Harmonic Inversion in Tidy3D

Alec Lau, Charles Wojcik

Abstract

Resonance is a ubiquitous topic in photonics, and Tidy3D's resonance finder plug-in makes it extremely easy to extract resonance information such as the amplitude, decay rate, phase, quality factor (Q), and estimation error of the resonances. Here we describe, from a high level, how Tidy3D finds resonances through Harmonic Inversion.

1 The Goal

Suppose we have a time-domain signal $c(t)$. The idea behind harmonic inversion is that we want to find ω_k, d_k such that

$$\sum_{k=1}^{K} d_k e^{-i\omega_k t} = c(t) \tag{1}$$

When this signal is defined over a suitably long time, we can use standard Fourier analysis to find these values. If, however, we don't have enough time points, this method fails. Thus for cases with limited time data, we need something different.

2 Harmonic Inversion Overview

Suppose further that $c(t)$ is known only at $N+1$ equally-spaced time points t_n. We will label these signals as c_n. Call the (constant) time different $t_{n+1} - t_n$ by dt, and

note that this means $t_n = ndt$. Then this problem gets recast as

$$\sum_{k=1}^{K} d_k e^{-i\omega_k ndt} = c_n \tag{2}$$

The key idea behind Harmonic Inversion is using the **Filter Diagonalization Method (FDM)**.

3 Filter Diagonalization Method

FDM takes inspiration from quantum mechanics: suppose we have some normalized quantum state $|\Phi\rangle$. The periodicity of $|\Phi\rangle$ would then depend on how much agreement $|\Phi\rangle$ has with a future version of itself. From the Schrödinger Equation, this future version (e.g. at time t) is $e^{-itH}|\Phi\rangle$ for some operator (i.e. Hamiltonian) H. The measurement of this agreement, given by $\langle\Phi| e^{-itH} |\Phi\rangle$, gives us information about the periodicity in time, and therefore resonances, of $|\Phi\rangle$. FDM considers c_n as this measurement:

$$c_n = (\Phi, e^{-itH}\Phi), \text{ for some operator } H \tag{3}$$

Note that here we use $(\cdot, \cdot)$ notation instead of bra-ket notation - this is because this is the complex symmetric inner product, not a Hermitian inner product.

The game then becomes a search for the appropriate Hamiltonian H. From our assumptions, we can already say some things about H that will make life easier. First of all, since our t_i points are evenly spaced, the time evolution for the first n timesteps is U^n, where $U = e^{-idtH}$, so it describes the evolution from time t_n to t_{n+1}:

$$c_n = (\Phi, U^n\Phi) \tag{4}$$

Diagonalizing U, we get, for $\{v_k\}$ U's orthonormal eigenbasis,

$$U = \sum_k v_k e^{-idt\omega_k}(v_k, \cdot) \tag{5}$$

Therefore, we have

$$c_n = (\Phi, \cdot)\sum_k v_k e^{-indt\omega_k}(v_k, \Phi) = \sum_k (v_k, \Phi)^2 e^{-indt\omega_k} := \sum_k d_k e^{-indt\omega_k} \tag{6}$$

which is our initial goal of expressing c_n. Thus we have $d_k := (v_k, \Phi)^2$ and w_k the eigenvalues (or energy values) of H.

Now our problem becomes the familiar matrix diagonalization problem. However, keep in mind that we haven't actually done any solving - this formulation depends solely on the choice of Φ and U, and we haven't made any such choices yet.

4 Solving the Eigenvalue Problem

To solve for U and its eigenvalues, we use their relationship we've defined with c_n, whose values are known, and a frequency range we'll define. For the latter, we'll choose J frequency values φ_i such that $dt\omega_{min} < \varphi_i < dt\omega_{max}$.

