FDTD: solving 1+1D delay PDE in parallel

Yao-Lung L. Fang[1,2]

[1]Department of Physics, Duke University, P.O. Box 90305, Durham, North Carolina 27708-0305, USA
[2]Computational Science Initiative, Brookhaven National Laboratory, Upton, NY 11973-5000, USA*

Dated: February 20, 2018

Abstract

We present a proof of concept for solving a 1+1D complex-valued, delay partial differential equation (PDE) that emerges in the study of waveguide quantum electrodynamics (QED) by adapting the finite-difference time-domain method (FDTD). The delay term is spatially non-local, rendering the conventional approaches such as the method of lines inapplicable. We show that by properly designing the grid and by using the (partial) exact solution as the boundary condition, the delay PDE can be numerically solved. In addition, we demonstrate that even with strong data dependency, such a problem can still be solved parallelly. Our code provides a numerically exact solution to the time-dependent multi-photon scattering problem in waveguide QED. The program is written in C and open-sourced on GitHub.

Keywords: Waveguide QED, Delay PDE, FDTD, Non-Markovianity

Contents

*Present address.

1 Overview

1.1 Introduction

Waveguide quantum electrodynamics (QED) concerns the interaction between one-dimensional (1D) waveguide photons and local emitters (atoms, qubits, etc.) [1–4]. In the cases where a coherent feedback loop is formed due to the presence of, for example, multiple distant emitters or a perfect mirror terminating the waveguide, the propagation of photons needs to be taken into account rigorously if the time of flight is non-negligible compared to the decay time [5–10].

As a concrete example, we consider a two-level system (2LS) coupled to a semi-infinite waveguide, one end of which is terminated by a perfect mirror [7, 11]. Denoting the atom-mirror separation as a, in the two-excitation sector we arrive at a 1+1D delay partial differential equation (PDE) that describes the (complex-valued) time-dependent wavefunction, $\psi(x,t)$, for the 2LS plus a photon at position x:

$$\begin{aligned}\frac{\partial}{\partial t}\psi(x,t) = &-\frac{\partial}{\partial x}\psi(x,t) - \left(i\omega_0 + \frac{\Gamma}{2}\right)\psi(x,t) + \frac{\Gamma}{2}\psi(x-2a,t-2a)\theta(t-2a)\\ &-\frac{\Gamma}{2}\Big\{\Big[\psi(-x-2a,t-x-a)-\psi(-x,t-x-a)\Big]\theta(x+a)\theta(t-x-a)\\ &\quad+\Big[\psi(2a-x,t-x+a)-\psi(-x,t-x+a)\Big]\theta(x-a)\theta(t-x+a)\Big\}\\ &+\sqrt{\Gamma}\Big[\chi(x-t,-a-t,0)-\chi(x-t,a-t,0)\Big],\end{aligned} \tag{1}$$

where ω_0 (Γ) is the 2LS frequency (decay rate), $\theta(x)$ is the step function, and $\chi(x_1,x_2,t)$ is the two-photon wavefunction (one at position x_1, another at x_2). By solving Eq. (1), the full dynamics of the system can be completely determined. For example, the time-dependent two-photon wavefunction is given by

$$\begin{aligned}\chi(x_1,x_2,t) = \chi(x_1-t,x_2-t,0) - \frac{\sqrt{\Gamma}}{2}\Big[&\psi(x_1-x_2-a,t-x_2-a)\theta(x_2+a)\theta(t-x_2-a)\\ &-\psi(x_1-x_2+a,t-x_2+a)\theta(x_2-a)\theta(t-x_2+a)+\big(x_2\leftrightarrow x_1\big)\Big].\end{aligned} \tag{2}$$

The detailed discussion of this problem is reported elsewhere [11].

While delay PDEs have emerged in many scientific and engineering contexts and been numerically studied using, for example, the method of lines [12], we emphasize that as far as we understand, those approaches cannot be directly applied to our problem, as the delay term in Eq. (1) lies in both x and t dimensions (i.e., it's a non-local delay), contrary to the common situation of delay PDE in which only one of the dimensions is delayed [12]. In other words, Eq. (1) cannot be converted to a system of ordinary differential equations (ODE) that are discretized in x and then solved by an ODE solver along t. Furthermore, to the best of our knowledge there is no general-purpose solvers for delay PDE. As a result, we implement the finite-difference time-domain (FDTD) method and demonstrate its validity in this paper.

The main purpose of this work is therefore three-fold: (i) to numerically solve Eq. (1) using FDTD; (ii) to provide a proof of concept that FDTD works well for tackling complex-valued, spatially non-local delay PDE and that certain degree of parallelism can be achieved; (iii) to present a numerically exact solution to the *time-dependent, multi-photon scattering* problem in waveguide QED.

1.2 Implementation and architecture

FDTD is widely used by engineers, especially those designing antennas, doing computational electrodynamics, studying plasmonics, etc. Below we briefly discuss how FDTD works, and refer interested readers to Refs. [13, 14] for the details.

Like most of differential-equation solving methods, FDTD discretizes the spacetime, and different discretizations have different pros and cons. For simplicity we choose a square lattice, and note in passing that in real FDTD applications the Yee (staggered) lattice is more common, as the conservation laws of EM fields are trivially true on the Yee lattice. In the following we set the ratio of the spatial step Δ_x to the temporal step Δ_t equal to the speed of light $c = 1$ so that $\Delta_t = \Delta_x = \Delta$. In FDTD, $\Delta_x/\Delta_t \geq c$ is called the Courant condition and is hold when the algorithm is stable. See Appendix A for a simplified discussion on stability.

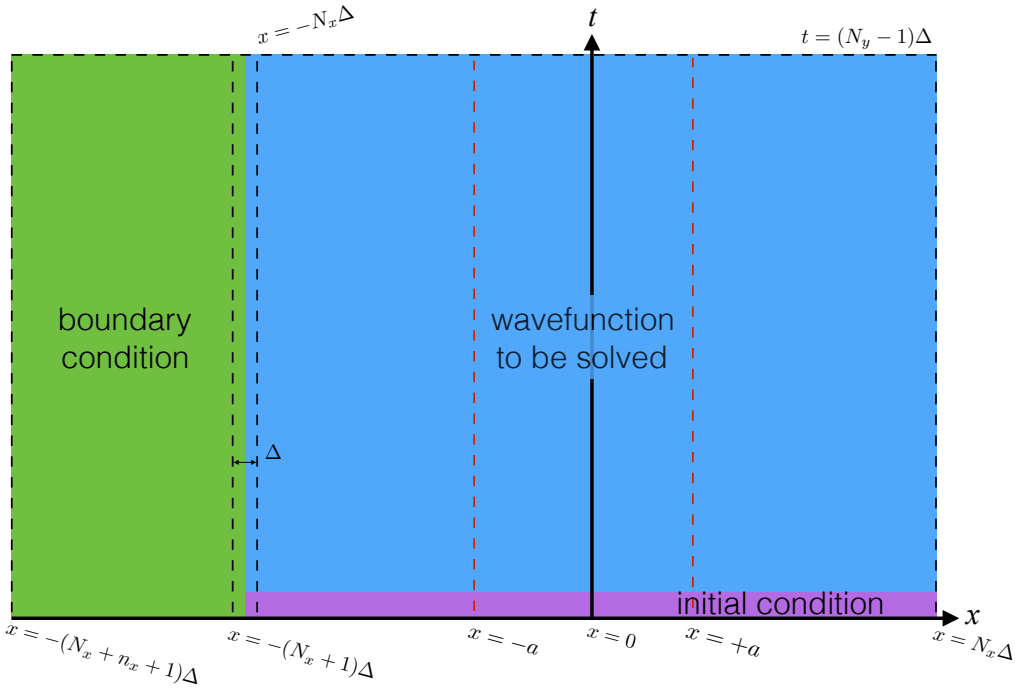

Figure 1: The schematic of the spacetime layout used in FDTD. Note that the lines $x = \pm a = \pm(n_x/2)\Delta$ locate in the blue region if $n_x \leq 2N_x$, and that $x = -a$ is at the center of the grid.

