NONLINEAR SCHRÖDINGER EQUATION: A COMPARISON OF PSUEDO-SPECTRAL METHODS AND A MACHINE LEARNING APPROACH

HANNAH POTGIETER AND SUKI SHERGILL

Key words. Physics-informed Networks, Nonlinear Schrödinger Equation, Strang Splitting, Predictive Modeling, Nonlinear Dynamics, Periodic solutions

AMS subject classifications. 65M70, 65K05

1. Introduction.

The cubic one-dimensional nonlinear Schrödinger equation (NLS) is

$$i\Phi_t + \alpha\Phi_{xx} + \gamma|\Phi|^2\Phi = 0 \tag{1.1}$$

where Φ is a complex-valued function, $\gamma = \pm 1$ denoting focusing or defocusing, and α is a real constant. We can classify NLS as a semilinear partial differential equation because the nonlinearity occurs on a term without any derivatives.

The cubic two-dimensional nonlinear Schrödinger equation is

$$i\Phi_t + \alpha_1\Phi_{xx} + \alpha_2\Phi_{yy} + \gamma|\Phi|^2\Phi = 0 \tag{1.2}$$

where Φ is a complex-valued function, $\gamma = \pm 1$ denoting focusing or defocusing, and α_1, α_2 are real constants. We restrict ourselves to the focusing case in this project.

In the context of this project, we solve on an interval $x \in [-L_x/2, L_x/2], y \in [-L_y/2, L_y/2]$ subject to periodic boundary conditions. All grids are uniformly spaced. When we have the assumption of a narrow bandwidth in comparison to the carrier wave, NLS is an applicable model. This means that the majority of the energy in Fourier space is concentrated near the spectral peak which corresponds to what we call the carrier wave.

We chose to study the nonlinear Schrödinger equation because it has applications in a variety of fields such as optical pulses, Langmuir waves in plasmas, and the propagation of ocean swell over long distances [2]. NLS is also different from PDEs we have seen thus far in lectures due to the non-linearity and complex valued coefficient. Additionally, we had existing Python code for the two methods (split-step and machine learning) in 1D as a starting point. The 1D machine learning code comes from maziarraissi.github.io/deephpms/. The existing 1D split-step code comes from part Hannah's APMA 930 project and undergraduate thesis. Sections 3.1-3.4 significantly overlap this previous work but generalize to 2D which is new for both of us. Different examples are chosen from those considered in Hannah's previous work. The APMA 930 project focused on soliton-soliton interactions and preservation of the Hamiltonian in the split-step method compared to a different numerical method. Hannah's undergraduate work used the 1D split-step code on experimental wave tank data [5].

A noteworthy property of NLS is that it has infinitely many conserved quantities. The three "trivial" integrals of motion are mass, linear momentum, and the Hamiltonian. Mass or the number of particles is given by

$$\mathcal{N}(\Phi) = \int_{-L_y/2}^{L_y,2}\int_{-L_x/2}^{L_x/2} |\Phi|^2 dxdy. \tag{1.3}$$

Linear Momentum in the x and y directions are given by

$$\mathcal{P}_x(\Phi) = i \int_{-L_y/2}^{L_y,2} \int_{-L_x/2}^{L_x/2} \Phi\bar{\Phi}_x - \Phi_x\bar{\Phi} dxdy \tag{1.4}$$

and

$$\mathcal{P}_y(\Phi) = i \int_{-L_y/2}^{L_y,2} \int_{-L_x/2}^{L_x/2} \Phi\bar{\Phi}_y - \Phi_y\bar{\Phi} dxdy \tag{1.5}$$

where bars denotes complex conjugate. The Hamiltonian which can be interpreted as the energy is given by

$$\mathcal{H}(\Phi) = \int_{-L_y/2}^{L_y,2} \int_{-L_x/2}^{L_x/2} \alpha_1|\Phi_x|^2 + \alpha_2|\Phi_y|^2 - \frac{\gamma}{2}|\Phi|^4 dxdy. \tag{1.6}$$

Integrals are extracted from [1] and [3]. We note that the form of the Hamiltonian appears differently from paper to paper. One can write down relatively simple expressions for some "nontrivial" integrals of motion, two of which are seen in [1] but we will only consider these three primary ones for the sake of conciseness and clarity.

2. Exact solutions.

2.1. 1D examples. First, we consider hyperbolic secant solutions given by

$$\Phi(x,t) = \pm\sqrt{2\frac{\alpha}{\gamma}} b \,\mathrm{sech}\,(bx) e^{i\alpha b^2 t} \tag{2.1}$$

where b is a real constant. Periodic boundary conditions are assumed.

We also consider Jacobi elliptic cnoidal solutions given by

$$\Phi(x,t) = \pm\sqrt{2\frac{\alpha}{\gamma}}\, b\, k\, \mathrm{cn}(bx, \sqrt{k}) e^{i\alpha b^2(2k^2-1)t} \tag{2.2}$$

where b is a real constant and $k \in [0,1]$. Jacobi elliptic cnoidal functions have periods given in terms of

$$\kappa(k) = \int_0^{\pi/2} \frac{1}{\sqrt{1-k^2\sin^2(t)}} dt \tag{2.3}$$

so the periodicity requirement is naturally satisfied if the domain is properly chosen.

Additionally, we ran simulations for an exact solution which is a soliton-soliton interaction but decided to omit this from the paper as results were very similar to the other examples. Further, Hannah's APMA 930 project focused on soliton interactions and we wanted to reduce overlap. The solutions given above can be found in [3].