Defining $z_j := e^{-i\varphi_j}$, we will define M basis vectors that will help us in our calculations:

$$\Psi_j := \sum_{n=0}^{M} e^{in\varphi_j} U^n \Phi = \sum_{n=0}^{M} (U/z_j)^n \Phi \tag{7}$$

where M is the number of Φ basis vectors required to form a complete basis. The relationship between these new Ψ_j vectors and our H's eigenbasis is just a linear

combination due to basic linear algebra. We will define this linear combination by

$$v_k = \sum_{j=1}^{J} B_{jk}\Psi_j \tag{8}$$

In defining a workable set of basis vectors, we'll redefine our U operators in this basis in order to recast our eigenvalue problem. Let $U^{(p)}$ and S be matrices such that

$$U^{(p)}_{jj'} := (\Psi_j, U^p\Psi_{j'}), \tag{9}$$

$$S := U^{(p=0)} \Rightarrow S_{jj'} := (\Psi_j, \Psi_{j'}) \tag{10}$$

Our eigenvalue problem is then

$$U^{(1)}B_k = e^{-idt\omega_k}SB_k \tag{11}$$

The reason why this formulation is advantageous is, since our $\{\Phi\}$ basis is all linear combinations of $\{U^n\Phi\}$ vectors, our $U^{(p)}$ matrices only depend on our c_n and φ data, which we know explicitly:

$$U^{(p)}_{jj'} = \sum_{n'=0}^{M}\sum_{n=0}^{M} z^{-n}(z')^{-n'}(\Phi, U^{n'}U^{n+p}\Phi) \tag{12}$$

$$= \sum_{n'=0}^{M}\sum_{n=0}^{M} (\frac{z'}{z})^n c_{n+n'+p}(z')^{-(n+n')}, \tag{13}$$

$$d_k = (\sum_{j=1}^{J} B_{jk}\sum_{n=0}^{M} c_n z_j^{-n})^2 \tag{14}$$

where $z = e^{-i\varphi_j}, z' = e^{-i\varphi_{j'}}$.

5 What is Needed from Tidy3D

The type of signal from which we'll extract our resonance data is the Tidy3D FieldTimeMonitor object.

We need the time points to be evenly spaced out (dt is constant), and the length of each non-time coordinate to be 1 ($Field(x, y, z, t) = Field_{xyz}(t)$). Here we'll label the number of frequencies as n_f and the number of time signals as n_t.

6 The Algorithm

We are now in a position where, given only our c_n signal and chosen frequency window $(\omega_{min}, \omega_{max})$, we can calculate our $U^{(p)}$ matrices, and thus our d_k and ω_k values. This is what we do:

1. Choose $J = 200$ frequency values evenly spaced between $(2\pi\omega_{min}dt, 2\pi\omega_{max}dt)$

2. Construct our $U^{(p)}$ matrices, with $M = (N-2)/2$. (Recall that N is the number of c_n points we have minus 1.)

3. Define our initial eigenvalues as $u_k := e^{-idt\omega_k}$ from the J frequencies we chose in Step 1.

4. Solve the eigenvalue problem $U^{(1)}B_k = u_k U^{(0)}B_k$ as well as solve for our d_k values. Additionally, we eliminate eigenvalues less than 1e-4 times the largest eigenvalue found - this makes the eigenvalue problem better conditioned.

5. Return to step 3 with updated u_k as these newly found eigenvalues. We iterate until the previous eigenvalues equal the newly solved eigenvalues, or after we've repeated this step $1 + 2J$ times.

Once we have our converged ω_k, d_k resonance and amplitude data, we can calculate the rest of the resonance data:

- Frequencies: $\frac{Re(\frac{i}{2\pi}\log(u_k))}{dt}$
- Decay values: $-\frac{Im(i\log(u_k))}{dt}$
- Q factors: $\pi\frac{|u_k|}{\text{decays}}$
- Amplitudes: $|d_k|$
- Phases: $\text{Angle}(d_k) = \arctan(\frac{Im(d_k)}{Re(d_k)})$
- Errors: $||(U^{(2)} - u_k^2 U^{(0)})B_k||$

API references

To see these principles in action, check out our Tidy3D documentation on field projection:

API Reference

Source code

Resonance Finder FAQ

Example notebooks

General Usage

Calculating Resonance Data

Acknowledgements

The Tidy3D harmonic inversion plugin was written by Charles Wojcik, Ph.D., Research Scientist at Flexcompute.

References

[1] V. A. Mandelshtam, H. S. Taylor. "Harmonic inversion of time signals and its applications." J. Chem. Phys. 107, 6756–6769 (1997)

[2] J. Main, V. A. Mandelshtam, G. Wunner, H. S. Taylor. "Harmonic inversion as a general method for periodic orbit quantization." Nonlinearity 11 1015 (1998)