We next express every term in Eq. (1) using finite differences. We use the "leapfrog" prescription that has $\mathcal{O}(\Delta^2)$ accuracy [15]:

$$\frac{\partial f(x+\Delta/2,t+\Delta/2)}{\partial t} \approx \frac{1}{\Delta}\left(\frac{f(x,t+\Delta)+f(x+\Delta,t+\Delta)}{2} - \frac{f(x,t)+f(x+\Delta,t)}{2}\right) \tag{3}$$

$$\frac{\partial f(x+\Delta/2,t+\Delta/2)}{\partial x} \approx \frac{1}{\Delta}\left(\frac{f(x+\Delta,t+\Delta)+f(x+\Delta,t)}{2} - \frac{f(x,t)+f(x,t+\Delta)}{2}\right) \tag{4}$$

$$f(x+\Delta/2,t+\Delta/2) \approx \frac{1}{4}\Big(f(x+\Delta,t+\Delta)+f(x+\Delta,t)+f(x,t+\Delta)+f(x,t)\Big) \tag{5}$$

In the code we call Eq. (5) a "square," because the value at the center of a square is approximated using the values at its four corners. These discretization rules apply to the terms in the first line of Eq. (1) so that given the values at three corners, the value at the top right corner of a square can be solved. As for those terms in the second and the third lines of Eq. (1), hereafter referred to as the source terms [11], we need a different representation:

$$f(x+\Delta/2,t) \approx \frac{1}{2}\Big(f(x,t)+f(x+\Delta,t)\Big), \tag{6}$$

which is referred to as a "bar" in the code. The reason for using bars over squares will become clear shortly.

Now we have everything we need to discretize Eq. (1). For putting the problem on a computer, we also need to draw a "box" as we cannot indefinitely walk through the entire spacetime. As a result, we need to specify 4 parameters to define the box geometry: `Nx`, `Ny`, `nx`, and `Delta`; see Table 1 for the list of accepted input parameters. The corresponding layout is shown in Fig. 1. Note that (a) the initial condition is given on the line $t = 0$ (the purple strip); (b) the boundary condition is given for not just one line, as needed for solving ordinary (space-local and time-local) PDEs, but for a wide area (the green region) because of the delay and source terms; (c) to reach $x \geq +a$ and to make the layout well-defined, we need n_x to be an integer multiple of 2 and $n_x \leq 2N_x$; (d) the step size Δ can be given *arbitrarily*, see Appendix B.

It is illustrative to see how the FDTD solves Eq. (1). Fig. 2 shows a snapshot of the FDTD solver which marches in a space-then-time manner. Since the delay PDE is *chiral* (unidirectional) [11], for a given time t the solver (conceptually represented by the black cross) moves from the edge of boundary condition toward positive x, then advances one step Δ in time and repeats. Each term in Eq. (1) has a different color for easy identification, and we use all previously solved values to solve for the top-right corner of the blue square, which is circled in red. Note that there are four colors that each has only two points (the "bars"), because when we Taylor-expand at the black cross, those terms are expanded at the center between the two points.

Furthermore, from Fig. 2 one can appreciate the fact that in general the system is highly "non-Markovian," a jargon widely used in the community of open quantum systems [16–18], since solving the wavefunction needs to look up the system's memory (solved values at earlier times); that is, the quantum system has a dependence on its past history. In terms of programming, this brings in a heavy burden because one can no longer discard values at earlier times by flushing them from memory to disk, as typically done in solving ordinary PDEs. Instead, one needs to keep all grid points in the memory, so the hardware capacity is an important factor; doing frequent I/O is not an efficient option. For example, as we are solving a complex wavefunction, depending on the grid size it can be very memory-consuming (each grid point stores a complex double number and thus takes 16 bytes) and space-consuming (the wavefunction is written to a plain txt when the calculation is done). So use the program with caution. A quick estimation for memory usage is roughly $32N_xN_y/1024^3$ (in GB), and for disk usage divided by `Tstep` + 1.

[1] Needed when `init_cond=3` and `identical_photons=0`.

[2] Needed when `init_cond=2`, and ineffective when `init_cond=1`.

[3] These options cannot be simultaneously turned off (set to 0), or no output will be generated.

[4] Requires `init_cond=2`.

Table 1: Summary of all input parameters accepted by our program.

Parameter	Description	Mandatory	Default
`Nx`	Defined such that total grid points (to be solved) along x is $2N_x+1$ (so $x/\Delta \in [-N_x, N_x]$)	Yes	n/a
`Ny`	Total grid points along t (so $t/\Delta \in [0, N_y-1]$)	Yes	n/a
`nx`	$n_x = 2a/\Delta$; n_x needs to be an integer multiple of 2 and $n_x \leq 2N_x$	Yes	n/a
`Delta`	Step size Δ (See Appendix B)	Yes	n/a
`k`	Driving frequency k (in units of Δ^{-1})	Yes	0
`k1, k2`	Incident frequencies for photon #1 (#2) (in units of Δ^{-1})	No[1]	0
`w0`	2LS frequency ω_0 (in units of Δ^{-1})	Yes	n/a
`gamma`	2LS decay rate Γ (in units of Δ^{-1})	Yes	n/a
`init_cond`	1: two-photon scattering (plane wave); 2: stimulated emission (one-photon exponential wavepacket); 3: two-photon exponential wavepacket	Yes	0
`alpha`	Wavepacket width α (in units of Γ)	No[2]	0
`alpha1, alpha2`	Wavepacket width for photon #1 (#2) (in units of Γ)	No[1]	0
`identical_photons`	whether or not the two incident photons are the same	No	1
`save_chi`	Output $\chi(a+\Delta, a+\Delta+\tau, t)$ as plain text	No[3]	0
`save_psi`	Output $\psi(x,t)$ as plain text	No[3]	0
`save_psi_binary`	Output $\psi(x,t)$ as binary	No[3]	0
`save_psi_square_integral`	Output $\int dx \lvert\psi(x,t)\rvert^2$ as plain text	No[3]	0
`measure_NM`	See Appendix D	No[3,4]	0
`Tstep`	Output the wavefunctions for every `Tstep` $+1$ temporal steps	No	0
`Nth`	Number of solvers	No	1

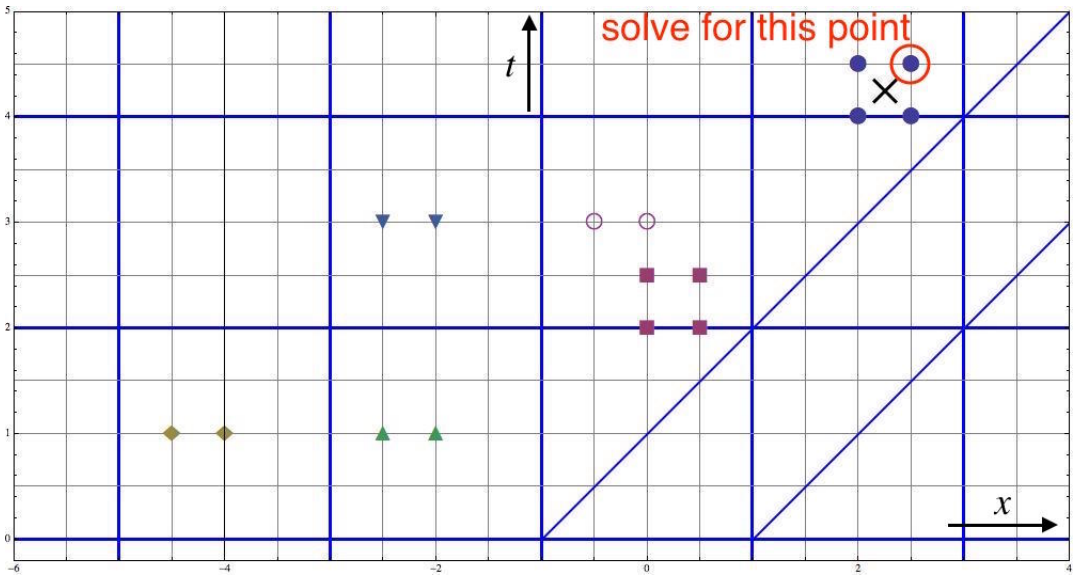

Figure 2: An example of calculating $\psi(x,t)$ for the circled point on the square lattice. The slant lines represent light cones extended from the coupling points at $x = \pm a$. n_x is chosen to be 4 for illustrative purposes, and the black cross denotes the Taylor-expansion point, referred to as the solver. The other colored points contribute to the point to be solved [each color corresponds to a term in Eq. (1)].

