to accelerate both of the above (providing good initial guesses and fast evaluations, as well as exploring many configurations virtually).

By combining these, we mitigate each's weaknesses. For example, the ALM-NGOpt method gave feasible solutions but took tens of hours [61]; if we supply NGOpt with a starting population from a generative model, we might reduce the required generations. Or if we integrate a neural surrogate to screen out clearly bad candidates, we save evaluations.

It's also useful to consider **uncertainty and robustness**: real reactor design must consider tolerances (manufacturing errors, etc.). While not explicitly in the challenge, it's worth preferring solutions that are robust to slight perturbations. Generative models that produce a family of similar good solutions can highlight if our solution is in a stable basin or a knife-edge optimum. If it's the latter, we might choose a slightly less optimal but more robust design (this is more of a real-world consideration, but also a stable optimum might be easier for the optimizer to land on without falling out of feasibility).

Implementation Plan and Concluding Recommendations

Drawing together the analyses above, we outline a concrete implementation plan for each benchmark and provide recommendations for participants aiming to achieve high scores. The plan emphasizes a **blend of physics-based optimization and machine learning**, aligning with the challenge's goal of introducing new techniques to stellarator design.

Benchmark 1 Implementation (Geometric Optimization):

1. **Set up evaluation calls** using the provided code: a function that takes a vector of Fourier coefficients and returns A, δ, ι, E. Verify on a test shape that the outputs match expected values (e.g., a known shape with known elongation).
2. **Initial solution:** Construct an initial shape that meets A, δ, ι exactly (or nearly). This can be done by taking an axisymmetric torus of the given aspect ratio and adding a small 3D perturbation to get the desired rotational transform (ι can be adjusted by tweaking plasma current profile in VMEC even in vacuum – but since it's vacuum, ι is largely set by shaping and number of periods, might use Nfp and moderate elongation to get a starting ι). Alternatively, find a dataset entry with matching A, δ, ι and use its shape coefficients.
3. **Optimization loop:** Use an Augmented Lagrangian approach around a derivative-free optimizer:
4. Initialize penalty factors for constraints small, and Lagrange multipliers = 0.

5. While not converged (or iter < max_iter):
    - Run an optimizer (e.g., COBYQA or CMA-ES or NGOpt) to minimize E + (Σ λ_i * constraint_i)^T + ρ * Σ(constraint_i)^2. Here constraint_i = (A - A_target, δ - δ_target, ι - ι_target). This inner optimization should be run until it converges or hits evaluation limit.
    - Update multipliers and increase ρ.
6. In practice, one might rely on an existing ALM implementation or just penalize heavily after a few steps (since equality constraints we want basically zero tolerance).
7. Each inner iteration can be parallelized: have the optimizer propose, say, 10 candidate shapes at a time (some optimizers allow batch suggestion or just ask for points in parallel) and evaluate them concurrently.
8. Monitor progress: check how E is dropping and constraints are holding. If ι or δ starts to drift, increase their penalty sooner.
9. **Optional surrogate assist:** If ALM is slow, at each outer iteration we can train a local surrogate around the current best and use a simple gradient descent on it to propose a candidate for the next generation. This might catch a descent direction faster than random mutation.
10. **Result verification:** Once the optimizer shows diminishing returns (E not improving more than, say, 1e-3 in successive iterations), take the best shape and run a high-resolution VMEC (maybe more Fourier modes internally or tighter tolerance) to get accurate metrics. Ensure A, δ, ι are within allowed tolerance (like 1%). If not, we may do a small manual tweak: for example, if ι came out a bit low, slightly adjust the Fourier coefficient that is known to raise ι (like torsional perturbation) and re-run VMEC.
11. **Output:** The final Fourier coefficients for submission, with an expected elongation significantly lower than the baseline (the baseline's geometric problem score was 0.969 which corresponds to some elongation – we aim for even closer to 1.0, meaning even smaller elongation) [71] .

Recommended Approach: We recommend the ALM + NGOpt (Nevergrad) as the primary solver based on its success in the baseline [46] . Simpler alternatives like SciPy's constrained optimizers are likely to fail or get stuck [83] . If computational resources are limited, one might simplify the problem further by reducing the number of free parameters (e.g., restrict to 2–3 dominant Fourier modes to adjust elongation while keeping others fixed). This reduces search dimensionality and can be effective because elongation is often mostly influenced by certain low-order modes (like the $m=2,n=0$ mode for vertical elongation). However, fixing too many parameters might make hitting the ι target impossible. So a balanced parameterization (maybe ~6–8 parameters) is a good compromise.

