applied sciences

MDPI

Article

FilterForge: An LLM-Based, Semi-Automated Agentic VS Code Extension for Microwave Bandpass Filter Design

Hüseyin Nuri Gülmez [1,*], Yunus Koç [2], Agah Oktay Ertay [3], Bora Döken [1] and Mesut Kartal [1]

[1] Faculty of Electrical and Electronics Engineering, Istanbul Technical University, Sariyer 34485, Istanbul, Türkiye; dokenb@itu.edu.tr (B.D.); kartalme@itu.edu.tr (M.K.)
[2] Informatics Institute, Istanbul Technical University, Sariyer 34485, Istanbul, Türkiye; kocyun@itu.edu.tr
[3] Faculty of Engineering and Architecture, Erzincan Binali Yildirim University, Yalnizbag Campus, Erzincan 24002, Erzincan Province, Türkiye; aoertay@erzincan.edu.tr
* Correspondence: gulmezh@itu.edu.tr

Abstract

We present FilterForge, a chat-driven VS Code environment that pulls the synthesis, analysis, simulation, and optimization stages of microwave bandpass filter design, normally coordinated by hand across tools written in different languages, into one workflow. A deployed Model Context Protocol (MCP) server exposes deterministic Python implementations of coupling-matrix synthesis, uniform predistortion, topology reconfiguration, a genetic-algorithm transmission-zero selector, a mode-matching engine for *H*-plane iris-coupled rectangular waveguide geometries, and a skill that generates PyAEDT/HFSS notebooks for various dimensioning design-curves. A language-model orchestrator turns natural-language requests into typed tool calls, while every reported quantity stays inside the deterministic kernels, so the numerics remain reproducible and model-agnostic. We evaluate the call layer on a 45-task benchmark across the five tool categories: gemini-3-flash reaches 96.3% tool-selection and 94.8% full-call accuracy with an 88.9% pass^3 rate, which an ablation traces to the curated tool-selection prompt rather than to raw model capability. The mode-matching engine is validated against full-wave HFSS on a six-pole 4 GHz Chebyshev filter tuned from the chat panel, and on an 8 GHz WR-112 counterpart taken end-to-end with no engineer in the loop, where a deterministic critique gates each round until a manufacturable geometry is reached. We then exercise the full workflow on two folded six-pole WR-90 cross-coupled filters at 10 GHz, a high-selectivity design synthesized against a stop-band mask and a group-delay-equalized variant whose positive cross-coupling uses a pair of side-wall irises, the latter settling to a peak-to-peak in-band group-delay ripple below 1.5 ns while recovering the synthesized return loss.

Keywords: microwave bandpass filter; coupling matrix synthesis; mode-matching method; large language model; computer-aided design automation

Check for updates

Academic Editors: Alexander Barkalov, Larysa Titarenko and Kazimierz Krzywicki

Received: 24 May 2026
Revised: 21 June 2026
Accepted: 22 June 2026
Published: 25 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Demand for microwave hardware that delivers tight spectral selectivity at higher frequencies and power levels has risen sharply over the past two decades, largely driven by satellite communications [1,2]. Broadband constellations, Earth-observation downlinks, and tighter spectrum-allocation rules [3] push much of the filtering burden onto the payload itself. Payload filters must then combine narrow guard bands, low insertion loss, and high power handling within tight mass and volume budgets [4,5].

For high-power, low-loss filtering up to roughly 30 GHz, rectangular waveguide bandpass filters remain the standard choice [6,7]. They appear both as standalone front-end filters and as channel filters inside input/output multiplexers (IMUX/OMUX) [1,4]. Low loss, high unloaded quality factor, good power handling, and thermal stability are the properties that recommend them for space payloads [5]. When such a filter is well tuned in simulation, the post-production tuning cycles stay short [8].

Much of the modern filter-synthesis literature applies directly to waveguide implementations. Cross-coupled topologies [9,10] for sharper selectivity, extracted-pole synthesis [11,12] for independent transmission zeros (TZs), frequency-dependent couplings (FDCs) [13,14] for asymmetric responses, and predistortion for finite-Q losses [15,16] all transfer naturally to waveguide-based coupled-resonator filters. High-precision CNC machining and, more recently, metal additive manufacturing [17,18] have also loosened many of the geometric constraints that once limited usable topologies.

Even so, the conventional waveguide filter design workflow remains slow and demanding. A practitioner must combine expertise in coupling-matrix synthesis, full-wave EM simulation, optimization, and post-fabrication tuning, and must trade bandwidth against selectivity, return loss against volume, and quality factor against mass, usually on the basis of experience rather than formal rules [1,5,19]. Much of this knowledge does not transfer cleanly from one specification to the next, so progress depends on design intuition that is hard to formalize. The result is a persistent bottleneck even for experienced designers [19,20].

Large language models (LLMs) are now being applied across many areas of electrical engineering [21,22]. In microwave and RF design, surrogate models and neural-network-assisted optimization were already speeding up filter synthesis and EM-driven tuning before the recent LLMs [20,23,24]. LLMs add something different: they carry broad domain knowledge that, inside an *agentic* workflow, lets the agent take on part of the design work rather than only the scripting around it [25,26]. How large that part is depends on the tools the agent is given. In Section 5.2, the agent takes a system-level brief and reaches a verified design on its own. In the other examples, it acts as an assistant, and the engineer keeps the judgment calls. Making this work in practice takes more than a strong model. The LLM has to live inside a purpose-built agent, and the agent needs a curated set of tools behind a defined interface [27]. Given a sufficient toolset and a fast model such as GPT-4o, a complete design problem can then be run as an agentic workflow, as has already been done in EDA automation, analog circuit synthesis, and RF amplifier design [28–31]. Recent infrastructure such as Model Context Protocol (MCP) servers [32] and reusable agent skills enables architectures that move past ad hoc tool calling, and we build on these in what follows.

To the best of the authors' knowledge, this is the first end-to-end LLM-driven agentic environment for microwave filter design. As a controlled testbed, we use the *H*-plane iris-coupled rectangular waveguide bandpass filter, a topology whose synthesis and EM behavior are well understood; this lets us put the design loop itself under test rather than the underlying physics. We call the resulting environment *FilterForge*. It makes three main contributions:

1. *A unified, chat-driven design environment built as a VS Code extension.* FilterForge brings the full filter design flow, including synthesis scripts written in different languages, EM simulation, and optimization, into a single environment. The designer interacts with the agent through VS Code's integrated chat panel in natural language, with dedicated UI panels for filter-specific tasks. Every module operates on a shared in-environment representation, so the format-compatibility and data-shuttling prob-

lems that usually appear when information is moved between separate tools do not show up here.

2. *An MCP server that exposes the filter-design modules to the agent.* Following the Model Context Protocol, the server gives the agent structured access to the synthesis, simulation, and analysis routines, as well as project resources and context. The agent's reasoning is thus separated from the specific implementations of the underlying engineering software: tools can be swapped, extended, or upgraded without rewriting the agent. Five categories of tool are exposed:
   - *Synthesis*: generalized-Chebyshev $\{E, F, P\}$ polynomial generation, transversal coupling-matrix synthesis, cascaded trisections, mixed n-tuplets, and uniform predistortion.
   - *Reconfiguration and editing*: Givens-rotation reduction of a coupling matrix to a target topology, direct in-place editing of coupling matrix for rapid what-if experiments.
   - *Analysis*: S-parameter and group-delay computation from a coupling matrix, and mode-matching analysis of H-plane geometries (applicable to inline topologies only).
   - *Optimization*: genetic-algorithm-based filter order selection and transmission zeros placement.
   - *Export*: Touchstone `.s2p`, MATLAB 2024b `.mat` files, and LaTeX/TikZ code for direct data inclusion in manuscripts.

   The complete catalogue with tool signature is given in Appendix A, Table A1.
3. *A bundled* skill *for design-curve generation.* Beyond the MCP tools, FilterForge ships with a skill: a structured, YAML-formatted procedure the agent can invoke much like a tool, but whose body is a higher-level recipe rather than a single function call. The skill instructs the agent to build a Jupyter notebook that drives HFSS through PyAEDT 1.1.0, sweeps the relevant geometric parameters, calls the modal analysis over the sweep, and produces the design curves the practitioner uses to seed dimensional optimization. The skill is written in the format consumed by GitHub Copilot agents, so it is portable beyond *FilterForge* and can serve as a template for further filter-design recipes contributed by other researchers.

Beyond exposing tools, the MCP layer also keeps the stochasticity of a general-purpose LLM out of the numerics. The agent does not produce numbers from the model's probabilistic priors; it calls deterministic, domain-specific tools and reasons over their outputs. This gives the LLM a specialized, deterministic context for a task it was never trained on. Its general capabilities are still in play, but only for the higher-level decisions where they belong.

FilterForge is deliberately *semi-automated*. Synthesis, reconfiguration, mode-matching analysis, GA-based optimization, and the bundled skill are all reachable by the agent and can be chained into a single chat session. A few steps are intentionally left to the engineer: fine-tuning of the filter response, running the agent-generated notebooks for initial dimensioning, and full-wave EM analysis of cross-coupled geometries. The goal is to remove the repetitive scripting and data-shuttling from the workflow, not to push the engineer out of decisions that need judgment.

2. System Design and Implementation

2.1. Overall Architecture

FilterForge is built around three decoupled layers, shown in Figure 1: a *presentation* layer that the engineer interacts with, an *orchestration* layer that turns natural-language

requests into tool calls, and a local *execution* layer that carries out the numerical work of Section 3. The interfaces between layers are deliberately narrow, so any stage of the design flow, including synthesis, predistortion, reconfiguration, mode-matching analysis, and genetic-algorithm-based optimization, can be invoked on its own or chained into an end-to-end session under chat control.

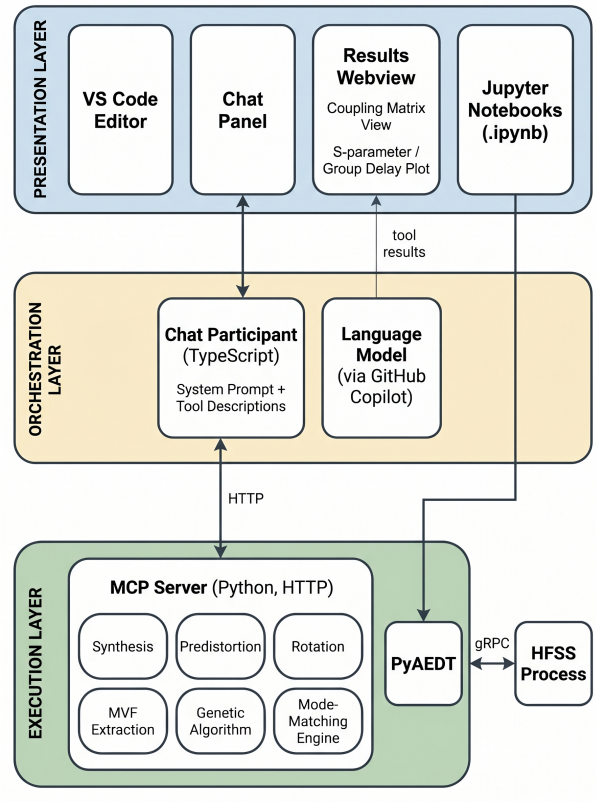


Figure 1. Three-layer architecture of FilterForge. The presentation layer (**top**) hosts the editor, chat panel, results webview, and notebooks. The orchestration layer (**middle**) is a chat participant that augments each user turn with a domain-specific system prompt and routes tool calls to a language model accessed through GitHub Copilot. The execution layer (**bottom**) is a local MCP server exposing the algorithms of Section 3; PyAEDT and HFSS run alongside it but are driven directly from the notebooks, so the full-wave EM stage remains optional.

The *presentation layer* is the only surface the engineer touches. It consists of the VS Code editor, the chat panel [33], a results webview that displays the current coupling matrix together with the corresponding S-parameter and group-delay response, and the bundled notebooks that drive HFSS through PyAEDT. The session state lives here: the current coupling matrix, its frequency response, the physical (f_0, BW) used for de-normalization, and the matrix origin (whether obtained by synthesis or manual edit). Because this state persists across chat turns, the engineer can synthesize a coupling matrix in one turn, ask the agent to rotate it to a folded topology in the next, and request a group-delay plot in a third without re-supplying the matrix.

The *orchestration layer* is a chat participant, registered under `@filterforge`, that augments every user turn with a domain-specific system prompt and forwards the result to a language model accessed through GitHub Copilot. The participant also serializes the model's natural-language tool requests into structured arguments before passing them to the execution layer. When a tool returns, the layer hands the numerical result back to the model for natural-language commentary, whose depth depends on how capable the underlying LLM is. No algorithmic work happens at this layer.

The model itself holds no hard-coded filter-design knowledge; all of it lives in the system prompt, which carries:

- the tool signatures exposed by the execution layer;
- the rules for moving between the normalized lowpass domain and the physical domain;
- the admissible coupling topologies and the conventions for transmission zeros;
- a terminology map for the synonyms common in the microwave-filter community.

Because the domain knowledge lives in the participant, smaller and cheaper LLMs already produce correct tool calls for routine synthesis and analysis. Stronger models mostly differ in how rich the surrounding commentary is.

The *execution layer* is a local Python 3.11 MCP server [32,34] that exposes synthesis, uniform predistortion, matrix reconfiguration, the GA-based transmission-zero selector, and the mode-matching engine as MCP tools, all implemented in Python. The server never sees the user's raw prompt; it only receives the typed arguments produced by the orchestration layer. Any irregularity on the user side is therefore handled upstream, and the server only ever sees a validated schema. An incomplete prompt such as `"order=6, return loss 20dB, no tz."` and a misspelled one such as `"predistorsion"` both reach the right tool without any change at the server. Every tool is *stateless*: all inputs are passed explicitly as arguments, and nothing is retained between calls. A separate set of design-curve notebooks, shipped with each FilterForge project and openly accessible to the user, drives HFSS through PyAEDT for the parameter sweeps of Section 4. These notebooks live next to the MCP server but rely on their own module instead of the server's tool implementations.

This layering buys two things. *Composability:* any stage of the pipeline can be entered on its own. The same matrix may be rotated to a new topology, edited by hand, or passed through the mode-matching engine without re-running synthesis. *Traceability:* every tool call and its arguments appear in the chat transcript next to the model's natural-language commentary, so the engineer can see both what was computed and how the agent interpreted it.

2.2. A Typical Session

Figure 2 shows FilterForge during an active synthesis dialogue, which makes the three-layer architecture easier to map onto what the engineer actually sees. The primary sidebar on the left holds the project entry points and two small status indicators: the API key and the MCP-server connection. At a glance, the engineer can tell whether the orchestration and execution layers are reachable. The central editor area is the surface of the presentation layer. It shows the current coupling matrix as an editable grid, with the corresponding S-parameter response plotted directly below; the two views stay in sync whenever a tool returns or the matrix is edited by hand. The chat panel that invokes the agent sits in the standard VS Code secondary sidebar, so an engineer already comfortable with the editor will not have to learn any new interface element.

Figure 3 traces a representative interactive synthesis as it unfolds in this chat panel. The user states the requirements in free form: a sixth-order filter with 20 dB return loss, a cascaded-triplet topology, and prescribed transmission zeros at $\Omega = 1.5j$ and $\Omega = 2.7j$. The agent does not call the synthesis tool right away. The topology rules in the system prompt tell it that triplet placement is not unique, so it first asks which trisection blocks should realize the two zeros. The options it lists are taken directly from those rules and not invented from prior knowledge. The user pins down the first triplet (R_1–R_2–R_3) for $\Omega = 1.5j$ and explicitly delegates the second to the agent. The agent then settles on R_3–R_4–R_5 for $\Omega = 2.7j$ and calls the synthesis tool, returning the resulting coupling matrix and a short summary of where the zeros ended up. The session shown runs on GPT-4o.

A stronger model produces the same tool calls and the same matrix on this example; only the surrounding commentary becomes longer and more discursive.

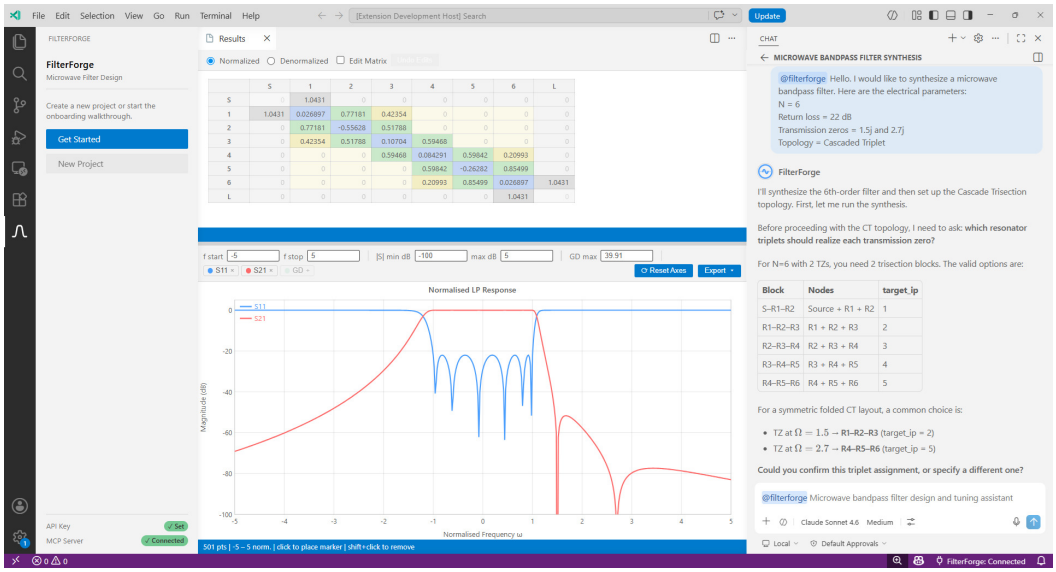
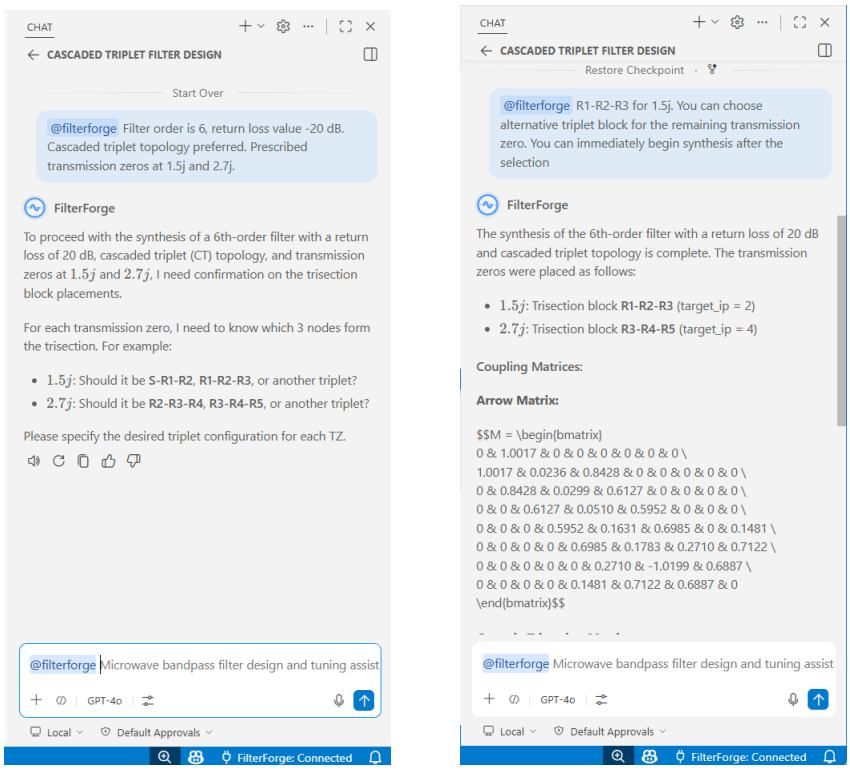


Figure 2. FilterForge during an active synthesis session. (**Left**): primary sidebar with project entry points and live API-key/MCP-server status. (**Center**): results panel with the editable coupling-matrix grid and synchronized normalized S-parameter plot. (**Right**): chat panel hosting the agent interaction.



(**a**) (**b**)

Figure 3. A typical interactive synthesis. (**a**) The user specifies the bandpass requirements; the agent, guided by the topology rules in the system prompt, asks which trisection blocks should realize the two prescribed zeros. (**b**) The user resolves one triplet explicitly and delegates the other to the agent.

The session need not end once the synthesized matrix is in place. Figure 4 shows a what-if experiment carried out in the results webview, without any further chat turn. Starting from the folded sixth-order matrix returned by the agent, the engineer enables the

Edit Matrix mode and sets the cross-coupling $M_{2,5}$ to zero. The webview re-evaluates the response on the fly and overlays it on the original, which is kept as a dashed reference for comparison. The two symmetric transmission zeros at $\pm 1.77j$ collapse, and the close-to-band selectivity degrades visibly. What that one cross-coupling was doing for the response is now visible on the screen. The in-place editing tool exists for this kind of by-hand probing: change a single matrix entry and watch how the response moves against the synthesized baseline.

