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

Hüseyin N. Gülmez *, Yunus Koç, Agah O. Ertay, Bora Döken, Mesut Kartal

Posted Date: 25 May 2026

doi: 10.20944/preprints202605.1627.v1

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

 doi:10.20944/preprints202605.1627.v1

Article

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

Hüseyin N. Gülmez [1,*], Yunus Koç [2], Agah O. Ertay [3], Bora Döken [1] and Mesut Kartal [1]

[1] Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Sarıyer 34485, Istanbul, Türkiye
[2] Istanbul Technical University, Informatics Institute, Sarıyer 34485, Istanbul, Türkiye
[3] Faculty of Engineering and Architecture, Erzincan Binali Yildirim Univeristy, Yalnizbag Campus 24002, Erzincan, Türkiye
* Correspondence: gulmezh@itu.edu.tr

Abstract

Microwave bandpass filter design typically requires manual coordination of synthesis, analysis, simulation, and optimisation tools written in different languages and exchanged through ad-hoc data formats. We present FilterForge, a chat-driven VSCode environment that pulls these stages into one workflow. A local Model Context Protocol (MCP) server exposes deterministic Python implementations of coupling-matrix synthesis, uniform predistortion, topology reconfiguration, a genetic-algorithm transmission-zero selector, and a mode-matching engine for *H*-plane iris-coupled rectangular waveguide geometries. A GPT-class language model sits on top as a chat participant that translates natural-language requests into typed tool calls, and a bundled skill generates PyAEDT/HFSS notebooks for design-curve sweeps. We validate the mode-matching engine on a six-pole, 4 GHz Chebyshev filter against full-wave HFSS; the two agree closely at a small fraction of the runtime. The synthesis and analysis tools themselves are unchanged from standalone use; the chat layer only adds an interface. An end-to-end walk-through on a folded 6-pole WR-90 cross-coupled filter at 10 GHz exercises the whole workflow from the chat panel: GA-based order and transmission-zero selection from a stop-band mask, folded coupling-matrix synthesis, design-curve dimensioning, HFSS modelling through the generated PyAEDT notebook, and a brief manual-tuning loop. A wideband sweep of the resulting geometry places the first spurious passband well above the operating band.

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

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

 doi:10.20944/preprints202605.1627.v1

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, optimisation, 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 formalise. 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 optimisation were already speeding up filter synthesis and EM-driven tuning before the recent LLMs [20,23,24]. LLMs add a different capability to this picture: they carry broad domain knowledge that, when used to drive an *agentic* workflow, can do a substantial fraction of the design work itself [25,26]. Making this work in practice takes more than a strong model. The LLM has to live inside a purpose-built agent, 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 behaviour 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 VSCode extension.** FilterForge brings the full filter design flow, including synthesis scripts written in different languages, EM simulation, and optimisation, into a single environment. The designer interacts with the agent through VSCode'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 problems 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*: generalised-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).
   - *Optimisation*: genetic-algorithm-based filter order selection and transmission zeros placement.
   - *Export*: Touchstone `.s2p`, MATLAB `.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

 doi:10.20944/preprints202605.1627.v1

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, sweeps the relevant geometric parameters, calls the modal analysis over the sweep, and produces the design curves the practitioner uses to seed dimensional optimisation. 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 specialised, 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 optimisation, 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 judgement.

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 optimisation, can be invoked on its own or chained into an end-to-end session under chat control. 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-normalisation, and the matrix origin (whether obtained by synthesis or manual edit). Because this state persists across chat turns, the engineer can synthesise 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 serialises 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 normalised 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 MCP server [32,34] that exposes synthesis, uniform predistortion, matrix reconfiguration, the GA-based transmission-zero selector, and the mode-matching engine

 doi:10.20944/preprints202605.1627.v1

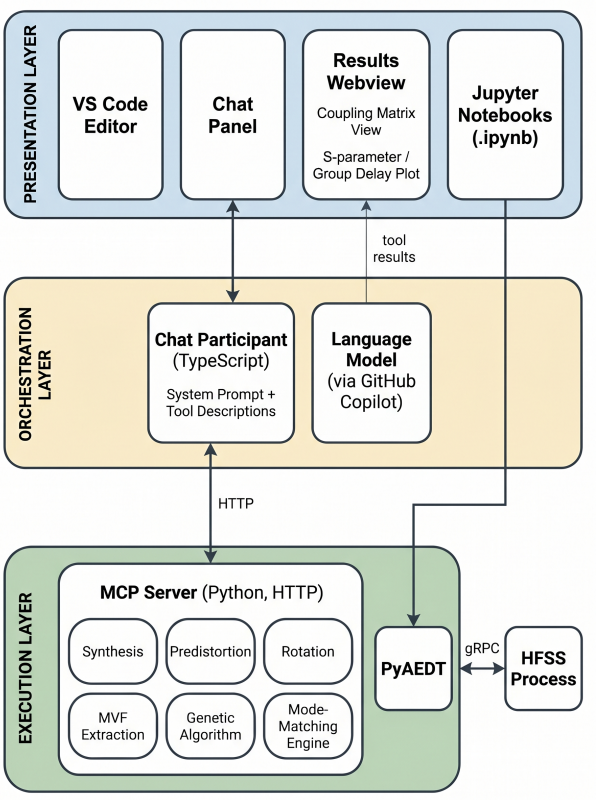


Figure 1. Three-layer architecture of FilterForge. The presentation layer (top) hosts the editor, chat panel, results webview, and notebooks. The orchestration layer (middle) is a chat participant that augments each user turn with a domain-specific system prompt and routes tool calls to a language model 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.

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

 doi:10.20944/preprints202605.1627.v1

sits in the standard VSCode secondary sidebar, so an engineer already comfortable with the editor will not have to learn any new interface element.