Also, participants should trust the physics: If you need to reduce elongation, typically you want to reduce the $m=2$ deformation in the cross-section. If the shape has strong $m=2$ components, try lowering those; if it's too elongated at certain toroidal angle, maybe increase a shaping at a different toroidal phase to counteract. These intuitive adjustments can guide the search or at least provide good initial guesses for the optimizer.

Benchmark 2 Implementation (Simple-to-Build QI):

1. **Data analysis & initialization:** Examine the dataset for trends between Δ<sub>QI</sub> and L<sub>grad</sub>. Identify a few candidates that have relatively low Δ<sub>QI</sub> and relatively high L<sub>grad</sub>. Even if none is optimal, they provide hints (like perhaps a certain configuration family – e.g., configurations with more gently rippled surfaces – have better coil simplicity). Use one or two of these as starting shapes. If, for example, W7-X like cases are present

(QI but complex coils), and another case is less QI but simpler coils, perhaps we can interpolate between them.

2. **Feasibility check:** Ensure the starting shape obeys A, E, M constraints. If not, adjust it manually (e.g., scale the aspect ratio down if needed, or reduce elongation by averaging shape with its rotated version). This gives a feasible or near-feasible baseline.
3. **Optimization algorithm:** Use Augmented Lagrangian with NGOpt or CMA-ES to minimize objective = Δ<sub>QI</sub> / L<sub>grad</sub> (or an equivalent formulation). Within the objective evaluation:
4. Compute Δ<sub>QI</sub>, L<sub>grad</sub> via VMEC++ and post-processing (provided by evaluation code).
5. If any constraint (A, E, M) is violated, add a large penalty (e.g., 100 * (fraction of violation)^2) to the objective so those points are disfavored.
6. Optionally, handle constraints via ALM outside the optimizer loop as well (similar to Problem 1).
7. Because this objective is non-linear and possibly noisy, a population-based search is wise. NGOpt will handle this by testing multiple strategy. We could also explicitly use **CMA-ES** which is known to perform well on continuous optimization by fitting a Gaussian to good candidates. CMA-ES can be seeded with multiple initial points (our starting candidates).
8. **Machine learning integration:** After a few generations of the optimizer (or even before starting it), train a quick surrogate model: e.g., a small neural network that predicts Δ<sub>QI</sub> and L<sub>grad</sub> from shape parameters. Use ~1000 random shapes from dataset (or Latin hypercube around initial shape) as training data. This surrogate can predict the objective very fast. Use it to scan a broad region around the initial guess to see if there are obvious directions where objective improves. For example, one might fix all but one coefficient and vary it to see sensitivity. This can supplement our understanding of the shape-effects: maybe increasing a particular helical mode drastically lowers Δ<sub>QI</sub> but also lowers L<sub>grad</sub>, etc. Feed this insight into the optimization – e.g., give CMA-ES a hint by widening search distribution along beneficial directions.
9. **Generative proposal:** If feasible, use the classifier+GMM method from the paper to generate, say, 100 feasible samples (feasible meaning they meet constraints) [75]. Evaluate those and take the top 5 by objective as additional starting points or to inject into the population. Diversity is good – perhaps one of these samples has a coil-friendly field the optimizer hadn't considered.
10. **Run optimization to convergence:** Because ALM-NGOpt took ~34 hours for this problem on 96 CPUs [61], budget accordingly. One can run for a fixed number of iterations or until improvement plateaus. If using a smaller setup, consider running longer but at lower fidelity (e.g., fewer VMEC radial grid points) to get a result, then refine that result with a few high-fidelity evaluations and local tweaks.
11. **Coil verification:** As a sanity check, take the optimized plasma shape and run a simple coil design code (perhaps included in SIMSOPT or use FOCUS code if available) to see the coil shapes required. This is not part of the competition scoring, but it validates the "simple-to-build" claim. If the coils still look very contorted, it might indicate L<sub>grad</sub> metric alone doesn't capture everything (like coil torsion or coil length). If so, we might augment the objective with a secondary metric (though none is provided, some might consider minimizing normal field error complexity further). But likely L<sub>grad</sub> is a good proxy [84].
12. **Final submission:** Provide the shape with its metrics. We expect to significantly improve upon the baseline score (which was ~0.431) [71]. A higher score indicates either a lower Δ<sub>QI</sub>, higher L<sub>grad</sub>, or both. The baseline likely had to compromise (they achieved Δ<sub>QI</sub> ~ 8.61 with norm. violations ~0 in Table 3 [53]; presumably lower is possible). With

our approach, we aim to either get an even lower QI residual without losing coil simplicity, or the same residual with simpler coils.