Currently the FDTD program can solve three classes of problems, and for each class the boundary condition (green in Fig. 1) is given by known analytical expressions [11]:

(a) *Two-photon scattering*: the incoming photons are described by continuous wave, $\chi(x_1,x_2,0) = A^2\exp[ik(x_1+x_2)]\theta(-a-x_1)\theta(-a-x_2)$, and the initial condition for ψ is simply $\psi(x,0)=0$ (the 2LS is initially in its ground state). Therefore, we need to supply three more parameters that are physics-related: `k`, `w0`, and `gamma`; see Table 1. So for this case the program needs 7 mandatory parameters in total.

In $x<-a$, the solution to Eq. (1) is

$$\psi(x,t) = \sqrt{2}Ae^{ik(x-t)}e_0(t), \tag{7}$$

where $e_0(t)$ is the 2LS wavefunction, solved in the one-excitation sector assuming $e_0(0)=0$ and $\phi(x,0) = Ae^{ikx}\theta(-a-x)$,

$$\begin{aligned} e_0(t) = &\frac{i\sqrt{\frac{\Gamma}{2}}Ae^{-ika}(e^{-ikt}-e^{-(i\omega_0+\Gamma/2)t})}{k-\omega_0+i\Gamma/2} \\ &- Ae^{-ika}\sum_{n=1}^{\infty}\frac{\left(\frac{\Gamma}{2}\right)^{n-1/2}}{n!p^{n+1}}\Big[p^{n+1}(t-2na)^n e^{-(i\omega_0+\Gamma/2)(t-2na)} \\ &+ i^n(k-\omega_0)\gamma(n+1,-ip(t-2na))e^{-ik(t-2na)}\Big]\theta(t-2na), \end{aligned} \tag{8}$$

where $p = k-\omega_0+i\Gamma/2$, and $\gamma(n,z)$ is the (lower) incomplete Gamma function [19]. We note that evaluating $\gamma(n,z)$ on the complex plane is in general a non-trivial task; see the discussion in Appendix C. The above expressions are used to generate initial and boundary conditions. Finally, we note that $A=1$ is set in the program for convenience.

(b) *Stimulated emission*: a single-photon exponential wavepacket

$$\varphi(x) = i\sqrt{\alpha\Gamma}e^{ikx+\alpha\Gamma(x+a)/2}\theta(-x-a), \tag{9}$$

is sent in, with the 2LS initially excited, so $\psi(x,0) = \varphi(x)$ and $\chi(x_1,x_2,0) = 0$. In this case, one also needs to specify the wavepacket width α in the input file in addition to the seven mandatory parameters for this case. In $x < -a$, the boundary condition is given by

$$\psi(x,t) = \varphi(x-t) \times e_1(t), \tag{10}$$

where $e_1(t)$ is again the 2LS wavefunction solved in the one-excitation sector assuming $e_1(0) = 1$ and $\phi(x,0) = 0$,

$$e_1(t) = e^{-(i\omega_0+\frac{\Gamma}{2})t} \sum_{n=0}^{\infty} \frac{1}{n!} \left[\frac{\Gamma}{2} e^{(i\omega_0+\frac{\Gamma}{2})2a}(t-2na)\right]^n \theta(t-2na). \tag{11}$$

(c) *Two-photon exponential wavepacket*: The corresponding initial conditions are $\psi(x,0) = 0$ and

$$\chi(x_1,x_2,0) = \frac{A}{\sqrt{2}} \left[\varphi_1(x_1)\varphi_2(x_2) + \varphi_1(x_2)\varphi_2(x_1)\right], \tag{12}$$

where A is the normalization constant such that $\iint dx_1 dx_2 |\chi(x_1,x_2,0)|^2 = 1$ and $\varphi_i(x)$ is the i-th photonic wavepacket, assumed of the form Eq. (9) with incident frequency k_i and width α_i. The normalization constant can be chosen to be positive without loss of generality:

$$A = \sqrt{\frac{4(k_1-k_2)^2 + (\alpha_1+\alpha_2)^2\Gamma^2}{4(k_1-k_2)^2 + (\alpha_1+\alpha_2)^2\Gamma^2 + 4\alpha_1\alpha_2\Gamma^2}}. \tag{13}$$

Following our standard procedure, we first solve for $\psi(x < -a, t)$ and then plug it into the FDTD code to solve in the region $x > -a$. We find that the solution is given by [20]

$$\psi(x < -a, t) = A\left[\varphi_1(x-t)e_0^{(2)}(t) + \varphi_2(x-t)e_0^{(1)}(t)\right], \tag{14}$$

where $e_0^{(i)}(t)$ is the qubit wavefunction in the one-excitation sector, solved subject to $e(0) = 0$ and $\phi(x,0) = \varphi_i(x)$:

$$\begin{aligned} e(t) = {} & \frac{\sqrt{\alpha\Gamma^2/2}(e^{-(i\omega_0+\Gamma/2)t} - e^{-(ik+\alpha\Gamma/2)t})}{p} - i\sqrt{\alpha\Gamma}\sum_{n=1}^{\infty} \frac{\left(\frac{\Gamma}{2}\right)^{n-1/2}}{n!} \Big[(t-2na)^n e^{-(i\omega_0+\Gamma/2)(t-2na)} \\ & + \frac{i^n(k-\omega_0-i\alpha\Gamma/2)}{p^{n+1}}\gamma(n+1,-ip(t-2na))e^{-(ik+\alpha\Gamma/2)(t-2na)}\Big]\theta(t-2na) \end{aligned} \tag{15}$$

with $p = k - \omega_0 + i\Gamma/2(1-\alpha)$. Note that when the two photons are identical, $\varphi_1 = \varphi_2 = \varphi$, the expression above reduces to the known form: $\psi(x < -a, t) = \sqrt{2}\varphi(x-t)e_0(t)$ [cf. Eq. (7)].

One could easily tweak the code to accept other kinds of initial conditions, but the strategy for solving Eq. (1) remains the same for all possible scenarios. It is important to note that if we cannot provide analytical solutions in $x < -a$ as the boundary condition, we cannot solve Eq. (1) numerically. But we do not need to know more either — the constraint $n_x \leq 2N_x$ guarantees that knowing the wavefunction in $x < -a$ is sufficient, as it makes the line $x = -a$ lie outside of the green region in Fig. 1 so that the boundary condition is completely determined.

Finally, to calculate various non-Markovian measures using the solution of ψ, instead of post-processing it is much easier to do most of the work *in situ*. To construct the geometric measure [21], two functions $\lambda(t)$ and $\mu(t)$ can be calculated [11].[5] The detail of the implementation is presented in Appendix D.

[5] For some reasons, in Ref. [11] $\mu(t)$ is called $c(t)$ and $\lambda(t)$ is called $\Delta(t)$, respectively. The author regrets the inconvenience.

1.3 Parallelization

As discussed above, due to the non-local delay term there is a strong data dependency — the present solution depends on its past history. However, even with such a severely constrained problem, interestingly the delay PDE can still be solved parallelly. By parallelism here we mean that the simultaneous presence of multiple FDTD solvers in the spacetime is allowed, see the schematic shown in Fig. 3. Great care must be taken, though, in order to satisfy the constraints set by the structure of the delay PDE (1). Specifically, each solver must be one temporal step above and at least n_x spatial steps behind its previous partner (called predecessor hereafter), or the causality would not be preserved correctly; that is, one solver may read the value at certain point which is yet to be solved by another solver. It turns out that it is beneficial to have a "cyclic lag" relation among the solvers: denoting the total number of solvers as `Nth`, which can be set in the input file, then #2 is lagged behind #1, #3 behind #2, etc., and finally #1 can be set behind #`Nth` once it reaches the grid boundary. We find that such a wrap-around can increase performance and is easy for coding.

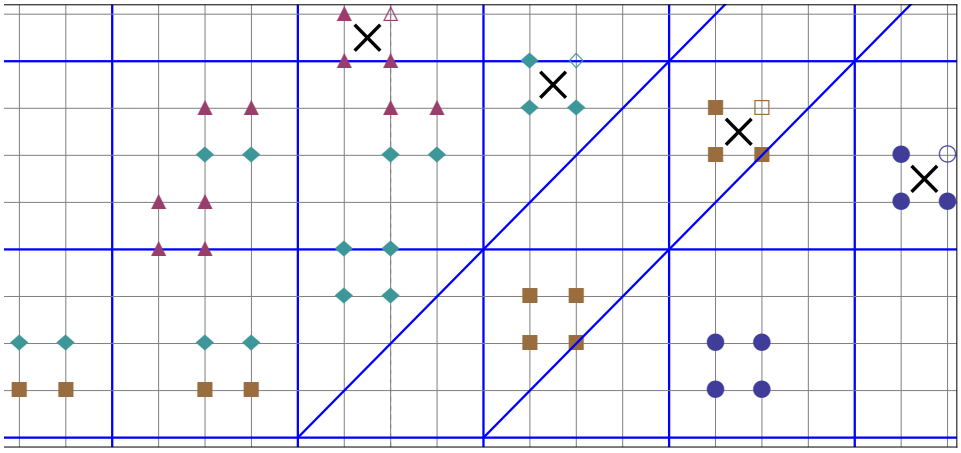

Figure 3: A snapshot of multiple marching FDTD solvers. Each color and shape corresponds to points read and written by a certain solver (thread). The points to be solved by each solver are empty, while known points are filled. Note that each solver is one step above and at least n_x steps behind its predecessor (here $n_x = 4$).