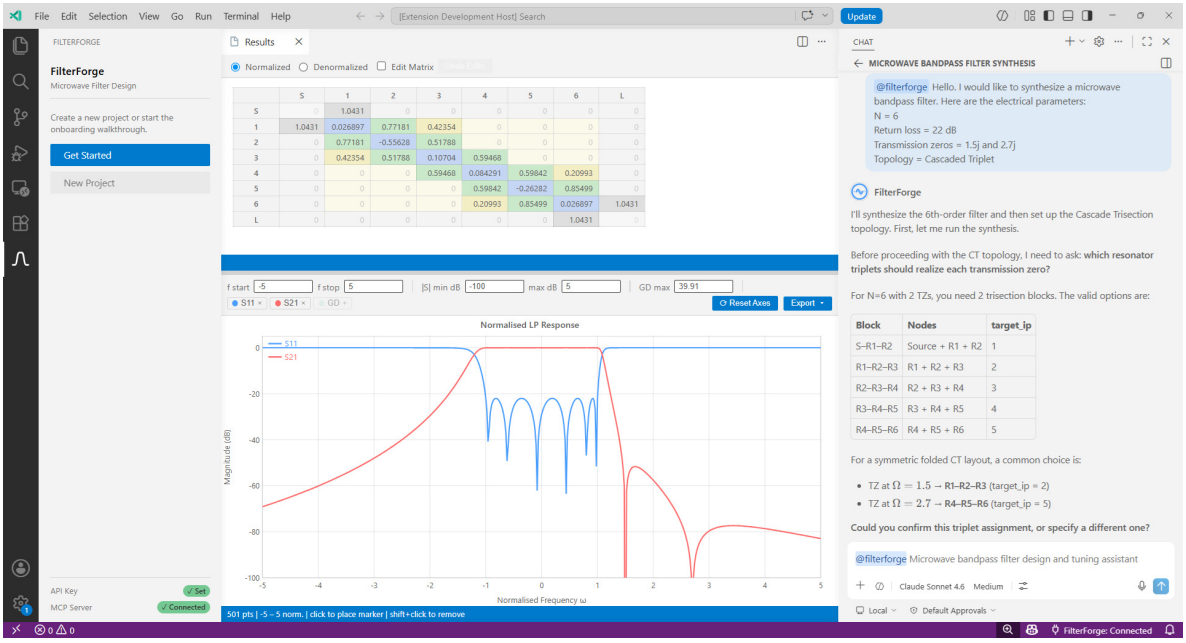
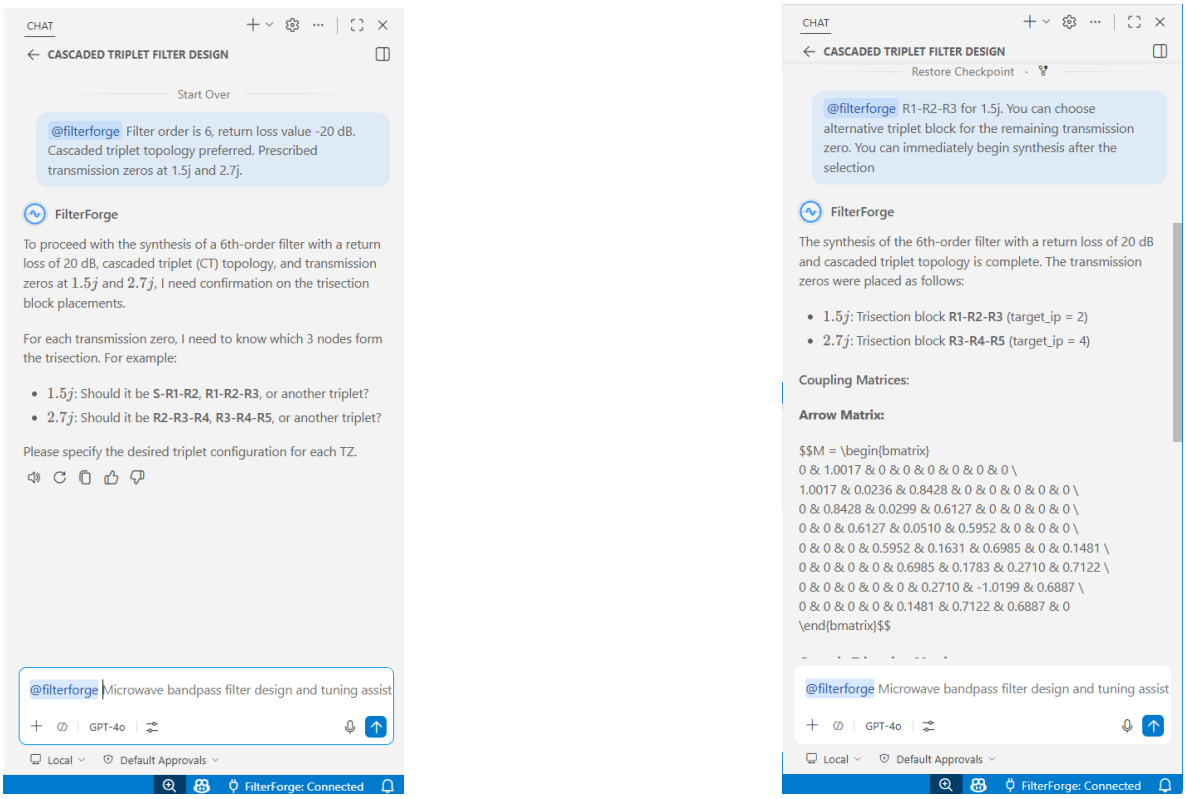


Figure 2. FilterForge during an active synthesis session. Left: primary sidebar with project entry points and live API-key / MCP-server status. Centre: results panel with the editable coupling-matrix grid and synchronised normalised S-parameter plot. Right: chat panel hosting the agent interaction.

Figure 3 traces a representative interactive synthesis as it unfolds in this chat panel. The user states the requirements in free form: a 6th-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 realise 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.

The session need not end once the synthesised 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 synthesised baseline.

 doi:10.20944/preprints202605.1627.v1



(**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 realise the two prescribed zeros. *(b)* The user resolves one triplet explicitly and delegates the other to the agent.

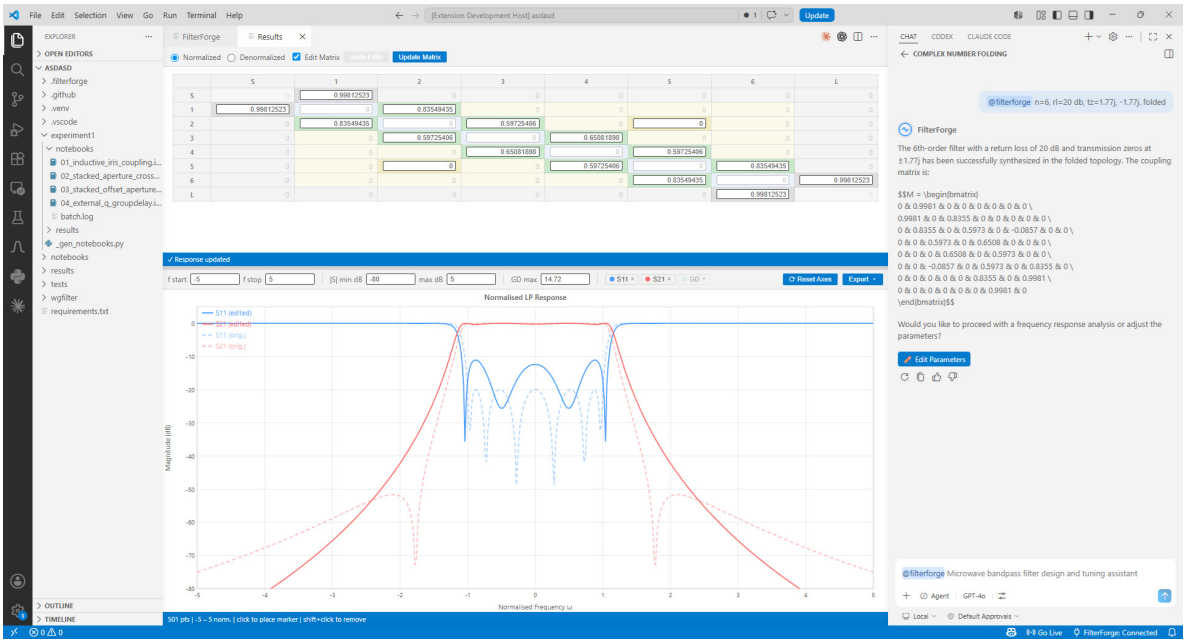


Figure 4. Editing a coupling in the interactive view. A 6th-order folded prototype with two finite transmission zeros is loaded into the grid, and the cross-coupling $M_{2,5}$ is then set to zero. The edited response is overlaid on the original one, and both transmission zeros vanish from S_{21}, as expected.

 doi:10.20944/preprints202605.1627.v1

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 synthesised in the normalised 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 generalised Chebyshev filter function

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

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 $j\lambda_k$ realises $\{E, F, P\}$ 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]

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

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

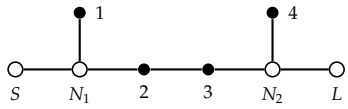
Although $\mathbf{M}_T$ is canonical, with every entry uniquely fixed by the specification, its transversal interconnection is rarely manufacturable: each resonator couples directly to both ports, with no inter-resonator paths, whereas physical waveguide layouts admit only a limited set of coupling routes. 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],

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

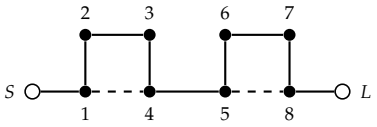
where $\mathbf{T}$ is a chain of Givens rotations $\mathbf{T}^{(i,j)}(\theta)$ with $S, L \notin \{i, j\}$. Each rotation annihilates one designated pivot entry of the matrix. Because zeroing one coupling generically reintroduces another, the rotation sequence must follow a *pivot schedule* specific to the target topology. The practical target

 doi:10.20944/preprints202605.1627.v1

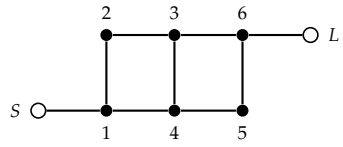
topologies for H-plane waveguide hardware are the inline, folded, extended-box, and cascaded extracted-pole forms, illustrated in Figure 5.



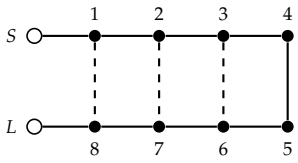
(**a**) Cascaded extracted-pole ($N = 4$, two NRNs)



(**b**) Cascaded quadruplet ($N = 8$)



(**c**) Extended-box ($N = 6$)



(**d**) Folded ($N = 8$)

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: each non-resonating node carries a dangling resonator that places one transmission zero independently of the main path. (**b**) Cascaded-quadruplet topology: 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: a two-path routing between adjacent box cells supports complex-axis zeros. (**d**) Folded canonical form: 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 Equalisation

A channel filter for a Ku-band satellite payload is synthesised 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 synthesised scattering response are reported below.

