A 2-3.5 GHz Automatically Tunable Bandpass Filter Using Deep Reinforcement Learning

***Abstract*— This paper presents a novel automatic tuning mechanism that eliminates hand-tuning and is suitable for electronically-tunable microwave filters. The proposed method is based on a deep Q-learning approach using physics-based filter characteristic parameters like resonant frequency, bandwidth, insertion loss, and return loss. The whole tuning process is done automatically and does not require any pre-tuning or human expertise. Furthermore, unlike single-frequency post-production tuning techniques, the presented methodology is applicable to continuously-tunable filters covering a wide frequency range. This method is experimentally demonstrated on a 2-3.5 GHz evanescent-mode electronically-tunable bandpass filter. To the best of our knowledge, this is the first demonstration of such an automatic tuning mechanism where the user can specify any frequency of interest and the filter tunes automatically to that frequency within the entire operating range of the filter.**

***Keywords*— automatic tuning, contactless cavity resonator, deep Q-learning, microwave tunable filter.**

I. Introduction

The main captivating feature of a typical software-defined cognitive radio transceiver is its ability to dynamically adjust its center frequency, bandwidth, and modulation type among many other characteristics [1]. For such an implementation several reconfigurable radio frequency (RF) components are required, including tunable filters preferably with automatic tuning. Evanescent (EVA) mode cavity resonator-based tunable filters have shown promising performance for the last decade. However, all these filters are manually tunable with various technologies such as piezoelectric disks, MEMS diaphragms, and commercial linear actuators [2]–[4].

While robotic tuning methodologies exist for screw-tuning filters, these techniques are slow (tuning takes several minutes) and are applicable to single-frequency post-production corrective tuning. They are typically not effective for applications that require continuously-tunable filters. The RoboCAT (Robotic Computer-aided Tuning) automation system developed by COM DEV for screw tuning microwave filters is such an example [5]. Similar single-frequency post-fabrication tuning techniques have bene presented in [6]–[8].

In this paper, we automate the filter tuning process by eliminating all hand-tuning for continuously-tunable filters using a deep Q-learning approach. Besides the main advantage of removing the need of a human expert during the tuning process, the presented deep Q-learning technique is powerful and capable of learning complex functions, making it robust to unintuitive tuning patterns, manufacturing imprecisions, and an increasing number of tuning elements. This work demonstrates the presented methodology on a two-pole widely tunable S-band filter as a proof-of-concept demonstrator. However, this algorithm is applicable to higher-order and higher-frequency filters.

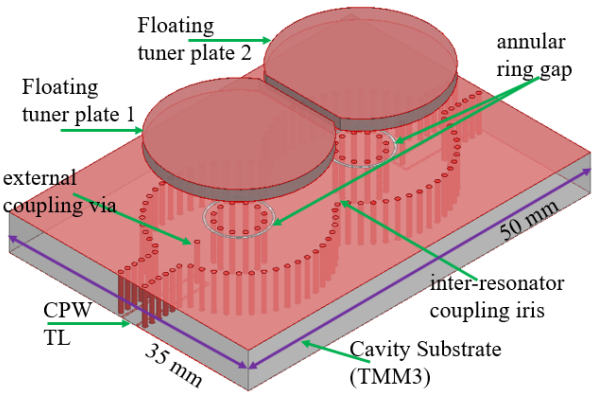

Fig. 1. HFSS simulated 3-D view of a second-order bandpass filter.

II. Filter Design and Tuning Process

A. Tunable Filter

A two-pole bandpass filter (BPF) at S-band has been designed using two contactless tunable EVA-mode cavity resonators [4]. Each resonator can be tuned individually by changing the air gap between the cavity substrate and the tuner plate. As a result, the filter can also be tuned by vertically moving the tuner plates. The coupling matrix used to design the two-pole filter with a 5% fractional bandwidth (FBW) is given below.

$$M = \begin{bmatrix} 0 & 0.8056 & 0 & 0 \\ 0.8056 & 0 & 0.708 & 0 \\ 0 & 0.708 & 0 & 0.8056 \\ 0 & 0 & 0.8056 & 0 \end{bmatrix} \quad (1)$$

A 3-D view of the second-order filter from HFSS simulation is shown in Fig. 1. The filter response type and FBW depend on the external and inter-resonator coupling values. The required external coupling to the grounded CPW transmission line (TL) and inter-resonator coupling are realized by using a coupling via and an iris, respectively. A full assembly of the fabricated second-order tunable BPF with M3L actuators is shown in

Fig. 2 (a). Two M3L actuators are used to vertically move the tuner plates, thus to tune the filter. The filter board and the tuner plates are fabricated using Rogers TMM3 substrates with thickness 0.200" and 0.060", respectively. The BPF can tune from 2 to 3.5 GHz with FBW of 5 to 6.5%. The measured insertion loss of the filter varies from 0.39 to 0.92 dB.