In the current implementation, we utilize the "wait-and-signal" mechanism provided by POSIX Threads (pthreads). Each solver is marched by an independent thread and has a lock for its position and a condition variable for waiting for its predecessor. Before attempting to solve at a certain point, it must look up its predecessor's position and see whether the constraint is satisfied or not. If not, then it must wait for the signal sent by its predecessor before proceeding. Using pthreads' locks not only allows a solver to communicate with and wait for its predecessor, but also preserves cache coherence, which is important since it is possible that a solver needs to access a value immediately after it's solved by another solver. In the end of the next section, we present a time measurement for a typical problem size required by the physics [11, 20, 22].

1.4 Quality control

User instruction for the program is given in the `README.md` file distributed along with the code. We have tested our program on both Linux and Mac OS X. There are at least three possible tests to pass for establishing the validity of this code: 1. ψ in $x < -a$, where we know the full, exact solution; 2. ψ in $-a < x < a$ because we can solve for the first few triangular tiles analytically [11]; 3. the two-photon wavefunction χ, because we know the steady-state result from scattering theory [7].

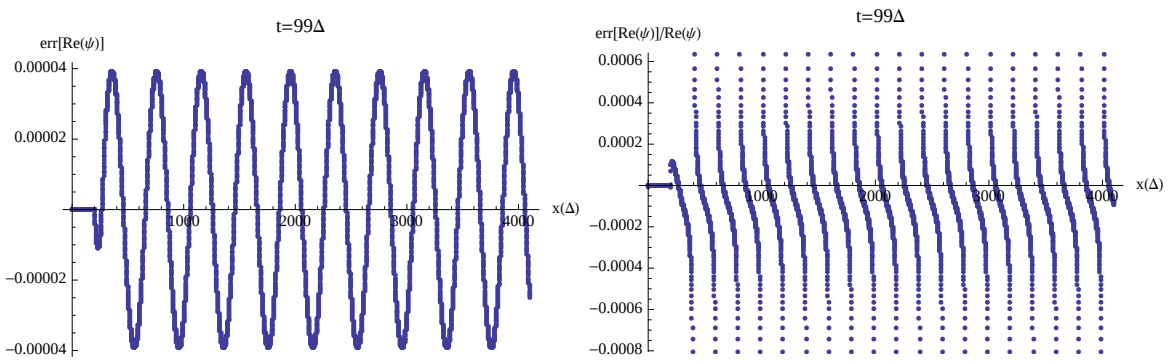

Figure 4: Absolute (left) and relative (right) errors for $\mathrm{Re}(\psi)$ as a function of spatial steps (in units of Δ) at $t = 99\Delta$. The last step corresponds to $x = -a$. Input parameters: $n_x = 200, N_x = 4000, N_y = 4 \times 10^4, \Delta = 10^{-2}, k = \omega_0 = \pi/2, \Gamma = \pi/40$.

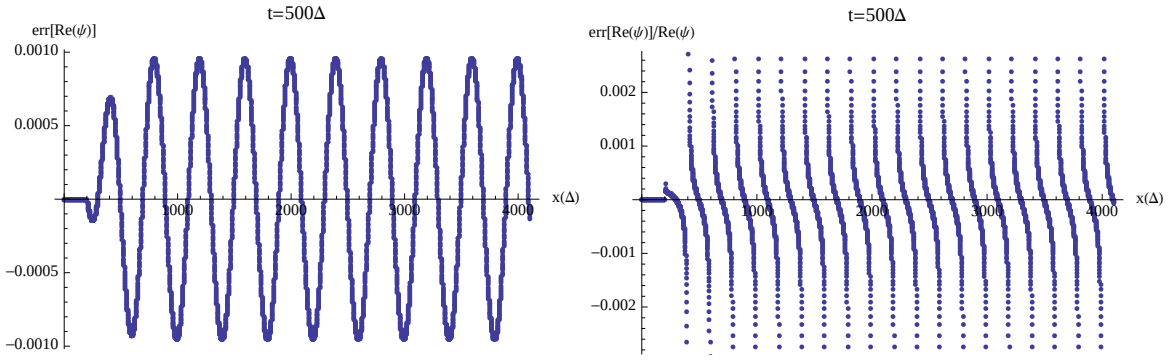

Figure 5: Absolute (left) and relative (right) errors for $\mathrm{Re}(\psi)$ as a function of spatial steps (in units of Δ) at $t = 500\Delta$. All parameters are the same as those in Fig. 4.

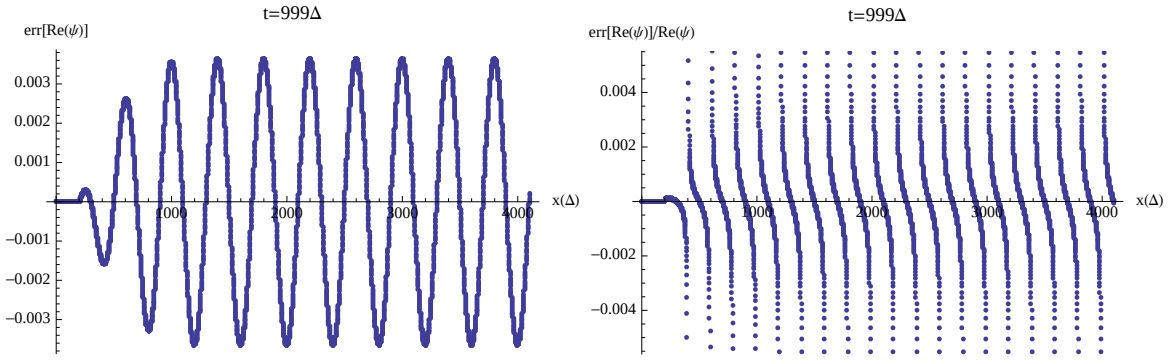

Figure 6: Absolute (left) and relative (right) errors for $\mathrm{Re}(\psi)$ as a function of spatial steps (in units of Δ) at $t = 999\Delta$. All parameters are the same as those in Fig. 4.

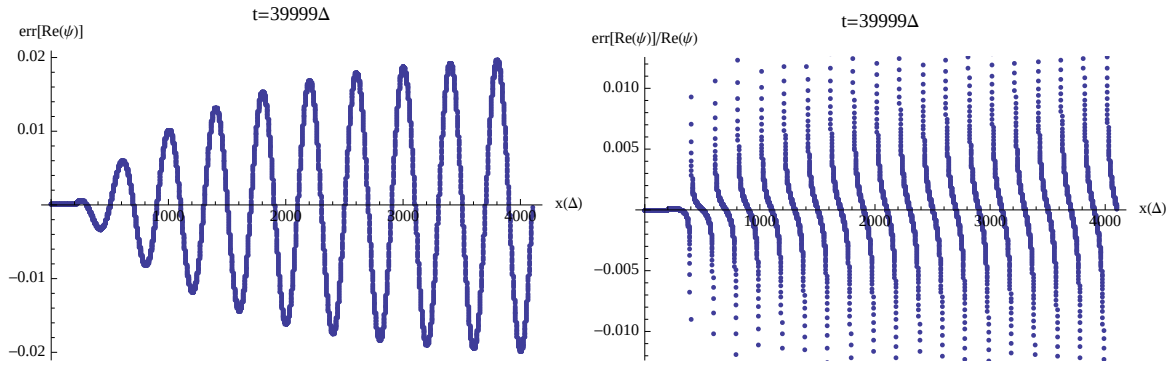

Figure 7: Absolute (left) and relative (right) errors for $\mathrm{Re}(\psi)$ as a function of spatial steps (in units of Δ) at $t = 39999\Delta$. All parameters are the same as those in Fig. 4.

