Supplementary Material

Dimitrios C. Tzarouchis,[1†] Brian Edwards,[1†] Nader Engheta[1*]

[1]Department of Electrical and Systems Engineering,
School of Engineering and Applied Sciences,
University of Pennsylvania, Philadelphia, 19104, U.S.A.
[†]These authors contributed equally to this work.
[*]To whom correspondence should be addressed; e-mail: engheta@seas.upenn.edu

July 31, 2024

1 Details for the root finding algorithm

In the following the lowercase quantities are vectors, the capitalized ones are matrices while Greek letters denote scalars - the subscripts follow a logical notation. The problem statement for the root finding procedure is

$$f(\mathbf{z}) = 0 \tag{1}$$

find $\mathbf{z} \in \mathbb{C}^{m\times 1}$ that satisfies the above equation (roots) of $f \in \mathbb{C}^{m\times 1}$. The above problem is solved using Newton's method for finding the root of a vector polynomial function [1]. For example we have

$$f(\mathbf{z}) = \begin{pmatrix} f_1(z_1,z_2,z_3,z_4,z_5) \\ f_2(z_1,z_2,z_3,z_4,z_5) \\ f_3(z_1,z_2,z_3,z_4,z_5) \\ f_4(z_1,z_2,z_3,z_4,z_5) \\ f_5(z_1,z_2,z_3,z_4,z_5) \end{pmatrix} \tag{2}$$

with $z_{1\cdot 5} \in \mathbb{C}$ or (equivalently)

$$f_1(\mathbf{z}) = (z_1 - r_1)(z_2 - 4.2i)(z_3 + 2)(z_4 - 5i)(z_5 - 3.5) \tag{3}$$
$$f_2(\mathbf{z}) = (z_1 - 3.9)(z_2 - r_2)(z_3 + 2.5i)(z_4 - 3.2i)(z_5 - 4.2) \tag{4}$$
$$f_3(\mathbf{z}) = (z_1 + 5.2i)(z_2 - 4)(z_3 - r_3)(z_4 - 4i)(z_5 - 7.1) \tag{5}$$
$$f_4(\mathbf{z}) = (z_1 - 3)(z_2 - 7i)(z_3 + 4)(z_4 - r_4)(z_5 - 5i) \tag{6}$$
$$f_5(\mathbf{z}) = (z_1 - 5.2i)(z_2 - 4)(z_3 + 4.75i)(z_4 - 8)(z_5 - r_4) \tag{7}$$

where

$$r = \frac{1}{4}\left(s_1 + c_1 i, -s_1 + c_1 i, -s_2 - c_2 i, s_2 - c_2 i, 1i\right)^T \tag{8}$$

with $c_1 = \cos(2\pi/5)$, $c_2 = \cos(\pi/5)$, $s_1 = \sin(2\pi/5)$, and $s_2 = \sin(4\pi/5)$. The point corresponds to the vertices of a regular pentagon. For the evaluation of Newton's method we need to calculate the Jacobian matrix, i.e., $J_{ij} = \frac{\partial f_i}{\partial x_j}$ (here i and j are indexes) or

$$J_f(\mathbf{z}) = \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \cdots & \frac{\partial f_1}{\partial z_5} \\ \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} & \cdots & \frac{\partial f_2}{\partial z_5} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_5}{\partial z_1} & \frac{\partial f_5}{\partial z_2} & \cdots & \frac{\partial f_5}{\partial z_5} \end{pmatrix} \tag{9}$$

therefore the root can be found as

$$\mathbf{z}_{n+1} = \mathbf{z}_n - \alpha J_f^{-1}(\mathbf{z}_n) f(\mathbf{z}_n) \tag{10}$$

where α is a relaxation constant. Here we used $\alpha = 0.2$. In terms of an algorithm, we have the following routine

Algorithm 1 Root finding with Newton's method

```
1: Initial guess for $\mathbf{z}_1 = [0,0,0,0,0]$
2: for $n = 1, \ldots, m$ do
3:     $J_f(\mathbf{z}_n)$
4:     $\alpha_\lambda = \frac{2}{\lambda_{min}+\lambda_{max}}$    ▷ Scaling factor: $\lambda_{mim/max}$ are the min/max eigenvalues of $J_f(\mathbf{z}_n)$
5:     $K_n = I - \alpha_\lambda J_f(\mathbf{z}_n)$    ▷ Kernel that is fed to DCM machine
6:     $\mathbf{d}_n = J_f^{-1}(\mathbf{z}_n) f(\mathbf{z}_n)$    ▷ Compute matrix inverse with the DCM machine
7:     $\mathbf{z}_{n+1} = \mathbf{z}_n - \alpha \mathbf{d}_n$
8: end for
```

2 Details on the inverse design algorithm

In this section we present the details for the inverse design algorithm implemented in text. The algorithm consist of a part of the DDA methodology for the quantification of the problem and an its adaptation to a Lagrange formalism for solving the require inverse scattering problem, the determination of the permittivity of the scatterers. Note that both methods are arguably the simplest methods to follow, since they offer an intuitive understanding on the formulated problem and the coorresponding inverse-design (constraints optimization) problem.

2.1 Notes on the DDA method

In this section we present a few details regarding the DDA method used in the main text. The details can be found also in [2, 3, 4, 5]. A similar methodological approach was also used in [6].

We start by assuming that each 2D scatterer (assuming a point in the x-y plane) acquires its z-oriented dipole moment due to the local electric field, i.e.,

$$p = \alpha e_{\text{loc}} \tag{11}$$

where the e_{loc} is the vector of local z-polarized electric fields at the center of each point and α is the polarizability that depends on the shape and the material composition of each 2D scatterer. The local field is the sum of the incident field and the secondary fields generated from all the other dipoles such that:

$$e_{\text{loc}} = e_{\text{inc}} + Gp \tag{12}$$

where e_{inc} is the incident field vector, p is the induced polarization vector and G is the 2D Green's function. In our case we consider a two-dimensional (2D) problem with a transverse electric (TE) excitation (the field is normal (z-direction) to the x-y plane). Therefore the corresponding Green's function reads

$$G = G(r_i - r_k) = -j\frac{k_0^2}{4\pi\varepsilon_0} H_0^{(2)}(k_0|r_i - r_k|) \tag{13}$$

where $H_0^{(2)}(k_0|r_i - r_k|)$ is the Hankel function of the second type (with the time harmonic convention $e^{+j\omega t}$) and 0-th order and $k_0 = \omega_0\sqrt{\mu_0\varepsilon_0}$ is the free-space wavenumber [7]. The G is a $\mathbb{C}^{N\times N}$ Toeplitz matrix with zero diagonal entries since the $|r_i - r_k|$ is treated as in (assuming a uniformly spaced discrete grid) [2, 3, 4, 5].