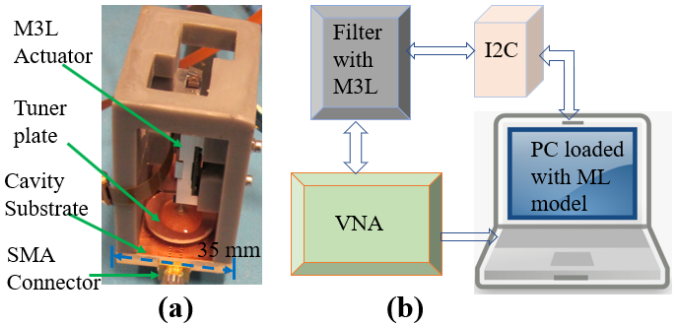

Fig. 2. (a) Complete assembly of a second-order filter with M3L actuators, (b) block diagram of automatic tuning system by machine learning approach.

B. Tuning Process

The manual tuning of this cavity filter is quite straightforward. First, an experienced operator connects the filter to a vector network analyzer (VNA) and manually inputs commands to change the air gap between the tuning plate and the cavity substrate based on their knowledge to tune the filter. There are two tuning plates for the second-order filter that need to change to specific positions to tune the filter at the desired frequency. This tuning process becomes more complex and time consuming for higher-order filters. Besides, manual tuning does not serve the actual purpose of the tunable filter for digital control.

Fig. 2(b) depicts the automatic tuning mechanism proposed in this paper. Once the filter is connected to the VNA, the S-parameters of the filter are measured by the VNA and read by a MATLAB script for the initial positions of the tuning plates via a GPIB cable. Then the machine learning (ML) algorithm, written in Python, determines the necessary gap changes for each resonator to improve the scattering characteristics towards the desired filter response. These calculated changes are sent to the M3L by USB through a Total Phase Aardvark I2C adapter. Each actuator moves to its new position, at which point MATLAB reads the updated filter response from the VNA. The process is automatically repeated until the desired S-parameter of the filter are obtained. In the whole process, the user only specifies the desired frequency and the filter is tuned automatically to that frequency.

III. Automatic Tuning Based on Machine Learning

In a reinforcement learning (RL) approach, we define our task through a reinforcement learning *environment*, and our tuning model is defined as an *agent* exploring the environment. The environment contains *states*, and *actions* from the agent cause transitions between states. The agent will learn to accomplish a task by following a *policy*, a mapping from states to actions. This policy is adjusted based on the amount of *reward* given for certain actions taken in certain states depending on how beneficial those actions were.

Q-learning is a method of training an agent to learn the optimal policy for accomplishing the task in the environment. A Q-learning agent analyzes the "quality" of every state-action pair to choose the best action given a state. Formally, a Q-value for state, s and action, a is computed as

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a') \tag{2}$$

where $(s,\ a)$ is the initial state-action pair and $(s',\ a')$ is the next iteration state-action pair. Here, $\gamma \in (0,1)$ is the decay rate, so decisions far into the future have less impact on the Q-value. This equation is known as the Bellman equation [9].

In a standard Q-learning environment, we can simply initialize all Q-values to some constant and update them iteratively according to the Bellman equation as the agent explores the environment. However, since the state space is often too big to deterministically evaluate every Q-value, we approximate $Q(s,a)$ with $\hat{Q}(s,a;\theta)$, where $\hat{Q}$ is a neural network called a deep Q-network [10].

A. Environment

We define our reinforcement learning environment using the following states, actions, and rewards.

- **States:** States are in the form of 3-tuples, (f, g_1, g_2), where f is the user-specified desired center frequency and g_1 and g_2 are the two gaps for the two tuning plates of the second-order BPF.
- **Actions:** There are six actions labeled 0 through 5. Actions 0 through 3 represent tuning gap 1 or 2 either up or down. Action 4 is an idle action, resulting in no change in state. Action 5 randomly reassigns the gaps, which improves exploration during the learning process and allows for quicker convergence if the gaps are extremely off.
- **Rewards:** The reward at each iteration is determined by computing the error between the ideal S-parameters and the measured S-parameters which is described in detail in the following subsection.