Recommended Approach: Based on our exploration, a **hybrid global-local strategy** is best. Use a global solver to handle competing influences of QI vs coil simplicity, then use local methods to fine-tune. Also, **pay attention to constraint trade-offs**: the limits on A, E, M mean at optimum they might all be active (i.e., the solution might sit right at one of those bounds). For example, perhaps the best solution makes use of the full allowed mirror ratio M_max and full allowed elongation E_max because those help lower Δ<sub>QI</sub> significantly [55]. If so, the optimal design will be a *boundary point* in constraint space. This is tricky for optimizers because they have to find the right balance where constraints are just satisfied. ALM is suited for that (Lagrange multipliers will converge to values indicating the trade-off cost of each constraint). Participants should monitor the active constraints: if, say, your solution isn't using all allowed elongation (E way below limit), maybe you can increase elongation a bit (which might improve QI) until it hits the limit, thereby getting a better objective. Conversely, if a constraint is only barely violated, a slight manual adjustment can fix it post-optimization.

Benchmark 3 Implementation (Multi-Objective Optimization):

1. **Decide target sweep range:** Determine a reasonable range for aspect ratio and coil complexity to explore. From engineering perspective, maybe A from ~4 up to ~12 covers "compact" to "large" stellarators. For coil simplicity, L<sub>grad</sub> has units (perhaps normalized to major radius) – we might see values ranging from, say, 0.1 (very tight coils) to 0.5 (very simple coils) in dataset. Instead of targeting L<sub>grad</sub> directly, we just let that float as we optimize for it. So primarily decide on discrete A values for single-objective runs: e.g., 4, 6, 8, 10, 12 (if time permits, more points like 5, 7, 9, 11 for finer resolution).
2. **Initial designs for each target A:** For each chosen A, find or create a starting shape. If A=4, we need a small compact initial guess – maybe scale down a known design or take NCSX shape (if available in dataset or literature). If A=12, use W7-X like shape (it has A~11, quasi-isodynamic and stable). For intermediate A, perhaps interpolate shape or use a generic parameterized family (like start with a scaled version of one shape).
3. **Single-objective optimization per A:** Use ALM + NGOpt (or CMA-ES) to **maximize L<sub>grad</sub>** subject to fixed A and the physics constraints (QI, W, C). This means:
4. Keep aspect ratio at the target (perhaps by holding major radius fixed and adjusting minor radius accordingly after each perturbation).
5. Penalize any Δ<sub>QI</sub> > Δ<sub>QI,max</sub> (design not sufficiently QI), any W < 0 (no magnetic well), any C > C_max.
6. The objective to maximize L<sub>grad</sub> (or minimize -L<sub>grad</sub>) will push for simpler coils, but the constraints will prevent it from taking shape changes that ruin confinement or stability.
7. This might be even harder than Problem 2 because now QI must be near-perfect *and* stability must be achieved. If the optimizer struggles (e.g., always violating W), we may soften constraints gradually: e.g., first optimize without the W constraint to get a QI shape, then introduce W constraint and optimize further from that result (maybe need to include some pressure or adjust iota to get a well).
8. Another trick: sometimes adding a small artificial pressure in VMEC can create a well (since finite beta can induce a well if magnetic shear is appropriate). But since we are doing vacuum design, better to shape the boundary to include a well. This often means an outwards bump or higher rotational transform at edge. We can enforce well by including that in objective (like add term +heavy penalty if W < 0).