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 synthesised 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 follow.

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.

 doi:10.20944/preprints202605.1627.v1

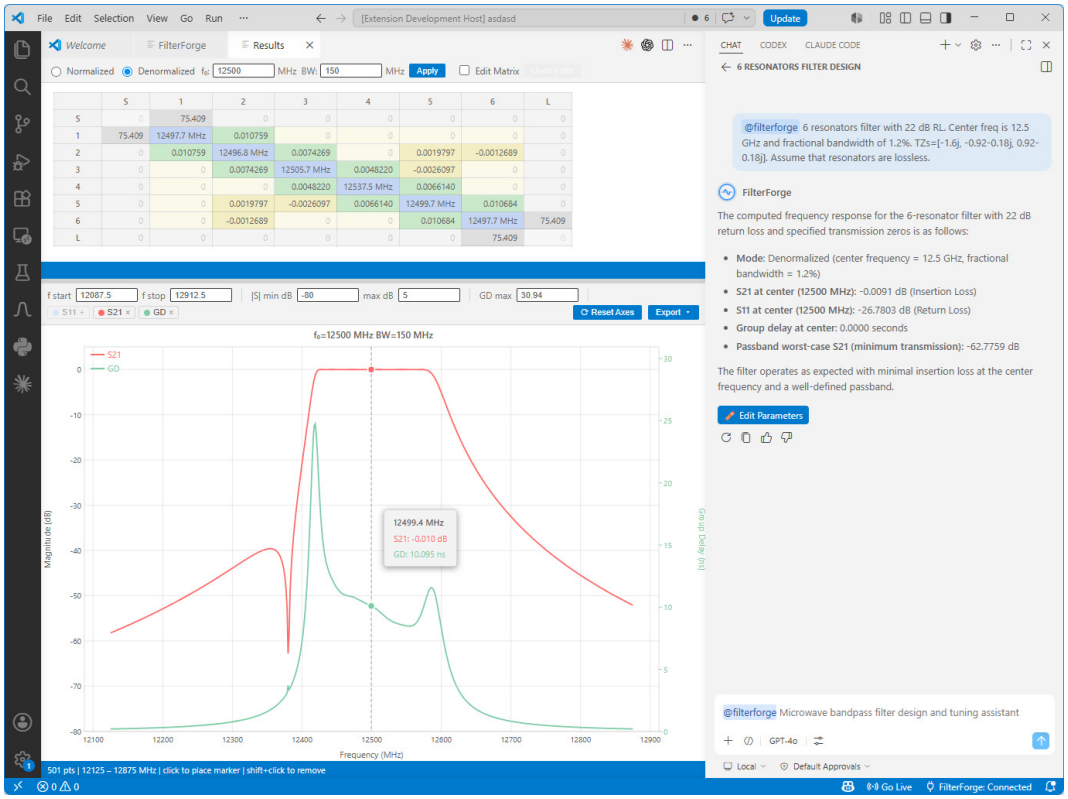


Figure 6. Synthesised scattering response and in-band group delay of the N=6 Ku-band folded channel filter with one imaginary-axis and one complex-conjugate transmission-zero pair.

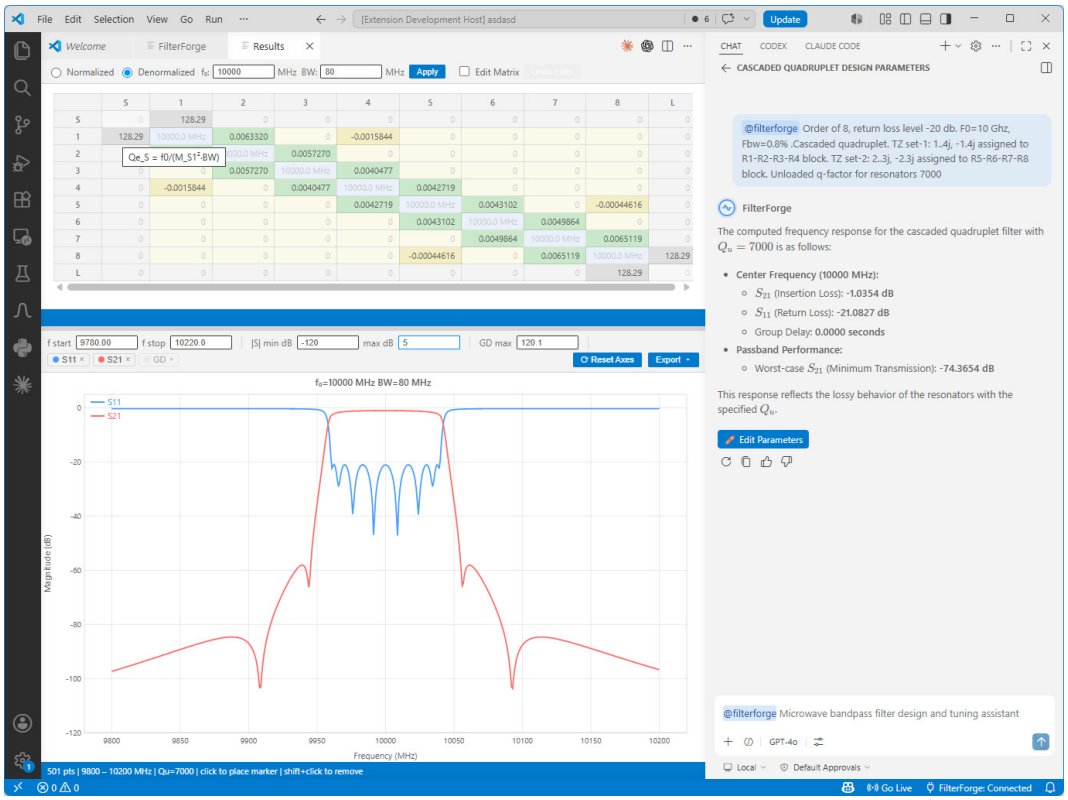


Figure 7. Synthesised scattering response of the N=8 X-band radar pre-selector in the cascaded-quadruplet topology, with two imaginary transmission-zero pairs at $\Omega = \pm 1.4j$ and $\Omega = \pm 2.3j$. The symmetric quasi-elliptic skirts deliver sharp rejection on either side of the passband, suppressing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

 doi:10.20944/preprints202605.1627.v1

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 synthesises a deliberately *worse* target response, chosen such that this translation restores the specification [1,15]. Fully cancelling σ_0 comes at a heavy in-band insertion-loss penalty, so in practice the compensation is only partial, toward an effective quality factor $Q_{\text{eff}} > Q_u$:

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

A representative choice from [1] is $Q_u = 4000 \rightarrow Q_{\text{eff}} = 10{,}000$, 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,

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

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 realise distinct physical networks. The selection is parametrised by [35]

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

and the two extremes pose a manufacturing trade-off for even-order folded topologies, summarised in Table 1. The choice is left to the designer: predistortion enters the pipeline as an optional stage activated by the triple $(Q_u, Q_{\text{eff}}, \mu)$.

Table 1. Manufacturing implications of the two extreme reflection-zero selections in even-order folded predistorted realisations.

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

3.2.1. Example Application: C-Band Satellite Payload Channel Filter

A 6th-order folded predistorted filter is synthesised 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 centre. The synthesised 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

 doi:10.20944/preprints202605.1627.v1

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 centre as the price of partial loss compensation toward $Q_{\text{eff}} = 6000$.