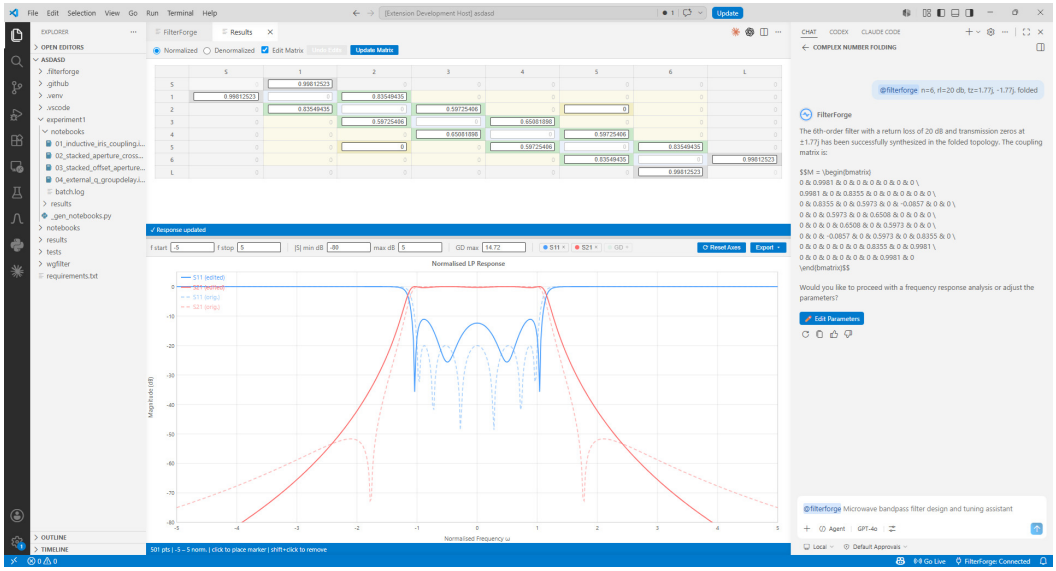


Figure 4. Live editing of a coupling coefficient in the interactive view. A sixth-order folded prototype with two finite transmission zeros is loaded into the grid, and the cross-coupling $M_{2,5}$ is then set to zero. The edited response is overlaid on the original one, and both transmission zeros vanish from S_{21}, as expected.

2.3. Evaluating the Agent Layer

FilterForge keeps the filter algorithms in deterministic MCP tools and leaves the language model one job: turning a free-form engineering request into the correct tool call with the correct arguments. Whether the synthesis polynomials, coupling matrices, and mode-matching responses are themselves computed correctly is a separate question, checked against reference results elsewhere in the paper; here, we test only the call layer.

From a designer's standpoint, that layer matters for two distinct reasons. First, the agent must pick the right tool out of the five categories of Section 2: a request for a Chebyshev synthesis must reach the synthesis tool, not the mode-matching analysis. Or, with the right tool chosen, it may mistranslate qualitative wording (*sixth-order*, 20 *dB return loss*, *transmission zero at* $\Omega = 1.5j$) into the typed arguments the tool expects. A wrong tool produces no result; a wrong argument silently produces the wrong filter.

2.3.1. Benchmark and Metrics

The benchmark has 45 single-turn tasks across the five tool categories of Section 2; the breakdown is in Table 1. Each task is a (prompt, *gold label*) pair, where the gold label is the tool call we would expect a competent designer to dictate: the tool name and the key arguments it should carry. A representative entry looks as follows.

Prompt: "`Synthesise a sixth-order generalised-Chebyshev filter with 20 dB return loss and a transmission zero at` $\Omega = 1.5j$."
Gold label: tool = `coupling_matrix_synthesis`, arguments = `{filter_order: 6, return_loss_dB: 20, transmission_zeros_s: [re:0, im:1.5]}`.

A run is graded by comparing the tool call the agent emits against this label. Prompts are written in both English and Turkish, the two languages the editor is used in.

Thirty-seven tasks are *canonical*: fully specified prompts an experienced designer would phrase without ambiguity. The remaining eight are stress cases of three kinds. *Perturbed* prompts are terse or misspelled versions of canonical tasks (`order = 6, return loss 20 dB, no tz`; `predistorsion`), each paired with a clean twin so the loss from noisy phrasing can be read off directly. *Clarify-expected* prompts are under-specified, where the right response is to ask back rather than call a tool with invented arguments, such as a cascaded-trisection request that does not say which resonators form the trisection (Section 2.2). *Decline-expected* prompts are out of scope and should be refused, such as a mode-matching analysis of a cross-coupled filter, which the engine does not support.

Table 1. Agent benchmark composition. The perturbed/edge column covers incomplete, misspelled, and synonym prompts together with clarify- and decline-expected tasks.

Tool Category	Tasks	Canonical	Perturbed/Edge
Synthesis	16	11	5
Reconfiguration	10	8	2
Analysis	9	8	1
Optimization	6	6	0
Export	4	4	0
Total	45	37	8

Each task is scored on two metrics, plus a repeatability check.

Tool-selection accuracy (TSA) is whether the agent picked the correct tool, or, on a clarify- or decline-expected task, correctly held back from calling one. It captures routing alone. A wrong tool either errors out or, worse, returns a plausible but irrelevant result, so a high TSA is the precondition for everything downstream.

Robustness (Δ_{rob}) is the change in full-call accuracy on the perturbed prompts relative to their clean twins, reported in percentage points. A value near zero means the agent reads a terse or misspelled prompt as well as a clean one; a large negative value means noisy phrasing breaks the call.

Full-call accuracy (FCA) is stricter metric. The tool must be correct and every key argument must lie within tolerance. The tolerances are 0.5 dB on return loss, 1% on prescribed transmission-zero positions, and exact equality on integer arguments such as filter order. The 0.5 dB slack on return loss is far below any change that would matter for the filter, and in practice it was never used: every model passed exactly the requested value, so the score equals exact matching. For the worked example above, the FCA-correct response is

`coupling_matrix_synthesis(filter_order = 6, return_loss_dB = 20, transmission_zeros_s = [{re:0, im:1.5}])`. A call with `filter_order = 6` but `return_loss_dB = 18` passes TSA, since the tool is right, and fails FCA, since the return-loss argument is out of tolerance, and it would build a filter that misses the specification.

Each task is run three times. *Pass*^{3} is the fraction the agent gets fully right on all three. Since the tools are deterministic, the same arguments always return the same matrix, so any variation between repeats sits in the language-model call layer and pass^3 exposes it.

2.3.2. Results

GPT-4o, used for the illustrative session of Section 2.2, was withdrawn from the in-editor model picker of the GitHub Copilot subscription powering FilterForge before the

benchmark was run, following a Copilot policy change. The evaluation therefore covers the two compact models that remained reachable from the same in-editor chat.

Table 2 compares the two. gemini-3-flash picks the correct tool on 96.3% of tasks (TSA), gets the arguments within tolerance too on 94.8% (FCA), and repeats across three runs on 88.9% (pass^3). Routine synthesis and analysis requests therefore do not need a frontier model; a compact production model already clears the bar. claude-haiku-4.5 trails, with FCA at 78.5%.

The two diverge most on the perturbed prompts. gemini-3-flash handles the noisy versions about as well as the clean twins, while claude-haiku-4.5 calls no tool on any of them, either asking back for parameters the tool does not need or replying conversationally without a call. On the clarify- and decline-expected tasks, both behave correctly: they ask which resonators form the trisection and refuse the unsupported cross-coupled mode-matching request, instead of fabricating arguments.

Table 2. Tool-calling accuracy across language models. TSA: tool-selection accuracy; FCA: full-call accuracy (tool and all key arguments within tolerance); Δ_{rob}: FCA change on perturbed versus canonical prompts; pass^3: fraction solved on all three repeats.

Model	TSA (%)	FCA (%)	Δ_{rob} (pp)	Pass^3 (%)
gemini-3-flash †	96.3	94.8	+16.7	88.9
claude-haiku-4.5	80.7	78.5	−100.0	73.3

† reference model.

Table 3 traces where the accuracy comes from. Stripping the tool-selection rules out of the system prompt drops the FCA of the reference model from 94.8% to 86.7%; the lost cases are almost all genetic-algorithm tasks, where the model otherwise reaches for a deprecated optimization tool the rule exists to forbid. Leaving only the bare tool schemas drops FCA to 82.2%, and the model also stops clarifying and declining when it should. So the agent layer, the curated system prompt and tool-selection rules around the language model, is what makes the workflow reliable, not the model on its own: a model that already calls tools competently does noticeably better once the domain rules are in place. The benchmark, the grader, the evaluated system prompt, and the raw per-model captures are archived together with the packaged extension in a single Zenodo artifact; Tables 2 and 3 regenerate from the captures with no installation and no API key.

Table 3. System-prompt ablation on gemini-3-flash. Removing the tool-selection rules, then all domain guidance, separates the contribution of the agent layer from raw model capability.

System-Prompt Variant	TSA (%)	FCA (%)
Full system prompt	96.3	94.8
Tool-selection block	87.4	86.7
Bare tool schemas	82.2	82.2

3. Theoretical Background

3.1. Coupling Matrix Synthesis and Topology Reconfiguration

An order-N bandpass filter with prescribed return loss RL, fractional bandwidth, and a set of finite-frequency transmission zeros is synthesized in the normalized lowpass domain ($s = j\Omega$, passband edges at $\Omega = \pm 1$) from the characteristic polynomials [1,10]. The lossless prototype scattering parameters are

$$S_{11}(s) = \frac{F(s)}{E(s)}, \qquad S_{21}(s) = \frac{P(s)}{\varepsilon E(s)}, \tag{1}$$

with F and P of degrees N and n_{fz}. The roots of F and P are the reflection and transmission zeros, respectively. E is a strictly Hurwitz polynomial of degree N, and ε sets the equiripple return-loss level via

$$\varepsilon = \frac{1}{\sqrt{10^{RL/10}-1}} \left| \frac{P(s)}{F(s)} \right|_{s=j}. \tag{2}$$

The polynomials F and P follow from the generalized Chebyshev filter function

$$\frac{F(\Omega)}{P(\Omega)} = \cosh\left[\sum_{k=1}^{N} \cosh^{-1}\left(x_k(\Omega)\right) \right], \qquad x_k(\Omega) = \frac{\Omega - 1/\Omega_k}{1 - \Omega/\Omega_k}, \tag{3}$$

where Ω_k is the position of the kth prescribed transmission zero ($\Omega_k = \pm\infty$ for zeros at infinity); the classical Chebyshev response $T_N(\Omega)$ is recovered when all transmission zeros lie at infinity. The polynomial E is obtained from $EE^* = FF^* + \varepsilon^{-2}PP^*$ by keeping its left-half plane roots.

Partial-fraction expansion of the short-circuit admittances about their common imaginary-axis poles $j\lambda_k$ realizes $\{E, F, P\}$ as a transversal network:

$$Y_{22}(s) = \sum_{k=1}^{N} \frac{r_{22,k}}{s - j\lambda_k}, \qquad Y_{21}(s) = \sum_{k=1}^{N} \frac{r_{21,k}}{s - j\lambda_k}. \tag{4}$$

The eigenvalues λ_k are the natural resonances of the N uncoupled transversal resonators, and the residues $r_{22,k}$ and $r_{21,k}$ set their source and load couplings. The $(N{+}2) \times (N{+}2)$ transversal coupling matrix $\mathbf{M}_T$ has entries

$$[\mathbf{M}_T]_{S,k} = \sqrt{r_{22,k}}, \quad [\mathbf{M}_T]_{k,L} = \frac{r_{21,k}}{\sqrt{r_{22,k}}}, \quad [\mathbf{M}_T]_{k,k} = -\lambda_k, \tag{5}$$

with all remaining entries zero; here S and L denote the source and load ports. The scattering response is then given by [10]

$$S_{11}(s) = 1 + 2j\,[\mathbf{A}^{-1}]_{1,1}, \;\; S_{21}(s) = -2j\,[\mathbf{A}^{-1}]_{N+2,1}, \;\; \mathbf{A} = s\mathbf{W} - j\mathbf{R} + \mathbf{M}_T, \tag{6}$$

with identity matrix $\mathbf{I} = \mathrm{diag}(1, 1, \ldots, 1, 1)$ and resistance matrix $\mathbf{R} = \mathrm{diag}(1, 0, \ldots, 0, 1)$.

Although $\mathbf{M}_T$ is canonical, with every entry uniquely fixed by the specification, its transversal interconnection is rarely manufacturable: each resonator couples directly to both ports, with no inter-resonator paths, whereas physical waveguide layouts admit only a limited set of coupling routes. The reconfiguration stage transforms $\mathbf{M}_0$ into a matrix $\mathbf{M}_1$ that produces the same response (6) but whose non-zero entries lie within a manufacturable target topology. $\mathbf{M}_1$ is obtained from $\mathbf{M}_0$ by an orthogonal similarity that fixes the source and load nodes [10],

$$\mathbf{M}_1 = \mathbf{T}^T \mathbf{M}_0 \mathbf{T}, \quad T_{i,i} = T_{j,j} = \cos\theta_r, \quad T_{i,j} = -T_{j,i} = \sin\theta_r, \tag{7}$$

where $\mathbf{T}$ is a chain of Givens rotations $\mathbf{T}^{(i,j)}(\theta)$ with $S, L \notin \{i, j\}$. Each rotation annihilates one designated pivot entry of the matrix. Because zeroing one coupling generically reintroduces another, the rotation sequence must follow a *pivot schedule* specific to the target topology. The practical target topologies for H-plane waveguide hardware are the inline, folded, extended-box, and cascaded extracted-pole forms, illustrated in Figure 5.

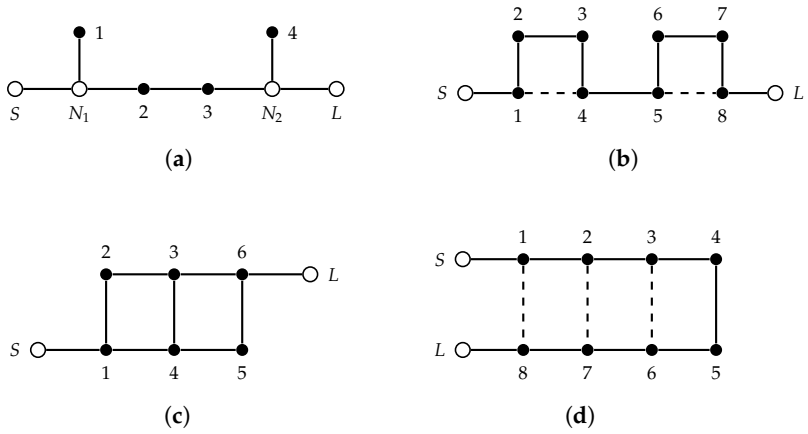


Figure 5. Coupling-routing diagrams of the manufacturable topologies discussed in the text. Filled circles denote resonators; hollow circles denote the source S, load L and non-resonating nodes N_i; solid lines are main-line couplings and dashed lines are cross couplings. (**a**) Cascaded extracted-pole topology ($N = 4$, two NRNs): each non-resonating node carries a dangling resonator that places one transmission zero independently of the main path. (**b**) Cascaded-quadruplet topology ($N = 8$): a cross coupling between the first and last resonator closes each four-resonator group and produces one prescribed transmission-zero pair per quadruplet. (**c**) Extended-box topology ($N = 6$): a two-path routing between adjacent box cells supports complex-axis zeros. (**d**) Folded canonical topology ($N = 8$): three cross couplings accommodate up to $N - 2$ prescribed transmission zeros.

The synthesis pipeline is illustrated below on two representative cases: a folded canonical example and a cascaded-quadruplet example.

3.1.1. Example Application 1: Ku-Band Satellite Channel Filter with Group-Delay Equalization

An example prompt that reproduces this case is available through the Prompt Cheat Sheet panel of the FilterForge extension shown in Appendix E. A channel filter for a Ku-band satellite payload is synthesized in the folded canonical topology, with order $N = 6$, centre frequency $f_0 = 12.5$ GHz, fractional bandwidth FBW $= 1.2\%$, and return loss $RL = 22$ dB. The transmission-zero set combines one purely imaginary zero at $\Omega = -1.6j$, which sharpens the lower-band skirt against an adjacent channel, with a complex conjugate pair at $\Omega = \pm 0.92 - 0.18j$ that flattens the in-band group delay without adding rejection. The coupling matrix and synthesized scattering response are shown in Figure 6

3.1.2. Example Application 2: X-Band Radar Receiver Pre-Selector

A higher-order pre-selector for the down-conversion stage of an X-band radar receiver is synthesized in the cascaded-quadruplet topology, with $N = 8$, $f_0 = 10$ GHz, FBW $= 0.8\%$, and $RL = 20$ dB. Two imaginary transmission-zero pairs at $\Omega = \pm 1.4j$ and $\Omega = \pm 2.3j$, one per quadruplet, produce sharp symmetric rejection on either side of the passband and suppress strong out-of-band interferers ahead of the mixer. The resulting coupling matrix and scattering response are shown in Figure 7.

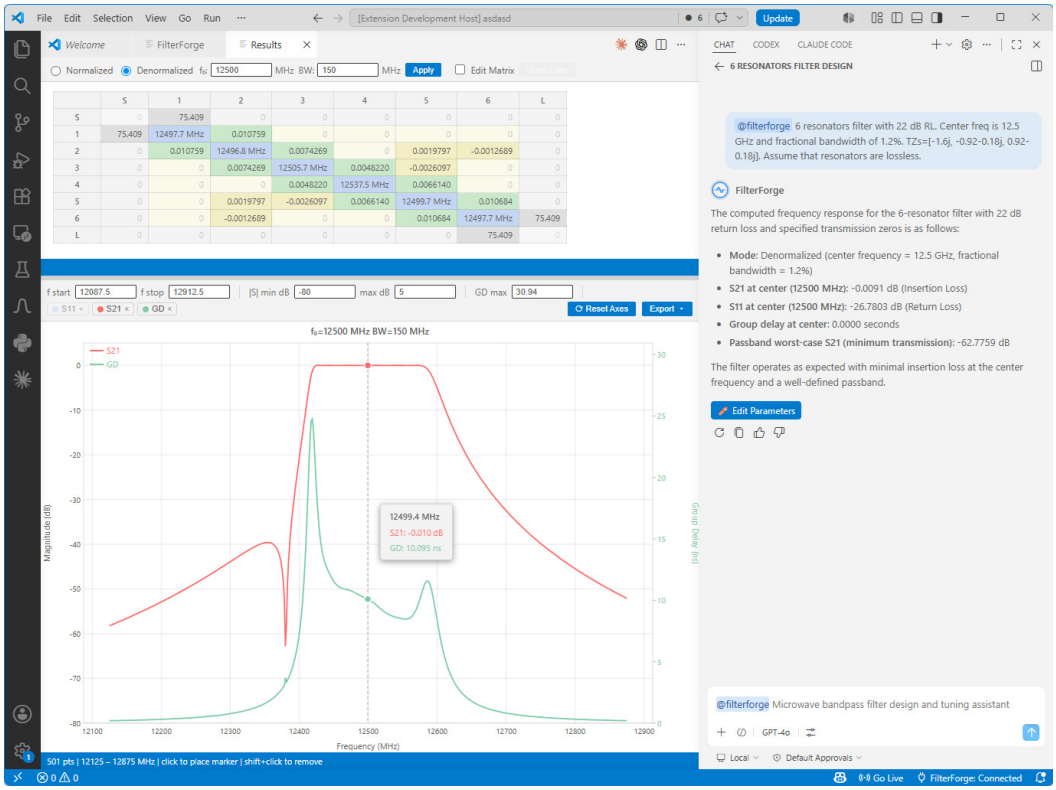


Figure 6. Synthesized scattering response and in-band group delay of the $N = 6$ Ku-band folded channel filter with one imaginary-axis and one complex-conjugate transmission-zero pair.

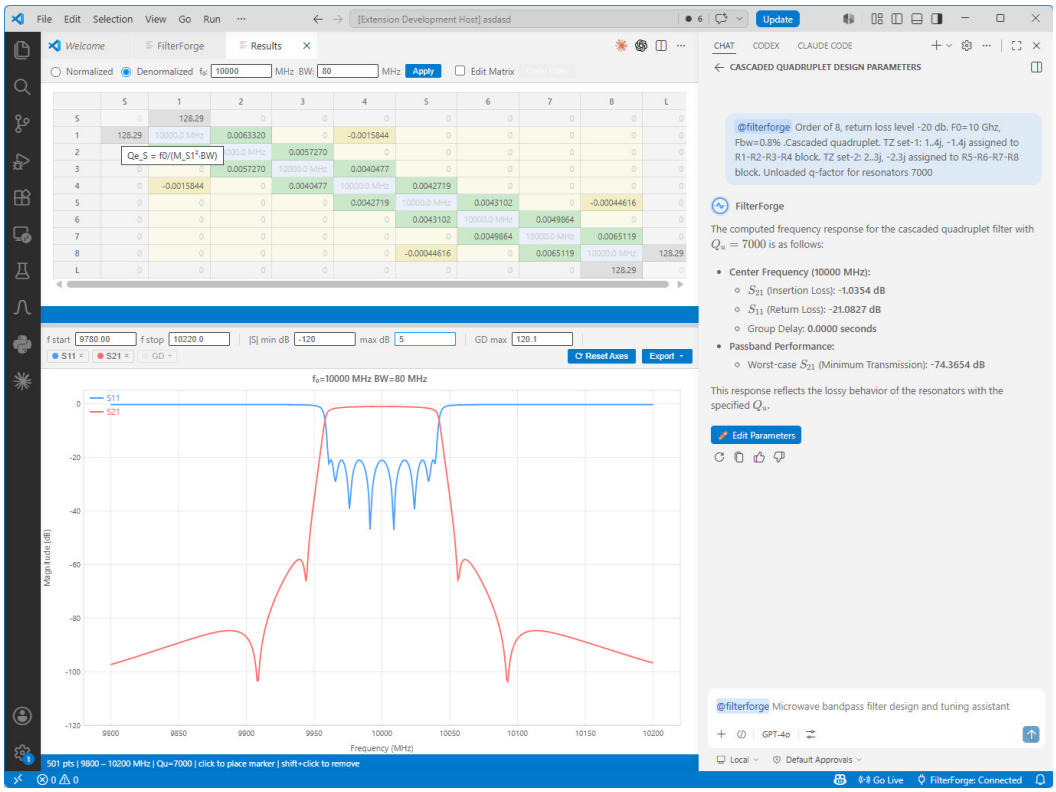


Figure 7. Synthesized scattering response of the $N = 8$ X-band radar pre-selector in the cascaded-quadruplet topology, with two imaginary transmission-zero pairs at $\Omega = \pm 1.4j$ and $\Omega = \pm 2.3j$. The symmetric quasi-elliptic skirts deliver sharp rejection on either side of the passband, suppressing strong out-of-band interferers ahead of the mixer.