2.2. 2D examples. First, we consider the Stokes' wave solution given by

$$\Phi(x,y,t) = Ae^{ibx+icy+idt} \tag{2.4}$$

where A is a complex constant, b, c are real constants, and

$$d = \gamma|A|^2 - \alpha_1 b^2 - \alpha_2 c^2$$

as is seen in [3]. We observe that the periods may differ in the x and y dimensions.

Next, we consider the generalization of the 1D Jacobi elliptic cnoidal solutions above which are given by

$$\Phi(x,y,t) = \pm\sqrt{2\frac{\alpha_1+\alpha_2}{\gamma}}\, b\, c\, k\, \mathrm{cn}(bx+cy,\sqrt{k})e^{i\alpha_1 b^2(2k^2-1)t+i\alpha_2 c^2(2k^2-1)t} \tag{2.5}$$

where b is a real constant and $k \in [0,1]$.

Any 1D solution can also be used as a 2D solution that is constant along one spatial axis. We ran our simulations on a 1D Jacobi elliptic cnoidal solution using the 2D code but excluded the result from the paper as results were similar to other examples presented.

2.3. Dispersion Relation. The dispersion relation for the NLS is interesting in the sense that applying the analysis we used in lecture can be tedious and for some cases of the NLS it isn't applicable. In this section, we will focus on the NLS equation of the following form

$$i\Phi_t = \Phi_{xx} + \gamma^2|\Phi|^2\Phi. \tag{2.6}$$

The following discussion on the dispersion relation is from [4], which is referred to it as the effective dispersion relation. For the linear Schrodinger equation given by

$$i\Phi_t = \Phi_{xx},$$

we can use the analysis we have previously seen. We substitute $\Phi = \exp(i(kt-\omega x))$ into the equation and get

$$\begin{aligned} i(ik)\exp(i(kt-\omega x)) &= -(i\omega)^2\exp(i(kt-\omega x)) \\ \Rightarrow k &= \omega^2. \end{aligned}$$

The phase velocity is given by

$$\frac{k}{\omega} = \omega,$$

and since it depends on ω the linear Schrodinger equation is dispersive.

For the nonlinear case we first consider the weakly NLS equation, where γ in (2.6) is really small, i.e. $\gamma \ll 1$. We can expand a NLS wave as

$$\Phi(x,t) = \sum_{n=-\infty}^{\infty} a_{\omega(t)}(t)e^{-i\omega_n x}$$

where $\omega_n = \frac{2\pi n}{L}$ is the wave number. This can be written as

$$\Phi(x,t) = \sum_{n=-\infty}^{\infty} a_{\omega_n}(0)\exp[-i(\omega_n x - [\omega_n^2 - \gamma^2(2\|a_\kappa(0)\|_2^2 - |a_{\omega_n}(0)|^2)]t)] + \mathcal{O}(\gamma^2)$$

where

$$\|a_\kappa(0)\|_2^2 = \|a_\kappa(t)\|_2^2 \equiv \sum_{m=-\infty}^{\infty}|a_{\omega_m}(0)|^2 = \sum_{m=-\infty}^{\infty}|a_{\omega_m}(t)|^2. \tag{2.7}$$

Then the frequency for the mode with wave number ω_n is given by

$$k(\omega_n) \approx \omega_n^2 - \gamma^2 \left(2\|a_\kappa(0)\|_2^2 - |a_{\omega_n}(0)|^2\right).$$

If the mode amplitudes are small compared to (2.7) then we get the following approximate dispersion relation

$$k(\omega) \approx \omega^2 - 2\gamma^2\|a_\kappa(0)\|_2^2 = \omega^2 - \frac{2\gamma}{L}\int_{-L/2}^{L/2} |\Phi(x,0)|^2 dx$$

where

$$\frac{1}{L}\int_{-L/2}^{L/2} |\Phi(x,0)|^2 dx \equiv \frac{1}{L}\|\Phi(x,0)\|_2^2.$$

By the conserved quantity for mass, we can let t be an arbitrary time instead of $t = 0$.

This dispersion relation can be extended to other problems, such as the strongly nonlinear NLS. An exact solution of the NLS is given by the following plane wave

$$\Phi(x,t) = A\exp[-i(\alpha x - (\alpha^2 - A^2\gamma^2)t)]$$

where A is the amplitude of the initial wave. As the wave propagates in the spatial domain some of the nodes will grow exponentially and this is due to instability of the nodes. The number of nodes that experience this instability is $2N + 1$ where N is the largest integer such that

$$1 \le N \le \frac{\sqrt{2}|A|\gamma L}{2\pi}.$$

When $|A|\gamma L$ is large, then the dispersion relation is given by a parabola, just like what we have seen in the previous two cases.

Lastly, we can find the the dispersion relation for waves with a coherent structure, meaning that the initial condition is a set of overlapping sech-shaped functions. But the analysis of this different from what we introduced in this section. Additionally, some NLS problems might not have a quadratic or a well defined dispersion relation. Formulating the dispersion relation for these types of problems goes beyond the scope of this project but the details of it are in [4].

3. Split-Step Method. The split-step scheme is a psuedo-spectral method. This means we assume our discrete solution can be written in terms of chosen basis functions. We work with a Fourier basis in space and a Strang splitting method in time. The conservation of the number of particles, $\mathcal{N}$, is intrinsic to split-step methods for NLS due to our ability to solve exactly in the Fourier domain in space. Conservation of the Hamiltonian, $\mathcal{H}$, is not enforced.