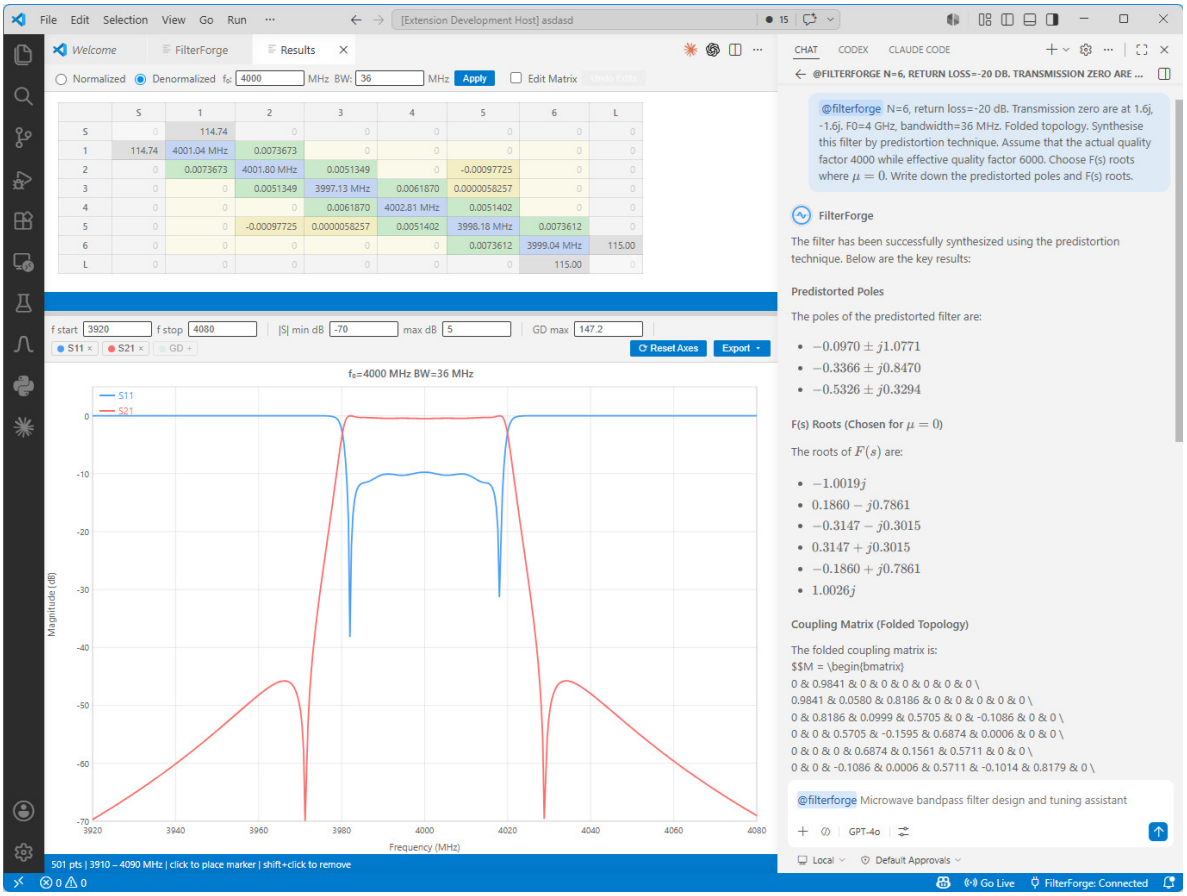


Figure 8. Synthesised coupling matrix and lossless scattering response of the 6th-order folded predistorted C-band channel filter ($f_0 = 4$ GHz, BW $= 36$ MHz, $RL = 20$ dB, zeros at $\Omega = \pm 1.6j$), with $Q_u = 4000$ compensated toward $Q_{\text{eff}} = 6000$ and reflection-zero selection $\mu = 0$. 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.

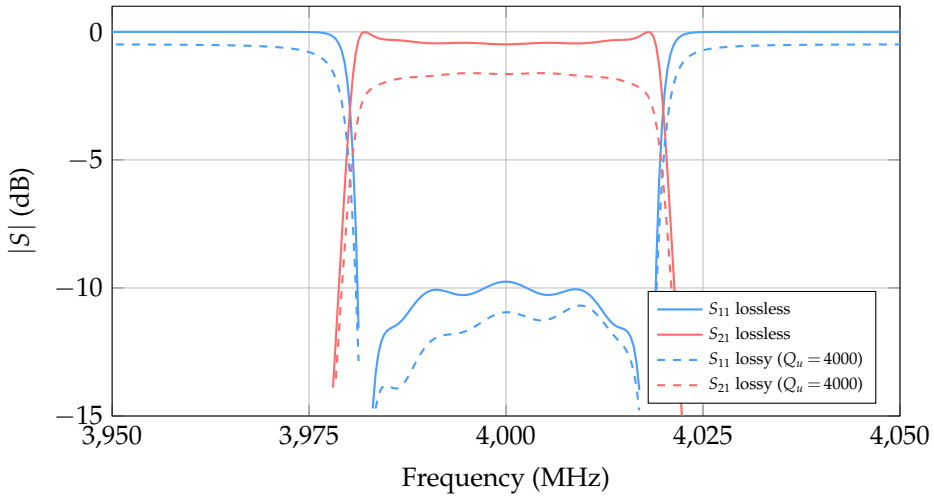


Figure 9. In-band close-up of the predistorted C-band filter, overlaying the lossless synthesised response (solid) with a forward analysis at $Q_u = 4000$ (dashed). Cavity dissipation flattens the in-band transmission and deepens the reflection notches a few dB back toward the -20 dB return-loss level.

 doi:10.20944/preprints202605.1627.v1

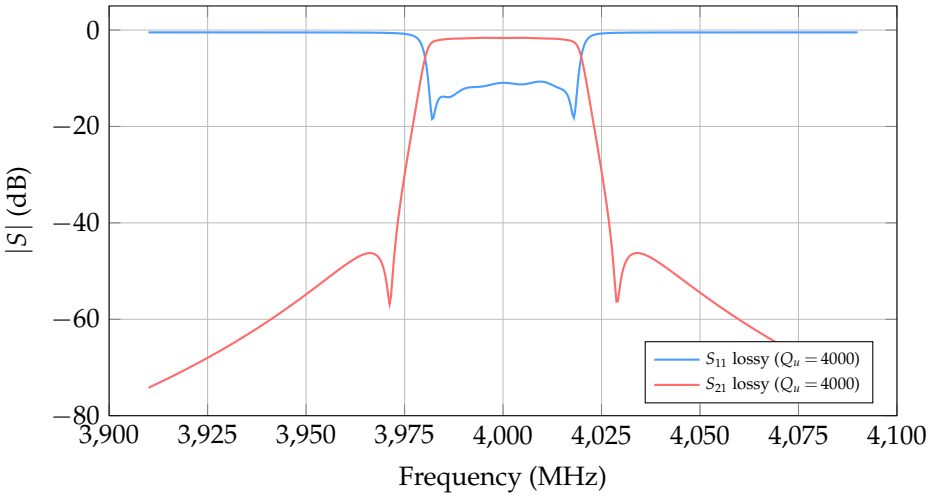


Figure 10. Frequency response view of the predistorted C-band filter under finite-Q operation ($Q_u = 4000$), showing the recovered passband. The insertion-loss penalty (2.31 dB) incurred at the passband centre is the price paid for the partial loss compensation toward $Q_{\text{eff}} = 6000$.

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, FilterForge automates the choice with a dedicated optimisation 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 equalisation) 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 minimise

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

where n_{sb} 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 generalised 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 12, 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 11 and overlays the result on the same template, as illustrated in Figure 13.

 doi:10.20944/preprints202605.1627.v1

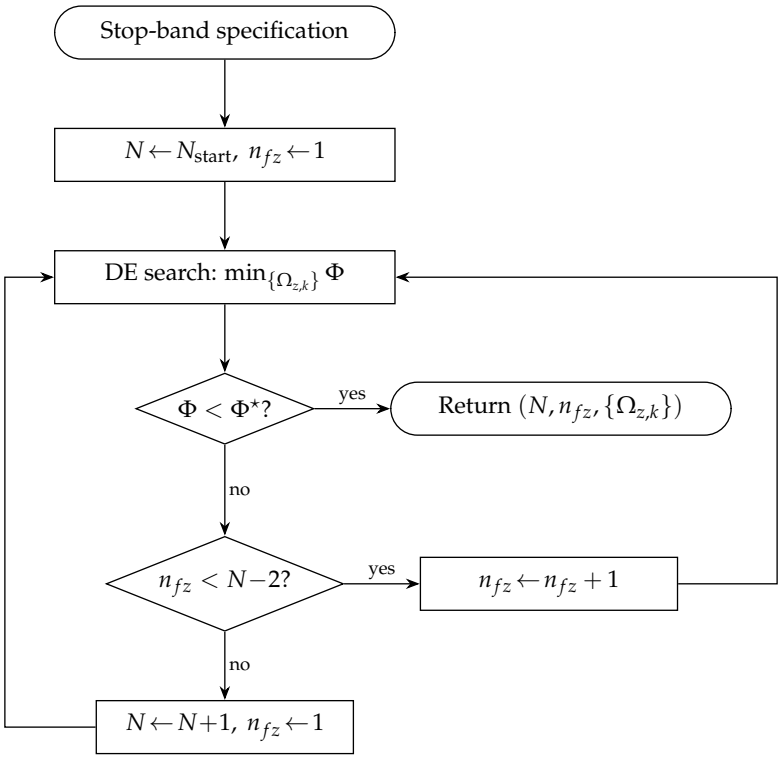


Figure 11. Outer-loop enumeration with inner differential-evolution search. The DE block returns the best $\{\Omega_{z,k}\}$ for the current (N, n_{fz}); if the resulting fitness falls below the feasibility threshold $\Phi^{\star}$ the configuration is accepted, otherwise n_{fz} is incremented or, on exhaustion, N is increased and the search restarted.