3.2. Uniform Predistortion for Finite Q

A channel filter on a satellite payload must hold in-band amplitude and group delay flat to within a fraction of a decibel, while fitting the tight payload mass and volume budgets. The two requirements pull in opposite directions: a small, lightweight cavity carries a modest unloaded quality factor Q_u, and the resulting dissipation rounds off the equiripple corners predicted by the lossless synthesis of Section 3.1 and leaves the in-band return-loss notches shallow. The distortion is most pronounced for narrow fractional bandwidths $\Delta = \mathrm{BW}/\omega_0$ and high prescribed return-loss levels, precisely the regime of interest in this work.

For a uniform Q_u across resonators, the lowpass-to-bandpass transformation shifts every root of F, P, E to the left in the s-plane by a common amount,

$$s \longrightarrow s + \sigma_0, \qquad \sigma_0 = \frac{1}{Q_u\,\Delta}, \tag{8}$$

so that the equiripple condition holds on the displaced line $s = j\Omega - \sigma_0$ rather than on the imaginary axis itself. Predistortion synthesizes a deliberately *worse* target response, chosen such that this translation restores the specification [1,15]. Fully canceling σ_0 comes at a heavy in-band insertion-loss penalty, so in practice the compensation is only partial, toward an effective quality factor $Q_{\mathrm{eff}} > Q_u$:

$$\sigma = \frac{1}{\Delta}\left(\frac{1}{Q_u} - \frac{1}{Q_{\mathrm{eff}}}\right). \tag{9}$$

A representative choice from [1] is $Q_u = 4000 \rightarrow Q_{\mathrm{eff}} = 10{,}000$, which recovers the sharp passband corners and deep return-loss ripple of the lossless prototype with limited insertion-loss penalty.

In coupling-matrix form, σ enters as an imaginary contribution to the resonator self-couplings,

$$\mathbf{M} \leftarrow \mathbf{M} - j\,\sigma\,\mathbf{U}, \qquad \mathbf{U} = \mathrm{diag}\big(0, \underbrace{1, \ldots, 1}_{N}, 0\big), \tag{10}$$

and leaves the source and load entries unchanged. Because $\mathbf{U}$ commutes with every similarity transformation that fixes the source and load nodes, predistortion can be applied before or after the Givens reconfiguration of Section 3.1; both orderings produce the same physical network.

The predistortion shift (9) translates the roots of $E(s)$ and $P(s)$ uniquely, but those of $F(s)$ must be recovered afresh from the Feldtkeller relation and, unlike E, are not constrained to be Hurwitz. The auxiliary product $F(s)F^*(-s)$ is of degree $2N$ and has roots in mirror pairs $\{s_k, -s_k^*\}$ about the imaginary axis [1]. Any selection of one root per pair yields a valid $F(s)$, and the resulting 2^N candidates have identical $|S_{11}|$ but realize distinct physical networks. The selection is parametrized by [35]

$$\mu = \left| \sum_{k=1}^{N} \mathrm{Re}(s_k) \right|, \tag{11}$$

and the two extremes pose a manufacturing trade-off for even-order folded topologies, summarized in Table 4. The choice is left to the designer: predistortion enters the pipeline as an optional stage activated by the triple $(Q_u, Q_{\mathrm{eff}}, \mu)$.

Table 4. Manufacturing implications of the two extreme reflection-zero selections in even-order folded predistorted realizations.

	$\mu = 0$	$\mu = \mu_{\max}$
Coupling matrix structure	symmetric about physical center	asymmetric
Resonator self-couplings	non-identical	all zero (synchronous tuning)
Manufacturing implication	identical iris widths, distinct cavity sizes	identical cavity sizes, distinct iris widths

ExampleApplication: C-Band Satellite Payload Channel Filter

A ready-to-use prompt for this predistortion example can be retrieved from the FilterForge Prompt Cheat Sheet illustrated in Appendix E. A sixth-order folded predistorted filter is synthesized for a representative C-band satellite payload channel, with $f_0 = 4$ GHz, BW $= 36$ MHz, $RL = 20$ dB, and transmission zeros at $\Omega = \pm 1.6j$. The cavity unloaded $Q_u = 4000$ is compensated toward an effective $Q_{\text{eff}} = 6000$, and the reflection-zero selection $\mu = 0$ yields a coupling matrix symmetric about the physical center. The synthesized matrix and lossless scattering response are shown in Figure 8: both S_{21} and S_{11} are deliberately distorted, with the transmission bowed in-band and the reflection notches lifted well above the prescribed 20 dB. Figure 9 overlays the same response against a forward analysis at $Q_u = 4000$; cavity loss flattens the transmission and deepens the reflection notches back toward specification. The corresponding wider-span lossy response is shown in Figure 10, where the recovered passband incurs a 2.31 dB insertion-loss penalty at the center as the price of partial loss compensation toward $Q_{\text{eff}} = 6000$.

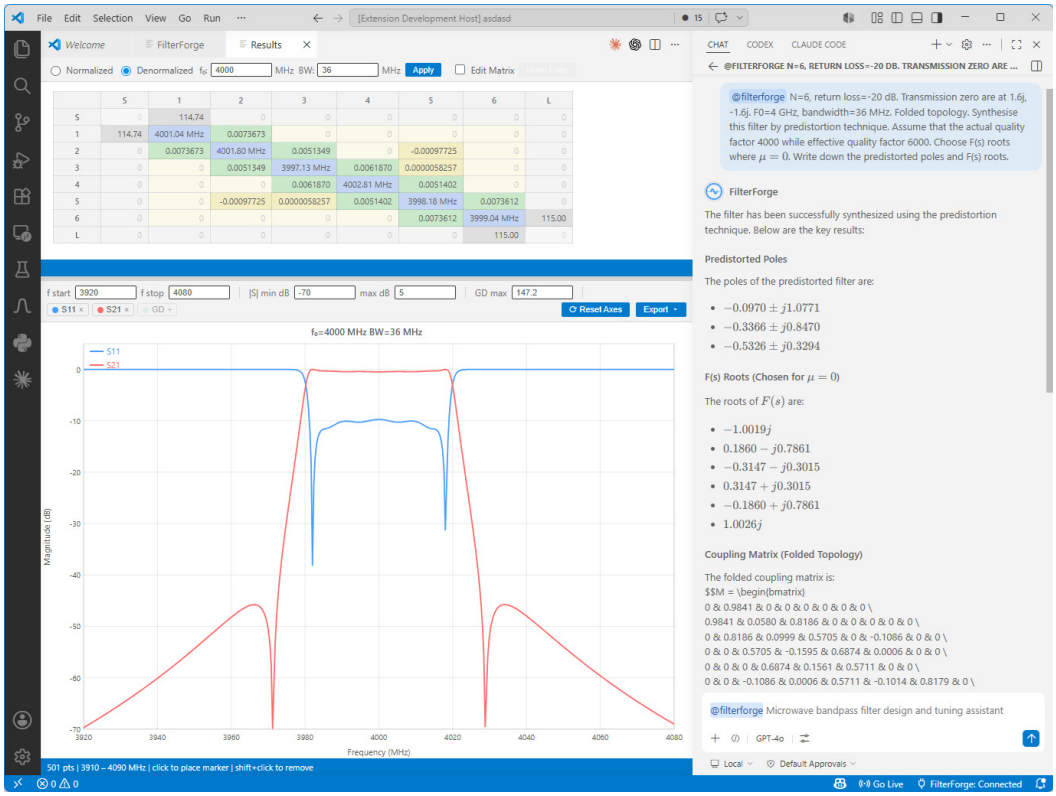


Figure 8. Synthesized coupling matrix and lossless scattering response of the sixth-order folded predistorted C-band channel filter ($f_0 = 4$ GHz, BW $= 36$ MHz, $RL = 20$ dB, zeros at $\Omega = \pm 1.6j$), with $Q_u = 4000$ compensated toward $Q_{\text{eff}} = 6000$ and reflection-zero selection $\mu = 0$. Both S_{21} and S_{11} are deliberately distorted: the transmission is bowed in-band and the reflection notches sit well above the prescribed 20 dB.

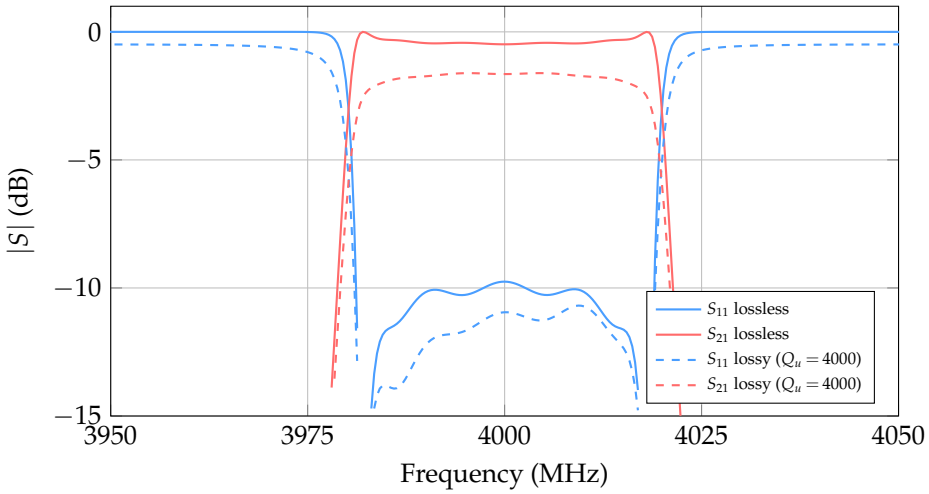


Figure 9. In-band close-up of the predistorted C-band filter, overlaying the lossless synthesized response (solid) with a forward analysis at $Q_u = 4000$ (dashed). Cavity dissipation flattens the in-band transmission and deepens the reflection notches a few dB back toward the 20 dB return-loss level.

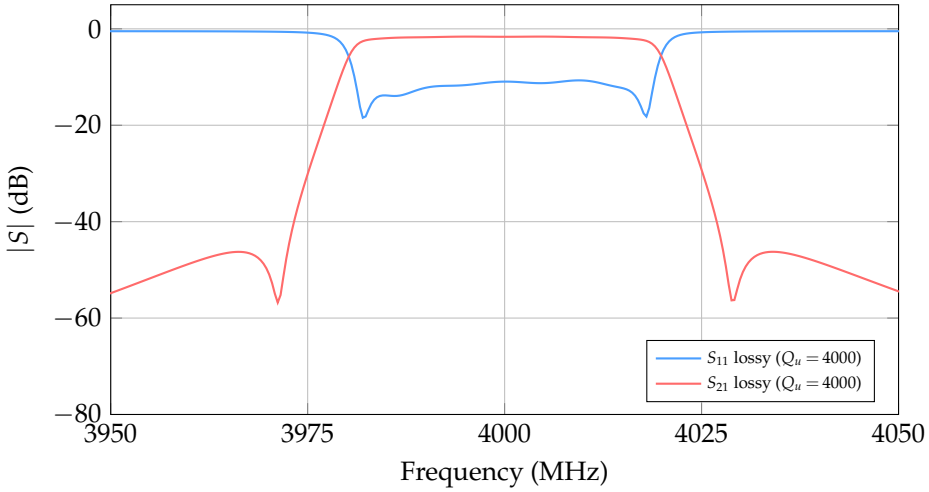


Figure 10. Frequency response view of the predistorted C-band filter under finite-Q operation ($Q_u = 4000$), showing the recovered passband. The insertion-loss penalty (2.31 dB) incurred at the passband centre is the price paid for the partial loss compensation toward $Q_{\text{eff}} = 6000$.

3.3. Genetic Algorithm for Transmission-Zero and Order Selection

The synthesis tools of mentioned in Section 3.1 take the filter order N and the finite transmission zeros $\{\Omega_{z,k}\}$ as inputs; they do not decide what these should be for a given stop-band specification. Pre-tabulated design charts cover the canonical cases (notably the unified design charts of Cameron [1], which relate order, bandwidth ratio, and rejection level for symmetric Chebyshev and quasi-elliptic responses), but only at the combinations enumerated in the source. For arbitrary stop-band masks set by the system engineer, Filter-Forge automates the choice with a dedicated optimization stage following the formulation of Qi et al. [36]: zero positions are found by a differential-evolution search [37], and N is fixed by an outer enumeration. The present implementation supports purely imaginary transmission zeros up to $n_{fz} = N - 2$ per design; complex-axis zeros (used for group-delay equalization) and fully canonical filters ($n_{fz} = N$) are out of scope.

For a candidate order N and transmission-zero count $n_{fz} \in \{1, \ldots, N-2\}$, the search varies the imaginary-axis ordinates $\{\Omega_{z,k}\}$ to minimize

$$\Phi(\{\Omega_{z,k}\}) = \sum_{i=1}^{n_{sb}} C_i \max_{\Omega \in \mathcal{S}_i} \left| S_{21}(\Omega; \{\Omega_{z,k}\}) \right|^2, \qquad C_i = 10^{(L_{s,i} - L_{s,\min})/10}, \tag{12}$$

where n_{sb} is the number of stopband attenuation specified, $\mathcal{S}_i$ is the ith stop-band region with required attenuation $L_{s,i}$, and S_{21} is evaluated from the generalized Chebyshev characteristic function (3) built from the candidate zeros. The weights C_i rescale every region to the deepest one, so that all attenuation targets enter the cost on the same logarithmic footing. An optional symmetry constraint reduces the search dimension by enforcing $\Omega_{z,k} = -\Omega_{z,N+1-k}$ for a symmetric response. The outer loop starts from a user-supplied N_{start} and increments N until at least one n_{fz} drives Φ below a feasibility threshold $\Phi^\star$ derived from the requested attenuations.

Within a chat session, the agent opens a dedicated webview, shown in Figure 11, in which the user enters f_0, BW and RL, and sketches the stop-band mask on a frequency-response template. Pressing *Optimize* executes the procedure of Figure 12 and overlays the result on the same template, as illustrated in Figure 13.

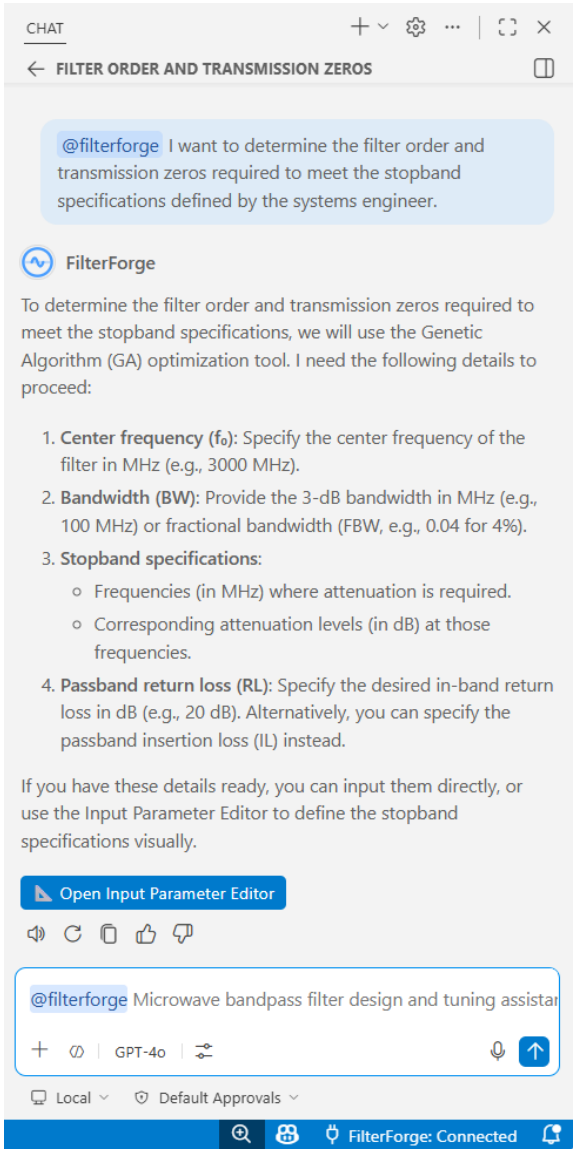


Figure 11. Chat request that opens the GA input parameter editor.

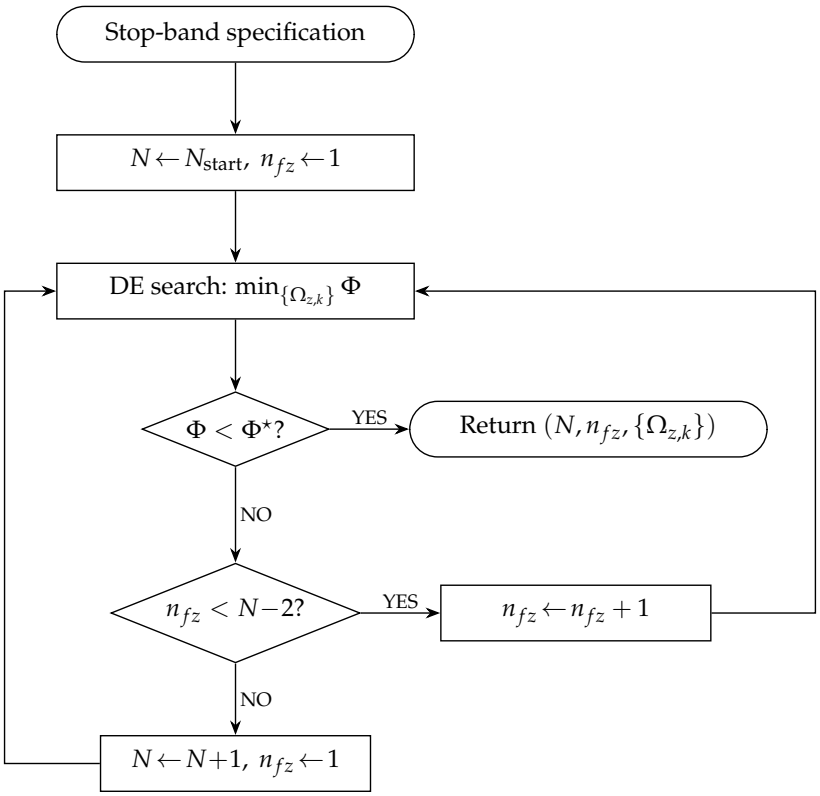


Figure 12. Outer-loop enumeration with inner differential-evolution search. The DE block returns the best $\{\Omega_{z,k}\}$ for the current (N, n_{fz}); if the resulting fitness falls below the feasibility threshold $\Phi^{\star}$ the configuration is accepted, otherwise n_{fz} is incremented or, on exhaustion, N is increased and the search restarted.

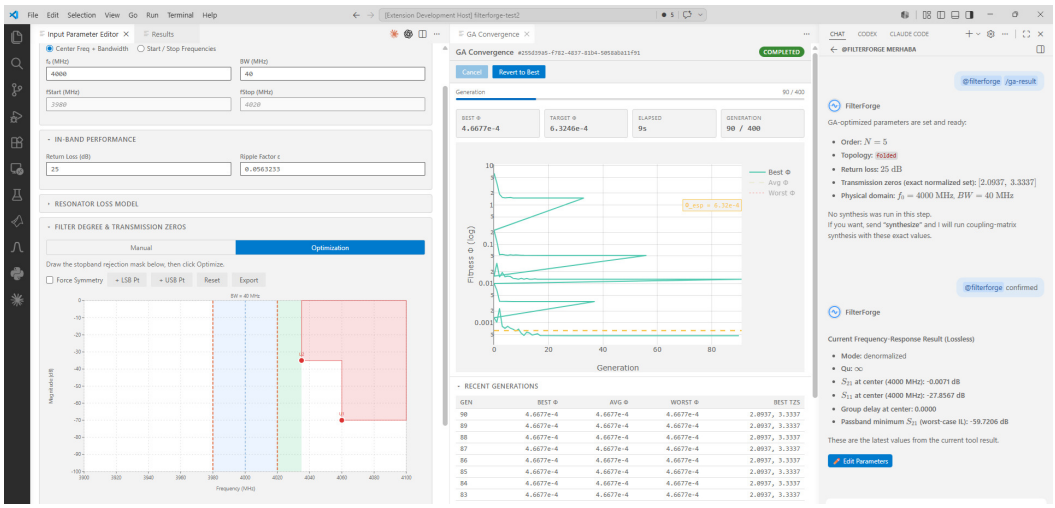


Figure 13. Input parameter editor with the interactive stop-band mask and the transmission zeros returned by the GA search.