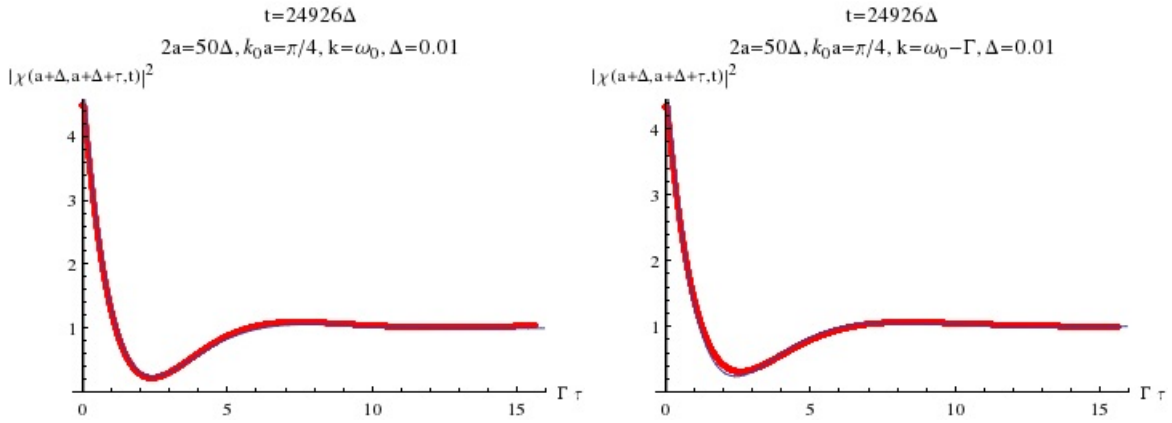

Figure 8: Comparison of calculated two-photon wavefunction $|\chi|^2$ (red dots) as a function of photon separation τ in the long-time limit for $k_0 a = \pi/4$ and (left) $k = \omega_0$ (right) $k = \omega_0 - \Gamma$. The blue curves are from scattering theory with Markovian approximation [7]. Input parameters: $n_x = 50, N_x = 10^4, N_y = 2.5 \times 10^4, \Delta = 10^{-2}, \omega_0 = 3.1415926536, \Gamma = 0.1570796327$.

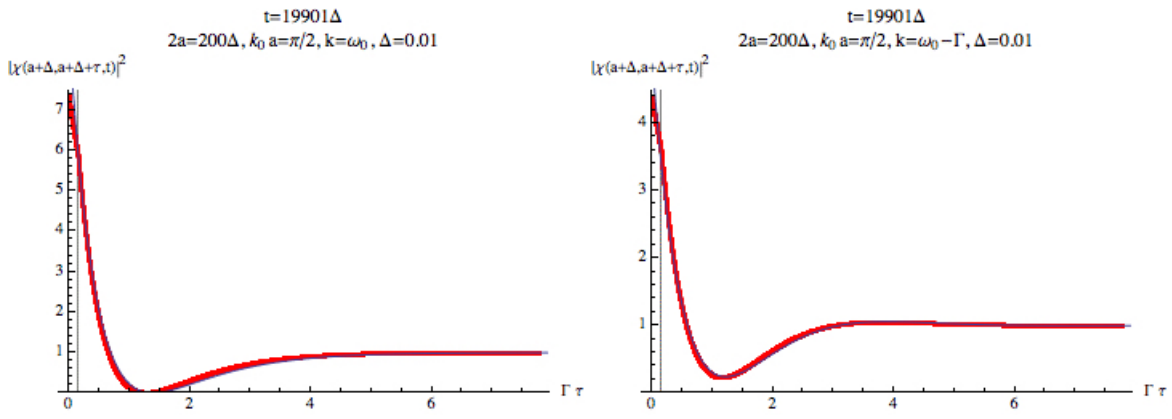

Figure 9: Comparison of calculated two-photon wavefunction $|\chi|^2$ (red dots) as a function of photon separation τ in the long-time limit for $k_0 a = \pi/2$ and (left) $k = \omega_0$ (right) $k = \omega_0 - \Gamma$. The blue curves are from scattering theory with Markovian approximation [7], and the vertical lines indicate $t = 2a$. Input parameters: $n_x = 200, N_x = 10^4, N_y = 2 \times 10^4, \Delta = 10^{-2}, \omega_0 = 1.5707963268, \Gamma = 0.0785398163$.

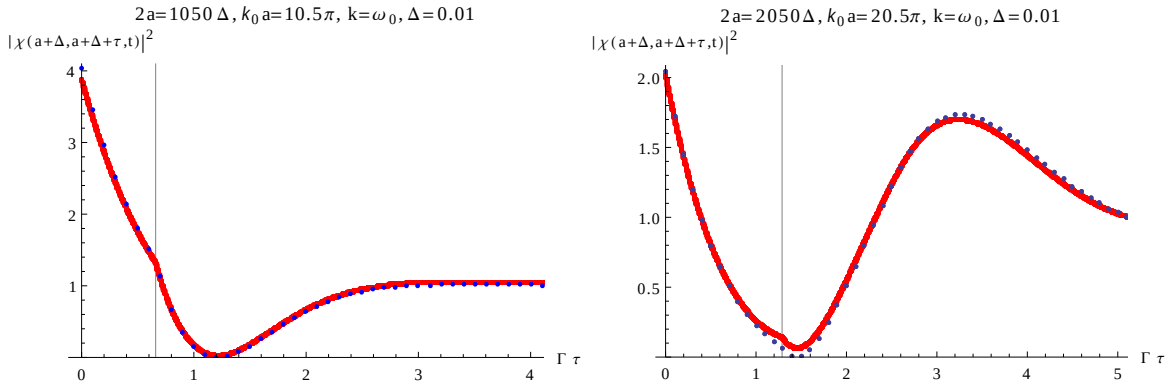

Figure 10: Comparison of calculated two-photon wavefunction $|\chi|^2$ (red dots) as a function of photon separation τ in the long-time limit for $k = \omega_0$ and (left) $k_0 a = 10.5\pi$ (right) $k_0 a = 20.5\pi$. The blue dots are from exact scattering theory [7], and the vertical lines indicate $t = 2a$. Input parameters: $N_x = 10^4$, $N_y = 5 \times 10^4$, $\Delta = 10^{-2}$, $k = \omega_0 = 6.2831853072$, $\Gamma = 0.0628318531$.

For the first test, in Figs. 4-7 we show several snapshots of the absolute and relative errors of the real part of ψ as a function of steps in the x-direction (and $x \leq -a$ in all plots). Note two common features in these plots: (a) the error is zero in the initial steps because it's where the boundary condition is; (b) errors are well-controlled in the sense that they are bounded in an oscillating envelope, so the program behaves correctly at least in $x < -a$.

For the second test, we generated the analytical expressions of ψ in the first four tiles (the fifth one took too much time to compute) in *Mathematica* [11] and then compared the values on the grid points of FDTD. The numerical result agreed well too (not shown).

Finally, to compare with the scattering theory we can construct the two-photon wavefunction $\chi(x_1, x_2, t)$ using the calculated ψ according to the formal solution of χ, Eq. (2), where the coordinates x_1 and x_2 should be chosen as $x_{1,2} \geq +a$ for capturing the outgoing fields. In the program, χ is calculated by setting $x_1 = a + \Delta$ and $x_2 = a + \Delta + \tau$, with $\tau = x_2 - x_1$ being the separation of the two detectors.[6] The results are shown in Figs. 8-10. The agreement is quite well, even in the non-Markovian regime (Fig. 10). Because evaluating Eq. (2) requires the full history of ψ, it is clear the program is valid for all regions in the spacetime. More results generated by our FDTD program are discussed in Refs. [11, 20, 22].

We next comment briefly on the performance of multi-thread support. In Fig. 11 we report the elapsed time for solving Eq. (1) as a function of the number of threads (specified by `Nth`) for the same input parameters, which are chosen to roughly match those used in Refs. [11, 20, 22]. It is clear that while the scaling is not ideal (i.e., not $1/N$, the black dashed curve), we do observe a speedup. In order to have a fair comparison, note that when single thread is used, the pthreads library would bring unnecessary overhead. Therefore, we also report in Fig. 11 the measurement without compiling with and linking to pthreads (light blue circles), compared with which the speedup is about 2x using 32 threads.