By combining Eqs. (11) and (12) we obtain the following expression, arranged using the matrix formulation as follows

$$p = \left(A^{-1} - G\right)^{-1} e_{\text{inc}} \tag{14}$$

where the lowercase quantities $p = [p_1, p_2, ..., p_N]^T$, $A = \text{diag}(\alpha)$, $\alpha = A_{\text{cell}}\varepsilon_0[\varepsilon_1 - 1, \varepsilon_2 - 1, ..., \varepsilon_N - 1]$ (A_{cell} is the cross-sectional area of a cylinders) and $e_{\text{inc}} = [e_1^{\text{inc}}, e_2^{\text{inc}}, ..., e_N^{\text{inc}}]^T$ are $\mathbb{C}^{N\times 1}$ vectors, diag($\cdot$) is the diagonal matrix operator.

Finally, the scattered field observed at M specified discrete detection points (in general $M \neq N$) is given by:

$$e_{\text{sca}} = G_{\text{pr}}\ p = G_{\text{pr}}\left(\text{diag}(\alpha^{-1}) - G\right)^{-1} e_{\text{inc}} \tag{15}$$

where $G_{\text{pr}} \in \mathbb{C}^{M\times N}$ is the "propagator" Green's function matrix. This propagator function connects the induced dipole polarization vectors of the scatterers with the desired detection (or objective) points.

The above matrix representation of the scattering problem allow us to have a clear inspection of the unknown quantities. These quantities are the ones that will be formulated as a Lagrange multiplier algorithm for the solution of the desired constrained optimization problem. We note here that, as seen in Eq. (15), the forward scattering problem requires a matrix inversion to evaluate the polarization density vectors induced in each scattering cell, as we have discussed in our previous work [6] in which we utilized the same DDA approach for the evaluation Eq. (14) and the matrix-vector operation of Eq. (15) for different excitation and for different scattering scenarios.

2.2 Notes on the Lagrange multiplier algorithm

For this, we utilize the DDA algorithm (where we closely follow the contrast source inversion method![8]) and the Lagrange multiplier method for applying the constraints and finding the optimal solution. First, in terms of the defined problem, we have that the polarization is connected with the following expressions

$$p = A(e_{\text{inc}} + Gp) \tag{16}$$

and

$$e_{\text{sca}} = G_{\text{pr}}p \tag{17}$$

A typical constrained minimization problem (primal) can be written as [9, 1, 10]

$$\begin{aligned} \min_{\substack{x,y \\ y\in\mathbb{R} \\ 0\leq y\leq 1}} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0 \end{aligned} \tag{18}$$

where $f(x,y)$ is the objective and $g(x,y)$ are the constraints also subject to further requirements of the problem such as $y \in \mathbb{R}$ and $0 \geq y \geq 1$ For such problems the dual Lagrangian problem is expressed as

$$\max_{\lambda} \min_{\substack{x,y \\ y\in\mathbb{R} \\ 0\leq y\leq 1}} \quad \mathcal{L}(x,y,\lambda) = f(x,y) + \lambda g(x,y) \tag{19}$$

which is essentially a dual unconstrained problem (since all the constraints are encapsulated to the λ term). It is worth noting that the Lagrange multiplier should be positive real, $\lambda \in R^+$. Finding an approximate solution to the primal inverse scattering problem is therefore reduced to finding a solution to the above dual problem. Notice that the Lagrange multiplier can be applied to either $f(x,y)$ or $g(x,y)$ without affecting the outcome of the overall process.

The algorithm for solving the above dual problem is the following:

- Step 0: initial x_0 and λ_0
- Step 1: minimize y_n, i.e., via $\nabla_y \mathcal{L}(x_{n-1}, y_n, \lambda_{n-1}) = 0$
- Step 2: project y_n into $y \in \mathbb{R}$ and $0 \leq y \leq 1$
- Step 3: minimize x_n, i.e., $\nabla_x \mathcal{L}(x, y_n, \lambda_{n-1}) = 0$
- Step 4: maximize λ_n, i.e., $\nabla_\lambda \mathcal{L}(x_n, y_n, \lambda) = 0$
- Step 5: Repeat steps 1-4 until the error is minimized

For our particular example we have that $x = p$, $y = A = \text{diag}(\varepsilon - 1)$, and $f(p, A) = 1/2||(A^{-1} - G)p - e_{\text{inc}}||^2$ and $g(p) = 1/2||G_{\text{pr}}p - e_{\text{obj}}||^2$, and Lagrange function reads

$$\mathcal{L}(p, A, \lambda) = ||(A^{-1} - G)p - Ae_{\text{inc}}||^2 + \lambda||G_p p - e_{obj}||^2 \tag{20}$$

The corresponding algorithmic steps are:

- Step 0: initial p_0 and λ_0
- Step 1: minimize A_n via $\nabla_A \mathcal{L}(p_{n-1}, A, \lambda_{n-1}) = 0$

  - We have that $\nabla_A \mathcal{L}(p_{n-1}, A, \lambda_{n-1}) = \nabla f(p, A)^* ||(A^{-1} - G)p - e_{\text{inc}}||$ ($*$ is complex conjugate). This expression lead to $A_n = p/(Gp - e_{\text{inc}})$. In practice this is a simple calculation since A is a diagonal matrix, i.e., $A = \text{diag}(\varepsilon - 1)$.

- Step 2: project A_n into $A_n \in \mathbb{R}$ and $0 \leq A \leq 4$ (for the range $\varepsilon \in [1, 5]$)
  - this is the point where essentially the required properties and bound of the permittivity can be implemented. These bounds or constrains can be general
  - the above projection is rather a simple projection that does not guarantee always the minimum within the projection domain. A more accurate projection would be of the form $A_n = \text{proj}[A_{n-1} - \eta \nabla_A ||(A_{n-1}^{-1} - G)p_{n-1} - e_{\text{inc}}||^2]$.

- Step 3: minimize p_n, i.e., $\nabla_p \mathcal{L}(p, A_n, \lambda_{n-1}) = 0$ (DCM metadevice).
  - $p_n = K_n^{-1} e_{\text{L}}^n$
  - $K_n = (A_n^{-1} - G)^*(A_n^{-1} - G) + \lambda_{n-1} G_{\text{pr}}^* G_{\text{pr}}$
  - $e_n^L = \lambda_{n-1} G_{\text{pr}}^* e_{\text{obj}} + (A_n^{-1} - G)^* e_{\text{inc}}$
  - The matrix inversion p_n is performed with our DCM metadevice
  - Due to noise error a simple weighted average filtering is applied, i.e., $p_n = (1 - \alpha_F)p_{n-1} + \alpha_F p_n$ with $\alpha_F = 0.25$

- Step 4: maximize λ_n, i.e., $\nabla_\lambda \mathcal{L}(A_n, p_n, \lambda) = 0$
  - This maximization can be calculated by a simple gradient descent, i.e., $\lambda_n = \lambda_{n-1} + \eta\left(\nabla_\lambda \mathcal{L}(p_n, A_n, \lambda) - \delta\right)$ or $\lambda_n = \lambda_{n-1} + \eta\left(||G_p p_n - e_{obj}||^2 - \delta\right)$
  - Notice that this is an gradient *ascent* since we assume $\eta > 0$, therefore we maximize the problem.