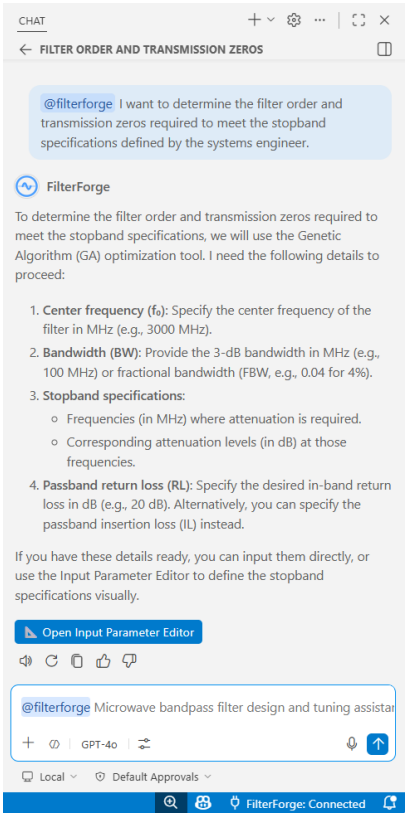


Figure 12. Chat request that opens the GA input parameter editor.

 doi:10.20944/preprints202605.1627.v1

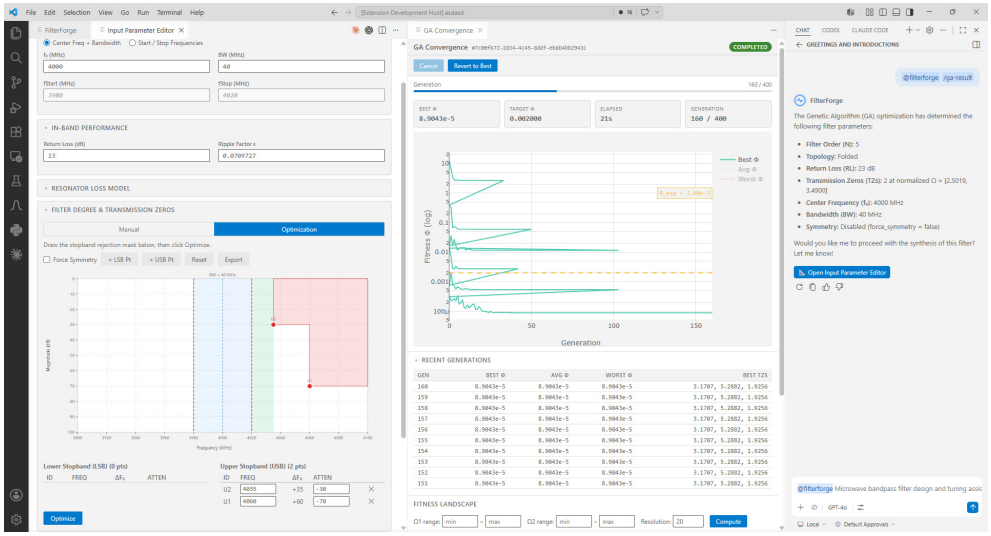


Figure 13. Input parameter editor with the interactive stop-band mask and the transmission zeros returned by the GA search.

3.3.1. Example Application: C-Band Channel with Stop-Band Mask

A representative C-band channel is specified by $f_0 = 4$ GHz, BW $= 40$ MHz, $RL = 25$ dB, and a symmetric stop-band requirement of 50 dB rejection at 3960 and 4040 MHz. With *Force Symmetry* enabled, the outer enumeration converges at $N = 6$ and $n_{fz} = 2$, with zeros at $\Omega_{z,1,2} = \pm 2.1107$. The synthesised coupling matrix 13 gives the scattering response of Figure 14, which meets the prescribed mask with margin.

$$M = \begin{bmatrix} 0 & 85.9 & 0 & 0 & 0 & 0 & 0 \\ 85.9 & 4000.7\ \text{MHz} & 0.00931 & 0 & 0 & 0 & 0 \\ 0 & 0.00931 & 4001.0\ \text{MHz} & 0.00600 & 0.00295 & 0.00050 & 0 \\ 0 & 0 & 0.00600 & 3990.3\ \text{MHz} & 0.00568 & 0 & 0 \\ 0 & 0 & 0.00295 & 0.00568 & 4000.3\ \text{MHz} & 0.00929 & 0 \\ 0 & 0 & 0.00050 & 0 & 0.00929 & 4000.7\ \text{MHz} & 85.9 \\ 0 & 0 & 0 & 0 & 0 & 85.9 & 0 \end{bmatrix} \tag{13}$$

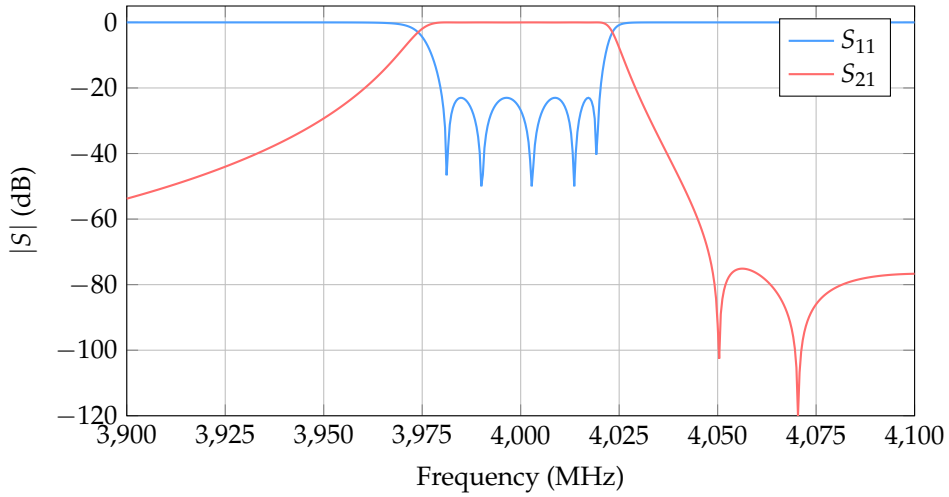


Figure 14. Scattering response of the coupling matrix synthesised from the GA-selected order and transmission zeros.

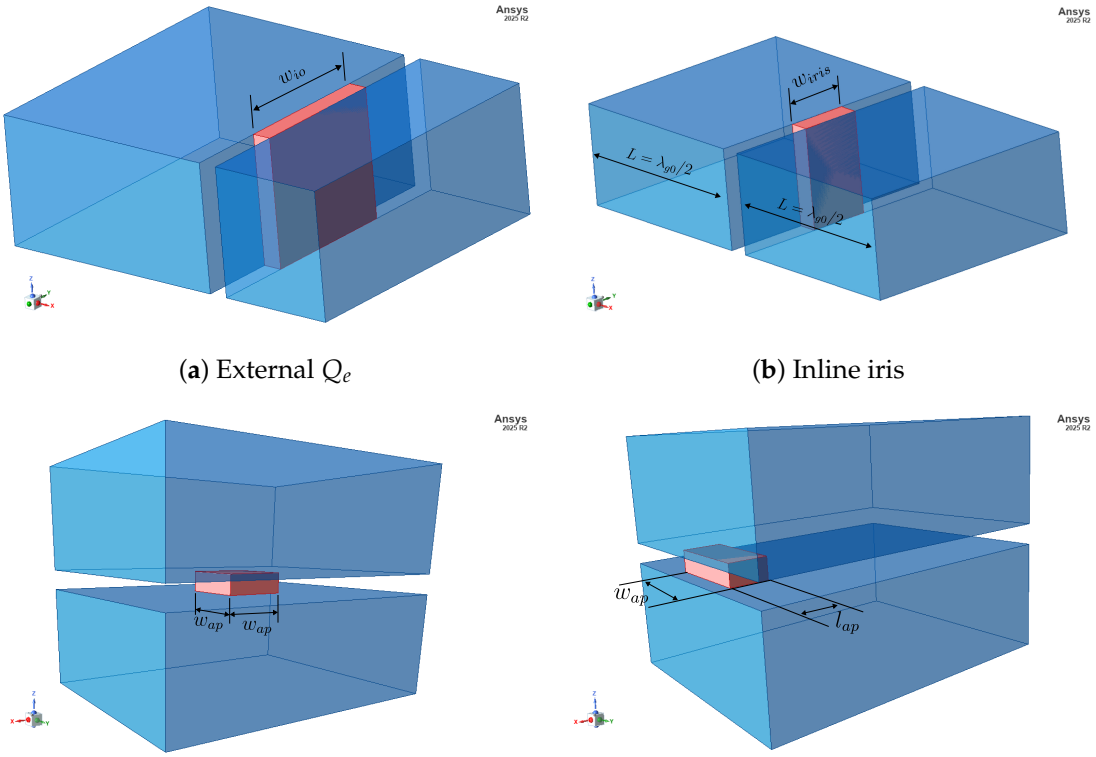
 doi:10.20944/preprints202605.1627.v1

4. Design-Curve Generation as an Agent Skill

The synthesised coupling matrix prescribes a set of normalised 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, and the curve is inverted at the synthesis target.