9. The turbulent transport proxy C is geometric; if a candidate violates it, likely it has a region of bad curvature compression. We might address that by adding specific shaping like a bump or shifting the plasma outwards in certain regions (to reduce compression). The optimizer with penalty on C will ideally find these.
10. Use moderately large populations (like 50) for the search because the feasible region might be small. Running on 96 cores, we can evaluate 50 at a time easily.
11. **Multi-objective refinement:** After these runs, we will have a set of designs at discrete A values. Now, run a **multi-objective GA (NSGA-II)** initialized with those designs plus some random ones around them, to see if it can discover any intermediate designs that dominate. NSGA might smooth out the curve between those points. Also, NSGA can adjust aspect ratio continuously, so maybe the true Pareto optimal A's are not exactly at our pre-chosen points. NSGA will find them if given a chance (it might e.g. find that A=9.5 yields a slightly better trade-off than both A=8 and A=10 cases).
12. Ensure NSGA-II considers constraints: we implement a rule where any individual violating constraints is considered dominated by any feasible individual. We could also implement constrained tournament selection (e.g., Deb's rules for constrained GA).
13. Use surrogate evaluation for NSGA for at least initial generations. For example, train surrogate models for W and C on data from our previous runs (which gives some sense of how shape affects them). If an NSGA individual is predicted infeasible by surrogate with high confidence, we skip actual evaluation to save time.
14. Evaluate and update NSGA population for maybe 20 generations.
15. **Pareto set assembly:** Combine the results from (3) and (4). Filter them to only the non-dominated ones. That is our candidate Pareto front. Evaluate each one with the highest fidelity (maybe more toroidal modes in VMEC, high resolution).
16. **Hypervolume check:** Compute the hypervolume with the given reference point to see if adding any point in between or extrapolating beyond improves it. For instance, if our lowest A point is A=4 and highest coil simplicity is at A=12, maybe hypervolume could improve if we add an even higher A=14 point (but perhaps A=14 is beyond practical and maybe not allowed if aspect ratio wasn't bounded – presumably it's not bounded above except by coil constraints and diminishing returns). If HV can improve by an extreme, we should consider it. Perhaps adding a trivial point like A=12 with super high L<sub>grad</sub> (if physically possible) could boost HV on one end. Or a very low coil complexity (high 1/Lgrad) at maybe A=3 might boost HV on the other end, but A=3 might be unachievable with QI+stability. We rely on physical intuition and surrogate: if surrogate says at A=14, W7-X scaled up, L<sub>grad</sub> could be extremely high and still QI stable, maybe it's worth including. But likely A=12 is already near that asymptote.
17. **Final selection of points:** We should end up with perhaps 5–7 points that nicely cover the curve. We might drop any point that is very close to another (dominance-wise) because it won't add HV. The challenge might have a limit on number of designs per submission – if so, we choose the best-spaced ones.
18. **Submission and interpretation:** Provide the set. For interest, we might analyze this Pareto set: e.g., how ι profiles or shape differ between the low-A design vs high-L<sub>grad</sub> design. Such analysis isn't required for the competition but could yield insights. For example, maybe the low-A design has more transform (more coils) to achieve stability, or uses a different field period number.

Recommended Approach: We strongly recommend **tackling multi-objective problems with multiple runs and an ensemble of methods**. Solely relying on a single NSGA-II run from scratch could be too slow and miss the mark. Breaking it into subproblems (fixing A or fixing coil complexity) and solving those gave the baseline at least some solutions [72]. We extend that with more points and then refine with NSGA-II to ensure nothing is left on the table. It's also recommended to **use HPC parallelism** heavily here – multi-objective needs many evaluations, so a cluster or at least multi-core machine is needed to do it in reasonable time.

One also should consider the **sensitivity of constraints**: if one of our final designs is only barely stable (W just above 0), small changes could break it. It might be beneficial to include a small safety margin (e.g., require W > 0.001 instead of > 0) to avoid submissions that teeter on the edge (since numerical error or slight difference in evaluation might mark it infeasible). Similarly, ensure Δ<sub>QI</sub> is well below threshold if possible, to be safe. A dominated stable design is still better than an unstable one which doesn't count at all.

Finally, as a reality check, imagine presenting these designs to stellarator engineers: A very compact stellarator that is QI and stable would be groundbreaking (it might have more complicated coils though – but that's the trade-off, it's one end of Pareto). A very coil-simple stellarator that is stable and QI would be like a "simplified W7-X" (bigger device but easier coils). Both are valuable paths. Depending on whether one prioritizes cost or performance, different points might be chosen for a real reactor. Our job in this benchmark is not to pick one, but to map out the possibilities.

Concluding Recommendations:

Through our deep dive into each benchmark, a clear overall strategy emerges: **integrate physics understanding with machine learning tools to navigate the stellarator design space efficiently.** We highlight the following recommendations for any team tackling these challenges:

- **Exploit the Dataset's Knowledge:** The provided dataset is a rich resource encoding decades of physics insight (through the QI optimizations used to generate it). Always start by analyzing and mining this data – it can guide where to search and help train surrogates that drastically speed up evaluations [75]. Blindly running physics codes without this prior will waste time. For example, use PCA on the dataset to find a low-dimensional subspace where most good designs lie, and restrict your optimization to that subspace (this cuts out a lot of bad/unphysical shapes).