3.1. Fourier Basis in Space. As aforementioned, we assume a Fourier basis in space. Given a spatial discretization with N grid points in the x-dimension and M grid points in the y-dimension, we assume that for a function f we have

$$f(x,y) = \sum_{\ell=-M/2}^{M/2-1} \sum_{k=-N/2}^{N/2-1} \hat{f}(k,\ell) e^{\frac{i2\pi kx}{L_x} + \frac{i2\pi \ell y}{L_y}} \tag{3.1}$$

where $f(x,y)$ has period L_x in x and period L_y in y, and

$$\hat{f}(k,\ell) = \frac{1}{L_x L_y}\int_{-L_y/2}^{L_y/2}\int_{-L_x/2}^{L_x/2} f(x,y) e^{-\frac{i2\pi kx}{L_x} - \frac{i2\pi \ell y}{L_y}} dxdy. \tag{3.2}$$

That is, we suppose functions can be represented by a finite number of Fourier modes where the number of modes kept is in accordance with the grid resolution. In the algorithm, the Fast Fourier Transform (FFT) is used as opposed to the Discrete Fourier Transform due to the computational advantage of being $\mathcal{O}(NM \cdot \text{ Log } (NM))$ as opposed to $\mathcal{O}(N^2 \cdot M^2)$.

3.2. Operator-Splitting in Time. The differential equation

$$\Phi_t = L(\Phi) + N(\Phi) \tag{3.3}$$

has a formal solution of

$$\Phi(x, y, t) = \Phi(x, y, 0)e^{(L+N)t} \tag{3.4}$$

where L and N and the linear and nonlinear operators respectively in a differential equation of form (3.3).

We can write 2D NLS in form (3.3), and see that

$$L(\Phi) = i\alpha_1\Phi_{xx} + i\alpha_2\Phi_{yy} \tag{3.5}$$

and

$$N(\Phi) = i\gamma|\Phi|^2\Phi. \tag{3.6}$$

The idea behind the Strang splitting technique is to approximate the time dependent part of the formal solution in such a way that the linear and nonlinear operators appear independently. When they appear independently, we can perform a sequence of solving the linear and nonlinear parts of the equation separately. Note that in contrast to working with numbers say a and b where we have $e^{(a+b)t} = e^{at}e^{bt}$, we are working with operators and this identity only holds in the operator sense when the operators commute.

The first approach is to do the naive thing and say

$$e^{(L+N)t} \approx e^{Lt}e^{Nt}. \tag{3.7}$$

What this means for our algorithm is that we can apply the routine for solving the linear equation followed by the routine for solving the nonlinear equation. It turns out that the naive approximation is first-order in t. In general, we can see that (3.7) is only accurate to the first order via Taylor series. Expanding as done in [9], we get

$$e^{t(L+N)} = I + (L + N)t + \frac{1}{2}(L^2 + LN + NL + N^2)t^2 + ... \tag{3.8}$$

and

$$e^{t(L)}e^{t(N)} = I + (L + N)t + \frac{1}{2}(L^2 + 2LN + N^2)t^2 + ... \tag{3.9}$$

which demonstrates that unless $LN = NL$ or in other words the operators commute, we are only first-order accurate in t.

Using the Taylor series expansions to inform the accuracy, we construct a second-order approximation given by

$$e^{(L+N)t} \approx e^{\frac{1}{2}Lt}e^{Nt}e^{\frac{1}{2}Lt}. \tag{3.10}$$

This is seen in [9]. What this means for our algorithm is that we can apply the routine for solving the linear equation with half the time step followed by the routine for solving the nonlinear equation and another half time step of the linear solver.

Similarly, we can construct higher-order splitting approximations by exploiting that

$$e^{Lt}e^{Nt} = e^{Zt} \tag{3.11}$$

where

$$\begin{aligned} Z = L + N + \frac{1}{2}[L,N] + \frac{1}{12}([L,L,N] + [N,N,L]) + \frac{1}{24}[L,N,N,L] \\ -\frac{1}{720}([L,L,L,L,N] + [N,N,N,N,L]) + \frac{1}{360}([L,N,N,N,L] \\ +[N,L,L,L,N]) + ... \end{aligned} \tag{3.12}$$

and

$$[L,N] = LN - NL \tag{3.13}$$

is the commutator of operators L and N [9]. The commutator is defined iteratively so that $[A,B,C] = [[A,B],C]$ for operators A, B, C and similarly for more than 3 operator inputs. Note that if L and N commute then $Z = L + N$ which is what we desire.

Although we can construct odd order approximations using (3.12), we are able to exploit symmetry for schemes of even order. Using (3.12), we can write

$$\begin{aligned} S_{2nd}(t) &= e^{\frac{1}{2}Lt}e^{Nt}e^{\frac{1}{2}Lt} \\ &= \exp\left[a_1 t + a_3 t^3 + \mathcal{O}(t^5) + ...\right] \end{aligned}$$

with $a_1 = L + N$ and $a_3 = \frac{1}{12}([N,N,L]) - \frac{1}{24}([L,L,N])$. Then, we assume that a fourth-order approximation can be constructed out of S_{2nd} using the same symmetry pattern seen between first-order and second-order. That is, we assume $S_{4th}(t) = S_{2nd}(b_1 Lt)S_{2nd}(b_0 Nt)S_{2nd}(b_1 Lt)$ for some constants b_0 and b_1. Then,

$$\begin{aligned} S_{4th}(t) &= S_{2nd}(b_1 Lt)S_{2nd}(b_0 Nt)S_{2nd}(b_1 Lt) \\ &= \exp\left[(b_0 + bx_1)a_1 t + (b_0^3 + 2b_1^3)a_3 t^3 + \mathcal{O}(t^5)\right] \end{aligned} \tag{3.14}$$

so we need $b_0 + 2b_1 = 1$ and $b_0^3 + 2b_1^3 = 0$ in order to obtain fourth-order accuracy in t. The nonlinear system of equations in b_0 and b_1 has a solution of $b_0 = \frac{-2^{1/3}}{2-2^{1/3}}$, $b_1 = \frac{1}{2-2^{1/3}}$. Mathematica was used to solve the nonlinear system of equations.