- Step 5: Repeat steps 1-4 until the error is minimized. In our case we used the following error
  - $||e_{\text{sca}} - e_{\text{obj}}||^2 / ||e_{\text{obj}}||^2$

Note that the quantities η and δ are the step and minimal error quantities that are user determined. The whole process stop either when λ reaches a plateau, or when the required error criterion is met. The optimization goal was set as $\frac{||e_{\text{sca}} - e_{\text{obj}}||^2}{||e_{\text{obj}}||^2} < \delta$, where $e_{\text{sca}} = G_p p_{\text{m}}$ with $p_{\text{m}} = (A_m^{-1} - G)^{-1} e_{\text{inc}}$ being the final m-th evaluation of the iteration.

Notice that our approach has several similarities with the contrast source inversion method and other similar inverse scattering methods [11, 8, 12, 13].

Undoubtedly this approach is only one of the available methods for approximating the inverse design problem. This is rather an attempt to showcase the ability of our device for performing inverse design with desired objectives and constraints by exposing the crucial parts of the algorithm, such as the matrix inversions. This part is usually implicit within commercially available FDTD or FEM software. Hence here we developed our own methodology so we can have deeper inspection to quantities. As a remark, the field of inverse design and inverse scattering is a very rich field with a plethora of methodologies that try to address similar problems [14].

Similar to Newton's method, matrix inversion represents the primary computational complexity. Gradients can be computed using automatic differentiation, akin to the methodology employed for calculating the Jacobian matrix.

3 RF Design, PCB, Device Implementation

A photograph of the experimental setup is shown in Fig S1, where all parts are designated accordingly.

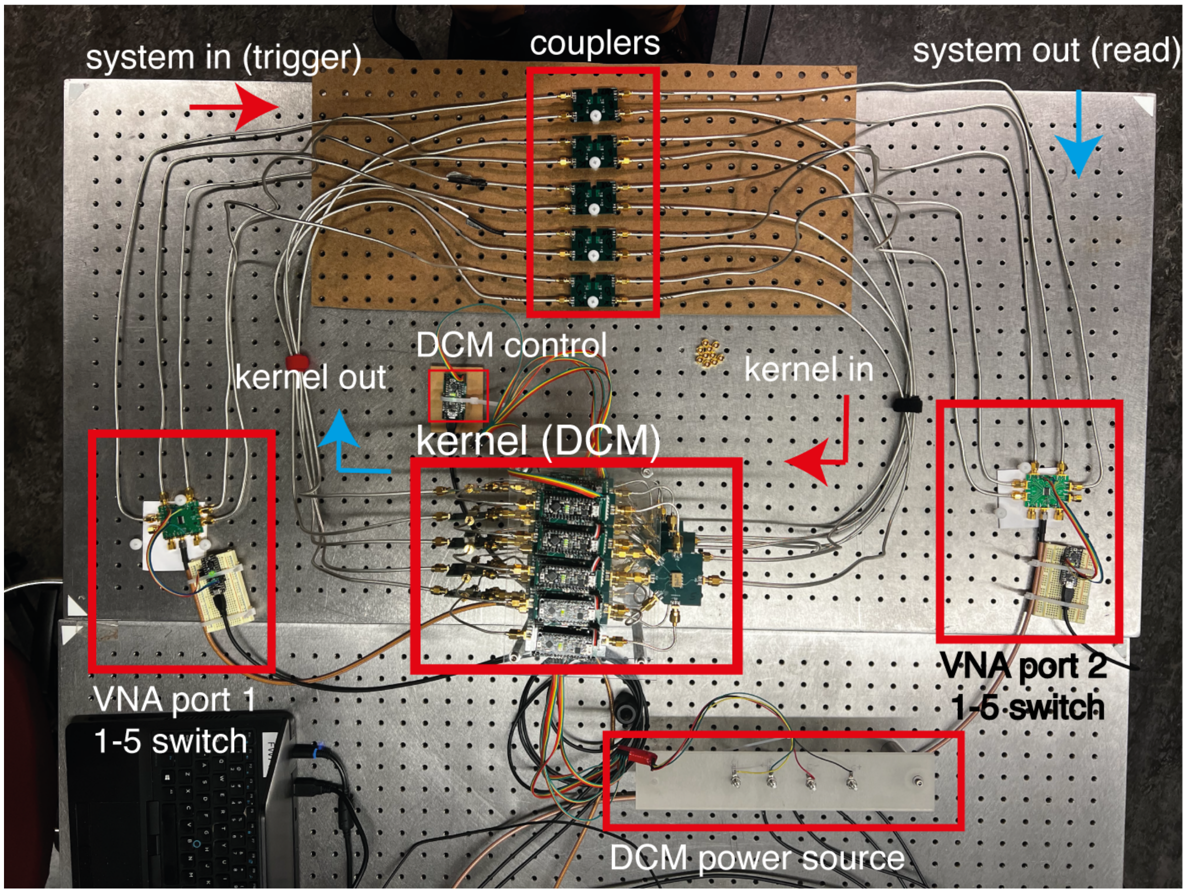

Figure S1: Photograph of the experimental setup with the corresponding components.

3.1 Measurement

Measurements were performed using an ENA-5071C two port VNA. In order to avoid the saturation of the amplifiers (multiplier module) the VNA power level was set to be -20dBm for the open loop configuration and -10dBm for the closed loop configuration. The VNA was set to have an IF bandwidth of 10 kHz. The single frequency measurements (1601 point) at 45MHz with averaging applied after obtaining the measured signal from VNA.

3.2 Multiplier

The schematic of the multiplier is depicted in Fig. S2. The multiplier was designed to perform multiplication on the incoming complex amplitude such that a new complex amplitude is rendered at the output. In other words, the output is $V_{\text{out}} = zV_{\text{in}}$. This involves changing both the amplitude and phase of the incoming signal. Phase change was performed using a pair of serially connected Minicircuit JSPHS-51+ Phase Shifters (PS). Each phase shifter provides slightly over 180 degrees of rotation. The amplitude change was performed using the Analog Devices AD603ARZ Variable Gain Amplifier (VGA). The Multiplier design contain the appropriate loads such that both the input and output of the device externally appears as 50 Ohm.