FilterForge exposes this workflow as an *agent skill* [39] rather than a deterministic MCP tool, invoked through the slash command `/design-curve` together with the desired centre frequency and waveguide cross-section, e.g. `/design-curve 10 GHz, WR-90, iris thickness 1.5 mm`. On invocation the agent reads the skill's `SKILL.md` (reproduced in Appendix B), copies the relevant notebook templates, and re-parametrises 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



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

(**c**) Centred 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**) Centred aperture, which produces capacitive (negative-sign) cross-coupling. (**d**) Offset aperture, which produces inductive (positive-sign) cross-coupling.

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

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.

 doi:10.20944/preprints202605.1627.v1

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 centre 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 the input iris and its cavity length are dimensioned together at inversion time.

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 synthesised target, fits a quadratic, and exposes an inversion call. 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 inversion 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 centred 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.

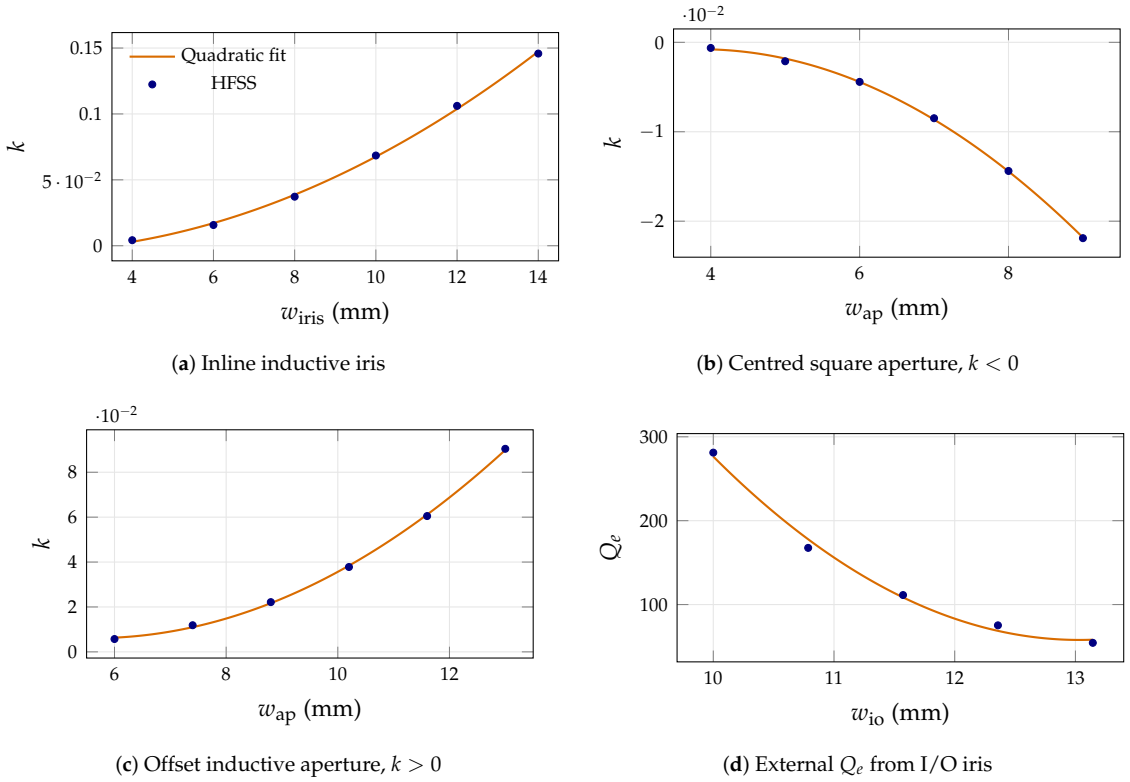
4.1. Example Application: Design-Curve Generation at 10 GHz in WR-90

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, centred at 10 GHz. A single invocation generated the four curves of Figure 16: the inline inductive iris over $w \in [4, 14]$ mm, the centred 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 parametrising 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 centred near 10 GHz in this waveguide-thickness combination can be dimensioned by inversion alone, without rerunning the full-wave sweep.

4.2. 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 polariser 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 generalised scattering matrix (GSM) [41,42].

 doi:10.20944/preprints202605.1627.v1



(**a**) Inline inductive iris

(**b**) Centred square aperture, $k < 0$

(**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).

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

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

 doi:10.20944/preprints202605.1627.v1

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.

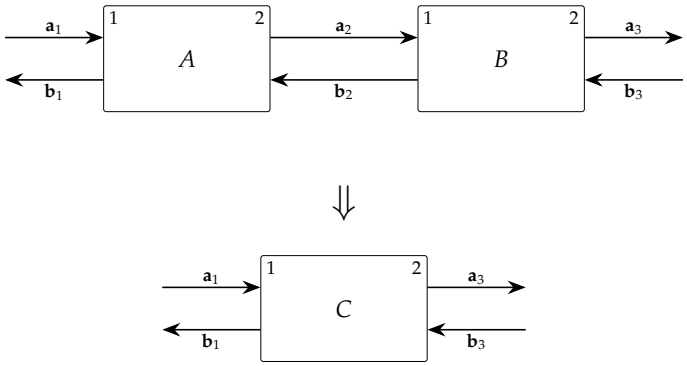


Figure 17. GSM cascade of two waveguide structures A and B that share a common interface, combined into a single composite structure C. The combination rule absorbs the intermediate modal amplitudes $\mathbf{a}_2, \mathbf{b}_2$ at the shared interface, so that C is described entirely by its external-port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

Example application: 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 $f_0 = 4\,\text{GHz}$, $\text{BW} = 40\,\text{MHz}$, $RL = -26\,\text{dB}$, no transmission zeros, in $a \times b = 50 \times 25\,\text{mm}$ rectangular waveguide with 2 mm-thick irises. The initial dimensions of Table 2 are taken from the design-curve inversions of Section 4. Figure 19 shows the resulting overlay: the synthesised 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.

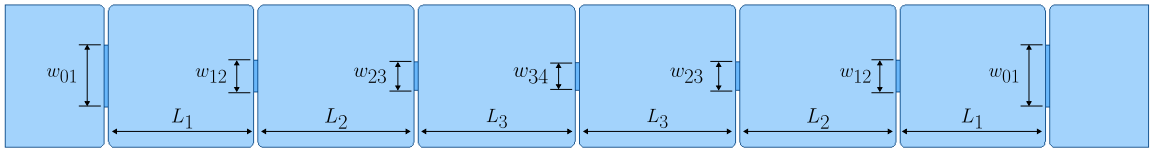


Figure 18. Top-view of the H-plane iris-coupled inline filter, with broad-wall dimension a, iris thickness t, iris widths $w_{k,k+1}$, and cavity lengths L_k labelled; the geometry is symmetric about the centre.