Similarly, we can construct a sixth-order scheme out of $S_{4th}(t)$. We have

$$\begin{aligned} S_{6th}(t) &= S_{4th}(c_1 Lt)S_{4th}(c_0 Nt)S_{4th}(c_1 Lt) \\ &= \exp\left[(c_0 + 2c_1)a_1 t + (c_0^5 + 2c_1^5)a_5 t^5 + \mathcal{O}(t^7)\right] \end{aligned} \tag{3.15}$$

so we need $c_0 + 2c_1 = 1$ and $c_0^5 + 2c_1^5 = 0$ in order to obtain sixth-order accuracy in t. The nonlinear system of equations in c_0 and c_1 has a solution of $c_0 = \frac{-2^{1/5}}{2-2^{1/5}}$, $c_1 = \frac{1}{2-2^{1/5}}$ which was also found via Mathematica. In our code, we use the second and sixth order schemes.

3.3. Solving the Linear and Nonlinear Parts. Now that we have a splitting technique which enables us to solve the linear and nonlinear parts of NLS separately, we need to know how to solve the two simplified equations.

The linear equation is given by

$$\Phi_t = L(\Phi) = i\alpha_1 \Phi_{xx} + i\alpha_2 \Phi_{yy} \tag{3.16}$$

which we can solve exactly in the Fourier domain. Evolving by Δt in time, we get the exact solution of

$$\hat{\Phi}_{k,\ell}(t + \Delta t) = e^{-i(k^2\alpha_1 + \ell^2\alpha_2)\Delta t} \hat{\Phi}_{k,\ell}(t) \tag{3.17}$$

where $\hat{\Phi}_{k,\ell}(t)$ denotes the Fourier coefficient, corresponding to wavenumber k in the x-dimension and wavenumber ℓ in the y-dimension, of $\Phi(x, y, t)$.

The nonlinear equation is given by

$$\Phi_t = N(\Phi) = i\gamma |\Phi|^2 \Phi \tag{3.18}$$

which is a nonlinear ODE. Luckily, we can (via Mathematica or by hand) solve the ODE exactly in physical space. Evolving by Δt in time, we get the exact solution of

$$\Phi(x, y, t + \Delta t) = e^{i\gamma |\Phi|^2 \Delta t} \Phi(x, y, t) \tag{3.19}$$

so there is no error introduced here aside from numerical round-off. For generalizations of NLS where there may not be an exact solution to the ODE, a numerical ODE solver such as a Runge-Kutta method may be used. Of course, this introduces error which depends on the order of the scheme employed.

3.4. Comments. The split-step methods are composed of the linear and nonlinear "exact" solutions which are only performing a phase rotation [1]. This means that the number of particles, $\mathcal{N}$, is well preserved by the method. Additionally, split-step methods preserve the dispersion relation for NLS as noted in [1]. This means that the solution error should not depend on speed on propagation and so split-step methods are applicable to varied dynamics. Note that group velocity and phase velocity can be obtained from the dispersion relation so the preservation of dispersion relation is needed to ensure accurate behaviour is captured by the numerical approximation.

For NLS, split-step methods must satisfy the time step restriction of

$$\Delta t \leq \frac{\Delta x^2}{\pi} \tag{3.20}$$

in order to guarantee stability. Stability analysis and derivation of this condition can be found in [7] and [8]. However, we often observe numerically that a larger time step may be acceptable but when the exact solution is not known it can be very dangerous to disobey the stability restriction.

Additionally, it should be noted that results are not grid-convergent in the typical sense. In finite difference methods for example, we would have some power of Δx multiplying the error estimate and see a clear rate of decay corresponding to that power for smooth solutions. For the psuedo-spectral split-step method, we usually observe improvements in the errors as we resolve the grid because we are able to capture more of the Fourier modes. Suitable grid resolution can be determined by the energy in higher harmonics as seen by examining the solution of interest in Fourier space.

dt	SS2 error rate (L^2)	SS2 error rate (H^1)
10^{-2}	NA	NA
5.99×10^{-3}	1.994	1.994
3.59×10^{-3}	1.997	1.997
2.15×10^{-3}	1.999	1.999
1.29×10^{-3}	2.001	2.001
dt	SS6 error rate (L^2)	SS6 error rate (H^1)
10^{-2}	NA	NA
5.99×10^{-3}	4.145	0.502
3.59×10^{-3}	-0.403	-0.102
2.15×10^{-3}	-0.021	-0.005

Table 3.1: Error convergence rates in computed using SS2 (Top) and SS6 (Bottom) versus exact hyperbolic secant solution at final time using $n = 2^7$ for the first few successive time step sizes.