Both of these devices are controlled using analog voltages with ranges of $[-0.5\text{V}, +0.5\text{V}]$ and $[0\text{V}, 12\text{V}]$ for the VGA and PS, respectively. In order to create a common control mechanism, op-amp level shifting circuits were used to put these on a common $[0\text{V}, 5\text{V}]$ interface. The Multiplier board has a connection that allows for a daughter board. The daughter board is supplied with 0V and +5V and is responsible for returning two control voltages in the range of $[0\text{V}, 5\text{V}]$.

This simple interface allows for a number of possible control schematics. At its most simple scenario, the control board can consist of a pair of potentiometers. However, we will present another control board which utilizes a microcontroller to receive UART input and render the two analog voltages.

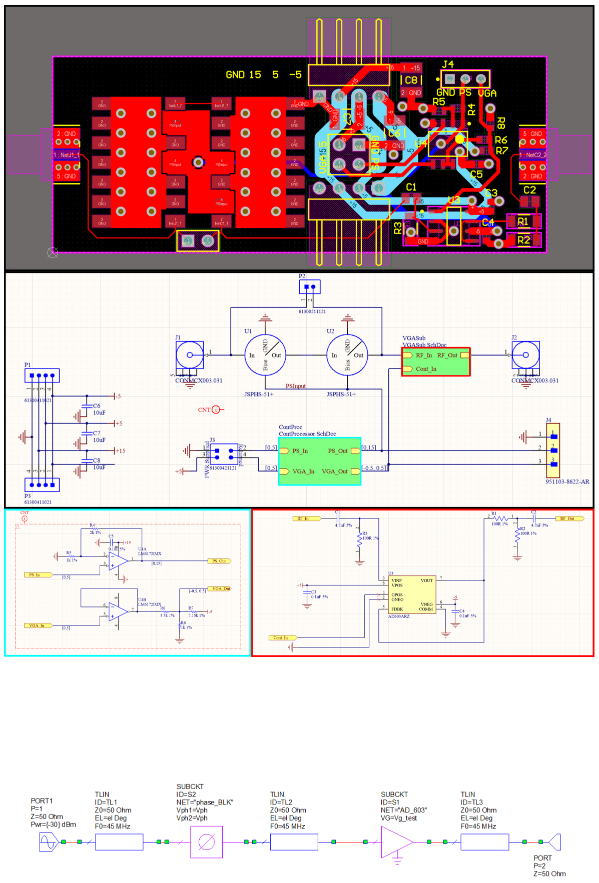

Figure S2: Schematics for the multiplier: The PCB layout design (top figure), and the corresponding subcircuit (pictures created with Altium®). Bottom figure represent the AWR Microwave Office® schematic with the realistic data

The VGA's dynamic range could be shifted using an external resistor. This was set so that the Multipliers's range (including load elements, PS losses, etc) was [-30dB − +17dB]. The multiplier effectively saturates if the input is greater than -10dBm. Therefore for all measurements the reference input signal that was used was -30dBm for avoiding any saturation effects.

It should be stated that the VGA imparts a varying phase change and the PS pair imparts an amplitude change. This will be addressed later.

3.3 1-5 splitter (5-1 combiner)

The schematic of the 1-5 splitter is depicted in Fig. S3. An ideal passive n-way splitter is comprised of a summation port and n feed ports. The scattering parameters are expected to be reciprocal such that for the i^{th} feed port $|S_{\text{S}i}|^2 = |S_{i\text{S}}|^2 = 1/n$ and all other elements within the matrix are zero. Due to losses, a real splitter will fall short of this precise definition. Our splitter was based on the Minicircuits AD5PS-1+, which yielded good performance at 45MHz with approximately -7.2dB split ratio for all outputs.Note that $1/5 \approx -7.0dB$.

3.4 Feedback coupler

The schematic of the feedback coupler is depicted in Fig. S4 The Feedback coupler must perform several tasks.

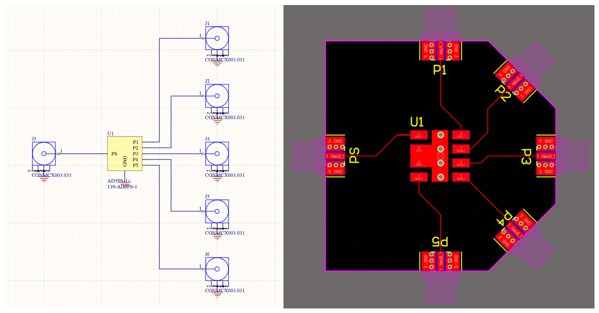

Figure S3: Schematic and layout for 1-5 splitter based on the Minicircuits AD5PS-1+ (pictures created with Altium®)

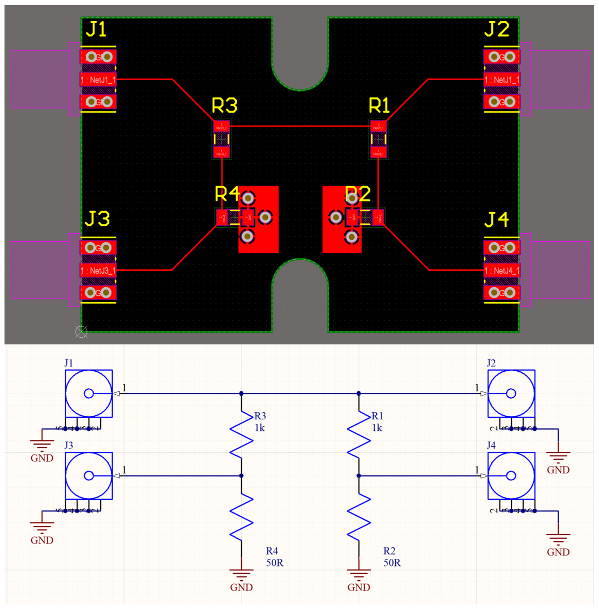

Figure S4: Schematic and layout for the feedback coupler (pictures created with Altium®)).

- Provide near unity feedback
- Introduce the input signal
- Sample the output signal

Therefore, the feedback coupler is a four-port device wherein the primary path has near unity transmission such that the feedback is strong.

3.5 Switches

In order to replicate having a 10 port VNA, we utilized two demo boards (EV1HMC253AQS24), which acted as RF SP8T RF switches, i.e., an analog multiplexer. For one SP8T, we utilized five of these ports for illuminating the bank of five couplers. The other SP8T was used to receive signals from the couplers. The remaining three ports on each were used for system sanity checks. Note that the stock high-pass 100pF capacitors on these boards were switched to 470pF for better transmission at 45MHz.

While the off ports were nominally matched to 50 Ohm from DC-2.5GHz, there was significant reflections. Deeper inspection of the datasheet indicated that the "off" ports were only matched above 500MHz. Measurements indicated that the off ports were approximately "open" at the design frequency and therefore reflections from the off ports could be significantly reduced with parallel 50Ohm terminations. However, this was not done as this it would have reduced power within the system on the "on" port. Rather, we note that any polluting signal from these "open" off ports will have crossed through the coupler twice. Due to the small coupling coefficient of the feedback coupler, these values will have become very small.