ExampleApplication: Asymmetric C-Band Channel with Stop-Band Mask

Consider a C-band channel with $f_0 = 4$ GHz, BW $= 40$ MHz, and $RL = 25$ dB, together with an asymmetric stop-band mask of 35 dB at 4035 MHz and 70 dB at 4060 MHz. With *Force Symmetry* disabled, the outer enumeration settles on $N = 5$ and $n_{fz} = 2$, and the zeros fall at $\Omega_z = 2.0937$ and 3.3337. Under the control parameters of Table 5, the differential-evolution search drives the fitness down to $\Phi = 4.67 \times 10^{-4}$, below the feasibility threshold $\Phi^{\star} = 6.32 \times 10^{-4}$ implied by the requested attenuations. Only $D = n_{fz} = 2$ zero ordinates are optimized in this case, so the population is $15D = 30$ individuals, and the search clears

the threshold within about 90 of the 400 permitted generations (roughly 9 s). The resulting coupling matrix, Equation (13), produces the scattering response in Figure 14. At band center, $S_{11} = -27.9$ dB and $S_{21} = -0.007$ dB (near-lossless), which meets the prescribed mask and the 25 dB return-loss target with margin.

$$M = \begin{bmatrix} 0 & 79.5 & 0 & 0 & 0 & 0 & 0 \\ 79.5 & 4000.8\ \text{MHz} & 0.00976 & 0 & 0 & 0 & 0 \\ 0 & 0.00976 & 4001.2\ \text{MHz} & 0.00593 & 0.00352 & 0.00070 & 0 \\ 0 & 0 & 0.00593 & 3988.6\ \text{MHz} & 0.00551 & 0 & 0 \\ 0 & 0 & 0.00352 & 0.00551 & 4000.2\ \text{MHz} & 0.00974 & 0 \\ 0 & 0 & 0.00070 & 0 & 0.00974 & 4000.8\ \text{MHz} & 79.5 \\ 0 & 0 & 0 & 0 & 0 & 79.5 & 0 \end{bmatrix} \quad (13)$$

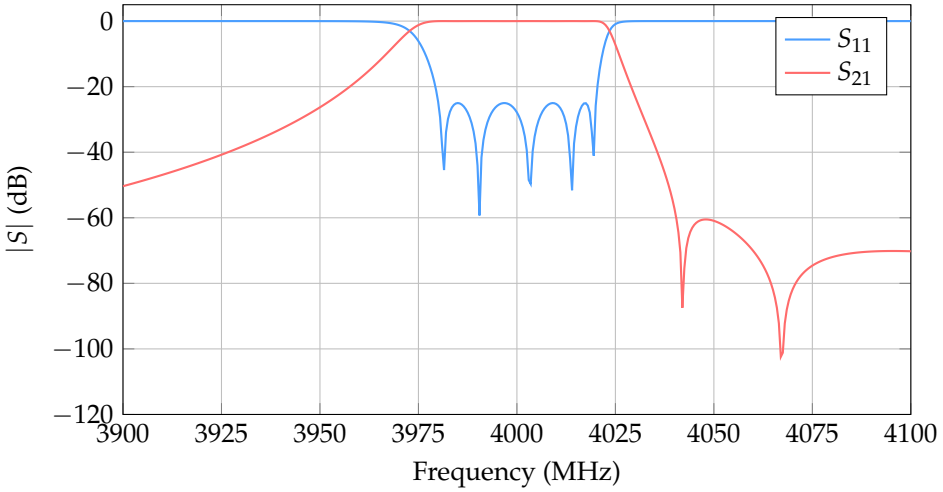


Figure 14. Scattering response of the coupling matrix synthesized from the GA-selected order and transmission zeros.

Table 5. Differential-evolution control parameters of the inner transmission-zero search (`scipy.optimize.differential_evolution`), identical for every (N, n_{fz}) tried by the outer loop.

Parameter	Value
Strategy	differential evolution, DE/best/1/bin
Population size	15 D (D = number of optimized zero ordinates)
Maximum generations	400
Mutation (differential weight F)	dithered, $F \sim \mathcal{U}[0.5, 1.5)$
Crossover probability CR	0.9
Initialisation	Latin-hypercube
Convergence tolerance	10^{-12} (then L-BFGS-B polish)
Random seed	42

4. Design-Curve Generation as an Agent Skill

The synthesized coupling matrix prescribes a set of normalized couplings $\{k_{i,i+1}\}$ and external quality factors $\{Q_e\}$. The last open task before full-wave verification is to turn these into physical iris and cavity dimensions. The procedure is well established [1,6,38]: a two-cavity eigenmode model isolates a single iris, a coarse sweep of the iris dimension produces a design curve with the swept dimension on the x-axis and the resulting coupling on the y-axis, and the dimension that realizes the synthesized coupling is then read off this curve at the target y.

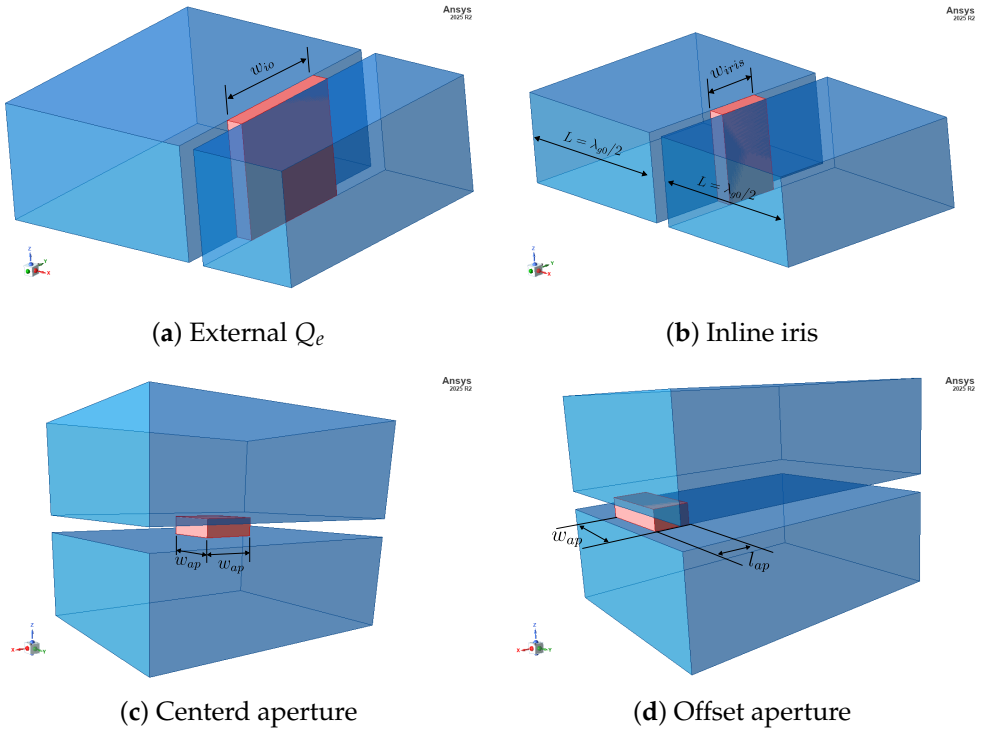
FilterForge exposes this workflow as an *agent skill* [39] rather than a deterministic MCP tool: instead of a fixed command, it is invoked in natural language through the `@filterforge` chat assistant by requesting design-curve notebooks for the desired cen-

ter frequency and waveguide cross-section, e.g., `@filterforge generate design-curve notebooks for WR-90 at 10 GHz, iris thickness 1.5 mm`. On invocation, the agent reads the skill's `SKILL.md` given in Listing 1, copies the relevant notebook templates, and re-parametrizes them against the user's licensed full-wave solver. The sweep range, EM setup, and polynomial fit stay in the generated notebook, where the user can edit them.

The skill bundles four parametric sub-structures, shown in Figure 15: the external coupling, the inline inter-resonator iris, and the two cross-coupling apertures of opposite sign. The coupling coefficient k is extracted with the eigenmode solver on the two-cavity model; its two lowest eigenfrequencies $f_1 < f_2$ give

$$k = \frac{f_2^2 - f_1^2}{f_2^2 + f_1^2}. \tag{14}$$

Because (14) is symmetric in f_1 and f_2, the cavity length can be held at its nominal $\lambda_g/2$ throughout the sweep: iris loading shifts both modes together and leaves k unbiased.



(**a**) External Q_e (**b**) Inline iris

(**c**) Centerd aperture (**d**) Offset aperture

Figure 15. HFSS sub-structures bundled with the `design-curves` skill. (**a**) Singly loaded cavity in driven mode, used to extract Q_e from the input group delay. (**b**) Inline H-plane iris between identical cavities in eigenmode, used to extract the direct coupling k. (**c**) Centered aperture, which produces capacitive (negative-sign) cross-coupling. (**d**) Offset aperture, which produces inductive (positive-sign) cross-coupling.

The external Q_e is extracted with the driven-modal solver on the singly loaded cavity, terminated in a wave port. The input group delay $\tau_{S_{11}}$ at the resonant frequency ω_0 gives

$$Q_e = \tfrac{1}{4}\,\omega_0\,\tau_{S_{11}}(\omega_0). \tag{15}$$

Here, ω_0 is the *loaded* resonance, which the input iris pulls well below the design center f_0 by an amount that grows with iris width. The notebook therefore sweeps a two-dimensional grid in iris width w and cavity length L, interpolates the length $L^\star(w)$ that brings the simulated resonance back to f_0, and reads Q_e at that length. Both $Q_e(w)$ and $L^\star(w)$ are

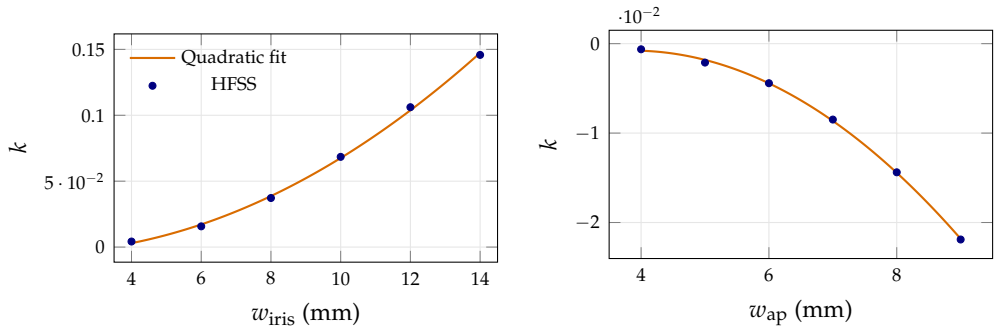
fitted with quadratics, so when the input iris width is read off at the synthesized Q_e target, the corresponding cavity length is read off at the same w in the same step.

All four sub-structures are simulated with perfect electric conductor walls and a lossless interior, which substantially reduces the per-sweep solve time. Each notebook sweeps the relevant iris dimension over a modest bracket around the synthesized target, fits a quadratic, and exposes a look-up call that returns the dimension corresponding to a requested coupling value on the curve. A degree-two polynomial is sufficient because the local dependence of k or Q_e on aperture geometry is smooth on the swept interval; the fixed order keeps the fit interpretable and the dimension read off the curve unique within the sweep range. Extrapolation is refused outright: a target outside the swept range raises an error rather than returning a silent guess.

Two practical caveats apply to the three k-extraction notebooks. Although k itself is unaffected by holding L at $\lambda_g/2$, each cavity in the assembled filter is loaded by two adjoining irises and resonates below f_0 unless shortened; an empirical per-iris correction $\Delta l(w)$ is therefore subtracted from the adjoining cavity lengths once the whole filter has been dimensioned. The cross-coupling notebooks further assume the sign of k by construction (capacitive negative for the centered aperture, inductive positive for the offset), and large geometric excursions can swap the two cavity modes and silently flip the extracted sign. When the fit looks anomalous, resolving the symmetric mode under perfect-electric and perfect-magnetic boundary terminations of the iris plane in turn fixes f_1 and f_2 unambiguously.

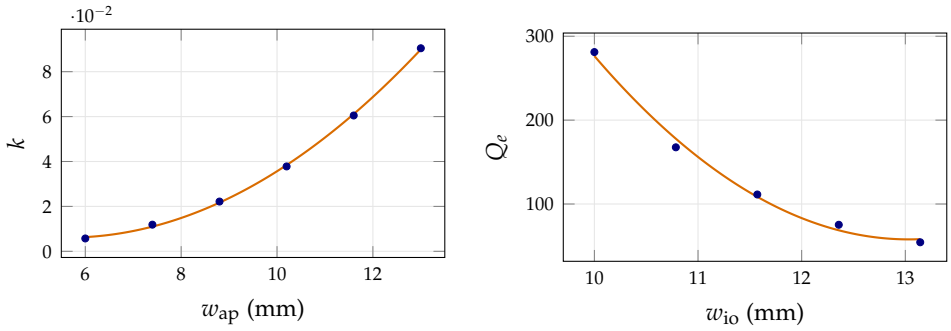
Example Application: Design-Curve Generation at 10 GHz in WR-90

The natural-language prompt used to drive this design-curve workflow is listed in the FilterForge Prompt Cheat Sheet reproduced in Appendix E. The skill was invoked on standard WR-90 waveguide ($a = 22.86$ mm, $b = 10.16$ mm) with an iris thickness of 1.5 mm, centered at 10 GHz. A single invocation generated the four curves of Figure 16: the inline inductive iris over $w \in [4, 14]$ mm, the centered square aperture over $w \in [4, 9]$ mm, the offset inductive aperture used in folded-section cross-couplings over $w \in [6, 13]$ mm, and the input/output iris parametrizing Q_e over $w \in [10.0, 13.1]$ mm. The four quadratic fits returned $R^2 \geq 0.994$ across their respective sweeps, so any subsequent filter centered near 10 GHz in this waveguide-thickness combination can be dimensioned by reading the required iris widths off these curves at their synthesized coupling targets, without rerunning the full-wave sweep.



(**a**) Inline inductive iris (**b**) Centered square aperture, $k < 0$

Figure 16. *Cont.*



(**c**) Offset inductive aperture, $k > 0$ (**d**) External Q_e from I/O iris

Figure 16. Design curves for WR-90 with 1.5 mm iris thickness at 10 GHz. Markers: HFSS samples; solid line: degree-two least-squares fit ($R^2 \geq 0.994$ across all four panels).

5. Mode-Matching Analysis of Cascaded Waveguide Structures

Mode matching (MM) is a fast and rigorous frequency-domain analysis technique for waveguide structures that decompose into uniform cylindrical sections joined by transversal discontinuities. Examples include H-plane iris-coupled bandpass filters, E-plane metal-insert filters, stepped impedance transformers, and the polarizer and OMT families [40,41]. The efficiency of MM rests on closed-form modal eigenfunctions in canonical cross sections, frequency-independent coupling integrals at each step, and a numerically stable cascade rule for the generalized scattering matrix (GSM) [41,42].

Across each uniform section, the transverse electric field expands in the TE, TM, and (where applicable) TEM modal eigenvectors $\mathbf{e}_n^{(g)}$ of the cross section [43],

$$\mathbf{E}_t^{(g)} \;=\; \sum_n \big(a_n^{(g)} + b_n^{(g)}\big)\, \mathbf{e}_n^{(g)}, \tag{16}$$

where $a_n^{(g)}$ and $b_n^{(g)}$ are the forward and backward modal amplitudes and the propagation constant $\gamma_n^{(g)}$ is imaginary for propagating modes and real for evanescent ones.

At a step between regions w and s with cross sections $A_w \supseteq A_s$, Galerkin testing of the tangential-field continuity on A_s together with the vanishing tangential electric field on $A_w \setminus A_s$ [41] gives

$$\mathbf{a}_w + \mathbf{b}_w \;=\; \mathbf{V}\big(\mathbf{a}_s + \mathbf{b}_s\big), \quad \mathbf{V}^T\big(\mathbf{a}_w - \mathbf{b}_w\big) \;=\; \mathbf{a}_s - \mathbf{b}_s, \tag{17}$$

where $\mathbf{V} = \mathbf{Y}_w^{1/2}\,\mathbf{C}\,\mathbf{Z}_s^{1/2}$ is the frequency-dependent dressing of the frequency-independent coupling matrix $\mathbf{C}$,

$$C_{ij} \;=\; \iint_{A_s} \mathbf{e}_i^{\,w} \cdot \mathbf{e}_j^{\,s}\, dA. \tag{18}$$

For rectangular, circular, coaxial, and elliptical cross sections, C_{ij} reduces to closed-form sinusoidal-integral or Bessel-function values at the modal cutoff wavenumbers [41,42]. Solving (17) for the outgoing amplitudes yields the GSM of the step.

The GSM of two structures A and B that share a common interface, illustrated in Figure 17, follows from the Redheffer star product [41]

$$\mathbf{S}^{AB} \;=\; \mathbf{S}^{A} \,\star\, \mathbf{S}^{B}, \tag{19}$$

which absorbs the intermediate modal amplitudes $\mathbf{a}_2, \mathbf{b}_2$ at the shared interface. A uniform connecting section of length ℓ enters the cascade through its diagonal propagation matrix

$$\mathbf{Y}(\ell) \;=\; \mathrm{diag}\big(e^{-\gamma_n \ell}\big). \tag{20}$$

Mode counts on either side of each step follow the fastest-convergence rule and are set by retaining all modes below a common cutoff frequency between ten and forty times the highest analysis frequency [41,44]. Losses are neglected, since the optimum lossless geometry coincides with the optimum lossy one [41]. FilterForge therefore exposes the MM engine in its lossless-PEC configuration.

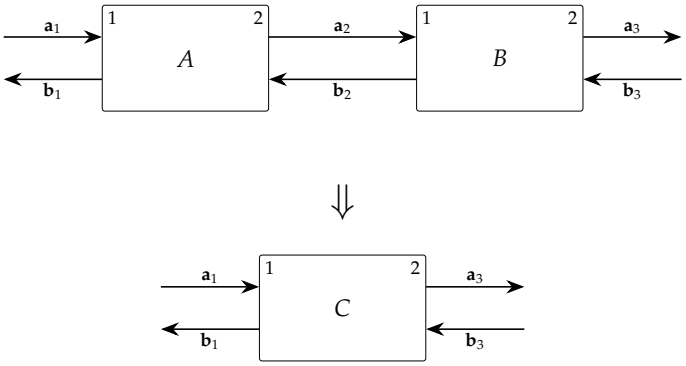


Figure 17. GSM cascade of two waveguide structures A and B that share a common interface, combined into a single composite structure C. The combination rule absorbs the intermediate modal amplitudes $\mathbf{a}_2, \mathbf{b}_2$ at the shared interface, so that C is described entirely by its external-port amplitudes $\{\mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_3, \mathbf{b}_3\}$. Uniform connecting sections enter the cascade as a third element through their propagation matrix $\mathbf{Y}(\ell)$.

5.1. Interactively Tuned H-Plane Iris-Coupled Inline Filter at 4 GHz

The canonical H-plane iris-coupled inline geometry, shown in Figure 18, is a chain of uniform broad-wall rectangular sections separated by thin inductive irises, and decomposes fully into the rectangular building blocks the MM engine supports. A six-pole instance is specified to FilterForge through the chat panel with $f_0 = 4$ GHz, BW $= 40$ MHz, $RL = 26$ dB, no transmission zeros, in $a \times b = 50 \times 25$ mm^2 rectangular waveguide with 2 mm-thick irises. The initial dimensions of Table 6 are obtained by reading each iris width off the design curves of Section 4 at its synthesized coupling target. Figure 19 shows the resulting overlay: the synthesized coupling-matrix response (dashed grey) and the MM analysis of the user-supplied geometry (solid) on the same axes. Each MM evaluation completes in well under a second, so the agent can request retuned dimensions and refresh the overlay within the conversation.

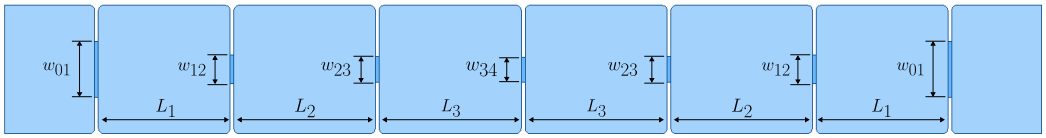


Figure 18. Top-view of the H-plane iris-coupled inline filter, with broad-wall dimension a, iris thickness t, iris widths $w_{k,k+1}$, and cavity lengths L_k labeled; the geometry is symmetric about the center.

A few rounds of agent-driven manual ad hoc adjustment produce the optimized dimensions of Table 6. To verify the MM engine itself against an independent reference, the optimized geometry was re-analyzed in HFSS; Figure 20 overlays the two S-parameter traces on the same axes. The analytical-MM and finite-element results agree across the passband and the close-in stopband to within the meshing tolerance of the latter.

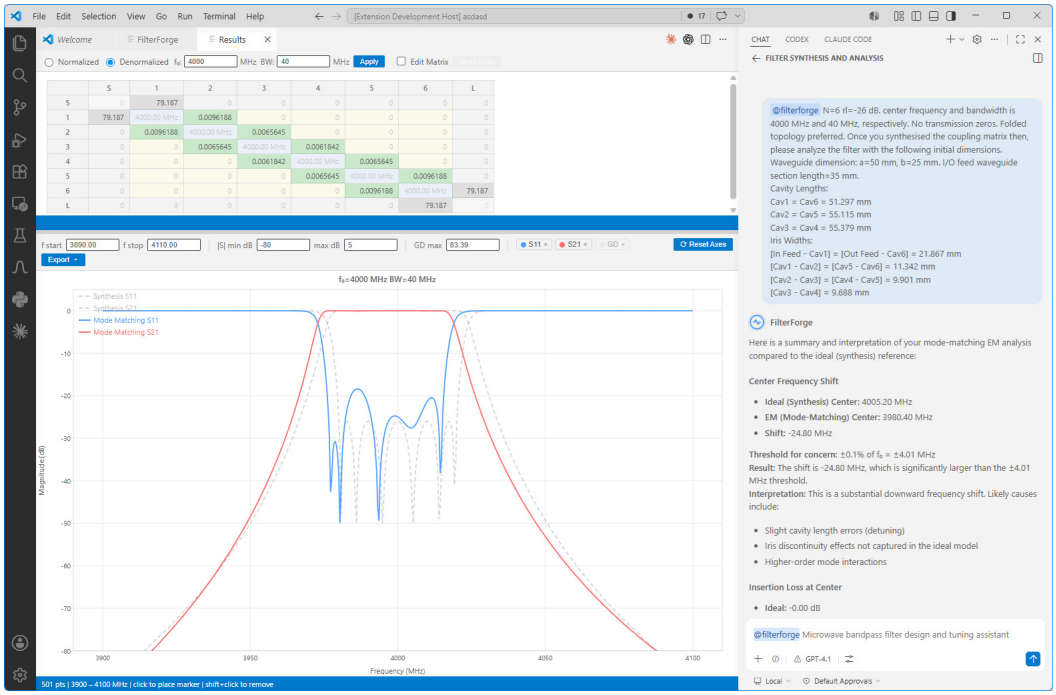


Figure 19. Interactive MM-driven tuning of the filter of Table 6 from the FilterForge chat panel: synthesized coupling-matrix response (dashed grey) and MM analysis of the user-supplied geometry (solid) on the same axes.