While a 2x speedup may not be as good as one expects, we emphasize that this is the *theoretical upper bound* for the parallelization approach that we introduced. It is because while solver #1 (or whoever at the bottom of the solver swarm) moves freely, all others have to wait for it either directly (such as #2) or indirectly (for example, #3 waits for #2, #4 waits for #3 and so on). So, if the message passing among the swarm has little to zero latency, i.e. signaling happens almost instantaneously compared to the actual computation, then the rest of the swarm moves collectively as one giant thread, leading to a factor of 2x. It is an open question whether there is room for further improvement, preferably without using condition variables or mutexes.

We note in passing that the pthreads overhead can be consistently estimated by either taking the dif-

[6] We note that by definition the two-photon correlation function is given by $g_2(\tau) = |\chi(x_d, x_d + \tau, t \to \infty)|^2$ with $x_d \geq +a$.

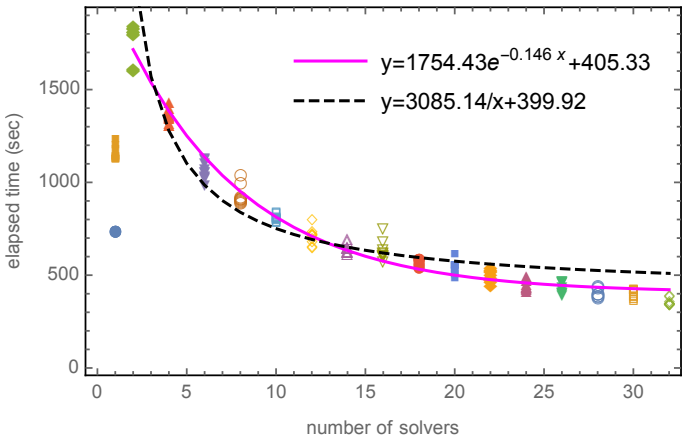

Figure 11: Measured elapsed time as a function of the number of solvers (threads) `Nth`. The test is run 10 times for each `Nth`. For single thread, the measurements without pthreads overhead (light blue circles) are also shown. The magenta curve is of the form $A\exp(-B\texttt{Nth})+C$ and the black dashed curve is $A/\texttt{Nth}+C$, both of which are fitted by all data points with $\texttt{Nth} = 1$ excluded. Input parameters: $N_x = 64800$, $N_y = 60000$, $n_x = 960$, $\Delta = 2.618 \times 10^{-4}$, `init_cond`$= 2$, $k = \omega_0 = 100$, $\Gamma = 5$, and $\alpha = 0.5$. Test machine: Intel Xeon CPU E5-2670 2.6 GHz with 32 logical cores and 128 GB physical memory.

ference between mean elapsed time with and without pthreads, or by the constant shift C from the fittings (curves in Fig. 11). Therefore, on a machine with many cores the elapsed time is dominated by the overhead only.

2 Availability

2.1 Operating system

Any modern OS that has C compilers supporting the C99 standard installed. This program has been tested on Linux + GCC and Mac OS X + Clang/LLVM.

2.2 Programming language

The FDTD code is written in C and conforms the C99 standard. Pre- and post-processing utility scripts are written in Python and *Mathematica*.

2.3 Additional system requirements

As discussed earlier, depending on the input parameters the memory and disk usages of the program can be excessive, so the users should choose the parameters wisely.

2.4 Dependencies

To enable the multi-thread support (which is turned on by default), compiling with and linking to pthreads are required. To explicitly disable it, use "make CFLAGS=-D__FDTD_NO_PTHREAD_SUPPORT__".

2.5 List of contributors

Yao-Lung Leo Fang (developer)

2.6 Software location

2.6.1 Archive

1. Name: Zenodo
2. Persistent identifier: https://doi.org/10.5281/zenodo.829809
3. Licence: WTFPL
4. Publisher: Yao-Lung Leo Fang
5. Version published: v1.0.0
6. Date published: July 15, 2017

2.6.2 Code repository

1. Name: GitHub
2. Persistent identifier: https://github.com/leofang/FDTD.git
3. Licence: WTFPL
4. Date published: July 14, 2017

2.7 Language

English.

3 Reuse potential

The program provides a numerically exact, time-dependent solution to the problem of a single 2LS placed in front of a mirror scattered by either a one- or two-photon initial state, so for waveguide-QED researchers this program can be very useful for solving the three classes of problems as discussed earlier. Arbitrary wavepackets or other forms of initial condition can be incorporated into the code with minor modification.

More flexibility, such as coupling a giant atom that has arbitrary number of legs to the waveguide [23–26], can be gained by re-implementing this program in an OO language (C++, Python, etc). In fact, the code has been written in the OO-style by (i) compacting all relevant fields into a C struct (roughly equivalent to a C++/Python class) named `grid`, (ii) passing "this" pointer to a `grid` instance when calling functions, as if they were member functions of the `grid` class, and (iii) following RAII. As a result, rewriting in C++, for example, should be of minimal work. Of course, the grid layout must be carefully designed in such a scenario to minimize memory usage.

Finally, this program serves as a proof of concept for mathematicians, scientists and engineers who are interested in solving complicated delay PDE using parallel FDTD. Important issues such as general proof of stability, multi-thread implementation, and robust algorithm for solving 1+1D delay PDE remain challenging. For development discussions, please either open an issue on GitHub or contact the authors. Users and developers who are benefited from this program are strongly encouraged to cite both this paper as well as Ref. [11].

3.1 Acknowledgments

We acknowledge financial support from U.S. NSF (Grant No. PHY-14-04125) and the Brookhaven National Laboratory's Laboratory Directed Research and Development project #17-029, and fruitful discussion with Harold U. Baranger, Francesco Ciccarello and Meifeng Lin.

3.2 Competing interests

The authors declare that they have no competing interests.

Appendix A Stability analysis

Following Ref. [15], we can perform a simple von Neumann stability analysis to check that our regularization scheme does not lead to amplitude divergence. First, let us consider the simplest ordinary PDE $\partial_x f + \partial_t f + W f(x,t) = 0$ [compared with Eq. (1), $W = \left(i\omega_0 + \frac{\Gamma}{2}\right)$]. If we write $\psi(m\Delta, n\Delta) = \xi^n e^{ik(m\Delta)}$, where ξ is called the amplification factor, then we have

$$\xi(k) = \frac{\left(\frac{1}{\Delta} - \frac{W}{4}\right) - \frac{W}{4}e^{ik\Delta}}{\left(\frac{1}{\Delta} + \frac{W}{4}\right) + \frac{W}{4}e^{-ik\Delta}} e^{-ik\Delta}. \tag{16}$$

It can be shown that $|\xi(k)| \leq 1$ as long as $\mathrm{Re}(W) > 0$, so our approach for solving this simple PDE is stable.

For Eq. (1) in $x < -a$, we can again insert the ansatz and obtain (assuming $\chi = 0$ for simplicity)

$$\begin{aligned}\left(\frac{1}{\Delta} + \frac{W}{4}\right)\xi^n e^{ikm\Delta} = \left(\frac{1}{\Delta} - \frac{W}{4}\right)\xi^{n-1}e^{ik(m-1)\Delta} - \frac{W}{4}\left(\xi^{n-1}e^{ikm\Delta} + \xi^n e^{ik(m-1)\Delta}\right) \\ + \frac{\Gamma}{8}\xi^{n-n_x-1}e^{ik(m-n_x-1)\Delta}\left(1 + e^{ik\Delta} + \xi + \xi e^{ik\Delta}\right).\end{aligned} \tag{17}$$

Note that the parameter n_x enters because of the delay term. Now this is a polynomial in ξ of order n_x, so there is no general solution for its roots. But empirically (i.e. by plotting) we still find that $|\xi(k)| \leq 1$; that is, our algorithm is stable in $x < -a$.

A more general stability analysis for Eq. (1) is challenging. However, we hope the above reasoning together with the validations discussed earlier are enough to convince the interested users that our FDTD program is stable.

Appendix B Conversions between physical quantities and simulation parameters

In the scattering problem, the three important parameters are $\mathcal{K}$ $(= k/\Gamma)$, $\mathcal{W}$ $(= \omega_0/\Gamma)$ and n $(= k_0 a/\pi)$. To satisfy the rotating-wave approximation $\mathcal{W} \gg 1$ is required, but how large $\mathcal{W}$ is should not matter. Once they are determined, the physics is determined. Here we describe how to express physical quantities in

terms of Δ, n_x, $\mathcal{K}$, $\mathcal{W}$ and n:

$$k = \frac{2n\pi\mathcal{K}}{n_x\mathcal{W}} \times \frac{1}{\Delta} \tag{18}$$

$$\omega_0 = \frac{2n\pi}{n_x} \times \frac{1}{\Delta} \tag{19}$$

$$\Gamma = \frac{2n\pi}{n_x\mathcal{W}} \times \frac{1}{\Delta} \tag{20}$$

$$\lambda_0 = \frac{n_x}{n} \times \Delta \tag{21}$$

A Python script is used to prepare the input parameters using the above relations.