The VNA was calibrated to the end of the switch ports. Measurements indicated that transmission through each of the switch ports was similar enough as to not warrant individual calibrations on each.

Each of the switches was actuated by three digital inputs to address the $2^3 = 8$ ports on each switch. These digital signals were created by a micro-controller which was programmed to respond to UART commands from an attached computer. Code is available at `github.com/brianedw/RFMath/Arduino/mcu_control_V2/mcu_control_V2.ino`.

3.6 Micro Controller Unit (MCU)

The two analog input control voltages for each Multiplier was created by an MCU Control Board, which attached directly to the Multiplier. The heart of this board is a Metro-Mini MCU.

Each control line was connected to both an 8-bit PWM DAC pin (labeled “fast”) and a 10-bit PWM DAC pin (labeled “slow”). While both pins connected to the control line through a high-pass filter, the fast DAC utilized a lower capacitance and resistance than that of the slow DAC. During a set operation, both pins would drive to their appropriate values, during this time, the behavior of the collective output would be dominated by the fast DAC and rapidly converge, but exhibit large ripples. After 20ms, the fast PWM DAC pin would switch to a high-impedance state, leaving the voltage to settle in the remaining difference utilizing the slow DAC alone. The high-pass filter was designed to maintain accuracy of 10bit. Since the Metro-Mini is a 5V compliant device, the generated voltages nicely matched to the expected inputs of the Multiplier.

Each MCU board had two 3-pin UART input connectors. These were shorted such that one could be used to receive a command from "upstream" while the other would effectively passively repeat the signal. Additionally, each MCU board had two 3-pin UART output connector which were similarly shorted together, allowing it to transmit the same message to two devices. Each MCU was programmed with a unique identification number. Upon receiving a UART command, it would either act on that command or repeat the command on its output UART pins for downstream devices. This input/output configuration created a lot of possibilities for control topologies. However, in practice we found that we could use a single MCU board (no multiplier attached), as a bridge between the computer and the array of MCU Boards and that this array could all be connected in parallel such that the output of the bridge was effectively driving 25 inputs. Note that the required time complexity is of the order of $\mathcal{O}(n^2)$. Possibly this complexity can be further reduced by implementing different connectivity schemes than the simple serial one that we used. Code is available at `github.com/brianedw/RFMath/Arduino/mcu_control_V2/mcu_control_V2.ino`.

The MCU stands out as the primary device in terms of both power consumption and reconfiguration speed. In our setup, the selected MCUs, while not optimal, result in an overall power consumption within the range of 1 watt, with a reconfiguration time on the order of milliseconds. Transitioning to a potential FPGA or ASIC implementation is anticipated to yield significant enhancements, potentially reducing power consumption to micro-watts and reconfiguration time to micro-seconds.

4 Tuning/Calibration

As stated in 3.2, the VGA has a minor effect on the phase and the PS has a minor effect on the amplitude. In other words, the phase and amplitude responses are coupled. Additionally, other systematic errors are present such as nonidealities in the level shifting circuits due to resistor tolerances. When connected in a network that includes RF jumper cables of varying length, there will also be phase shifts that naturally arise. In short, the relationship between the control voltages and the response of the Multiplier *in situ*, are repeatable, but difficult to predict without developing a more complex model.

We found that an effective strategy to capture, model, and invert the relationship between control voltages and system response goes as follows.

1. A collection of Multipliers are swept across their input values to map the relationship between control voltage and complex multiplier response.
2. These responses were analyzed using Principle Component Analysis (PCA) [15].

3. The multipliers were assembled into the open-loop configuration and the response of the entire open-loop network was measured under many sets of input control voltages.

4. These results were compared to a theoretical model of the network wherein the weights of the components could be adjusted until the theoretical results matched the experimental results.

5. With accurate PCA weights in hand, the Multipliers can be immediately adjusted to achieve a desired multiplication factor by inverting the model to achieve any open-loop kernel.

6. Additional refinement can be obtained by changing the device configuration into the closed loop, which now includes the feedback couplers. Again, we measure the response of the closed-loop network under many input conditions.

7. We further refine the PCA weights of each multiplier to match this more demanding data set. This becomes our final device model for both the open- and closed-loop configurations.

We will go into detail on each one of these items in the following sections.

4.1 Multiplier PCA

A collection of 35 multipliers were each mapped using the MCU control boards, capable of 10-bit resolution on both control voltages. The mapping occurred with a grid of values based on [0, 11, ..., 1012, 1023] on both controls. Ideally, the mapping of two Multipliers would yield identical responses. However, for all the reasons stated above, they do not. All of the mappings were compared using a complex domain PCA analysis. Typically, in PCA, one would examine deviations from the mean, but here we take another approach. Rather, the collection of mappings were analyzed directly to yield a set of 4 PCA components. The response of any individual Multiplier could then be found as the linear superposition of these components given by:

$$m(d_{\text{vga}}, d_{\text{PS}}) = \sum_{i=0}^{3} w_i c_i(d_{\text{vga}}, d_{\text{PS}})$$

The term $c_0(d_{\text{vga}}, d_{\text{PS}})$ is effectively the "average" response scaled by a complex factor, while the next several components represent likely deviations due to the systematic errors described above. Within a PCA analysis the final PCA components (i.e. $c_{34}(d_{\text{vga}}, d_{\text{PS}})$, not shown) should be nearly pure noise. We found that only the first four terms were needed to effectively model any given Multiplier.

Given any randomly chosen multiplier, we can find the complex valued PCA weights w_i through a least-squares analysis. As opposed to the "deviation from the mean" approach, the above formulation is particularly useful for RF engineering. While the Multipliers were measured directly at their input and output ports and analyzed as such, the model can easily account for the addition of RF cables which would provide attenuation and phase rotation. These will appear as a complex scaling of all of the components weights and the Multiplier's behavior (RF jumpers cables included) can still be captured as the simple linear superposition of the PCA components. In fact, any losses or phase rotations along the Multipliers flow path can be incorporated into these weights. Therefore, we do *not* characterize the individual multipliers, but delay this until the architecture is fully assembled, as described in the next section.

Regardless, we will use least-squares to find the set of w_i which characterizes the average multiplier response. We call these the "base weights".

4.2 Open-Loop Device Fitting

The goal of the this section is to determine the PCA weights that characterize each Multiplier *in situ*, so that the system errors can be captured and modeled. The open-loop DCM system was fully assembled including jumper cables, splitters, and couplers. All multipliers within the array were set to the same input value $(d_{\text{vga}}, d_{\text{PS}})$ pair. The transmission matrix of the system was then measured. This was repeated for all possible combinations of 10 evenly spaced values in the range $[0, 1023]$ to yield 100 measured transmission matrices, $\mathcal{T}_{\text{meas}}$. Note that not all of these 100 transmission matrices represent "passive" operators.