Table 6. Initial and optimized dimensions (mm) of the *H*-plane iris-coupled inline filter of Figure 19. The geometry is symmetric, so only the independent parameters are listed: cavity lengths L_k and iris widths w_k.

	$L_1 = L_6$	$L_2 = L_5$	$L_3 = L_4$	$w_{01} = w_{67}$	$w_{12} = w_{56}$	$w_{23} = w_{45}$	w_{34}
Initial	51.297	55.115	55.379	21.867	11.342	9.901	9.688
Optimised	51.430	55.112	55.369	21.373	10.990	9.618	9.244

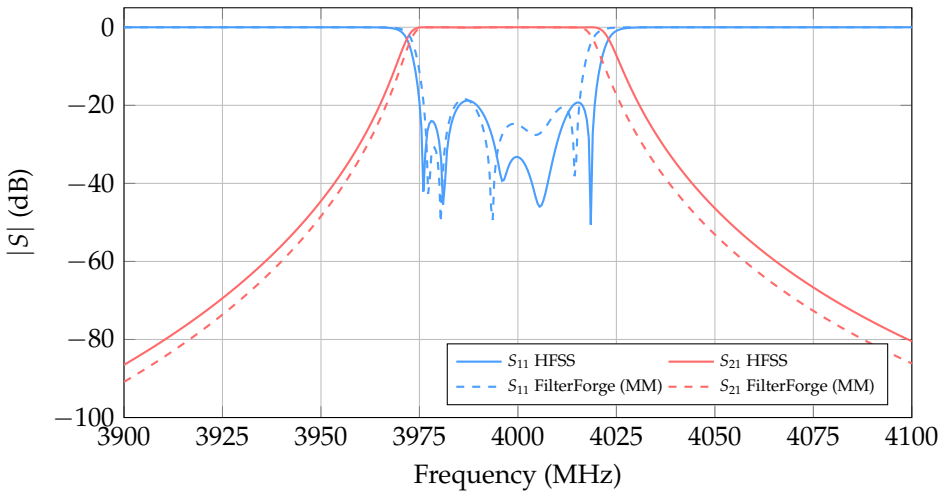


Figure 20. *S*-parameters of the optimized *H*-plane iris-coupled inline filter of Table 6: HFSS full-wave reference (solid) and FilterForge mode-matching analysis (dashed).

The MM engine is restricted by construction to geometries that decompose into uniform sections joined by concentric transversal steps. Bifurcations, offset cascades, tuning screws, rounded corners, and out-of-plane discontinuities fall outside this class and are handled through the HFSS/PyAEDT route of Section 4. For the *H*-plane iris-coupled topology,

however, the speed and rigor of MM close the loop between synthesis and dimensioning inside a single interactive session.

5.2. Agentically Designed H-Plane Iris-Coupled Inline Filter at 8 GHz

A ready-to-use prompt that triggers this agentic design workflow can be found in the FilterForge Prompt Cheat Sheet illustrated in Appendix E. The example is an H-plane iris-coupled inline filter centered at $f_0 = 8\,\text{GHz}$ with passband BW $= 300\,\text{MHz}$ and return-loss target $RL = 20\,\text{dB}$, in standard WR-112 waveguide with 1.5 mm iris thickness. The stopband mask requires at least 20 dB rejection at the symmetric offsets $f_0 \pm 320\,\text{MHz}$, and no transmission zeros are prescribed. Unlike the agent-assisted manual tuning of the previous case, here the *fully autonomous* design loop given in Figure 21 *system-level* brief, with the filter order left unspecified. Through the chat panel illustrated in Figure 22, the designer enters only the specifications above; the agent derives a starting order from the classical Chebyshev formula and raises it until the stopband mask is met.

Each round performs a closed-form K-inverter direct synthesis. With the TE_{10} guided wavelength and reactance slope parameter

$$\lambda_{g0} = \frac{\lambda_0}{\sqrt{1-(\lambda_0/2a)^2}}, \qquad x = \frac{\pi}{2}\left(\frac{\lambda_{g0}}{\lambda_0}\right)^2, \tag{21}$$

where $\lambda_0 = c/f_0$, the impedance inverters follow from the Chebyshev low-pass prototype $g_0, \ldots, g_{N+1}$ (fixed by the return loss):

$$K_1 = \sqrt{\frac{x\,\text{BW}}{g_0 g_1 f_0}}, \quad K_{j,j+1} = \frac{x\,\text{BW}}{\sqrt{g_j g_{j+1} f_0}}, \quad K_{N+1} = \sqrt{\frac{x\,\text{BW}}{g_N g_{N+1} f_0}}. \tag{22}$$

Each iris width is then found by a single-iris mode-matching search that matches the inverter target

$$|S_{21}(f_0)| = \frac{2}{K + K^{-1}}, \tag{23}$$

and the resonator lengths close the synthesis as

$$l_i = \frac{\lambda_{g0}}{4\pi}\,(\phi_i + \phi_{i+1}), \tag{24}$$

where $\phi_k = \angle S_{11}(f_0)$ is the converged iris reflection phase. The synthesized geometry is verified by a full-wave mode-matching (MM) sweep and checked against the specification by a deterministic evaluator; when a round fails, one bounded corrective action is selected according to the critique logic of Figure 21.

The loop converges in three rounds, traced in Table 7. Round 1 misses the in-band return-loss goal, so the synthesis return loss is tightened from 20 to 22 dB. This is a deliberate design margin: the direct synthesis rests on an idealized circuit model, so the realized iris-coupled filter matches slightly worse than the synthesis target, and requesting a tighter return loss makes the *realized* response meet the original 20 dB requirement. By Round 2 the return loss is met (20.02 dB), but the fourth-order filter reaches only 16.68 dB of rejection at $f_0 \pm 320\,\text{MHz}$, short of the required 20 dB, so the order is raised to $N = 5$. The fifth-order filter clears both criteria (20.60 dB in-band, 27.71 dB stopband) and is accepted. The order increment, not the return-loss margin, is what fixes the rejection: tightening the return loss sharpens the in-band match but does not steepen the skirts. The stopband rejection in fact drops slightly from Round 1 to Round 2 (18.61 $\rightarrow$ 16.68 dB), so only a higher order can satisfy the requirement. The designer never specifies the order; the agent determines the minimum order that meets the stated stopband. The measured passband center stayed

within tolerance throughout ($|f_c - f_0| < \tau$, with a largest deviation of 10.7 MHz), so the recentering branch of Figure 21 was not exercised in this run. Figure 22 shows the accepted design: the synthesized coupling matrix and the ideal response (dashed) overlaid with the full-wave mode-matching analysis (solid), alongside the realized geometry. Table 8 contrasts the initial fourth-order synthesis with the accepted fifth-order design; the order increment introduces the center resonator L_3 and slightly retunes the remaining sections.

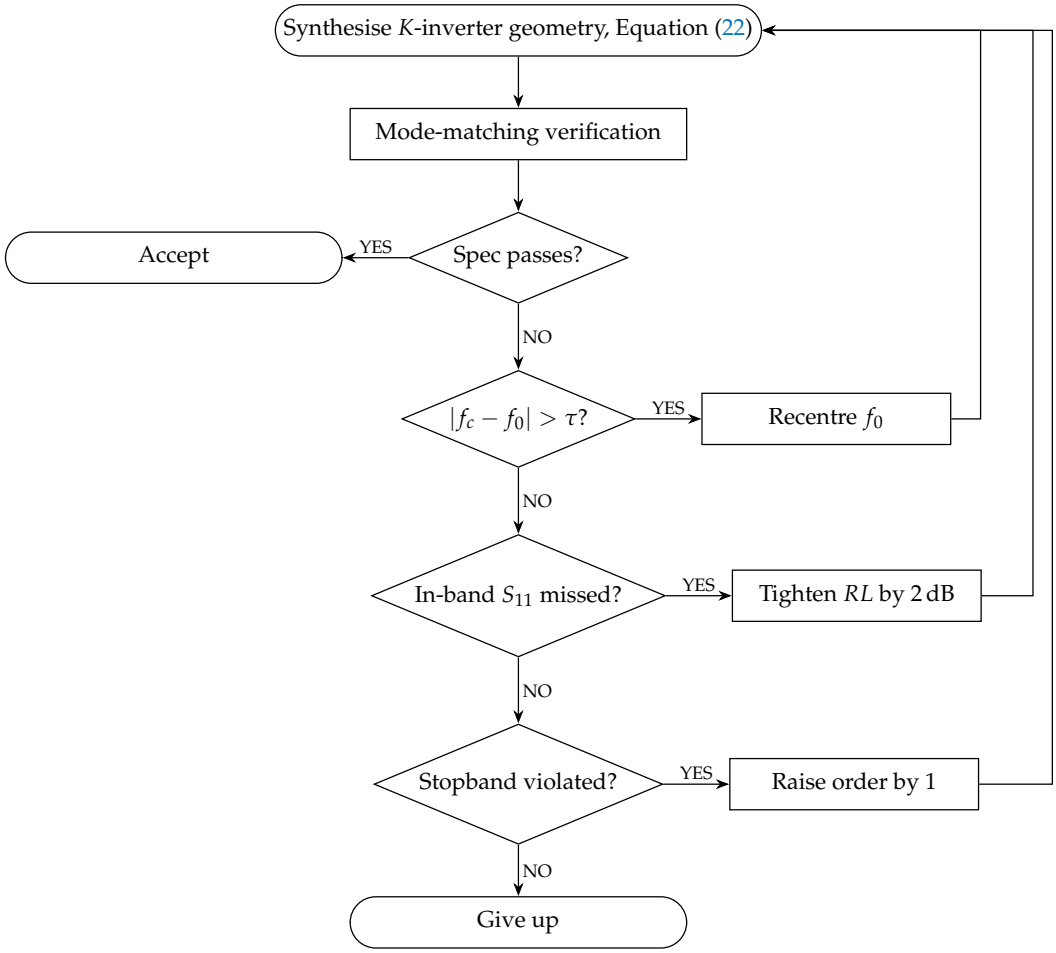


Figure 21. Deterministic critique applied once per round. f_c is the measured passband center and $\tau = 0.05\,\mathrm{BW} = 15$ MHz the center tolerance; the RL margin and the order increment are bounded to $+8$ dB and $+2$ above the original specification, respectively.

Table 7. Autonomous decision trace for the 8 GHz design.

Round	N	RL_{syn} (dB)	max $\lvert S_{11}\rvert$ (dB)	min Stopband (dB)	Action
1	4	20	−18.67	18.61	RL margin
2	4	22	−20.02	16.68	Order increment
3	5	22	−20.60	27.71	Accept

Table 8. Initial (Round 1, $N = 4$) and optimized (accepted, $N = 5$) dimensions (mm). The geometry is symmetric, so only the independent cavity lengths L_k and iris widths $w_{k,k+1}$ are listed, using the fifth-order labeling. The center resonator L_3 does not exist at the fourth order, so its initial entry is marked "–"; it is the resonator introduced by the order increment.

	$L_1 = L_5$	$L_2 = L_4$	L_3	$w_{01} = w_{56}$	$w_{12} = w_{45}$	$w_{23} = w_{34}$
Initial ($N = 4$)	20.693	22.893	–	13.687 *	8.817	8.065
Optimized ($N = 5$)	20.643	22.943	23.192	13.795	8.805	7.879

* At $N = 4$ only $w_{01} = 13.687$ applies; the iris w_{56} does not exist.

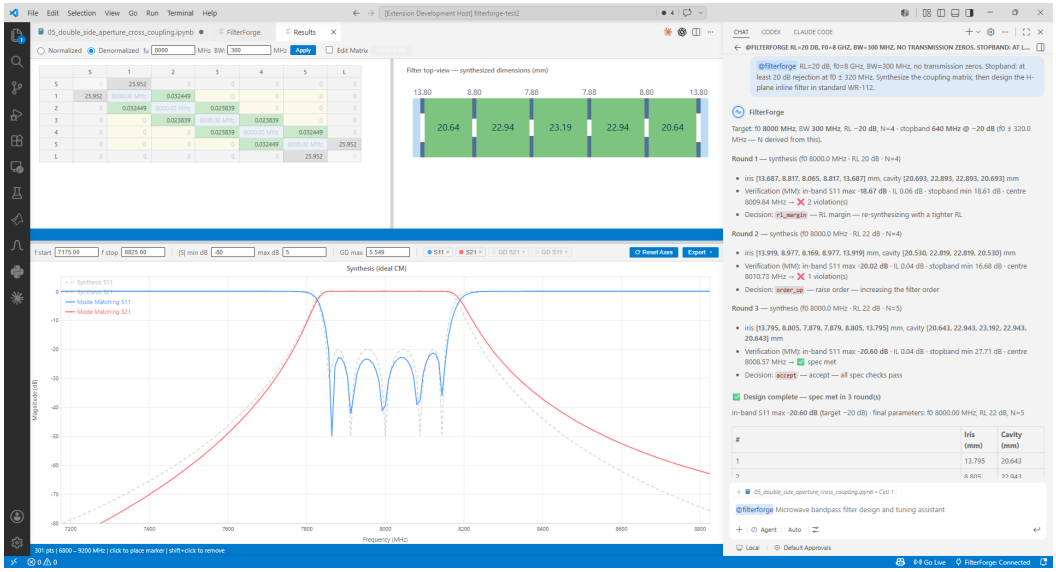


Figure 22. Chat-first agentic design of the 8 GHz WR-112 filter. (**Right**): the designer's natural-language, system-level specification (no filter order) and the agent's streamed per-round decisions (RL margin → order increment → accept). (**Left**): the resulting Results panel showing the synthesized coupling matrix, the realized geometry, and the ideal coupling-matrix response (dashed) overlaid with the full-wave mode-matching analysis (solid).

A runtime comparison on the same workstation (Intel Core i7, 4 cores, 64 GB RAM) is given in Table 9. The MM solver evaluates the dominant TE_{10} scattering parameters as a multi-mode generalized-scattering-matrix cascade, using between 6 and 20 TE_{m0} modes per segment (auto-set by a relative-convergence rule); it requires no meshing and no iterative field convergence. A single 601-point analysis of the accepted filter completes in 2.0 s, against ≈180 s for HFSS (601 points, interpolating sweep, Max $\Delta S = 0.2\%$ convergence), a speed-up of roughly 90×. Because each verification is so cheap, the entire autonomous design (three corrective rounds of synthesis and full-wave verification) finishes in 3.0 s, faster than a single HFSS sweep. That low cost is what puts full-wave verification inside the corrective loop of Figure 21 rather than at the end: every round is re-checked against the real physics at negligible cost.

Table 9. Computational cost of the mode-matching (MM) verification used in the agentic loop versus a reference HFSS analysis of the accepted $N = 5$ filter, measured on the same workstation (Intel Core i7, 4 cores, 64 GB RAM).

	Mode Matching	**HFSS**
Convergence control	Fixed TE_{m0} modes/segment	Max $\Delta S = 0.2\%$
Frequency sweep	Direct, per point	Interpolating
Frequency points	601	601
Modes per segment	20 (cavities), 6–10 (irises)	—
Runtime by stage		
Direct K-inverter synthesis (per round)	<0.1 s	—
MM verification sweep (301 pts, per round)	0.9 s	—
Single S-parameter analysis (601 pts)	2.0 s	≈180 s
Autonomous tuning, 3 rounds (total)	3.0 s	—

The agent does something a classical optimizer cannot. The corrective loop of Figure 21, together with the synthesis, mode-matching, and compliance kernels, is fully deterministic and classical, so every run remains reproducible and physically auditable. The agent sits above this inner loop. It converts a free-form, system-level brief into a complete, validated design specification: it infers the unstated defaults, resolves WR-112 to its waveguide

dimensions, and recognizes that a stopband mask was supplied in place of an order. It then selects the appropriate procedure among the specialized design agents. A classical optimizer also iterates, but only once a human has handed it a fully formalized problem of variables, bounds, objective, and constraints. Here, the agent performs that formalization itself, order selection included.

6. Demo Applications

6.1. 6-Pole 10 GHz High Selectivity Waveguide Filter

The example is a cross-coupled folded E-plane bandpass filter with passband centered at $f_0 = 10\,\text{GHz}$ and a stopband mask of at least 40 dB attenuation at the symmetric offsets 9.92 GHz and 10.08 GHz. With the mask entered through the chat panel, the GA-based order-and-zero selector of Section 3.3 returns the minimum-order solution that satisfies it: a six-pole prototype with a symmetric pair of transmission zeros at $\Omega_z = \pm 1.68\,j$ on the normalized lowpass axis, return-loss target $RL = 26\,\text{dB}$, and a topology which folded along the E-plane, since the zero pair requires a negative cross-coupling. The passband bandwidth is $\text{BW} = 100\,\text{MHz}$, a fractional bandwidth of one per cent.

Coupling-matrix synthesis with the prescribed zeros returns the external quality factor and the inter-resonator couplings of the folded matrix:

$$\begin{aligned} Q_e &= 80.252, \quad k_{12} = k_{56} = 9.47\times10^{-3}, \\ k_{23} &= k_{45} = 6.31\times10^{-3}, \quad k_{34} = 7.06\times10^{-3}, \\ k_{25} &= -1.21\times10^{-3}. \end{aligned} \tag{25}$$

The negative sign of k_{25} is the cross-coupling that produces the symmetric zero pair.

Each entry of (25) is mapped to a physical dimension through the design curves of Section 4: the four inline inductive irises against the inline-iris curve, the offset rectangular aperture at the bend (which carries k_{34}) against the offset-aperture curve with fixed aperture length $\ell_{\text{ap}} = 3.5\,\text{mm}$ and 1.0 mm end-wall clearance, the centered square cross-coupling aperture (which carries k_{25}) against the square-aperture curve, and the input/output coupling against the Q_e curve. Both the cavity length $L_1 = L_6$ and the I/O iris width $w_{01} = w_{67}$ are read jointly from the two-dimensional Q_e grid, which retunes each candidate cavity for resonance at f_0 before reporting Q_e. The inner cavities 2–5 are initialized at the unloaded $\lambda_g/2 = 19.854\,\text{mm}$, minus the 0.5 mm extraction-time pre-correction built into the curves and a further 0.25 mm per adjoining inductive iris. The result is $L_2 = L_3 = L_4 = L_5 = 19.354\,\text{mm}$. The folded topology imposes mirror symmetry about the center, so the independent parameters listed in Table 10 fully describe the geometry; cavity inner-corner radii are neglected.

The MM engine of Section 5 does not yet support cross-coupled geometries, so the full filter is built and analyzed directly in HFSS through a PyAEDT notebook emitted by the `design-curve` skill. The resulting model, shown in Figure 23, places cavities 1–3 in one row and cavities 4–6 in the parallel row; the bend offset aperture occupies the shared end wall between cavity 3 and cavity 4, and the centered square aperture occupies the wall shared by cavity 2 and cavity 5.

HFSS analysis of the initial geometry returns the response of Figure 24a. The return-loss target is missed substantially, and the two transmission skirts roll off at noticeably different rates, with the lower skirt steeper than the upper. The skirt asymmetry indicates that the three resonator pairs $(1,6)$, $(2,5)$, and $(3,4)$ are not individually tuned to f_0: the centered square aperture loads cavities 2 and 5 more heavily than the uniform per-iris correction anticipates, and the bend offset aperture loads cavities 3 and 4 more heavily than an ordinary inline iris does.

Table 10. Initial dimensions of the folded six-pole filter (mm), synthesized at $f_0 = 10$ GHz in WR-90 with 1.5 mm wall thickness throughout. Only independent parameters are listed; mirror symmetry about the center gives $L_k = L_{7-k}$ and $w_{k,k+1} = w_{6-k,7-k}$.