Table 2. Initial and optimised 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

 doi:10.20944/preprints202605.1627.v1

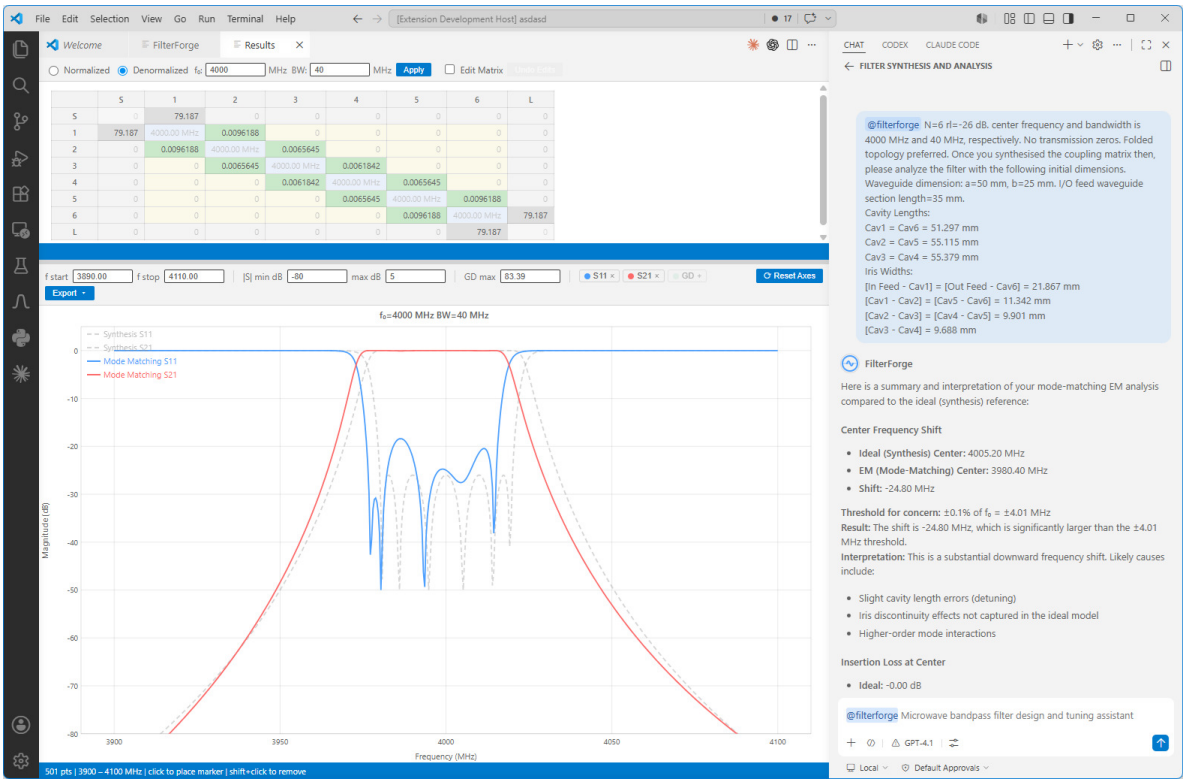


Figure 19. Interactive MM-driven tuning of the filter of Table 2 from the FilterForge chat panel: synthesised coupling-matrix response (dashed grey) and MM analysis of the user-supplied geometry (solid) on the same axes.

A few rounds of agent-driven adjustment produce the optimised dimensions of Table 2. To verify the MM engine itself against an independent reference, the optimised geometry was re-analysed 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.

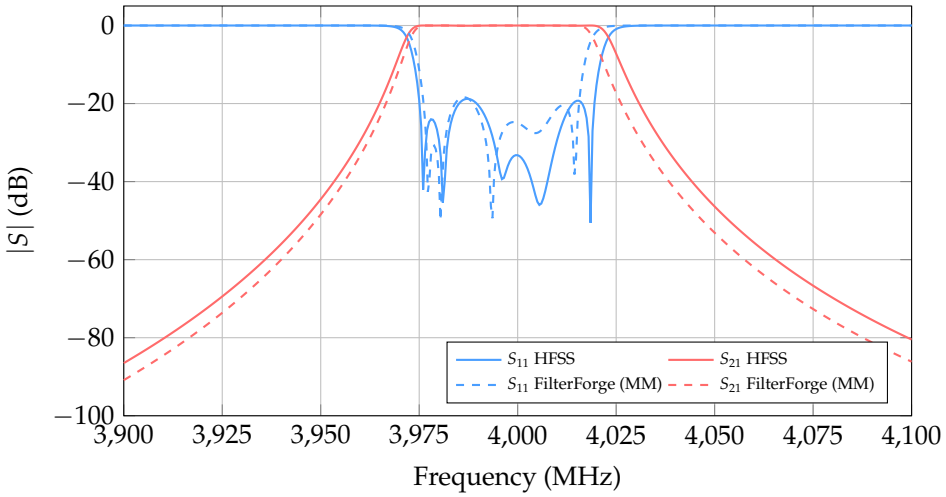


Figure 20. S-parameters of the optimised H-plane iris-coupled inline filter of Table 2: HFSS full-wave reference (solid) and FilterForge mode-matching analysis (dashed).

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 rigour of MM close the loop between synthesis and dimensioning inside a single interactive session.

 doi:10.20944/preprints202605.1627.v1

5. End-to-End Demonstration: a Folded 6-pole WR-90 Filter

The example is a narrowband WR-90 bandpass filter with passband centred 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 normalised lowpass axis, return-loss target $RL = -26\,\text{dB}$, and a folded topology, 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:

$$\begin{aligned} Q_e &= 80.252, \quad k_{12} = k_{56} = 9.47\times10^{-3}, \\ k_{23} &= k_{45} = 6.31\times10^{-3}, \quad k_{34} = 7.06\times10^{-3}, \\ k_{25} &= -1.21\times10^{-3}. \end{aligned} \tag{21}$$

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

Each entry of (21) 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 centred 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 initialised 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 centre, so the independent parameters listed in Table 3 fully describe the geometry; cavity inner-corner radii are neglected.

Table 3. Initial dimensions of the folded 6-pole filter (mm), synthesised at $f_0 = 10\,\text{GHz}$ in WR-90 with 1.5 mm wall thickness throughout. Only independent parameters are listed; mirror symmetry about the centre 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

 doi:10.20944/preprints202605.1627.v1

The MM engine of Section 4.2 does not yet support cross-coupled geometries, so the full filter is built and analysed directly in HFSS through a PyAEDT notebook emitted by the `/design-curve` skill. The resulting model, shown in Figure 21, 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 centred square aperture occupies the wall shared by cavity 2 and cavity 5.