The same system was modeled using Scikit-RF, wherein the following assumptions were made:

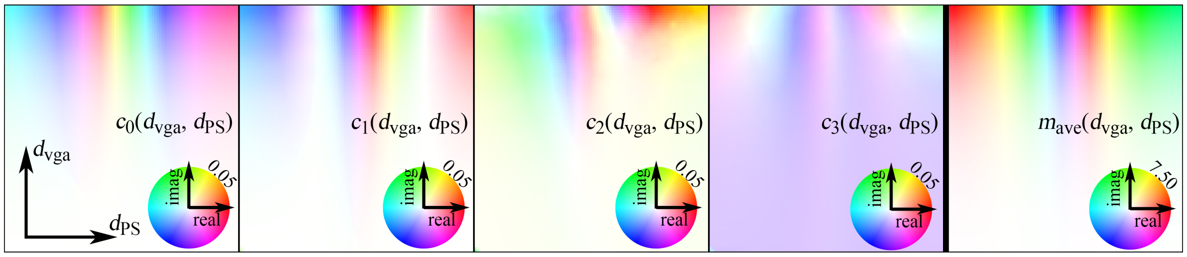

Figure S5: PCA Components and Average Multiplier Response. The first four panels represent $c_0(d_{\mathrm{vga}}, d_{\mathrm{PS}})$, $c_1(d_{\mathrm{vga}}, d_{\mathrm{PS}})$, $c_2(d_{\mathrm{vga}}, d_{\mathrm{PS}})$, and $c_3(d_{\mathrm{vga}}, d_{\mathrm{PS}})$ and have a maximum saturation of 0.05. The final image shows the response of the "average" Multiplier with a maximum saturation at 7.5

- The 5-1 splitters were ideal such that power was evenly split with no phase
- All jumpers were zero-length
- The coupler feedback path was ideal with no power removed
- The multipliers were all assumed to be "average" and the their responses were assumed to be given by the "base weights".

The system was simulated using SciKit-RF for each input pair $(d_{\mathrm{vga}}, d_{\mathrm{PS}})$ to yield 100 measured transmission matrices $\mathcal{T}_{\mathrm{sim}}(\overline{w})$, which are naturally a function of each Multipliers PCA weight. We can then define an error $\mathrm{error}(\overline{w}) = |\mathcal{T}_{\mathrm{sim}}(\overline{w}) - \mathcal{T}_{\mathrm{meas}}|^2$ and optimize $\overline{w}$ until that error is minimized. It should be noted, that with only four PCA weights per Multiplier, in theory, only 4 transmission matrices are required to fully define the system. Using 100 helps guarantee that normal measurement noise does not unduly influence the fitting. Additionally, if a low error can be achieved across 100 measurements using only 4 weights, then we can be confident that the model was sufficient to capture the entire open loop system response, $\mathbb{K}$.

4.3 Setting the Open Loop System Response

Given a desired open-loop system response, $\mathbb{K}$, we need to calculate the necessary multiplier values for the DCM architecture, $m_{i,j}$, gathered to form $\mathbb{M}$. In this case, the simplicity of the DCM architecture makes this trivial. If we assume an idealized passive five port splitters such that given an input of 1W at the summation port, s, we will observe 1/5W on each branch port, i. Put in terms of Scattering Parameters, $S_{\mathrm{s},i} = 1/\sqrt{5}$ and via reciprocity $S_{i,\mathrm{s}} = 1/\sqrt{5}$. Since we have such splitters at the input and output of the Multiplier array, $\mathbb{K} = (1/\sqrt{5})\mathbb{M}(1/\sqrt{5})$ and therefore $\mathbb{M} = 5\mathbb{K}$. Note that since we fitted the PCA weights of the Multipliers under the assumption of ideal components, it is appropriate to assume ideal components here.

With each of the desired $m_{i,j}$ in hand to achieve a given $\mathbb{K}$, the next step is to determine the required $(d_{\mathrm{vga}}, d_{\mathrm{PS}})$. This can be done using a number of function inversion schemes such a gradient descent. In practice, this could be very fast as it is likely that in many applications, each new $\mathbb{M}$ will be a small step from the previous $\mathbb{M}$ and therefore each multiplier will change only slightly.

4.4 Closed-Loop Device Fitting

Due to the recursive nature of the closed-loop configuration (Matrix Inversion), the accuracy requirements are more stringent than for the open-loop configuration. Moreover, additional degrees of freedom are introduced in the form of coupler coefficients. These can be considered part of $\overline{w}$. In short, the devices must be fitted again.

We will employ a similar strategy as was used in the *Open-Loop Device Fitting*. Using the open-loop calibrated device models, a sequence of randomly generated passive transmission matrices, $\mathbb{K}$, are shown to the system. Note, unlike the open-loop matrices, in order to guarantee convergence, these matrices must be passive. We model the closed-loop system using Scikit-RF. Using the open-loop weights as a starting point, we optimize the multiplier weights and coupling coefficients until the simulated $\mathcal{T}_{\text{sim}}(\overline{w})$ matches the measured $\mathcal{T}_{\text{meas}}$. This represents a small, but necessary, refinement from the open-loop device model and can be used for both open- and closed-loop applications.

4.5 Setting the Closed Loop System Response

Setting the closed-loop system response, $\mathbb{K}$ is identical to setting the open-loop response. In both cases, each desired $m_{i,j}$ is used to find the required $(d_{\text{vga}}, d_{\text{PS}})$ using a function inversion scheme.

5 System Accuracy

We performed an open loop measurement on 100 complex-valued random matrices with (eigenvalues) values within the unit circle. For these, we configured the open-loop with the target (or ideal kernel) A_e and retrieved the measured results A_m. We define as error the quantity

$$\frac{||A_m - A_e||^2}{||A_e||^2} 100\% \tag{21}$$

In Fig. S6, we can see the difference between the two matrices for 100 random cases. We observe that all the results are within a $0.05 - 0.3\%$ percent error. Similarly, we performed the same error analysis for the same 100 random matrices, only this time on a closed-loop setup (matrix-inversion). The results (Fig. S7) reveal that the error can climb up to 20%, but for most of the results, we get a matrix inversion with less than 2% error. Finally, we assess the matrix inversion fidelity by evaluating the trace of the $A_m^{-1} A_e$ product. Ideally the trace of the product $\text{tr}(A_m^{-1} A_e)/5 = 1$. In Fig. S8, we observe that this product spans between $0.5 - 1.5$. However, for the particular examples we used in the manuscript, this accuracy can be maintained at reasonably high levels once error-correcting and filtering techniques are applied. Note that for the closed-loop case, the level of the measured voltage is in the order of μV, very close to the noise floor of the VNA device we used. For the open loop operation, the measured voltage was hundreds of mV.