B. Reward Function

The measured S-parameters (S_{11} and S_{21}) of the filter are passed back to the environment. Instead of using the entire curves, which is computationally expensive and may result in overfitting, we only look at 7 specific points in each curve. The 7 points (M_1 through M_7) for S_{11} and S_{21} are at the desired center frequency (f), $f \pm bw/2$, $f \pm (bw/2 + bw_{max})$, and $f \pm (bw/2 + 4 * bw_{max})$.

We compute losses for S_{11} and S_{21} as $L^{(1)}$ and $L^{(2)}$ as

$$L^{(1)} = -\lambda_1 \log(1 - M_1) - \sum_{i=2}^{7} \lambda_i \log(M_i) \tag{3}$$

$$L^{(2)} = -\sum_{i=1}^{3} \lambda_i \log(M_i) - \sum_{i=4}^{7} \lambda_i \log(1 - M_i) \tag{4}$$

$$L = L^{(1)} + L^{(2)} \tag{5}$$

Here, L represents the total loss and λ_i represents a weight for the ith point of the curves, chosen as a hyperparameter. This loss resembles cross-entropy loss, where we attempt to maximize the values at certain points and minimize the values at others to match a perfect curve. A perfect S_{11} curve for this loss calculation has a magnitude of $-\infty$ at M_1 and a magnitude of 0 elsewhere. A perfect S_{21} curve has a magnitude of 0 at M_1, M_2, and M_3 and a magnitude of $-\infty$ elsewhere.

Let L_t define the loss computed at iteration t, where $L_0 = 0$. Using this loss, our reward function is computed as

$$r_t = L_{t-1} - L_t \tag{6}$$

which is simply the improvement in loss since the last iteration. This ensures that we are rewarding improvements in the curves and penalizing bad actions.

C. Intelligent Tuning Algorithm

Tuning begins by initializing gaps uniformly at random within the domain of all air gaps. The air gap domain is defined by the filter designer, and in this specific filter, it was from 50 to 450 μm. Each iteration t of our tuning procedure proceeds as follows.

1. Given current state s_t, choose an action from the network using an annealing ε-greedy strategy. That is, with probability $\varepsilon = \frac{1}{\log_2(t+1)}$ (which decreases over time), choose an action at random. Otherwise, choose
$$a_t = \arg\max_a \hat{Q}(s_t, a; \theta) \tag{7}$$
which is the action with the highest Q value as recommended by the network.
2. Execute chosen action and receive next state s_{t+1} and reward r_t as feedback. Store tuple (s_t, a_t, s_{t+1}, r_t) into experience-replay memory, E.
3. Sample B, a batch of b tuples from E. Here, b is a model hyperparameter set to 64 in our experiments. The most recent $\frac{b}{8}$ tuples added to E are guaranteed to be sampled to ensure that newer information is used in learning, and the remaining tuples are sampled uniformly at random with replacement from E.
4. Perform a forward pass with $\hat{Q}$ on the samples of B. Perform gradient descent on the network with a loss function, which is define as
$$loss = \left[r_t + \gamma \max_a \hat{Q}(s_{t+1}, a; \theta) - \hat{Q}(s_t, a_t; \theta)\right]^2 \tag{8}$$
with hyperparameter γ. Note that this is simply the mean square error (MSE) between the predicted Q value and the expected Q value according to the Bellman equation.

Our network architecture consists of a multilayer perceptron (MLP) with input size 3, one hidden layer of size 64, and output size 6.

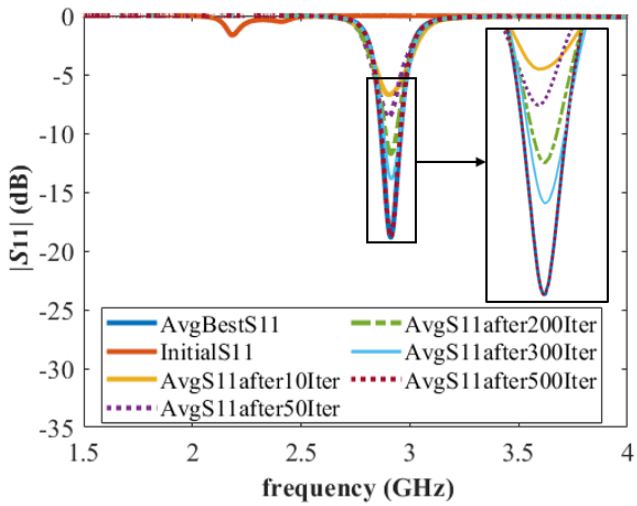

Fig. 3. Average S_{11} measurement responses after N number of iterations for 20 trials.