We remark the role of Δ, which represents both length and time in dimensional analysis (recall $c = 1$). Thus, we have a degree of freedom (dof) to choose its value without affecting the physics, as if we were changing the length unit. We emphasize this notion because there is in fact a deeper reason for having this dof: Eq. (1) is a wave equation, and thus *scale invariant* [27]. It is well-known that FDTD is quite suitable for scale-invariant problems, and the step size Δ is just a conceptual quantity for proper discretization and is irrelevant of the physics. What matters is the dimensionless quantities such as $(\omega_0\Delta)$ and $(\Gamma\Delta)$, not Δ itself.

The current implementation is not yet strictly scale invariant — we still explicitly keep Δ as a mandatory parameter, whose value does not affect the result (providing rounding errors are negligible), but it is not necessary if the code is designed in a different way. What really controls the (relative) step size, and therefore the error, is the ratio of n_x/n; see Eq. (21). As a result, we need $n_x \gg n$ to have many steps per wavelength. Empirically we find $\Delta/\lambda_0 \leq 1/100$ is preferable.

Appendix C Evaluating incomplete Gamma functions on the complex plane

In the present work, the incomplete Gamma function $\gamma(n, z)$ naturally emerges from our equations. While there are several open-sourced math libraries such as GSL and Boost that implement the real-valued version, evaluating $\gamma(n, z)$ for complex-valued argument z is a very tricky task and to our knowledge no open-sourced code provides such an implementation. Commercial softwares like Mathematica does provide a routine for arbitrary arguments, but its efficiency is nonideal for simulation purposes. As a by-product, therefore, we provide an implementation of the incomplete Gamma function for nonzero positive integers $n = 1, 2, \cdots$, and complex-valued z.

Specifically, we implement the normalized lower incomplete Gamma function [19]

$$P(n, z) = \frac{\gamma(n, z)}{\Gamma(n)} = \frac{\gamma(n, z)}{(n-1)!}. \tag{22}$$

Other members in the incomplete-Gamma family can be easily obtained if $P(n, z)$ is known. The most challenging part is when z is on the negative real axis. Fortunately, this problem is recently tackled in Ref. [28], and we incorporate their findings into our implementation, which works well even when z has a tiny, nonzero imaginary part. Our implementation is summarized in Fig. 12. We evaluate $P(n, z)$ according to four different expressions for z in different regions on the complex plane:

(a) Continuous fraction; see Eq. (6.2.7) in Ref. [15]:

$$Q(n, z) = 1 - P(n, z) = \frac{e^{-z}z^n}{\Gamma(n)} \left(\frac{1}{z+1-n-} \frac{1\cdot(1-n)}{z+3-n-} \frac{2\cdot(2-n)}{z+5-n-} \cdots \right); \tag{23}$$

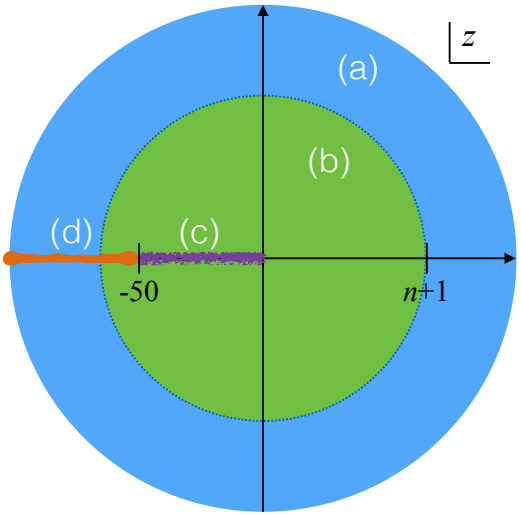

Figure 12: Evaluating $P(n,z)$ for complex-valued z. Depending on $|z|$, we use different formulae, labeled from (a) to (d), to achieve fast and accurate convergence. Note that (c) and (d) are employed when $\mathrm{Re}(z) < 0$ and $|\,\mathrm{Im}(z)| \lesssim 10^{-16}$.

(b) Series expansion; see Eq. (6.2.5) in Ref. [15]:

$$P(n,z) = e^{-z} z^n \sum_{i=0}^{\infty} \frac{z^i}{\Gamma(n+i+1)}; \tag{24}$$

(c) Series expansion for γ^*; see Eq. (6) in Ref. [28]:

$$P(n,-z) = (-z)^n \gamma^*(n,-z) \sim \frac{(-1)^n e^z}{\Gamma(n)} \sum_{i=0}^{\infty} (1-n)_i z^{n-i-1}; \tag{25}$$

(d) Poincaré-type expansion; see Eq. (29) in Ref. [28]:

$$P(n,z) = z^n \gamma^*(n,z) = \frac{z^n}{\Gamma(n)} \sum_{i=0}^{\infty} \frac{(-z)^i}{i!(i+n)}. \tag{26}$$

We note that combining (a) and (b) gives the standard approach to real-valued z [15], and that for very large $|z|$ the computation may not converge [28], but the convergence range is large enough for our purposes.

Appendix D Non-Markovian measures

In this Appendix we illustrate the computation of the non-Markovian (NM) geometric measure [21] quantifying the single-photon scattering process in this system. Specifically, we consider initially the waveguide has a single-photon exponential wavepacket of the form Eq. (9), and calculate two functions, $\mu(t)$ and $\lambda(t)$,[5] for constructing the geometric measure. The detailed discussion is reported elsewhere [11], and here we simply quote the results. The computation can be performed by setting an arbitrarily positive `alpha`, `init_cond=2`, and `measure_NM=1` in the input file.

Denoting $\phi(x,t)$ as the photon wavefunction in the one-excitation sector, we can define a complex function $\mu(t)$ as

$$\mu(t) \equiv \int_{-\infty}^{\infty} dx\, \phi^*(x,t)\psi(x,t). \tag{27}$$

Similarly, we define a real function $\lambda(t)$ as

$$\lambda(t) = \int_{-\infty}^{\infty} dx\, |\psi(x,t)|^2 - |e_0(t)|^2, \tag{28}$$

where $e_0(t)$ is given by the solution of single-photon scattering, Eq. (S27) in Ref. [11]. After calculating $\psi(x,t)$, the program will compute $\mu(t)$, $\lambda(t)$, $e_0(t)$ and $e_1(t)$ [Eq. (11)], and output the results as plain text. The geometric measure is defined as [21]

$$\mathcal{N}_{\text{geo}} = \int_{\frac{d|\det M_t|}{dt}>0} \frac{d|\det M_t|}{dt}\, dt, \tag{29}$$

and elsewhere [11] we show that for our system $\det M_t = |\mu(t)|^2\lambda(t)$. Therefore, once the two functions $\lambda(t)$ and $\mu(t)$ are known, the geometric measure can be calculated easily.[7]

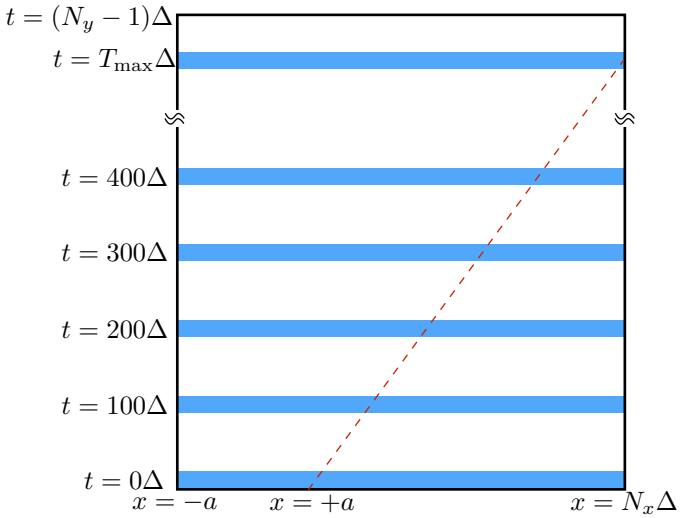

Figure 13: Spacetime layout of the FDTD output files. Information for $x < -a$ is not written to the disk in order to reduce the file size. During post processing, one can load data for every 100 steps into memory (blue strips). The red dashed line represents the light cone extended from the second 2LS at $x = +a$, and the time at which it intersects with the box boundary is denoted $t = T_{\max}\Delta$.