Parameter	Symbol	Initial	Tuned
Broad-wall dimension (WR-90)	a	22.86	22.86
Narrow-wall dimension (WR-90)	b	10.16	10.16
Cavities 1, 6 length	$L_1 = L_6$	16.698	16.498
Cavities 2, 5 length	$L_2 = L_5$	19.354	19.354
Cavities 3, 4 length	$L_3 = L_4$	19.354	19.45
I/O iris width	$w_{01} = w_{67}$	11.738	11.6
Iris 1–2 (=5–6) width	$w_{12} = w_{56}$	5.043	5.043
Iris 2–3 (=4–5) width	$w_{23} = w_{45}$	4.575	4.5
Bend offset aperture width	w_{34}	6.407	6.407
Bend aperture length	ℓ_{ap}	3.5	3.5
Bend aperture end-wall clearance	d_{ap}	1.0	1.0
Cross-coupling aperture side	w_{25}	4.595	4.595

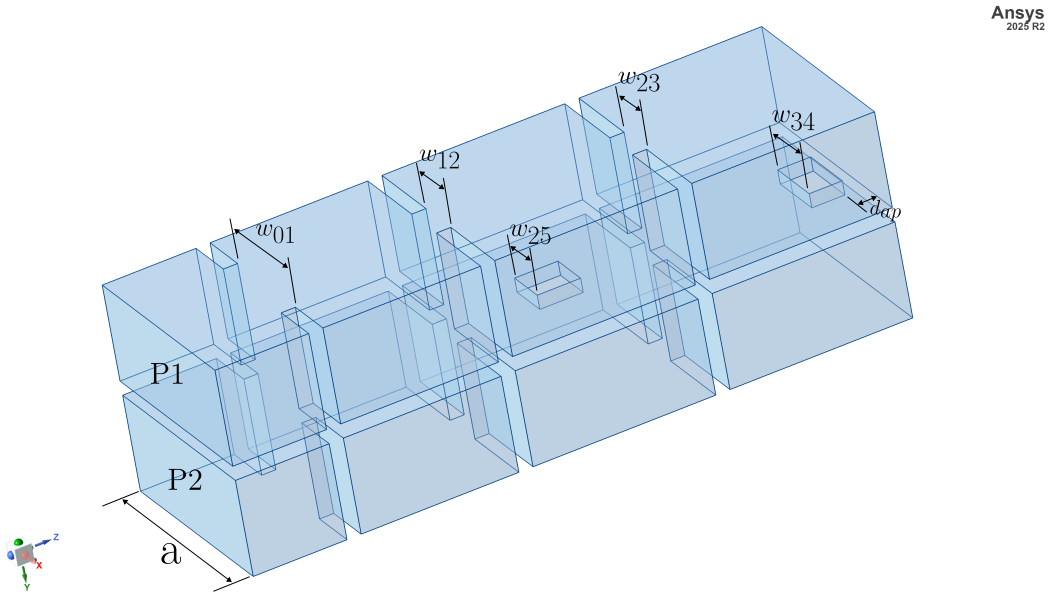
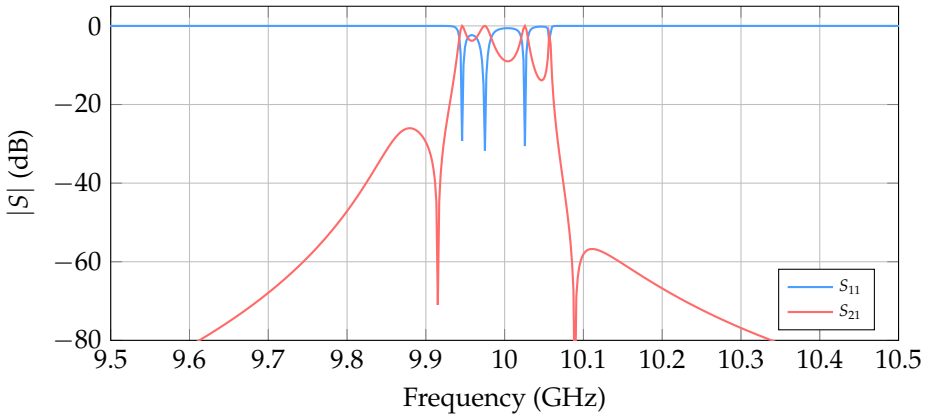


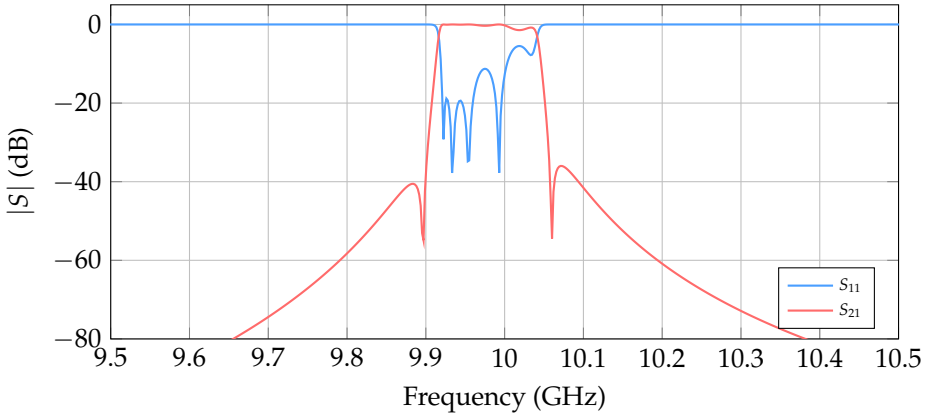
Figure 23. HFSS model of the folded six-pole filter, generated automatically through the PyAEDT notebook emitted by the `design-curve` skill from the dimensions of Table 10.

The two affected cavity pairs are dimensioned separately. The single inner-cavity length parameter is split in two, one for the $(2, 5)$ pair and one for the $(3, 4)$ pair, with each still initialized at 19.354 mm but tuned independently against the response. A short manual-tuning loop takes the response from Figure 24a to that of Figure 24b: the two skirts come into symmetry, the transmission zeros are restored, and the in-band return loss settles at the synthesized level.

The wideband sweep of Figure 25 shows that the first spurious passband, driven by the higher-order cavity modes, lies around 16 GHz, well above the operating band. The residual in-band mismatch (the $|S_{11}|$ ripples sit above the synthesized -26 dB target) could be closed with coupling-matrix-extraction-based computer-aided tuning, which is outside the scope of this paper. The example nonetheless covers the full FilterForge workflow on a non-trivial cross-coupled geometry, end to end and inside the chat panel.



(**a**) Initial dimensions



(**b**) After length-pair retuning

Figure 24. In-band S-parameters of the folded six-pole filter. (**a**) Response from the dimensions read off the design curves at the synthesized coupling targets (Table 10), with skirt asymmetry from differential aperture loading. (**b**) Response after the inner-cavity length is split into independent $(2,5)$ and $(3,4)$ parameters and a short manual-tuning loop is applied.

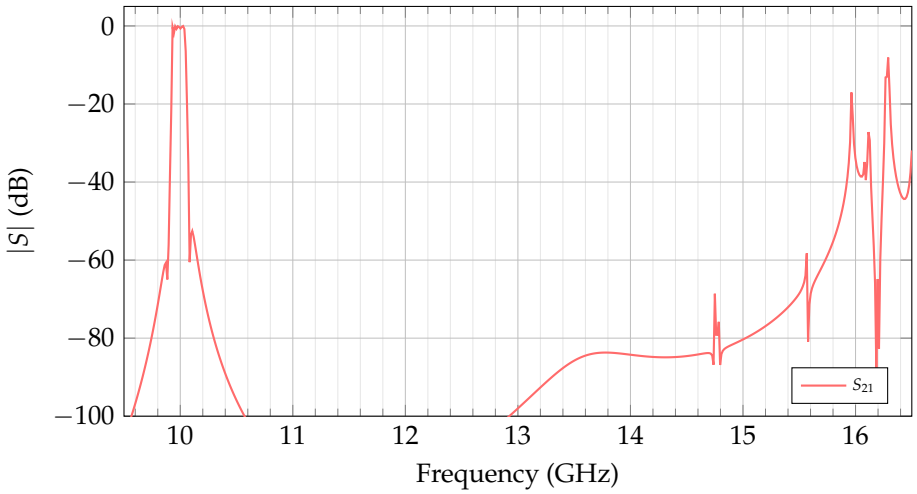


Figure 25. Wideband S-parameter response of the initial filter geometry. The first spurious passband, driven by higher-order cavity modes, lies around 16 GHz, well above the operating band at $f_0 = 10$ GHz.

6.2. 6-Pole 10 GHz Group-Delay-Equalized Waveguide Filter

The second end-to-end example uses the same WR-90 platform as Section 6.1 but a different goal: a flat in-band group delay instead of steep selectivity. The specification entered

through the chat panel is $f_0 = 10\,\text{GHz}$, BW $= 150\,\text{MHz}$, return-loss target $RL = 20\,\text{dB}$, with group-delay equalization requested across the passband. The GA selector of Section 3.3 does not choose the zeros this time. Its stop-band search places only imaginary-axis zeros, which sharpen the skirts, but equalization needs real-axis zeros, and those add no rejection. The real-axis pair is set instead by an optimization-based equalization method [1] that moves the zeros until the in-band group delay is as close to linear as it can get. It returns a six-pole prototype with a symmetric pair at $\Omega_z = \pm 1.12$, which flattens the passband delay at a small return-loss cost. The topology is folded along the E-plane as in Section 6.1; the one change is that the diagonal cross-coupling k_{25} is now positive, which is what puts the zeros on the real axis.

The positive cross-coupling cannot be realized with the centered square aperture used in Section 6.1, whose loading is intrinsically capacitive. Two narrow inductive irises cut symmetrically on the side walls of cavities 2 and 5, where the magnetic field is maximum, deliver positive coupling instead. The sign is verified at extraction time with the symmetry-plane test described in Section 4: an eigenmode analysis of the two-cavity coupled model returns the two lowest eigenfrequencies, and a second analysis with a perfect-electric sheet placed on the symmetry plane returns the odd-mode frequency unambiguously, so the remaining mode is the even one. The coupling coefficient,

$$k = \frac{f_{\text{odd}}^2 - f_{\text{even}}^2}{f_{\text{odd}}^2 + f_{\text{even}}^2} \tag{26}$$

is positive as long as $f_{\text{odd}} \geq f_{\text{even}}$; a sign flip during the sweep indicates that the sub-structure has left the magnetic-coupling regime and electric-coupling dominates. The double-side iris sub-structure and the resulting design curve at 10 GHz in WR-90 are shown in Figure 26.

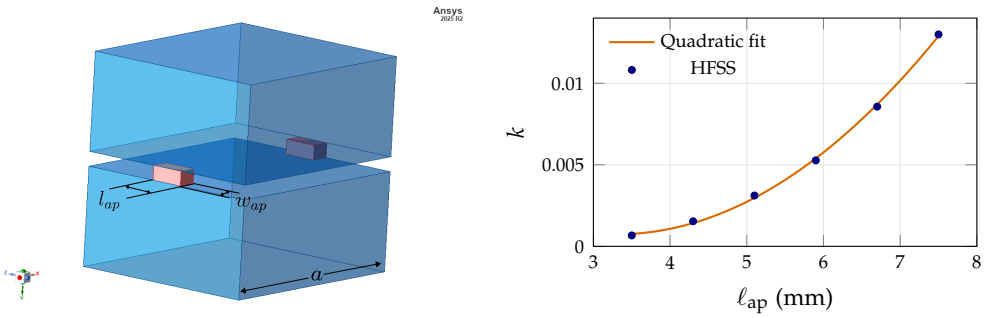


Figure 26. Double-side iris cross-coupling sub-structure used for the group-delay-equalized filter at 10 GHz in WR-90. (**Left**) HFSS sub-structure. (**Right**) Resulting positive (magnetic) coupling k versus aperture length ℓ_{ap} ($R^2 = 0.999$). Markers: HFSS samples; solid line: degree-two least-squares fit $k(\ell_{\text{ap}}) = 6.885 \times 10^{-4}\,\ell_{\text{ap}}^2 - 4.541 \times 10^{-3}\,\ell_{\text{ap}} + 8.227 \times 10^{-3}$. Inversion at the synthesized target $k_{25} = 1.83 \times 10^{-3}$ returns $\ell_{\text{ap}} \approx 4.557\,\text{mm}$.

Coupling-matrix synthesis with the prescribed real-axis zeros returns the external quality factor and the inter-resonator couplings of the folded matrix:

$$\begin{aligned} Q_e &= 65.617, \quad k_{12} = k_{56} = 12.8\times10^{-3}, \\ k_{23} &= k_{45} = 9.18\times10^{-3}, \quad k_{34} = 7.27\times10^{-3}, \\ k_{25} &= +1.83\times10^{-3}. \end{aligned} \tag{27}$$

Compared with the high-selectivity case of (25), the wider passband and looser return-loss target inflate all four inline couplings and Q_e, and k_{25} now carries the opposite sign.

Each entry of (27) is mapped to a physical dimension through the same design curves of Section 4 used for the high-selectivity filter, with the centered-aperture curve replaced by the double-side iris curve of Figure 26(right). The four inline inductive iris widths are read off the inline-iris curve at their $k_{i,i+1}$ targets, the bend offset aperture width is read off the offset-aperture curve at fixed $\ell_{\mathrm{ap}} = 3.5\,\mathrm{mm}$ and 1.0 mm end-wall clearance, and the input/output coupling is read off jointly with the $L_1 = L_6$ cavity length on the two-dimensional Q_e grid. The positive cross-coupling k_{25} is realized by two side-wall irises of length $\ell_{25} = 4.557\,\mathrm{mm}$ and fixed width $w_{25} = 2\,\mathrm{mm}$, read directly off Figure 26(right). The inner cavities are initialized at the unloaded $\lambda_g/2 = 19.854\,\mathrm{mm}$ minus the 0.5 mm extraction-time pre-correction and a further 0.25 mm per adjoining inductive iris. Mirror symmetry about the center reduces the geometry to the independent parameters listed in Table 11.

Table 11. Initial and tuned dimensions of the group-delay-equalized folded 6-pole filter (mm), synthesized at $f_0 = 10\,\mathrm{GHz}$ in WR-90 with 1.5 mm wall thickness and 9.927 mm I/O feed waveguide throughout. Only independent parameters are listed; mirror symmetry about the center gives $L_k = L_{7-k}$ and $w_{k,k+1} = w_{6-k,7-k}$.

Parameter	Symbol	Initial	Tuned
Broad-wall dimension (WR-90)	a	22.86	22.86
Narrow-wall dimension (WR-90)	b	10.16	10.16
Cavities 1, 6 length	$L_1 = L_6$	15.884	15.342
Cavities 2, 5 length	$L_2 = L_5$	19.354	18.925
Cavities 3, 4 length	$L_3 = L_4$	19.354	19.286
I/O iris width	$w_{01} = w_{67}$	12.692	12.692
Iris 1–2 (=5–6) width	$w_{12} = w_{56}$	5.614	5.539
Iris 2–3 (=4–5) width	$w_{23} = w_{45}$	5.063	4.91
Bend offset aperture width	w_{34}	6.524	6.368
Bend aperture length	ℓ_{ap}	3.5	3.5
Bend aperture end-wall clearance	d_{ap}	1.0	1.0
Cross-coupling iris length	ℓ_{25}	4.557	4.157
Cross-coupling iris width	w_{25}	2.000	2.000

The full filter is built and analyzed in HFSS through the PyAEDT notebook emitted by the `design-curve` skill. The resulting model, shown in Figure 27, retains the inline layout of the high-selectivity case, with cavities 1–3 and 4–6 in two parallel rows; the only difference is that the centered-aperture shared wall between cavities 2 and 5 is replaced by the pair of side-wall irises that carry the positive cross-coupling.

HFSS analysis of the initial geometry returns the response of Figure 28. The group-delay trace already shows the equalization lobe imposed by the real-axis zeros, but it is off-center for the same reason seen in Section 6.1: the cross-coupling and bend apertures load their cavities more than a plain inline iris would, so the inner resonators are pulled off f_0 and the return-loss target is missed.

The two affected inner-cavity pairs are dimensioned separately. Following the same strategy as in Section 6.1, the single inner-cavity length is split into independent $(2, 5)$ and $(3, 4)$ parameters, both still initialized at 19.354 mm but tuned independently against the response. A semi-automated tuning loop in HFSS/Filterforge, driven by coupling-matrix extraction [45], takes the response from Figure 28 to that of Figure 29: the equalization lobe centers on the passband, the peak-to-peak in-band group-delay ripple settles below 1.5 ns across the central $\pm 60\,\mathrm{MHz}$ of the passband, with a midband group delay of about 10 ns at f_0, and the return loss settles at the synthesized 20 dB level. This example runs the same end-to-end workflow as Section 6.1 on a topology with the opposite cross-coupling sign.

Adding a single new sub-structure to the `design-curve` skill was therefore enough to carry the demo from selectivity-driven to group-delay-driven specifications, with none of the surrounding steps changed.

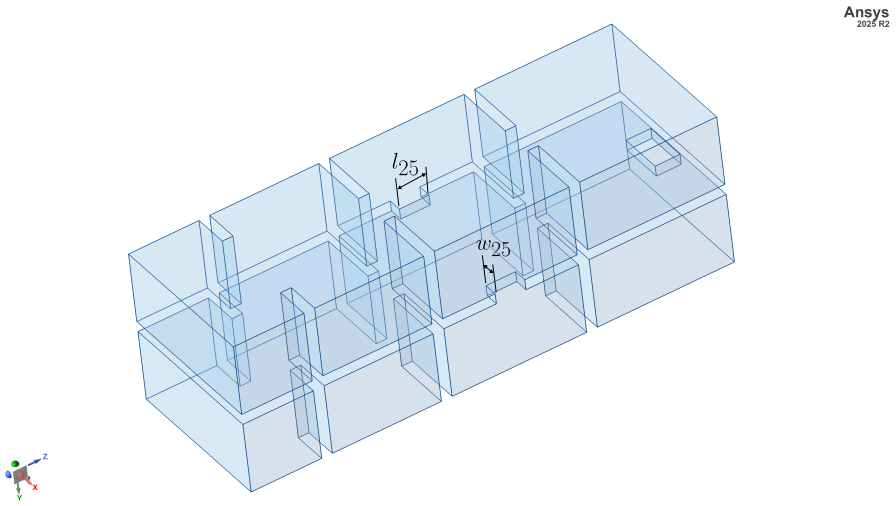
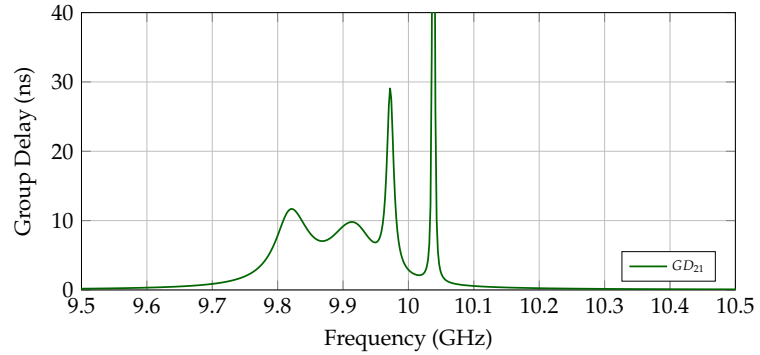
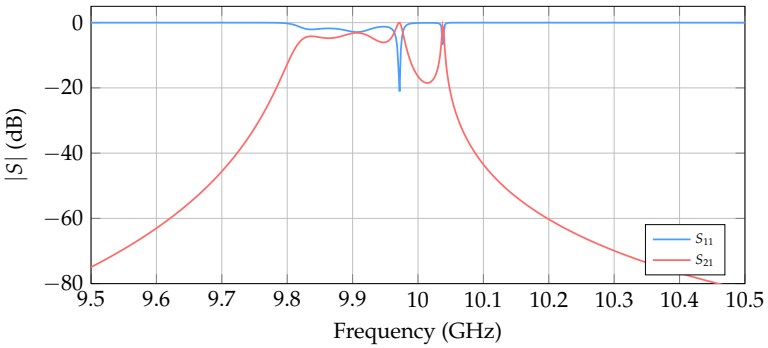


Figure 27. HFSS model of the group-delay-equalized folded six-pole filter, generated automatically through the PyAEDT notebook from the dimensions of Table 11. The double-side iris cross-coupling replaces the centered square aperture of Figure 23.



(**a**) Group delay



(**b**) *S*-parameters

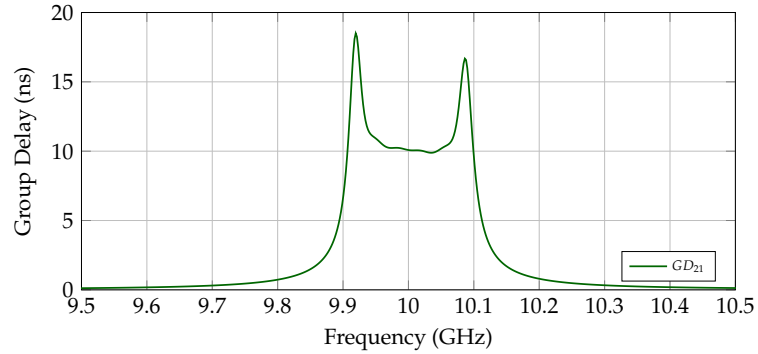
Figure 28. In -band response of the group-delay-equalized filter from the dimensions read off the design curves at the synthesized coupling targets, listed in Table 11. (**a**) S_{21} group delay, showing the equalization lobe imposed by the real-axis zeros; (**b**) $|S_{11}|$ and $|S_{21}|$. The coupling apertures load the inner cavities unevenly, so they are detuned, the return-loss target is missed, and the lobe is offset from the band center.

The complete tuning record (parameters touched, in situ sensitivities, and the intermediate dimensional and coupling-matrix states) is reproduced in Appendix D. Eight

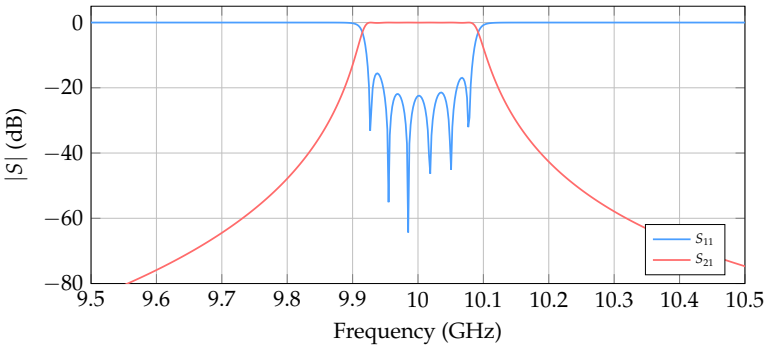
HFSS iterations take the geometry from the synthesized start (S0) of Table 11 to the tuned state (S8). Each iteration is a driven-modal solve on an Intel Core i7 4-core workstation with 64 GB of RAM, with the adaptive mesh-refinement convergence criterion set to $\Delta S = 0.4\,\%$, and takes about 5 min; the loop therefore dominates the roughly 40 min of end-to-end wall-clock time. Within this workflow, the agent already automates synthesis, GA-based order and transmission-zero selection, design-curve generation and the look-up of dimensions on those curves at the synthesized coupling targets, and the construction of the initial HFSS project through the emitted PyAEDT notebook; the iterative tuning loop is the only step still driven by the designer, as summarized in Table 12. The residual manual cost is bounded by two missing components: the in-house mode-matching engine does not yet cover cross-coupled topologies, and coupling-matrix extraction is performed through an external MATLAB package implementing the model-based vector-fitting method of [45] rather than a native MCP tool. Both are listed as future work; adding them would let the agent close the tuning loop end-to-end at a small fraction of the present cost.