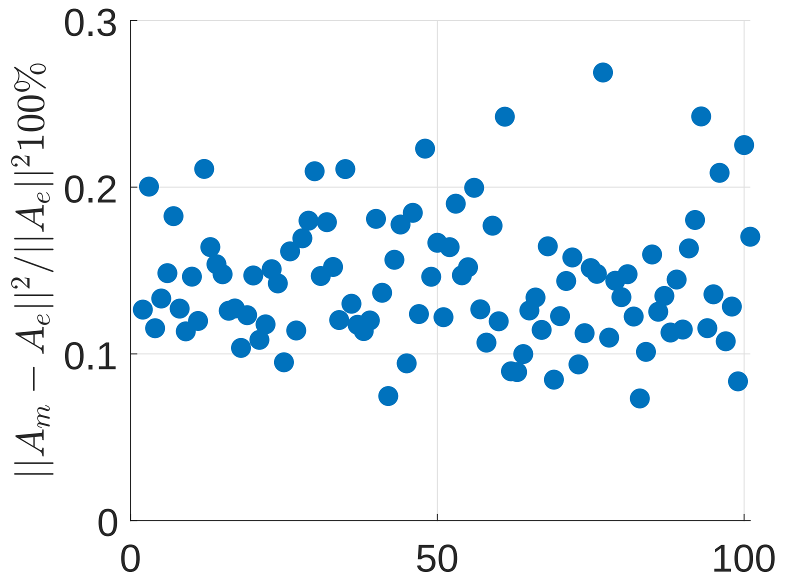

Figure S6: The error between the exact and the measured matrices, open loop configuration, for 100 random complex matrices.

6 System Transient Analysis

6.1 Single Multiplier

In terms of the time response of the multiplier module the transient analysis reveal (Fig. S9) that the module obtained the desired value approximately within 3-4 signal periods, i.e., $T = 22.2$ns. The

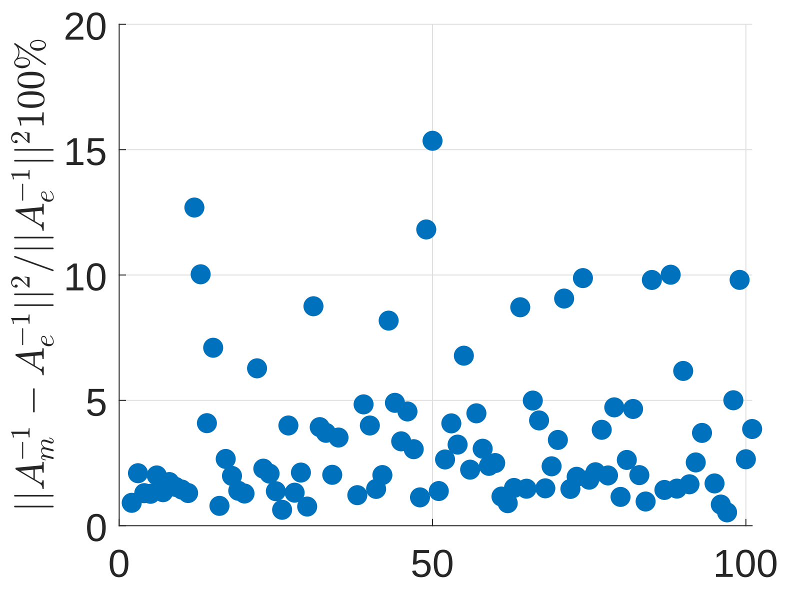

Figure S7: The error between the exact and the measured matrices, closed loop configuration (matrix inversion), for 100 random complex matrices.

measurements were performed using the RIGOL DG4062 pulse generator (15 sinusoidal pulses at 45MHz), and the measured response extracted with the RIGOL DS1104 oscilloscope.

The open loop response is therefore assumed to be very close to the single multipliers response since both splitters and connecting cables introduce a small phase shift to the signal. The closed loop transient response is affected by both the multiplier timing and the condition number of the input matrix (kernel) as shown in [16].

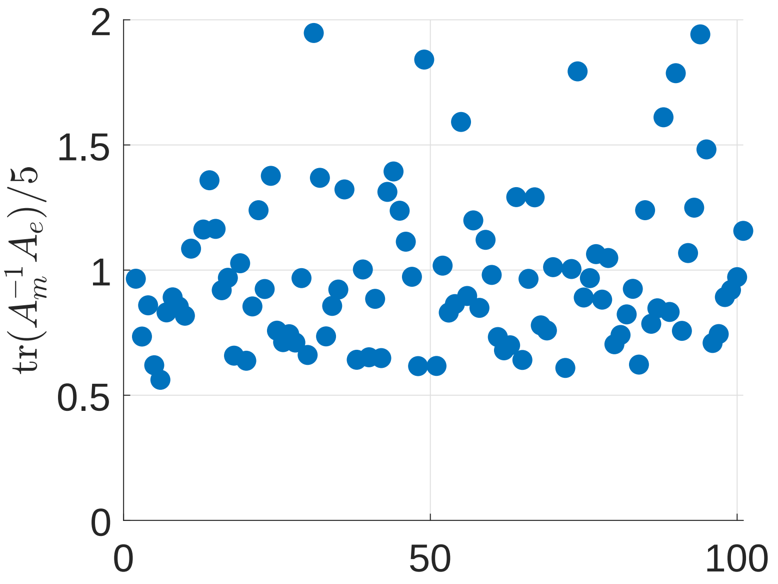

Figure S8: The fidelity of the matrix inversion expressed in terms of the normalized trace of the $A_m^{-1}A_e$ product for 100 random complex matrices.

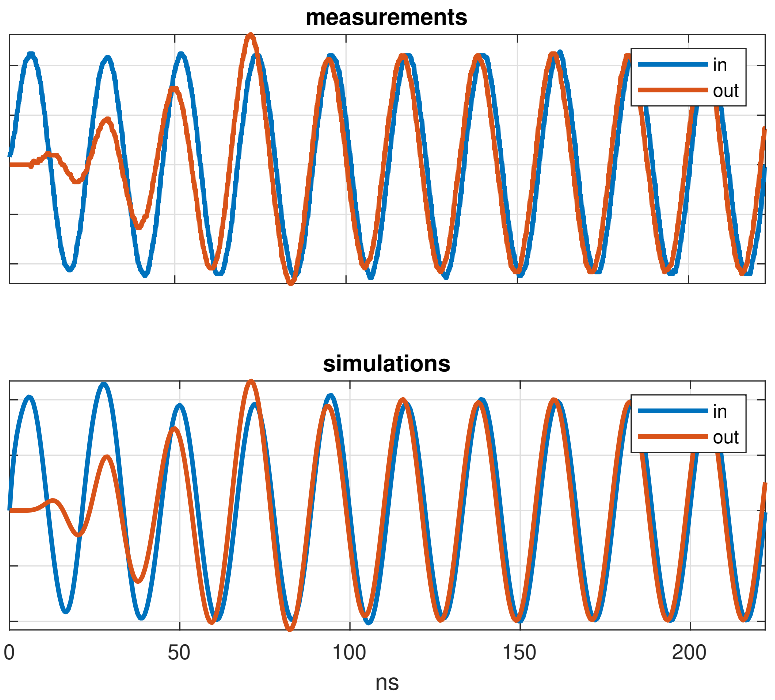

Figure S9: The transient response of a single multiplier module. The blue curves correspond to the input signal, while the red curves are the measured (top) and simulated (bottom) using AWR Microwave Office® results. The agreement is excellent. It is evident that it takes approximately 3 to 4 signal periods for the multiplier to obtain the desired output signal. Here we assumed small signal amplification (VGA voltage is $+.05$) and the phase shift voltage is 0V.