3.5. Error Analysis in 1D. The discussion in the previous subsection tells us we can expect the split step method to do a good job of preserving the number of particles, $\mathcal{N}$, but we hold no expectations on its ability to preserve the other integrals of motion . In time, we expect to see L^2 grid norm convergence which has the same order as the splitting provided sufficient regularity and grid resolution. We may take derivatives in Fourier space simply by scaling using the coefficients. Therefore, we also track the H^1 grid norm errors to see if they are converging at the same rate as the L^2 errors.

First, we consider a hyperbolic secant solution (2.1) given by

$$\Phi(x,t) = \sqrt{2\frac{\alpha}{\gamma}}2\,\mathrm{sech}\,(2x)e^{i\alpha 4t} \tag{3.21}$$

with $\alpha = 1$ and $\gamma = 1$. We impose the spatial period $L = 20$. We solve up to time $t_f = 1$ on an grid with $n = 2^7$ spatial points and again with $n = 2^{10}$ and vary dt in order to examine convergence. The smallest time step we may take with ensured stability is $dt = \frac{(20/n)^2}{\pi}$ but in practice we observe that a bit larger is also alright and is in fact necessary to see the convergence. We take 10 different dt values ranging from 10^{-2} to 10^{-4} for both n values.

Figures 3.1 ($n = 2^7$) and 3.2 ($n = 2^{10}$) show plots of the errors in the hyperbolic secant solution and conserved quantities for the different dt values. We notice that the second-order split step method (SS2) L^2 and H^1 errors both exhibit second-order numerical convergence. However, the sixth-order split step method (SS6) errors only decay initially and then level off. The value at which the errors level off depends on the grid resolution. This is an artifact of the Fourier truncation because it sets the minimum error in the computed solution. Error in the conservation of the Hamiltonian appears to be comparable to the solution error and the other two integrals of motion are preserved more or less to numerical precision. Tables 3.1 ($n = 2^7$) and 3.2 ($n = 2^{10}$) show the initial convergence rates for the solution error.

Next, we try out a Jacobi elliptic cnoidal solution (2.2) given as

$$\Phi(x,t) = \sqrt{2\frac{\alpha}{\gamma}}\,0.8^2\,\mathrm{cn}(bx, 0.8)e^{i\alpha(2(0.8)^2-1)t} \tag{3.22}$$

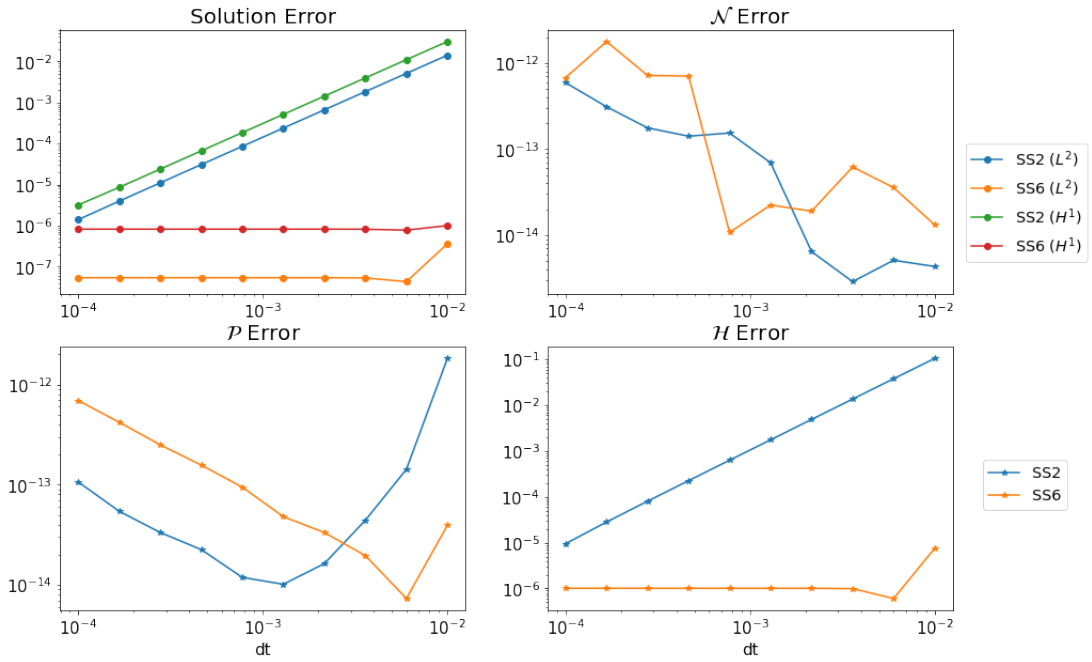

Fig. 3.1: Errors in computed versus exact hyperbolic secant solution at final time using $n = 2^7$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times.

with $\alpha = 0.5$ and $\gamma = 1$. We have spatial period $4\,\kappa(0.8) \approx 9$ using (2.3). Note that unlike the hyperbolic secant example we cannot adjust this period and expect the method to succeed. We solve up to time $t_f = \pi/2$ on a grid with $n = 2^5$ spatial points and again with $n = 2^9$ and vary dt in order to examine convergence. The smallest time step we may take with ensured stability is $dt \approx \frac{(9/n)^2}{\pi}$ but again in practice we observe that a bit larger is also fine for this example. We take 10 different dt values ranging from 10^{-1} to 10^{-3} for both n values.