Table 12. Automation breakdown of the end-to-end workflow for the group-delay-equalized filter.

Workflow Step	Driver
Coupling-matrix synthesis with real-axis zeros	Agent (MCP tool)
GA-based order and transmission-zero selection	Agent (MCP tool)
Design-curve generation for all five sub-structures	Agent (`design-curve` skill)
Dimension look-up on the design curves at synthesized coupling targets	Agent (MCP tool)
HFSS project construction from initial dimensions	Agent (PyAEDT notebook)
HFSS driven-modal solve (∼5 min per iteration)	HFSS (FEM)
Coupling-matrix extraction from the simulated response	External MATLAB package
Selection of correction dimension at each stage	Designer



(**a**) Group delay



(**b**) *S*-parameters

Figure 29. In -band response of the group-delay-equalized filter after the same length-pair retuning loop used in Section 6.1. (**a**) S_{21} group delay, with the equalization lobe centered on the passband and the in-band ripple more compressed relative to Figure 28a; (**b**) $|S_{11}|$ and $|S_{21}|$.

7. Limitations and Future Work

FilterForge is semi-automated by design. The agent formalizes a system-level brief and calls deterministic tools; it does not exercise learned judgment over the raw physics. That split keeps the results reproducible, and it also fixes the boundaries of the current version.

Two limits dominate. The mode-matching engine handles only inline geometries—uniform sections joined by concentric transversal steps—so cross-coupled, offset, and out-of-plane apertures fall outside its scope. The folded examples (Sections 6.1 and 6.2) therefore fall back on HFSS and an engineer-driven tuning loop, whereas the inline filters run end-to-end inside a single chat session. The matching diagnostic step, coupling-matrix extraction, also lives outside the environment in a MATLAB package (Table 12, Appendix C), which is what stops the agent from tuning cross-coupled filters on its own. Two minor caveats: all of the agent's domain knowledge sits in the system prompt, with no fine-tuning and nothing carried between sessions, and a single agent runs the whole flow with no independent check on its output.

Closing the mode-matching gap is the immediate next step. Extending the scattering-matrix cascade to non-concentric offset and side-wall apertures brings cross-coupled and coaxial-cavity topologies inside the engine, after which the manual HFSS round disappears. Its natural companion is a native coupling-matrix-extraction tool; the diagnostic-to-correction procedure is already written out as a chain of single-variable, sensitivity-driven corrections in Appendix C, so this is implementation, not research. Together they would let the agent tune cross-coupled hardware the same way the critique loop of Figure 21 already tunes inline filters.

Further out, we want to split the single orchestrator into several agents that check each other across synthesis, dimensioning, and verification. Mutual-critique setups of this kind are starting to appear in nearby RF-design work [31]. The deterministic kernels stay authoritative, so reproducibility is safe; the open problem is keeping the agents from endorsing each other's mistakes.

One choice we are not revisiting is the separation of domain knowledge into the system prompt and numerics into the tools. It lets us swap the underlying LLM, keeps general-purpose guesses out of the numbers, and leaves the tools portable. Lighter domain adaptation, retrieval over the filter-design literature, and a memory of past designs all fit inside this split, as long as the numerics stay in the tools.

We see this semi-automation as the right operating point, not a way station. The judgment-heavy decisions in high-power microwave hardware carry real manufacturing risk; handing them to the agent before the verification loop is both fast and complete across every supported topology would trade away more than it gains.

8. Conclusions

We have introduced FilterForge, a semi-automated VS Code extension that places coupling-matrix synthesis, predistortion, topology reconfiguration, GA-based order and transmission-zero selection, and modal analysis under a single chat-driven agent backed by an MCP server. Splitting an LLM orchestrator from deterministic tools keeps every reported quantity reproducible while natural-language prompts drive a full design session. A 45-task benchmark over the five tool categories (Section 2.3) shows the call layer is reliable on production-class models: gemini-3-flash reaches 96.3% tool-selection accuracy, 94.8% full-call accuracy, and an 88.9% pass^3 rate, which an ablation traces to the curated tool-selection block rather than to raw model capability.

The mode-matching engine reproduces full-wave HFSS responses for the six-pole *H*-plane iris-coupled filter at a fraction of the cost, confirming that deterministic tools can supply the numerical backbone an LLM lacks while the agent contributes orchestration and

interpretation. A 4 GHz inline design tuned from the chat panel validates that workflow against HFSS (Section 5), and an 8 GHz WR-112 counterpart is taken end-to-end without an engineer in the loop (Section 5.2): the agent runs a deterministic critique, gates each candidate, and converges on a manufacturable geometry in one chat session. The same GA kernel selects order and zero placement for an asymmetric C-band channel directly against a stop-band mask (Section 3.3).

A demonstration on a folded six-pole WR-90 cross-coupled filter exercises the rest of the workflow inside the chat panel: zero selection from a stop-band mask, folded-matrix synthesis, design-curve dimensioning, HFSS modeling through the generated PyAEDT notebook, and a short length-pair tuning loop that resolves the skirt asymmetry from uneven aperture loading (Section 6.1). A second demonstration on the same platform, a group-delay-equalized filter whose positive cross-coupling uses a pair of side-wall irises (Section 6.2), shows the same skill chain handling a very different specification by changing only the synthesis target and the cross-coupling sub-structure.

For cross-coupled hardware still outside the mode-matching engine, the procedure an engineer follows during the HFSS round is written out in Appendix C as single-variable, sensitivity-driven coupling-matrix corrections, ready to become a native skill once the engine is extended. The benchmark, grader, system prompt, packaged extension, and per-model captures are released in a single Zenodo artifact, so both the evaluation and the demonstrations regenerate without installation. The remaining limits and the roadmap that follows (cross-coupled mode matching, native coupling-matrix extraction, and a multi-agent decision layer) are discussed in Section 7.

Author Contributions: Conceptualization, H.N.G.; methodology, H.N.G.; software, H.N.G. and Y.K.; validation, H.N.G., Y.K., A.O.E. and B.D.; formal analysis, H.N.G.; investigation, Y.K.; resources, H.N.G. and A.O.E.; data curation, B.D.; writing—original draft preparation, H.N.G.; writing—review and editing, H.N.G., A.O.E. and M.K.; visualization, H.N.G. and Y.K.; supervision, B.D. and M.K.; project administration, M.K.; funding acquisition, H.N.G. and M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Scientific Research Project Department of Istanbul Technical University (ITU-BAP) [Grant No. 43462]; the Scientific and Technological Research Council of Türkiye (1002-B) [Grant No. 123E680] and the YÖK 100/2000 Ph.D. Programme.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Everything needed to reproduce this work is archived in a single Zenodo artifact (DOI: https://doi.org/10.5281/zenodo.20754038): the packaged FilterForge extension (`.vsix`), everything required to run the interactive chat workflow, and the agent benchmark—the 45-task gold set, the grader and table generator, the evaluated system prompt, and the raw per-model tool-call captures. Tables 2 and 3 regenerate from the released captures with `python grade.py` and `python tables.py`, in plain Python with no installation.

Acknowledgments: The authors thank the faculty staff for administrative and technical support during this work.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
BW	Bandwidth
CNC	Computer Numerical Control

DE	Differential Evolution
EDA	Electronic Design Automation
EM	Electromagnetic
FBW	Fractional Bandwidth
FDC	Frequency-Dependent Coupling
GA	Genetic Algorithm
GPT	Generative Pre-trained Transformer
GSM	Generalized Scattering Matrix
HFSS	High Frequency Structure Simulator
IDE	Integrated Development Environment
IMUX	Input Multiplexer
LLM	Large Language Model
MCP	Model Context Protocol
MM	Mode Matching
OMUX	Output Multiplexer
PEC	Perfect Electric Conductor
PyAEDT	Python Ansys Electronics Desktop Toolkit
RF	Radio Frequency
RL	Return Loss
TE	Transverse Electric
TEM	Transverse Electromagnetic
TM	Transverse Magnetic
TZ	Transmission Zero
UI	User Interface
VS Code	Visual Studio Code
WR	Rectangular Waveguide
YAML	YAML Ain't Markup Language

Appendix A

This appendix lists every MCP tool exposed by the FilterForge server in the present implementation. For each tool, the principal required inputs and the principal fields of the returned dictionary are given; JSON-schema-level details (optional fields, default values, units of secondary parameters) are omitted to keep the table readable and are documented in the source repository.

Table A1. Complete MCP tool catalog. Required inputs are listed; structured outputs are summarized by their principal fields. Units: MHz for physical frequencies, dB for return- and insertion-loss specifications, normalized lowpass Ω or complex s for prototype-domain quantities.

Tool	Purpose	Key Inputs	Key Outputs
Synthesis			
`chebyshev_poly_synthesis`	Generalized Chebyshev E, F, P polynomials of order N	N, RL, $\{s_{z,k}\}$	ε, ε_R, coefficients of E, F, P, reflection and transmission zero locations
`coupling_matrix_synthesis`	Direct $(N+2)$ transversal CM, optional folded reconfiguration	N, RL, $\{s_{z,k}\}$	$\mathbf{M}_T$, optional $\mathbf{M}_{\text{folded}}$, eigenvalues, source/load couplings, ε, ε_R
`cascade_trisection_synthesis`	Inline cascade of trisections, one imaginary-axis TZ per block	N, RL, imaginary-axis TZs, trisection specifications	$\mathbf{M}_{\text{arrow}}$, $\mathbf{M}_{\text{CT}}$
`cascaded_ntuplets_synthesis`	Inline cascade of triplet and quadruplet blocks, complex-axis TZs	N, RL, $\{s_{z,k}\}$, block sequence	$\mathbf{M}_{\text{arrow}}$, $\mathbf{M}_{\text{ntuplet}}$
`predistortion_synthesize`	Predistorted CM for finite Q_u	N, RL, $\{s_{z,k}\}$, Q_u, μ-choice, domain, topology	$\mathbf{M}_T$, optional $\mathbf{M}_{\text{folded}}$, ε, ε_R

Table A1. *Cont.*

Tool	Purpose	Key Inputs	Key Outputs
Reconfiguration			
`matrix_reconfiguration`	Givens-rotation reconfiguration into a target topology	$(N+2) \times (N+2)$ matrix, target topology	Reconfigured matrix, topology label
Analysis			
`frequency_response_calculate`	S_{11}, S_{21}, group delay from CM or polynomials	CM or polynomials, mode, f_0, BW	Frequency grid, $\lvert S_{11}\rvert$, $\lvert S_{21}\rvert$, group delay, passband metrics
`mode_matching_analyze`	Full-wave MM analysis of H-plane iris-coupled geometry	Cavity and iris segment list, waveguide height, frequency range	Complex S-parameters, frequency grid, mode counts used
Optimization (genetic algorithm)			
`ga_optimize_filter_order`	Synchronous DE search for minimum N and TZ positions meeting stop-band masks	f_0, stop-band mask, BW or FBW, RL	N, n_{fz}, TZ locations (normalized and physical), fitness values, verification flag
`start_ga_optimization`	Launches the GA in a background thread	Same as `ga_optimize_filter_order`	`run_id`
`get_ga_progress`	Polls the live GA state	`run_id`	Status, current N, best fitness, generation, final result on completion
`cancel_ga_optimization`	Cancels a running GA	`run_id`	Cancellation acknowledgment
`compute_fitness_landscape`	2D fitness map over (n_{fz}, TZ position)	f_0, stop-band mask, BW, RL, N, n_{fz}	TZ grid, fitness grid, best TZ, minimum fitness
Export			
`touchstone_export`	Renders CM and physical parameters as a Touchstone `.s2p` file	CM, f_0, BW	Suggested filename, file contents, number of points
`coupling_matrix_export_mat`	Serializes CM and metadata to a MATLAB Level-5 `.mat` file	CM, node labels, N	Suggested filename, base64 payload, payload size
`tikz_export`	Renders CM topology and/or scattering response as standalone LaTeX/TikZ source	CM, f_0, BW, figure type	Suggested filename, TikZ source code

Appendix B

This appendix reproduces verbatim the `SKILL.md` manifest bundled with the `design-curve` skill discussed in Section 4. The file is written in the GitHub-flavored Markdown format consumed by Claude Code and Copilot agents: a YAML frontmatter block declares the skill's identifier and trigger description, followed by a free-form body that scopes when the skill applies, lists the invariants the agent has to respect, sets out the workflow, and points to the bundled references and notebooks.

Listing A1: SKILL.md manifest bundled with the design-curve skill

```
---
name: design-curves
description: >-
  Generate PyAEDT/HFSS design-curve Jupyter notebooks for waveguide bandpass
  filter dimensioning -- coupling coefficient k and external Q versus a physical
  dimension. Use when the user asks for filter dimensioning / design-curve
  notebooks for a rectangular-waveguide filter (e.g., "WR90 12 GHz H-plane
  iris-coupled filter dimensions"), inter-resonator iris coupling, electric
  cross-coupling apertures, external Q (I/O iris), or~a full folded-filter
  S-parameter response. Builds on the bundled wgfilter library; reuses it,
  never rewrites the EM physics.
---

# Waveguide Filter Design-Curve~Notebooks

Generate Jupyter notebooks that drive Ansys HFSS (via PyAEDT) to extract
**design curves** for waveguide bandpass-filter dimensioning: the mapping from
a physical dimension (iris/aperture width, cavity length) to a coupling
```

```
coefficient ‘k‘ or external quality factor ‘Qe‘, fitted as a 2nd-degree
polynomial and invertible to answer *"what dimension gives this synthesised
coupling?"*.

## When to~use

Trigger when the user requests dimensioning / design-curve notebooks for a
rectangular-waveguide filter: inter-resonator iris coupling, cross-coupling
apertures, external Q, or~a folded-filter response -- for any waveguide
standard and centre~frequency.

## Hard~rules

- **Reuse, never rewrite.** All EM physics lives in the bundled ‘wgfilter‘
  package. Generated notebooks **import and call** it; you never reimplement
  geometry, eigenmode math, the~sweep driver, or~curve fitting.
- **Adapt a bundled notebook.** Copy the matching archetype from
  ‘assets/notebooks/‘ and re-parametrise only its params cell + output
  filenames. Keep cell order and ‘wgfilter‘ calls verbatim.
- **Prerequisite check.** Generated notebooks ‘import wgfilter‘, so the
  FilterForge project must contain the scaffold from ‘assets/‘ (the ‘wgfilter/‘
  package as a sibling of ‘notebooks/‘, plus ‘requirements.txt‘). If~it is
  missing, tell the user to scaffold it (see the package README) rather than
  inlining the library.
- **AEDT version.** ‘wgfilter/config.py‘ pins ‘AEDT_VERSION = "2025.2"‘; the
  only required edit for a different install is that string -- surface this.
- **Read ‘references/gotchas-and-tuning.md‘ before emitting**, and~distil its
  run-order + mode-swap + tuning warnings into a markdown cell at the top of
  every notebook you generate (add the mode-swap guard cell to eigenmode
  archetypes).

## Workflow

1. **Parse the spec:** waveguide standard (-> ‘a, b‘ from ‘filter-theory.md‘),
   centre ‘f0‘, fractional bandwidth, order ‘N‘, return loss/ripple, and~   topology (
     inline iris-coupled / cross-coupled / folded). Ask only if a
   load-bearing field is missing.
2. **Get the targets:** if the user supplies a coupling matrix or ‘k_ij‘/‘Qe‘,
   use it. Otherwise synthesise the Chebyshev prototype and compute
   ‘k_{i,i+1}‘, ‘Qe_in/out‘ per ‘references/filter-theory.md‘ (state that
   synthesis is an overridable input).
3. **Select archetypes** by topology and coupling sign
   (‘references/notebook-archetypes.md‘, sign rules in ‘filter-theory.md‘ S3):
   - inline iris-coupled -> **01** (inter-resonator iris) + **04** (I/O Qe);
   - + electric cross-coupling -> add **02** (centred square aperture);
   - + folded inductive bridge -> add **03** (offset aperture);
   - full assembled response -> **08** (6-pole folded, CSV-tuned).
4. **Generate notebooks:** copy each chosen bundled notebook, re-parametrise
   the params cell from the spec -- ‘a, b‘; ‘f0‘; ‘L = config.te101_length_mm(a,
   f0)‘ (do **not** reuse ‘config.CAVITY_LENGTH‘ unless ‘f0 == 10‘ GHz);
   ‘fed_len‘/section lengths from ‘config.waveguide_lambdas‘; eigenmode
   ‘min_freq ~ f0 - 2‘; modest sweep ranges that bracket the targets; unique
   ‘results/‘ filenames and ‘meta‘. Wire each inverse cell to its synthesised
   target.
5. **Explain the run procedure:** venv + ‘requirements.txt‘; confirm
   ‘AEDT_VERSION‘; which notebooks are fully scripted (01/02/03) vs.
   build-and-save -> **manual HFSS GUI solve** -> read (04, 08); how ‘invert‘
   yields the dimensions; recommended end-to-end order from
   ‘gotchas-and-tuning.md‘ S1.

## References (read as needed)

- ‘references/wgfilter-api.md‘ -- signature-level contract for every ‘wgfilter‘
```

```
  function + global conventions (units, axes, single-solve rule, sign).
- ‘references/notebook-archetypes.md‘ -- the 5 canonical notebooks (01/02/03/04/
  08), cell-by-cell, with˜what to re-parametrise.
- ‘references/filter-theory.md‘ -- spec -> prototype -> ‘k‘/‘Qe‘; coupling-sign
  convention; rectangular-waveguide ‘a,b‘/band table; cavity/wavelength
  relations.
- ‘references/pyaedt-hfss-setup.md‘ -- environment, eigenmode vs modal, the˜  single-
     Optimetrics-solve license rule, the˜manual-solve pattern, error table.
- ‘references/gotchas-and-tuning.md‘ -- **read first**; run order, EM-geometry
  intent, one-parameter-per-trial tuning discipline, mode-swap guard, other
  pitfalls.
- ‘references/recipe-wr90-iris-coupled.md‘ -- fully worked end-to-end example
  for the canonical "WR90 12 GHz H-plane iris-coupled"˜request.

## Bundled˜assets

‘assets/wgfilter/‘ (the library, vendored verbatim), ‘assets/notebooks/‘
(01-04 + 08 + ‘08_folded_filter_params.csv‘), ‘assets/tests/test_designcurve.py‘
(AEDT-free math validation), ‘assets/requirements.txt‘. These are the
FilterForge new-project scaffold *and* the templates you copy from.
```

Appendix C

Dimensional synthesis only gives a starting geometry: each coupling aperture sets an inter-resonator coupling and simultaneously reactively loads the adjacent cavities, so the as-built EM response differs from the prototype, the worst case being narrowband filters at high frequency. The procedure below corrects the full-wave model against the target coupling matrix in a few iterations and leaves only the final equiripple adjustment to a conventional EM optimizer. The key idea is to use coupling-matrix extraction as a diagnostic, which turns blind multi-dimensional optimization into a sequence of single-variable corrections.

The extracted normalized $N+2$ matrix $\mathbf{M}$ is read on two levels: the off-diagonal entries M_{ij} are the realized couplings, while the diagonal entries M_{ii} are the self-detuning of each resonator,

$$\Delta f_i \;=\; M_{ii}\,\frac{\mathrm{BW}}{2} \;=\; M_{ii}\,\frac{\mathrm{FBW}}{2}\,f_0, \tag{A1}$$

where f_0 is the center frequency and $\mathrm{FBW} = \mathrm{BW}/f_0$ the fractional bandwidth. The normalized couplings and the external quality factor follow as

$$k_{i,i+1} \;=\; \mathrm{FBW}\,M_{i,i+1}, \qquad Q_\mathrm{e} \;=\; \frac{1}{\mathrm{FBW}\,M_{S1}^2}, \tag{A2}$$

so the extracted $\mathbf{M}$ can be compared with the prototype entry by entry.

Each correction is set by a local sensitivity. An initial estimate for a length-controlled resonator frequency comes from the TE_{101} resonance of a rectangular cavity of broad-wall width a and length d,

$$f_{101} \;=\; \frac{c}{2}\sqrt{\frac{1}{a^2}+\frac{1}{d^2}}, \qquad \frac{\partial f}{\partial d} \;=\; -\,\frac{c^2}{4\,f\,d^3}, \tag{A3}$$

which for an unloaded X-band cavity ($a = 22.86\,\mathrm{mm}$, $d \approx 19.3\,\mathrm{mm}$) gives $\partial f/\partial d \approx -310\,\mathrm{MHz/mm}$. In the assembled filter the irises load the walls and shift this slope by 10–20 %, so the sensitivity is better measured in place from two solved points,

$$s_p \;=\; \frac{f^{(2)}-f^{(1)}}{p^{(2)}-p^{(1)}}, \qquad \Delta p_\mathrm{corr} \;=\; \frac{f_0-f^{(\mathrm{cur})}}{s_p}, \tag{A4}$$

and the same relation applies to couplings with $s_w = \Delta M_{ij}/\Delta w$ and $\Delta w_{\text{corr}} = (M_{ij}^{\text{ideal}} - M_{ij}^{\text{cur}})/s_w$.

Two rules keep the loop well behaved: only one parameter is changed per simulation, and the sensitivity is treated as local and re-measured as the operating point drifts. A wrong step does no lasting harm, since the new extraction and the previous one together yield the two points needed to compute the true slope. Corrections are ordered to exploit a one-way dependence: apertures reload the resonators strongly whereas small length corrections barely move the couplings, so the diagonal is nulled first; the off-diagonal entries are then corrected through s_w, and the small residual diagonal shift is absorbed by a short return to the frequency step, settling within about two passes. When every matrix entry is within 1–2 % of target, a gradient-based EM optimizer polishes the response to equiripple, minimizing the in-band max $|S_{11}|$ error. The narrow fractional bandwidth magnifies every error: from (A1) a 30 MHz shift in an FBW $\approx$ 1.5 % design (about 0.1 mm of cavity length) uses a quarter of the band, so dimensional control better than 50 µm and a well-converged adaptive mesh are required.