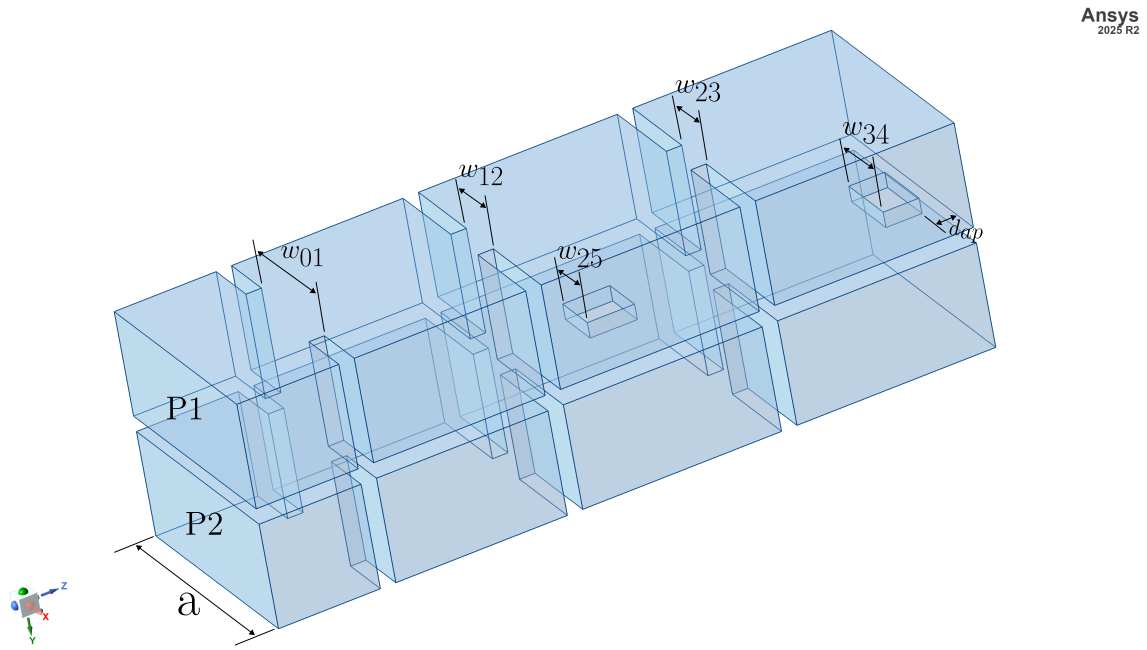


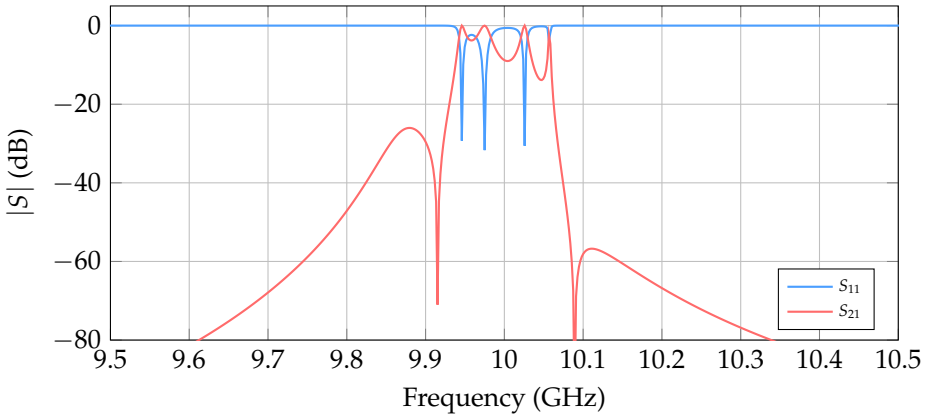
Figure 21. HFSS model of the folded 6-pole filter, generated automatically through the PyAEDT notebook emitted by the `/design-curve` skill from the dimensions of Table 3.

HFSS analysis of the initial geometry returns the response of Figure 22(a). 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 centred 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.

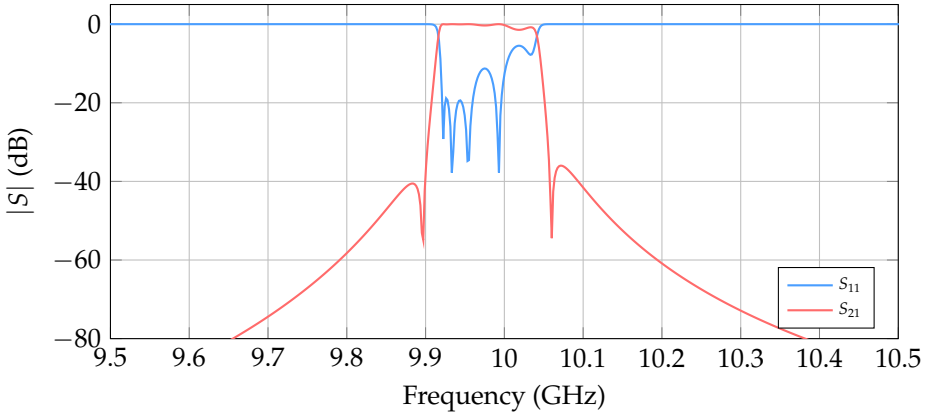
The two affected cavity pairs are dimensioned separately. The single inner-cavity length parameter is split in two, one for the $(2,5)$ pair and one for the $(3,4)$ pair, with each still initialised at 18.354 mm but tuned independently against the response. A short manual-tuning loop takes the response from Figure 22(a) to that of Figure 22(b): the two skirts come into symmetry, the transmission zeros are restored, and the in-band return loss settles at the synthesised level.

The wideband sweep of Figure 23 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 synthesised -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.

 doi:10.20944/preprints202605.1627.v1



(**a**) Initial dimensions



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

Figure 22. In-band S-parameters of the folded 6-pole filter. (**a**) Response from the design-curve inversions of Table 3, 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.

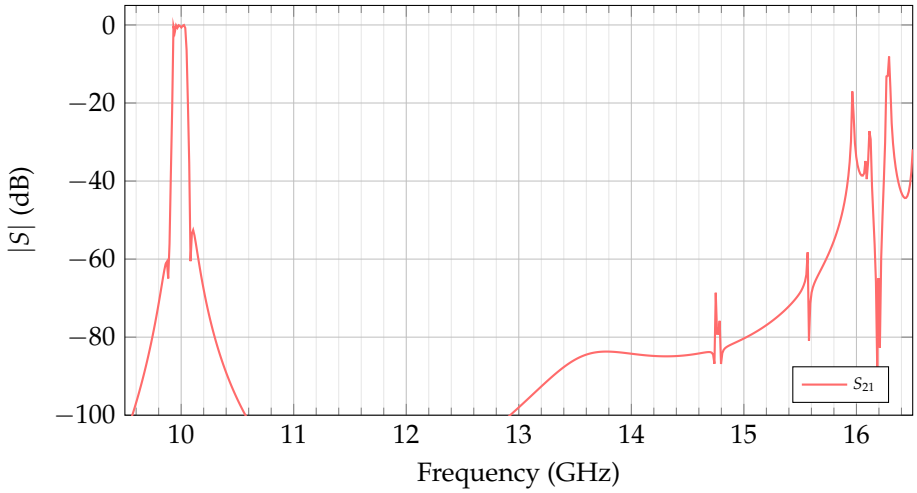


Figure 23. Wideband S-parameter response of the initial filter geometry. The first spurious passband, driven by higher-order cavity modes, lies around 16 GHz, well above the operating band at $f_0 = 10$ GHz.

6. Conclusions

We have introduced FilterForge, a semi-automated VSCode extension that places coupling-matrix synthesis, predistortion, reconfiguration, GA-based transmission-zero selection, and modal analysis under a single chat-driven agent backed by an MCP server. The mode-matching engine reproduces full-wave HFSS responses for a six-pole H-plane iris-coupled bandpass filter at a fraction of the cost,

 doi:10.20944/preprints202605.1627.v1

confirming that deterministic, domain-specific tools can supply the numerical backbone an LLM lacks while the agent contributes orchestration and interpretation. An end-to-end demonstration on a folded 6-pole WR-90 cross-coupled filter further exercises the whole workflow inside the chat panel: GA-driven order and transmission-zero selection from a stop-band mask, folded-matrix synthesis, design-curve dimensioning, HFSS modelling through the generated PyAEDT notebook, and a short manual-tuning loop that resolves the skirt asymmetry caused by the differential loading of the cross-coupling apertures. A wideband sweep of the resulting geometry places the first spurious passband well above the operating band. Future work will extend the mode-matching engine to cross-coupled and coaxial cavity filter topologies, and add coupling-matrix extraction and adaptive predistortion as new capabilities exposed through the FilterForge MCP server. The supported coupling-topology set will also be broadened so that the agent can reconfigure and synthesise a wider range of canonical and non-canonical filter forms within the same chat-driven environment.

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: A demo version of the proposed extension is available from the first author upon reasonable request.

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	Generalised 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

 doi:10.20944/preprints202605.1627.v1

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
VSCode	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 catalogue. Required inputs are listed; structured outputs are summarised by their principal fields. Units: MHz for physical frequencies, dB for return- and insertion-loss specifications, normalised lowpass Ω or complex s for prototype-domain quantities.

Tool	Purpose	Key inputs	Key outputs
Synthesis			
`chebyshev_poly_synthesis`	Generalised 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
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
Optimisation (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 (normalised 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

 doi:10.20944/preprints202605.1627.v1

Table A1. *Cont.*

Tool	Purpose	Key inputs	Key outputs
`cancel_ga_optimization`	Cancels a running GA	`run_id`	Cancellation acknowledgement
`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`	Serialises 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-flavoured 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.

```
---
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.
```

 doi:10.20944/preprints202605.1627.v1


```
- **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/conFigurepy` 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 = conFigurete101_length_mm(a,
   f0)` (do **not** reuse `conFigureCAVITY_LENGTH` unless `f0 == 10` GHz);
   `fed_len`/section lengths from `conFigurewaveguide_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.
```

 doi:10.20944/preprints202605.1627.v1

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), 10 March 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.

 doi:10.20944/preprints202605.1627.v1

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. Computer-Aided Design of Int. Cir. and Systems* **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 *Proc. IEEE MTT-S Int. Microw. Symp.*, San Francisco, CA, USA, June 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; ISBN 978-0-471-97069-9.
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; ISBN 978-0879422370.
44. Mittra, R.; Lee, S.W. *Analytical Techniques in the Theory of Guided Waves*; Macmillan: New York, NY, USA, 1971; ASIN B0006C0E1M.