Figures 3.3 ($n = 2^5$) and 3.4 ($n = 2^9$) show plots of the errors in the Jacobi elliptic cnoidal solution and conserved quantities for the different dt values. The second-order split step method (SS2) L^2 and H^1 errors both exhibit second-order numerical convergence. Again, the sixth-order split step method (SS6) errors only decay initially and then level off. There is however sixth-order numerical convergence for about the first half of time step sizes when using $n = 2^9$. Error in the conservation of the Hamiltonian again appears to be comparable to the solution error and the other two integrals of motion are preserved more or less to numerical precision. Tables 3.3 ($n = 2^7$) and 3.4 ($n = 2^9$) show the initial convergence rates for the solution error.

3.6. Error Analysis in 2D. The first 2D example we run is the Stokes's solution (3.22) of

$$\Phi(x, y, t) = e^{ix+iy+idt} \tag{3.23}$$

dt	SS2 error rate (L^2)	SS2 error rate (H^1)
10^{-2}	NA	NA
5.99×10^{-3}	1.994	1.994
3.59×10^{-3}	1.997	1.997
2.15×10^{-3}	1.999	1.999
1.29×10^{-3}	2.001	2.001
dt	SS6 error rate (L^2)	SS6 error rate (H^1)
10^{-2}	NA	NA
5.99×10^{-3}	5.725	2.272
3.59×10^{-3}	3.736	7.469
2.15×10^{-3}	0.173	-1.864

Table 3.2: Error convergence rates in computed using SS2 (Top) and SS6 (Bottom) versus exact hyperbolic secant solution at final time using $n = 2^{10}$ for the first few successive time step sizes.

dt	SS2 error rate (L^2)	SS2 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	1.977	1.978
3.59×10^{-2}	2.014	2.014
2.15×10^{-2}	2.002	2.002
dt	SS6 error rate (L^2)	SS6 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	6.126	6.403
3.59×10^{-2}	1.636	0.589
2.15×10^{-2}	-0.071	-0.069

Table 3.3: Error convergence rates in computed using SS2 (Top) and SS6 (Bottom) versus exact Jacobi elliptic cnoidal solution at final time using $n = 2^5$ for the first few successive time step sizes.

dt	SS2 error rate (L^2)	SS2 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	2.146	2.149
3.59×10^{-2}	1.965	1.966
2.15×10^{-2}	2.014	2.014
dt	SS6 error rate (L^2)	SS6 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	6.769	7.006
3.59×10^{-2}	5.899	5.964
2.15×10^{-2}	6.031	6.032

Table 3.4: Error convergence rates in computed using SS2 (Top) and SS6 (Bottom) versus exact Jacobi elliptic cnoidal solution at final time using $n = 2^9$ for the first few successive time step sizes.

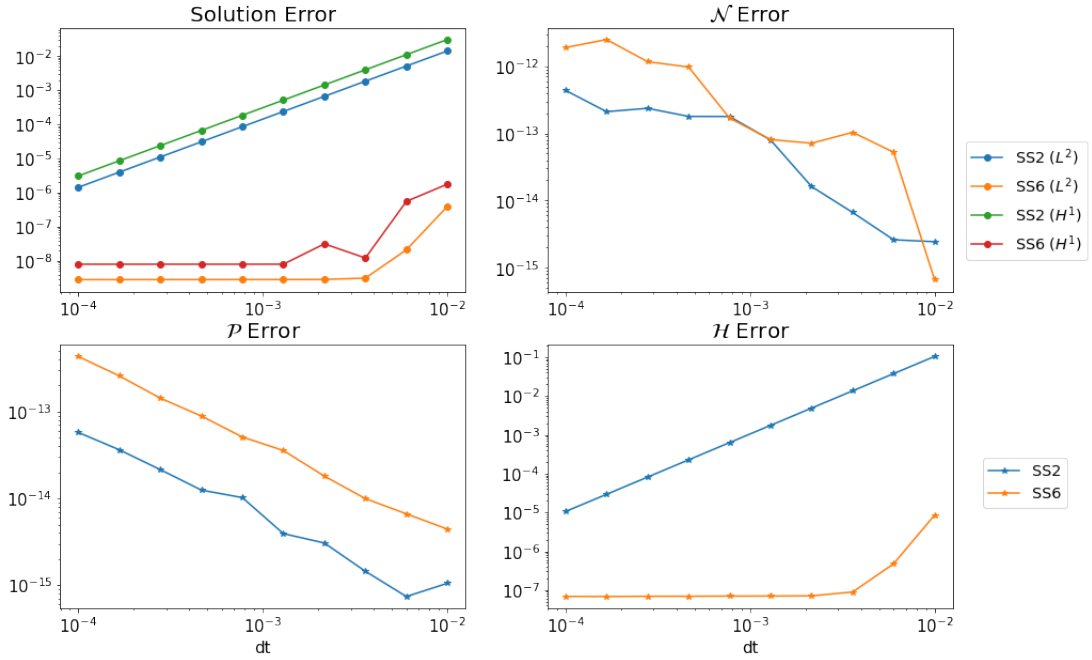

Fig. 3.2: Errors in computed versus exact hyperbolic secant solution at final time using $n = 2^{10}$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times.

where $d = \gamma - 4\alpha_1 - \alpha_2$and we take $\alpha_1 = \alpha_2 = 1, \gamma = 1$. This gives the periods $L_x = 2\pi$ and $L_y = 2\pi$. We solve up to time $t_f = \pi/2$ on an $n \times n$ grid with $n = 2^2$ and again with $n = 2^6$ and vary dt in order to examine convergence. The smallest time step we may take is $dt = \frac{(2\pi/n)^2}{\pi}$ so as to ensure stability but as in 1D we observe that a bit larger is also alright here. We take 10 different dt values ranging from 10^{-1} to 10^{-3} for both n values. Figure 3.5 shows the initial and final solution profiles for context.