IV. EXPERIMENTAL VALIDATION

To evaluate the proposed automated tuning mechanism, a two-pole BPF has been manufactured using the Printed Circuit Board (PCB) technology. A picture of the fabricated filter is shown in Fig. 2(a), where two M3L actuators are used to provide the vertical movement of the tuners. The filter responses were measured using a Keyshight N5230C VNA. The communication between the filter and tuning system is described in Section 2B.

Fig. 3 and Fig. 4 show the average measurement responses per iteration over 20 trials for S_{11} and S_{21}, respectively. We set the desired frequency at 2.9 GHz and ran the Python ML code for 20 trials, each with 500 iterations. From Fig. 3 and Fig. 4, we observe that within 10 iterations, the filter is tuned from an arbitrary frequency position to the desired frequency position. Within 50 iterations, the return loss drops below

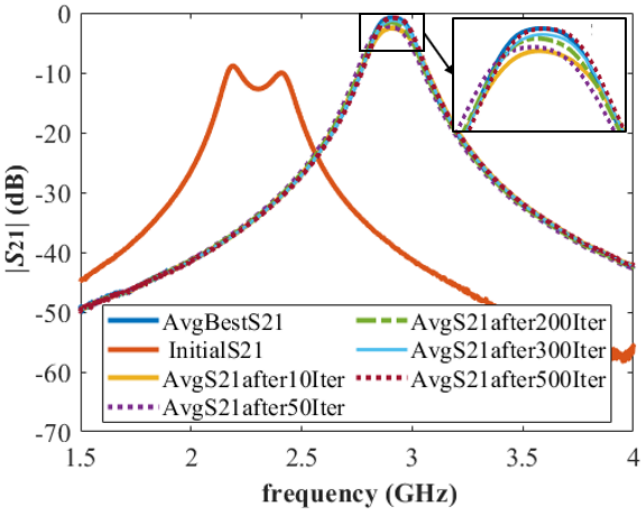

Fig. 4. Average S_{21} measurement responses after N number of iterations for 20 trials.

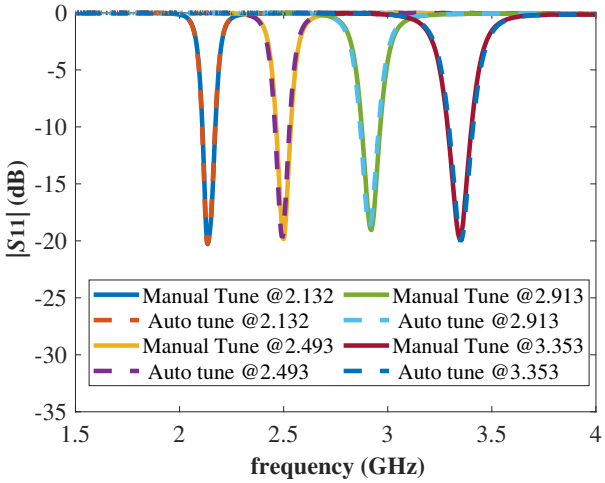

Fig. 5. Comparison of measured S_{11} responses by the auto-tune and hand-tune approaches at four different example frequencies.

$-10dB$, a reasonable standard for most applications. The next few hundred iterations improve insertion loss and drop the return loss even further.

We have tested the tuning algorithm for the entire tuning range of the filter. Fig. 5 and Fig. 6 show the measured S_{11} and S_{21} at four example frequencies within the filter's range. For each desired frequency, we ran the code for 300 iterations. At the end of 300 iterations for each trial, the filter is tuned perfectly at the desired location. A perfect agreement between the automatic tuning and manual tuning results is shown in Fig. 5 and Fig. 6.