For users interested in computing the two functions themselves, we note that the spatial integrals in Eqs. (27) and (28) need some care, which is explained in the rest of this section. (In our code they are computed according to the trapezoidal rule.) One of the issues for numerically computing these integrals is that the amount of data generated by the FDTD program can be huge, as previously explained. Depending on the number of grid points, it can be easily over tens of GB and more. It is certainly inconvenient to write this much of data into a file and subsequently load it into the memory for post processing.

We suggest a manageable way to reduce the disk and memory usages. First, note that the analytical solution in $x < -a$ is known, so do not write data in this region into files. The resulting data layout is shown in Fig. 13. Doing this will cut half of the disk usage (compare Fig. 13 with Fig. 1). Next, depending on the system time scale, one can selectively load data for every, say, 100 steps (blue strips in Fig. 13; set `Tstep=99`). This will significantly reduce the memory usage. One can keep loading data into the memory

[7] $\lambda(t)$ and $\mu(t)$ can also be used to construct other NM measures; detail in progress will be reported elsewhere.

until reaching $t = (N_y - 1)\Delta$, the max simulation time in FDTD, but for the purpose of calculating the integrals, one may only use data up to $t = T_{\text{max}}\Delta$, where

$$T_{\text{max}} = \min(N_y - 1, N_x - n_x/2). \tag{30}$$

The reason is that the time $T_{\text{max}}\Delta$ is where the second light cone intersects with the box boundary. If data beyond this limit is used, the wavefront will go outside of the box and the integrals would be underestimated. Thus, instead of positive infinity, numerically the upper bound for those integrals should be the x-coordinate that intersects with $t = T_{\text{max}}\Delta$ at the 2nd light cone. Finally, we note that `Tstep` should be wisely chosen so that after interpolation the output data can faithfully represent the result.

References

[1] Lodahl P, Mahmoodian S, and Stobbe S, 2015, Interfacing single photons and single quantum dots with photonic nanostructures, Rev Mod Phys 87(2): 347, DOI: http://dx.doi.org/10.1103/RevModPhys.87.347.

[2] Roy D, Wilson C M, and Firstenberg O, 2017, Colloquium: Strongly interacting photons in one-dimensional continuum, Rev Mod Phys 89: 021001, DOI: http://dx.doi.org/10.1103/RevModPhys.89.021001.

[3] Noh C and Angelakis D G, 2017, Quantum simulations and many-body physics with light, Reports on Progress in Physics 80(1): 016401, DOI: http://dx.doi.org/10.1088/0034-4885/80/1/016401.

[4] Liao Z, Zeng X, Nha H, and Zubairy M S, 2016, Photon transport in a one-dimensional nanophotonic waveguide QED system, Physica Scripta 91(6): 063004, DOI: http://dx.doi.org/10.1088/0031-8949/91/6/063004.

[5] Zheng H and Baranger H U, 2013, Persistent Quantum Beats and Long-Distance Entanglement from Waveguide-Mediated Interactions, Phys Rev Lett 110(11): 113601, DOI: http://dx.doi.org/10.1103/PhysRevLett.110.113601.

[6] Tufarelli T, Kim M S, and Ciccarello F, 2014, Non-Markovianity of a quantum emitter in front of a mirror, Phys Rev A 90(1): 012113, DOI: http://dx.doi.org/10.1103/PhysRevA.90.012113.

[7] Fang Y L L and Baranger H U, 2015, Waveguide QED: Power spectra and correlations of two photons scattered off multiple distant qubits and a mirror, Phys Rev A 91(5): 053845, DOI: http://dx.doi.org/10.1103/PhysRevA.91.053845. *ibid.* **96**, 059904(E) (2017).

[8] Grimsmo A L, 2015, Time-Delayed Quantum Feedback Control, Phys Rev Lett 115(6): 060402, DOI: http://dx.doi.org/10.1103/PhysRevLett.115.060402.

[9] Ramos T, Vermersch B, Hauke P, Pichler H, and Zoller P, 2016, Non-Markovian dynamics in chiral quantum networks with spins and photons, Phys Rev A 93: 062104, DOI: http://dx.doi.org/10.1103/PhysRevA.93.062104.

[10] Pichler H and Zoller P, 2016, Photonic Circuits with Time Delays and Quantum Feedback, Phys Rev Lett 116(9): 093601, DOI: http://dx.doi.org/10.1103/PhysRevLett.116.093601.

[11] Fang Y L L, Ciccarello F, and Baranger H U, 2017, Non-Markovian dynamics of a qubit due to single-photon scattering in a waveguide. 1707.05946.

[12] Zubik-Kowal B, 2008, Delay partial differential equations, Scholarpedia 3(4): 2851, DOI: http://dx.doi.org/10.4249/scholarpedia.2851.

[13] Taflove A and Hagness S C, 2005, *Computational Electrodynamics: the Finite-Difference Time-Domain Method*, Artech House, Norwood, MA, 3rd edition.

[14] Schneider J B, 2010, *Understanding the Finite-Difference Time-Domain Method*. http://www.eecs.wsu.edu/~schneidj/ufdtd/.

[15] Press W H, Teukolsky S A, Vetterling W T, and Flannery B P, 2007, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 3rd edition.

[16] Rivas Á, Huelga S F, and Plenio M B, 2014, Quantum non-Markovianity: characterization, quantification and detection, Rep Prog Phys 77(9): 094001, DOI: http://dx.doi.org/10.1088/0034-4885/77/9/094001.

[17] Breuer H P, Laine E M, Piilo J, and Vacchini B, 2016, Colloquium: Non-Markovian dynamics in open quantum systems, Rev Mod Phys 88(2): 021002, DOI: http://dx.doi.org/10.1103/RevModPhys.88.021002.

[18] de Vega I and Alonso D, 2017, Dynamics of non-Markovian open quantum systems, Rev Mod Phys 89(1): 015001, DOI: http://dx.doi.org/10.1103/RevModPhys.89.015001.

[19] Olver F W J, Lozier D W, Boisvert R F, and Clark C W, 2010, *NIST Handbook of Mathematical Functions*, Cambridge University Press.

[20] Ciccarello F, Calajò G, Fang Y-L L, and Baranger H U, in preparation.

[21] Lorenzo S, Plastina F, and Paternostro M, 2013, Geometrical characterization of non-Markovianity, Phys Rev A 88(2): 020102, DOI: http://dx.doi.org/10.1103/PhysRevA.88.020102.

[22] Fang Y L L and Baranger H U, 2017, Erratum: Waveguide QED: Power spectra and correlations of two photons scattered off multiple distant qubits and a mirror [Phys. Rev. A 91, 053845 (2015)], Phys Rev A 96: 059904, DOI: http://dx.doi.org/10.1103/PhysRevA.96.059904.

[23] Gustafsson M V, Aref T, Kockum A F, Ekström M K, Johansson G, and Delsing P, 2014, Propagating phonons coupled to an artificial atom, Science 346(6): 207, DOI: http://dx.doi.org/10.1126/science.1257219.

[24] Frisk Kockum A, Delsing P, and Johansson G, 2014, Designing frequency-dependent relaxation rates and Lamb shifts for a giant artificial atom, Phys Rev A 90(1): 013837, DOI: http://dx.doi.org/10.1103/PhysRevA.90.013837.

[25] Guo L, Grimsmo A, Kockum A F, Pletyukhov M, and Johansson G, 2017, Giant acoustic atom: A single quantum system with a deterministic time delay, Phys Rev A 95: 053821, DOI: http://dx.doi.org/10.1103/PhysRevA.95.053821.

[26] Fang Y-L L, unpublished.

[27] Joannopoulos J D, Johnson S G, Winn J N, and Meade R D, 2008, *Photonic Crystals: Molding the Flow of Light*, Princeton University Press, 2nd edition.

[28] Gil A, Ruiz-Antolín D, Segura J, and Temme N M, 2016, Algorithm 969: Computation of the Incomplete Gamma Function for Negative Values of the Argument, ACM Trans Math Softw 43(3): 26:1, DOI: http://dx.doi.org/10.1145/2972951.