Figures 3.6 ($n = 2^2$) and 3.7 ($n = 2^6$) show plots of the errors in the Stokes solution and conserved quantities for the different dt values. Both the second-order split step method (SS2) and the sixth-order split step method (SS6) errors are immediately at more or less machine precision for the solution and for conserved quantities. This makes sense because the truncation of Fourier modes is not an approximation for this example. Tables not included as there are no convergence rates.

The second example we run is the 2D generalization of the Jacobi elliptic cnoidal solution (2.5) of

$$\Phi(x, y, t) = \sqrt{2\frac{\alpha_1 + \alpha_2}{\gamma}}\,(0.8)^2\,\mathrm{cn}(x + y, 0.8)e^{i\alpha_1(2(0.8)^2-1)t+i\alpha_2(2(0.8)^2-1)t} \tag{3.24}$$

where we take $\alpha_1 = \alpha_2 = 0.5, \gamma = 1$. This gives the periods $L_x = L_y = 8\,\kappa(0.8) \approx 18$ using (2.3) to get 2 peaks and troughs in the profile. We solve up to time $t_f = \pi/2$ on an $n \times n$ grid with $n = 2^5$

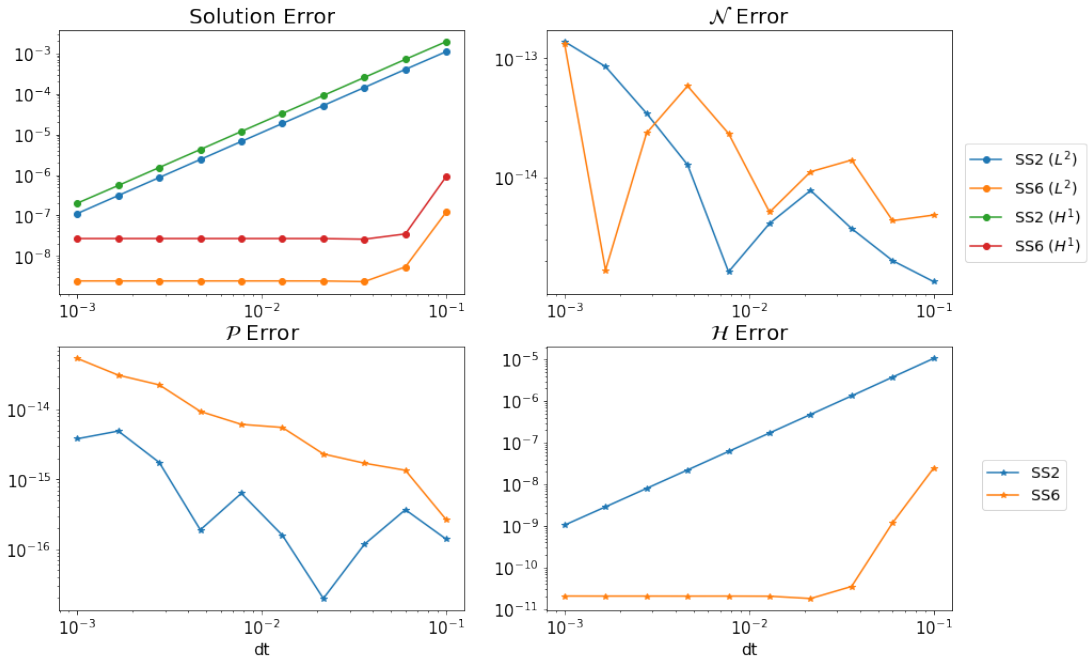

Fig. 3.3: Errors in computed versus exact Jacobi elliptic cnoidal solution at final time using $n = 2^5$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times.

and again with $n = 2^7$ and vary dt in order to examine convergence. The smallest time step we may take is $dt \approx \frac{(18/n)^2}{\pi}$ so as to ensure stability but we can get away with larger in this case. We take 10 different dt values ranging from 10^{-1} to 10^{-3} for both n values.

Figures 3.8 ($n = 2^5$) and 3.9 ($n = 2^7$) show plots of the errors in the hyperbolic secant solution and conserved quantities for the different dt values. We notice that the second-order split step method (SS2) L^2 and H^1 errors both exhibit second-order numerical convergence at least initially. Likewise, for $n = 2^7$ the sixth-order split step method (SS6) errors decay initially and then level off but are already leveled off for the largest time step using $n = 2^5$. Error in the conservation of the Hamiltonian appears to be comparable to the solution error and the other two integrals of motion are preserved more or less to numerical precision. Tables 3.5 ($n = 2^5$) and 3.6 ($n = 2^7$) show the initial convergence rates for the solution error.