We have also tested the convergence rate of the tuning algorithm for the filter. Fig. 7 shows the total S-parameter loss and cumulative reward versus the number of iterations. From the Fig. 7, it is clearly evident that the loss and reward are quite stable after 250 iterations, indicating that the filter can

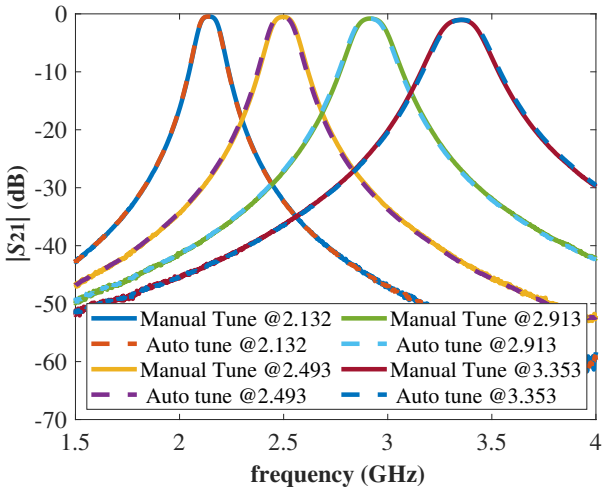

Fig. 6. Comparison of measured S_{21} responses by the auto-tune and hand-tune approaches at four different example frequencies.

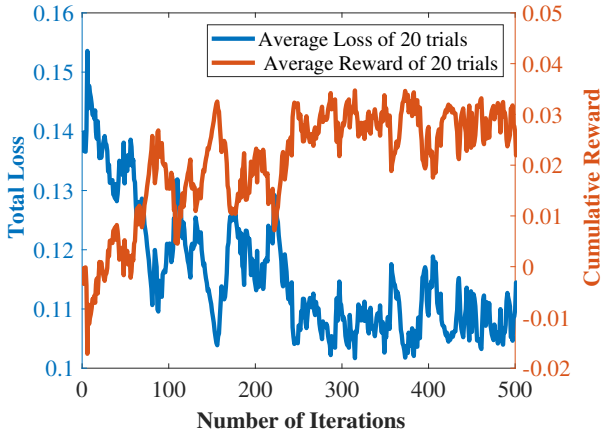

Fig. 7. Total loss and Cumulative reward versus the number of iterations.

reliably be tuned to its best settings within 250 iterations.

V. Conclusion

This work presents a novel automatic tuning mechanism for continuously-tunable filters suitable for wide tuning range applications. The tuning mechanism is based on a deep Q-learning approach that takes into account physically-meaningful parameters such as resonant frequency, bandwidth, insertion loss, and return loss. Furthermore, it does not require any expert human input or pre-tuning. Measured S-parameter responses show excellent agreement between hand-tuned and auto-tuned states. The practicality of this approach is expected to enable a new generation of automatically-tunable filters based on physics-based machine-learning algorithms.

References

[1] I. F. Akyildiz, W. Lee, M. C. Vuran, and S. Mohanty, "A survey on spectrum management in cognitive radio networks," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 40–48, April 2008.

[2] T. Lee, B. Lee, S. Nam, Y. Kim, and J. Lee, "Frequency-tunable tri-function filter," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 11, Nov 2017.

[3] P. Adhikari, W. Yang, and D. Peroulis, "A 20–26.5-ghz pcb bandpass filter tuned with contactless tuners," *IEEE Microwave and Wireless Components Letters*, vol. 29, no. 8, pp. 513–515, Aug 2019.

[4] M. Abdelfattah and D. Peroulis, "High- q tunable evanescent-mode cavity siw resonators and filters with contactless tuners," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 9, pp. 3661–3672, Sep. 2019.

[5] C. D. Ltd., "Robotic computer-aided tuning," in *Microwave Journal*, March 2006.

[6] R. V. P. Harscher and S. Amari, "Automated test and tuning system for microwave filters," in *IEEE MTT-S International Microwave Symposium Digest (Cat. No.01CH37157)*, vol. 3, Phoenix, AZ, 2001, pp. 1543–1546.

[7] G. Pepe, F. . Gortz, and H. Chaloupka, "Computer-aided tuning and diagnosis of microwave filters using sequential parameter extraction," in *2004 IEEE MTT-S International Microwave Symposium Digest (IEEE Cat. No.04CH37535)*, vol. 3, June 2004, pp. 1373–1376 Vol.3.

[8] V. Miraftab and R. R. Mansour, "Automated microwave filter tuning by extracting human experience in terms of linguistic rules using fuzzy controllers," in *2006 IEEE MTT-S International Microwave Symposium Digest*, June 2006, pp. 1439–1442.

[9] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602