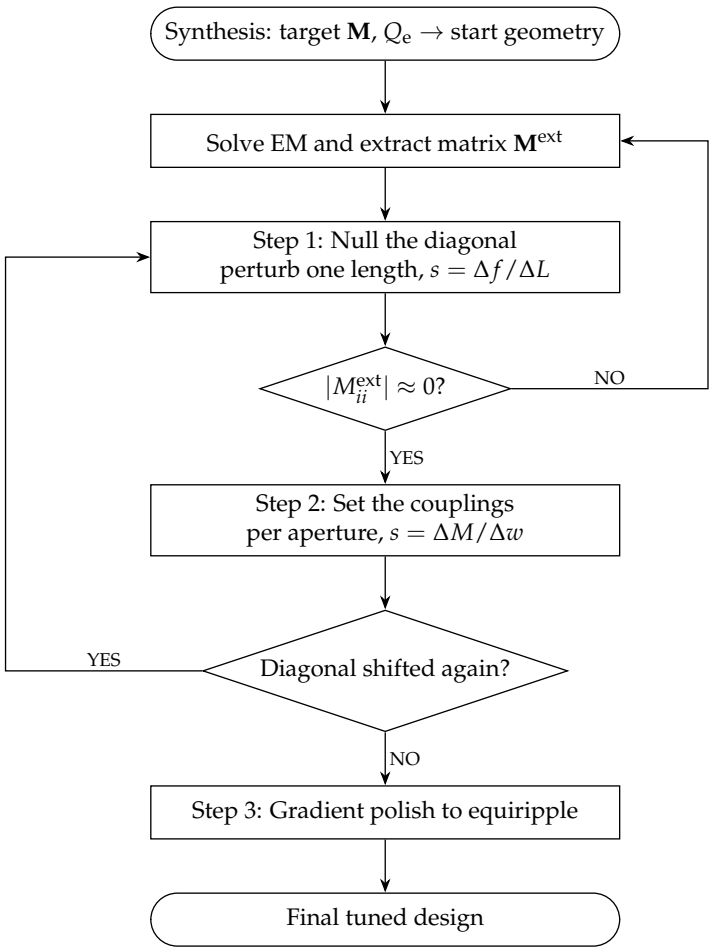


Figure A1. Coupling-matrix-guided tuning loop. The diagonal is nulled first, the couplings are corrected next, and a gradient-based EM optimizer polishes to equiripple. Each step changes one parameter and measures its in situ sensitivity.

Appendix D

Each stage compares the extracted matrix with the previous extraction and the target, computes the in situ sensitivity from the two extractions, and applies a one-shot dimensional correction. Note that during the procedure, self couplings, M_{ii} treated as denormalized (in MHz) while inter-cavity couplings normalized. Also note that, there could

be minor differences in between reciprocal couplings, $M_{ij}^{ext} \neq M_{ji}^{ext}$ couplings, arithmetic average of the coupling values are considered.

Table A2. Dimensional convergence across tuning states (mm). S0: synthesized start. S1–S3: frequency correction. S4: M_{23}. S5–S6: M_{34}. S6: M_{25} via ℓ_{25}. S7: M_{12} and $f_{1,6}$. S8: final micro-tuning of the filter. S9: tuned geometry of the group-delay-equalized variant (Section 6.2, Table 11), listed for comparison with the same parameter set.

Parameter	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
L_1	15.884	16.100	15.000	15.411	15.411	15.411	15.411	15.379	15.379	15.342
L_2	19.354	19.726	18.861	18.890	18.890	18.890	18.890	18.890	18.948	18.925
L_3	19.354	19.424	19.157	19.286	19.286	19.286	19.286	19.286	19.286	19.286
w_{12}	5.614	5.614	5.614	5.614	5.614	5.614	5.614	5.539	5.539	5.539
w_{23}	5.063	5.063	5.063	5.063	4.910	4.910	4.910	4.910	4.875	4.910
w_{34}	6.524	6.524	6.524	6.524	6.524	6.120	6.368	6.368	6.294	6.368
ℓ_{25}	4.557	4.557	4.557	4.557	4.557	4.557	4.157	4.157	4.157	4.157

Table A3. Coupling-matrix convergence across tuning states. Diagonal rows are the extracted resonator self-resonances (MHz); off-diagonal rows are normalized couplings. The last column is the synthesized target.

Quantity	S0	S1	S2	S3	S4	S5	S6	S7	S8	Ideal
$f_{1,6}$ (MHz)	10,124	10,155	9908	10,012	10,016	10,015	10,016	10,003	10,003	10,000
$f_{2,5}$ (MHz)	10,115	10,202	9993	10,005	9991	9995	9991	9986	10,002	10,000
$f_{3,4}$ (MHz)	10,022	10,030	9972	10,009	9999	9993	9999	9999	9997	10,000
M_{S1}	0.967	0.937	1.053	1.013	1.008	1.012	1.008	1.018	1.020	1.008
$M_{12,56}$	0.822	0.768	0.868	0.887	0.901	0.894	0.901	0.859	0.854	0.854
$M_{23,45}$	0.658	0.624	0.703	0.677	0.626	0.627	0.628	0.625	0.626	0.612
M_{34}	0.530	0.474	0.538	0.543	0.505	0.442	0.505	0.504	0.484	0.485
M_{25}	0.126	0.108	0.173	0.163	0.118	0.161	0.118	0.118	0.119	0.122

Appendix D.1. StateS2: Resonance Frequency Tuning

The S0→S1 step lengthened all three cavities, yet every resonance rose rather than fell, showing the in situ length sensitivity to be *positive* (opposite to the empty-cavity estimate). The two extractions provide the slope, and the correction reverses direction.

Table A4. State S2: matrix comparison (diagonal in MHz, off-diagonal normalized).

Quantity	S0 (Prev.)	S1 (Cur.)	Ideal	Action
$f_{1,6}$	10,124	10,155	10,000	correct L_1
$f_{2,5}$	10,115	10,202	10,000	correct L_2
$f_{3,4}$	10,022	10,030	10,000	correct L_3
M_{S1}	0.967	0.937	1.008	hold
$M_{12,56}$	0.822	0.768	0.854	hold
$M_{23,45}$	0.658	0.624	0.612	hold
M_{34}	0.530	0.474	0.485	hold
M_{25}	0.126	0.108	0.122	hold

$$s_{L_2} = \frac{10202 - 10115}{19.726 - 19.354} = +234\ \text{MHz/mm}, \quad \Delta L_2 = \frac{10000 - 10202}{234} = -0.86\ \text{mm}, \quad \text{(A5)}$$

$$s_{L_3} = \frac{10030 - 10022}{19.424 - 19.354} = +114\ \text{MHz/mm}, \quad \Delta L_3 = \frac{10000 - 10030}{114} = -0.26\ \text{mm}, \quad \text{(A6)}$$

$$s_{L_1} = \frac{10155 - 10124}{16.100 - 15.884} = +144\ \text{MHz/mm}, \quad \Delta L_1 = \frac{10000 - 10155}{144} = -1.08\ \text{mm}. \quad \text{(A7)}$$

Table A5. State S2: Dimensional update (mm).

Parameter	S1	S2	Δ
L_1	16.100	15.000	−1.100
L_2	19.726	18.861	−0.865
L_3	19.424	19.157	−0.267

Appendix D.2. State S4: Coupling Coefficient Tuning (M_{23} via w_{23})

With the diagonal nulled, the off-diagonal entries are corrected. Narrowing w_{23} lowers M_{23} and yields the coupling slope; the result lands within 2 % of target and is held, the small residual deferred to the micro-tuning.

Table A6. State S4: matrix comparison.

Quantity	S3 (Prev.)	S4 (Cur.)	Ideal	Action
$M_{23,45}$	0.677	0.626	0.612	corrected, hold
$M_{12,56}$	0.887	0.901	0.854	reloaded (later)
$f_{2,5}$ (MHz)	10,005	9991	10,000	reloaded (minor)

$$s_{w_{23}} = \frac{\Delta M_{23}}{\Delta w_{23}} = \frac{0.626 - 0.677}{4.910 - 5.063} = +0.333\ \text{mm}^{-1}, \tag{A8}$$

$$\Delta w_{23,\text{corr}} = \frac{0.612 - 0.626}{0.333} = -0.042\ \text{mm} \quad (\text{within } 2\,\%; \text{ deferred}). \tag{A9}$$

Table A7. State S4: Dimensional update (mm).

Parameter	S3	S4	Δ
w_{23}	5.063	4.910	−0.153

Appendix D.3. State S7: Coupling Coefficient and Resonance Frequency Tuning (M_{12} and $f_{1,6}$)

The input region couples two quantities: narrowing w_{12} lowers M_{12} but also detunes resonator 1, while L_1 sets $f_{1,6}$. Two single-parameter probes give both slopes, and the w_{12}-induced frequency shift is compensated in the L_1 correction.

Table A8. State S7: Matrix comparison.

Quantity	S6 (Cur.)	Probe	Ideal	Action
$M_{12,56}$	0.901	0.838 ($w_{12} = 5.514$)	0.854	correct w_{12}
$f_{1,6}$ (MHz)	10,016	10,032 ($L_1 = 15.48$)	10,000	correct L_1

$$s_{w_{12}} = \frac{0.838 - 0.901}{5.514 - 5.614} = +0.63\ \text{mm}^{-1}, \qquad \Delta w_{12} = \frac{0.854 - 0.901}{0.63} = -0.075\ \text{mm}, \tag{A10}$$

$$s_{L_1} = \frac{10032 - 10016}{15.480 - 15.411} = +232\ \text{MHz/mm}. \tag{A11}$$

The w_{12} change also lowers $f_{1,6}$ by about 9 MHz, leaving +7 MHz for L_1:

$$\Delta L_1 = \frac{10000 - 10007}{232} = -0.03\ \text{mm}. \tag{A12}$$

Table A9. State S7: Dimensional update (mm).

Parameter	S6	S7	Δ
w_{12}	5.614	5.539	−0.075
L_1	15.411	15.379	−0.032

After S7 every matrix entry lies within 1–2 % of target. State S8 applies a final micro-tuning ($L_2\uparrow$, $w_{34}\downarrow$, $w_{23}\downarrow$), after which the in-band return loss meets the 20 dB specification and the cross-coupling M_{25} flattens the group delay across the passband

Appendix E

The FilterForge extension exposes a library of ready-to-use example prompts through its *Prompt Cheat Sheet* panel. Each card lists a representative natural-language request together with the agent skill it is expected to trigger (e.g., coupling-matrix synthesis, pre-distortion, design-curve generation, mode-matching analysis). The cards can be inspected, copied, and pasted directly into the chat input, providing a low-friction entry point for new users to reproduce the example applications reported throughout this paper. A screenshot of the panel is shown in Figure A2.

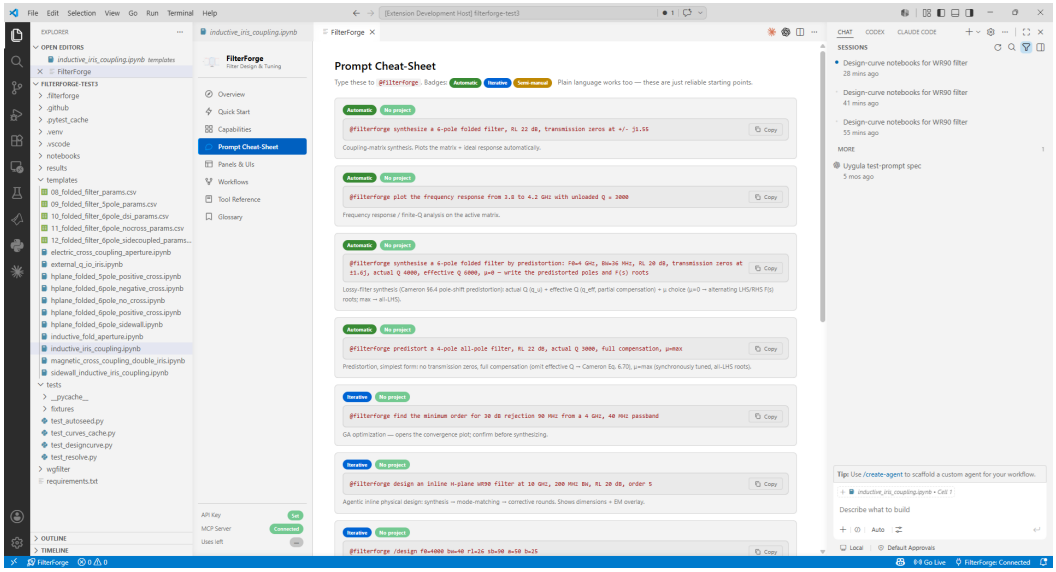


Figure A2. Prompt Cheat Sheet panel of the FilterForge extension. Each card pairs a representative natural-language prompt with the agent skill it invokes, and can be copied directly into the chat input to reproduce the example applications reported throughout this paper.

References

1. Cameron, R.J.; Kudsia, C.M.; Mansour, R.R. *Microwave Filters for Communication Systems: Fundamentals, Design, and Applications*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2018.
2. Maral, G.; Bousquet, M.; Sun, Z. *Satellite Communications Systems: Systems, Techniques and Technology*, 6th ed.; Wiley: Chichester, UK, 2020.
3. International Telecommunication Union. *Radio Regulations, Edition of 2020*; ITU: Geneva, Switzerland, 2020.
4. Kudsia, C.; Cameron, R.; Tang, W.-C. Innovations in microwave filters and multiplexing networks for communications satellite systems. *IEEE Trans. Microw. Theory Tech.* **1992**, *40*, 1133–1149.
5. Boria, V.E.; Gimeno, B. Waveguide filters for satellites. *IEEE Microw. Mag.* **2007**, *8*, 60–70.
6. Matthaei, G.L.; Young, L.; Jones, E.M.T. *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*; Artech House: Norwood, MA, USA, 1980.
7. Pozar, D.M. *Microwave Engineering*, 4th ed.; Wiley: Hoboken, NJ, USA, 2011.
8. Accatino, L. Computer-aided tuning of microwave filters. In Proceedings of the IEEE MTT-S International Microwave Symposium Digest, Long Beach, CA, USA, 2–4 June 1986; pp. 249–252.
9. Atia, A.E.; Williams, A.E. Narrow-bandpass waveguide filters. *IEEE Trans. Microw. Theory Tech.* **1972**, *20*, 258–265.

10. Cameron, R.J. Advanced coupling matrix synthesis techniques for microwave filters. *IEEE Trans. Microw. Theory Tech.* **2003**, *51*, 1–10.
11. Rhodes, J.D.; Cameron, R.J. General extracted pole synthesis technique with applications to low-loss TE_{011} mode filters. *IEEE Trans. Microw. Theory Tech.* **1980**, *28*, 1018–1028.
12. Tamiazzo, S.; Macchiarella, G. Synthesis of cross-coupled filters with frequency-dependent couplings. *IEEE Trans. Microw. Theory Tech.* **2017**, *65*, 775–782.
13. He, Y.; Macchiarella, G.; Wang, G.; Wu, W.; Sun, L.; Wang, L.; Zhang, R. A direct matrix synthesis for in-line filters with transmission zeros generated by frequency-variant couplings. *IEEE Trans. Microw. Theory Tech.* **2018**, *66*, 1780–1789.
14. Szydłowski, Ł.; Leszczynska, N.; Mrozowski, M. Generalized Chebyshev bandpass filters with frequency-dependent couplings based on stubs. *IEEE Trans. Microw. Theory Tech.* **2013**, *61*, 3601–3612.
15. Williams, A.E.; Bush, W.G.; Bonetti, R.R. Predistortion technique for multicoupled resonator filters. *IEEE Trans. Microw. Theory Tech.* **1985**, *33*, 402–407.
16. Yu, M.; Tang, W.-C.; Malarky, A.; Dokas, V.; Cameron, R.; Wang, Y. Predistortion technique for cross-coupled filters and its application to satellite communication systems. *IEEE Trans. Microw. Theory Tech.* **2003**, *51*, 2505–2515.
17. Guo, C.; Shang, X.; Lancaster, M.J.; Xu, J. A 3-D printed lightweight X-band waveguide filter based on spherical resonators. *IEEE Microw. Wirel. Compon. Lett.* **2015**, *25*, 442–444.
18. Booth, P.; Lluch, E.V. Enhancing the performance of waveguide filters using additive manufacturing. *Proc. IEEE* **2016**, *105*, 613–619.
19. Swanson, D.G., Jr. Narrow-band microwave filter design. *IEEE Microw. Mag.* **2007**, *8*, 105–114.
20. Koziel, S.; Ogurtsov, S. *Antenna Design by Simulation-Driven Optimization*; Springer: Cham, Switzerland, 2014.
21. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–12 December 2020; Volume 33, pp. 1877–1901.
22. Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y.T.; Li, Y.; Lundberg, S.; et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv* **2023**, arXiv:2303.12712.
23. Zhang, Q.J.; Gupta, K.C.; Devabhaktuni, V.K. Artificial neural networks for RF and microwave design—From theory to practice. *IEEE Trans. Microw. Theory Tech.* **2003**, *51*, 1339–1350.
24. Liu, B.; Yang, H.; Lancaster, M.J. Global optimization of microwave filters based on a surrogate model-assisted evolutionary algorithm. *IEEE Trans. Microw. Theory Tech.* **2017**, *65*, 1976–1985.
25. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
26. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Front. Comput. Sci.* **2024**, *18*, 186345.
27. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 10–16 December 2023.
28. Liu, M.; Ene, T.-D.; Kirby, R.; Cheng, C.; Pinckney, N.; Liang, R.; Alben, J.; Anand, H.; Banerjee, S.; Bayraktaroglu, I.; et al. ChipNeMo: Domain-adapted LLMs for chip design. *arXiv* **2023**, arXiv:2311.00176.
29. Wu, H.; He, Z.; Zhang, X.; Yao, X.; Zheng, S.; Zheng, H.; Yu, B. ChatEDA: A large language model powered autonomous agent for EDA. *IEEE Trans. Comput.-Aided Des. Int. Cir. Syst.* **2024**, *43*, 3184–3197.
30. Lai, Y.; Lee, S.; Chen, G.; Poddar, S.; Hu, M.; Pan, D.Z.; Luo, P. AnalogCoder: Analog circuit design via training-free code generation. *arXiv* **2024**, arXiv:2405.14918.
31. Lu, H.; Li, G.; Chen, Q.; Gao, H.; Wang, S.; He, X.; Liu, Y.; Chen, G.; Li, N.; Qi, X.; et al. RFAmpDesigner: A self-evolving multi-agent LLM framework for automated radio frequency amplifier design. *arXiv* **2026**, arXiv:2605.10093.
32. Anthropic. Introducing the Model Context Protocol. Available online: https://www.anthropic.com/news/model-context-protocol (accessed on 1 May 2026).
33. Microsoft. Chat Participant API—Visual Studio Code Extension Guides. Available online: https://code.visualstudio.com/api/extension-guides/ai/chat (accessed on 15 May 2026).
34. Anthropic. Model Context Protocol Specification (2025-03-26). Available online: https://modelcontextprotocol.io/specification/2025-03-26 (accessed on 15 May 2026).
35. Yu, M.; Cameron, R.J.; Yu, J.; Wang, Y. Symmetrical realization for predistorted microwave filters. In *IEEE MTT-S International Microwave Symposium Digest, 2005*; IEEE: New York, NY, USA, 2005; pp. 245–248.
36. Qi, L.-H.; Xing, D.-Q.; Wang, R.; Li, Y.-N. Filter order and transmission zeroes calculation of cross-coupled resonator filters. *J. Phys. Conf. Ser.* **2020**, *1570*, 012068. https://doi.org/10.1088/1742-6596/1570/1/012068.
37. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. https://doi.org/10.1023/A:1008202821328.

38. Hong, J.-S.; Lancaster, M.J. *Microstrip Filters for RF/Microwave Applications*; John Wiley & Sons: New York, NY, USA, 2001. https://doi.org/10.1002/0471221619.
39. Anthropic. Equipping Agents for the Real World with Agent Skills. Available online: https://www.anthropic.com/news/skills (accessed on 15 May 2026).
40. Ruiz-Cruz, J.A.; Garai J.R.M.; Rebollar, J.M. Computer aided design of waveguide devices by mode-matching methods. In *Passive Microwave Components and Antennas*; Zhurbenko, V., Ed.; IntechOpen: Rijeka, Croatia, 2010; pp. 117–146. https://doi.org/10.5772/9403.
41. Conciauro, G.; Guglielmi, M.; Sorrentino, R. *Advanced Modal Analysis: CAD Techniques for Waveguide Components and Filters*; John Wiley & Sons: Chichester, UK, 2000.
42. Patzelt, H.; Arndt, F. Double-plane steps in rectangular waveguides and their application for transformers, irises, and filters. *IEEE Trans. Microw. Theory Tech.* **1982**, *30*, 771–776. https://doi.org/10.1109/TMTT.1982.1131135.
43. Collin, R.E. *Field Theory of Guided Waves*, 2nd ed.; IEEE Press: New York, NY, USA, 1991.
44. Mittra, R.; Lee, S.W. *Analytical Techniques in the Theory of Guided Waves*; ASIN B0006C0E1M; Macmillan: New York, NY, USA, 1971.
45. Zhao, P.; Wu, K.-L. Model-based vector-fitting method for circuit model extraction of coupled-resonator diplexers. *IEEE Trans. Microw. Theory Tech.* **2016**, *64*, 1787–1797.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.