7 De-embedding the solution

7.1 Open Loop

Let us define the open-loop response as

$$V_{\text{out}} = \mathbb{K}V_{\text{in}}$$

Note that this includes not only the DCM architecture (multipliers, splitters, jumpers), but also the through channel of the input/output coupler. In other words, the open-loop is defined using all of the components of the closed-loop. However, the loop has been broken "open" just after the coupler array and measured at this point. Since in the closed configuration, these measurement planes were coincident, upon "closing" the loop, these measurement will then represent the complete response of the loop. While a minor perturbation to the results, this definition assumes that the weakly coupled additional ports on the coupler are properly terminated.

Let us further define response of only the DCM architecture as $\mathbb{K}'$. When the system is in a closed loop configuration, this relates the vector exiting the coupler array (V_4) to the vector incident on the coupler array (V_2).

$$V_2 = \mathbb{K}'V_4$$

The coupler array introduces a small loss as the input is introduced and the output is measured. The near unity transmission is named α_1. It is clear then that $\mathbb{K} = \alpha_1\mathbb{K}'$.

7.2 Closed Loop

The closed loop response is fully defined by the open-loop response and the definition of the scattering parameters of the coupler.

$$V_2 = \mathbb{K}'V_4 \tag{22}$$
$$V_3 = \alpha_2 V_1 + \beta V_2 \tag{23}$$
$$V_4 = \beta V_1 + \alpha_1 V_2 \tag{24}$$

Our goal is to solve the equations for V_4, which represents the vectorial solution of the problem in question. For the *expected solution*, this should be done such that the solution depends only on the kernel $\mathbb{K}$ and the input vector (V_1). For the *measured solution*, this should be only in terms of the measured results (V_3) and the known input (V_1).

7.3 Expected Solution

We begin by applying the definitions above

$$V_4 = \beta V_1 + \alpha_1 V_2$$
$$V_4 = \beta V_1 + \alpha_1 \mathbb{K}'V_4$$
$$V_4 = \beta V_1 + \mathbb{K}V_4$$

and then solve the final equation for the V_4.

$$V_4 = (I - \mathbb{K})^{-1}\beta V_1$$

7.4 Measured Solution

We begin with Eq 23:

$$V_3 = \alpha_2 V_1 + \beta V_2$$

and then solve it for V_2

$$V_2 = \frac{1}{\beta}V_3 - \frac{\alpha_2}{\beta}V_1$$

and then substitute the above into 24

$$V_4 = \beta V_1 + \alpha_1(\frac{1}{\beta}V_3 - \frac{\alpha_2}{\beta}V_1)$$

and then simplify

$$V_4 = (\beta - \frac{\alpha_1\alpha_2}{\beta})V_1 + \frac{\alpha_1}{\beta}V_3$$

Note that in many real world cases, the coupler will be defined such that we can assume $\alpha_2 \to 0$

$$V_4 = \beta V_1 + \frac{\alpha_1}{\beta}V_3$$

References

[1] D. P. Bertsekas, *Nonlinear programming.* Belmont, Mass. :: Athena Scientific,, 1995.

[2] E. M. Purcell and C. R. Pennypacker, "Scattering and Absorption of Light by Nonspherical Dielectric Grains," *Astrophys. J.*, vol. 186, p. 705, dec 1973.

[3] B. T. Draine and P. J. Flatau, "Discrete-Dipole Approximation For Scattering Calculations," *J. Opt. Soc. Am. A*, vol. 11, no. 4, p. 1491, 1994.

[4] M. A. Yurkin and A. G. Hoekstra, "The discrete dipole approximation: An overview and recent developments," *J. Quant. Spectrosc. Radiat. Transf.*, vol. 106, no. 1-3, pp. 558–589, 2007.

[5] S. P. Groth, A. G. Polimeridis, and J. K. White, "Accelerating the discrete dipole approximation via circulant preconditioning," *J. Quant. Spectrosc. Radiat. Transf.*, vol. 240, p. 106689, 2020.

[6] V. Nikkhah, D. C. Tzarouchis, A. Hoorfar, and N. Engheta, "Inverse-designed Metastructures Together with Reconfigurable Couplers to Compute Forward Scattering," *ACS Photonics*, jul 2022.

[7] C. A. Balanis, *Advanced engineering electromagnetics.* John Wiley & Sons, 1999.

[8] P. M. van den Berg and R. E. Kleinman, "A contrast source inversion method," *Inverse Probl.*, vol. 13, no. 6, pp. 1607–1620, 1997.

[9] D. Bertsekas, *Convex optimization theory*, vol. 1. Athena Scientific, 2009.

[10] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization.* Cambridge University Press, 2004.

[11] R. E. Kleinman and P. den Berg, "A modified gradient method for two- dimensional problems in tomography," *J. Comput. Appl. Math.*, vol. 42, no. 1, pp. 17–35, 1992.

[12] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, vol. 93 of *Applied Mathematical Sciences.* New York, NY: Springer New York, 2013.

[13] S. Boutami and S. Fan, "Efficient pixel-by-pixel optimization of photonic devices utilizing the dyson's equation in a green's function formalism: Part i implementation with the method of discrete dipole approximation," *Journal of the Optical Society of America B*, vol. 36, p. 2378, 9 2019.

[14] Z. Li, R. Pestourie, Z. Lin, S. G. Johnson, and F. Capasso, "Empowering metasurfaces with inverse design: Principles and applications," *ACS Photonics*, vol. 9, pp. 2178–2192, 7 2022.

[15] I. T. Jolliffe, *Principal component analysis for special types of data.* Springer, 2002.

[16] D. C. Tzarouchis, M. J. Mencagli, B. Edwards, and N. Engheta, "Mathematical operations and equation solving with reconfigurable metadevices," *Light Sci. Appl.*, vol. 11, p. 263, Sep 2022.