- **Balance Objectives and Constraints via Penalties or Multi-stage Optimizations:** Many of these problems have conflicting requirements. Approaches like Augmented Lagrangian (which gradually enforce constraints) have been proven effective [47] [46]. We recommend using ALM for benchmarks with strict constraints (Problem 2 and 3). For multi-objective (Problem 3), break it into manageable pieces and then combine. It's easier to handle two objectives at a time than five things at once; our plan isolates physics constraints via penalty and focuses on the main objectives in the outer loop.

- **Use HPC and Parallelism:** Stellarator optimization is computationally heavy; the baseline used ~30–90 CPU-hours for single solutions [61]. Our approach, with added ML, offsets some cost but will still require substantial compute, especially for multi-objective. We advise using parallel VMEC evaluations extensively (the evaluation scripts likely support multiprocessing). If available, use a cluster to run multiple optimization instances in parallel (e.g., different initial seeds or different fixed parameters) and then take the best result.

- **Validate and Iterate:** Always validate the final candidates with independent means if possible. For instance, run a higher-resolution VMEC or a different equilibrium code (like an initial-value MHD stability code or SPEC) to ensure the solution is truly stable and quasi-isodynamic. The scoring uses

VMEC++, so that's primary, but a design that's only marginally stable in VMEC might actually be unstable under a slight change. If we found one design to be sensitive, we might attempt a small robustness optimization: e.g., maximize the stability margin (W) without losing much coil simplicity, to ensure a safety buffer.

- **Leverage Modern ML Libraries:** Tools like PyTorch and JAX can be game-changers if used wisely. Training a neural net surrogate can turn a problem that requires 1000 physics evaluations into one that requires 100 (for training) + a few for final validation. JAX could even enable autodiff through simplified physics for gradient-based refinement. We encourage experimenting with these – for example, differentiating through a simplified coil calculation to directly adjust shape for coil simplicity could augment the strategy.

- **Human insight matters:** While automation and ML are powerful, human insight into the plasma physics should guide the setup. For instance, understanding that to get a magnetic well one might need to increase ι or indent the plasma can inform the parameterization or initial guess for Problem 3. Recognizing the patterns of quasi-isodynamic fields (like the need for mirror wells with poloidally closed contours) helps in setting up objectives and interpreting results. A purely black-box approach might produce a solution, but a human-in-the-loop can often diagnose why an optimizer is failing (e.g., always violating a certain constraint) and then adjust the strategy (maybe the parameterization doesn't allow what's needed for that constraint – add more freedom, etc.).

In conclusion, the ConStellaration Fusion Challenge benchmarks cover a comprehensive set of requirements for a fusion-ready stellarator: from pure geometry shaping to advanced physics-performance and engineering trade-offs. Our plan addresses each with a combination of *rigorous optimization algorithms* and *data-driven model assistance*, underpinned by *stellarator physics principles*. By following this plan, one can systematically approach each benchmark, satisfying all constraints and optimizing the objectives to achieve solutions that **surpass the baselines** and advance the state-of-the-art in stellarator design. The convergence of fusion physics and machine learning in this challenge exemplifies how next-generation tools can accelerate progress towards practical fusion energy [85] [86] – participants in this challenge are not only competing for high scores, but also contributing to the innovative design methodologies that could one day **shape the first commercially viable stellarator power plant**.

Sources: This report is based on the ConStellaration challenge description and dataset documentation [87] [88], the accompanying research preprint [26] [19] [89], and baseline optimization results reported by Proxima Fusion [43] [46]. The strategies combine insights from these sources with established techniques in optimization and ML for engineering design [47] [75]. All calculations and proposals adhere to the problem definitions and publicly provided evaluation criteria [39] [29].

---

[1] [2] [3] [4] [5] [6] [8] [12] [13] [14] [36] [50] [54] [85] [87] Proxima and Hugging Face Announce the ConStellaration Challenge: Help Us Optimize the Future of Fusion Energy with Machine Learning
https://www.proximafusion.com/press-news/proxima-and-hugging-face-announce-the-constellaration-challenge-help-us-optimize-the-future-of-fusion-energy-with-machine-learning

[7] [86] Bringing Fusion Down to Earth: ML for Stellarator Optimization
https://huggingface.co/blog/cgeorgiaw/constellaration-fusion-challenge

9 10 11 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 51 52 53 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 89 [2506.19583] ConStellaration: A dataset of QI-like stellarator plasma boundaries and optimization benchmarks
https://ar5iv.labs.arxiv.org/html/2506.19583

88 proxima-fusion/constellaration · Datasets at Hugging Face
https://huggingface.co/datasets/proxima-fusion/constellaration