4. Physics Informed Neural Networks. We give an overview of the physics informed neural network approach for 1D NLS shown in [6] and extend the framework for 2D NLS. In general, consider a PDE of the form

$$u_t + \mathcal{N}u = 0$$

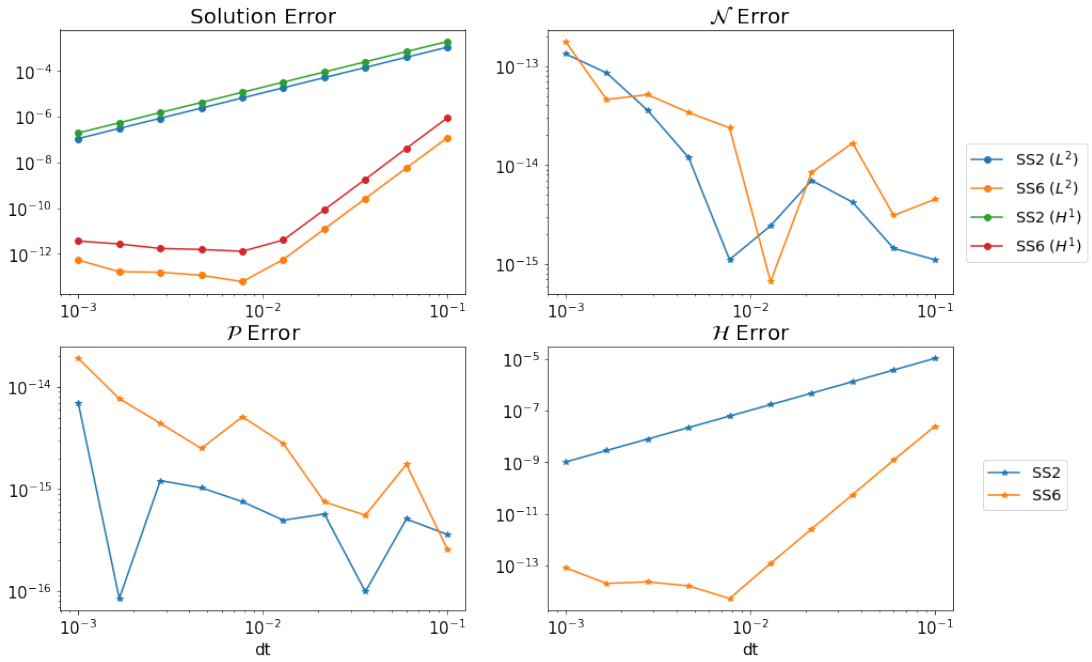

Fig. 3.4: Errors in computed versus exact Jacobi elliptic cnoidal solution at final time using $n = 2^9$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times.

where $\mathcal{N}$ is a potentially nonlinear operator acting on u. Let $f(x,t)$ be the LHS of the PDE above, so we get

$$f := u_t + \mathcal{N}u \tag{4.1}$$

The parameters of the neural network are learned by minimizing the mean squared error (MSE) loss. The MSE for u is

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} \left|u(t_u^i, x_u^i) - u^i\right|^2$$

where N_u is the number of training data for u. This gives the loss of the initial and boundary data. The MSE for f is

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left|f(t_f^i, x_f^i)\right|^2$$

where N_f is the number of collocation points. This MSE enforces the structure of (4.1). All together the total MSE for system is given by

$$MSE = MSE_u + MSE_f.$$

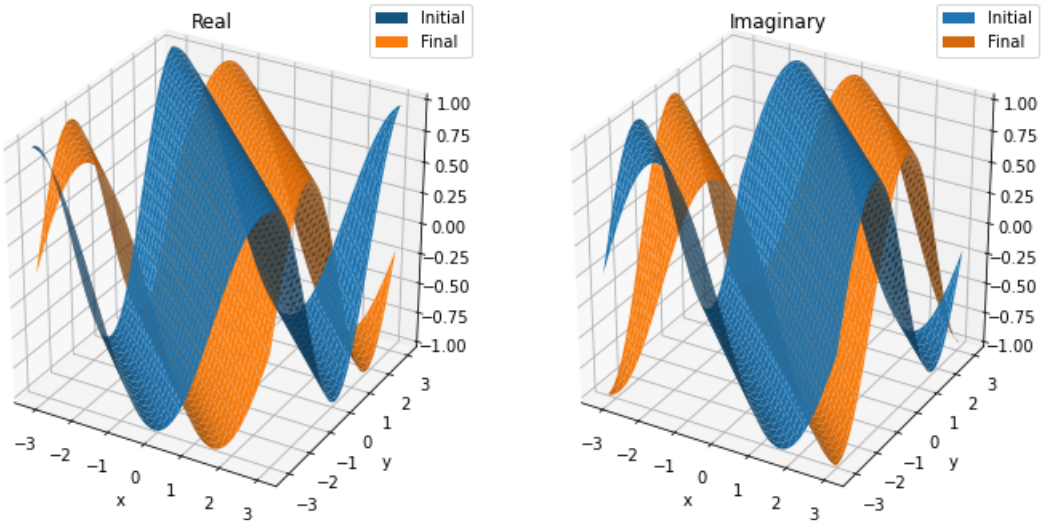

Fig. 3.5: We plot the initial and final profiles of the exact Stokes solution separated into the real part (Left) and imaginary part (Right).

dt	SS2 error rate (L^2)	SS2 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	2.232	2.210
3.59×10^{-2}	2.197	2.065
2.15×10^{-2}	2.083	1.494
dt	SS6 error rate (L^2)	SS6 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	-0.072	-0.003
3.59×10^{-2}	-0.083	-0.003

Table 3.5: Error convergence rates in computed using SS2. (Top) and SS6 (Bottom) versus exact 2D Jacobi elliptic cnoidal solution at final time using $n = 2^5$ for the first few successive time step sizes.

The errors we are prescribing to be minimized can be customized depending on the nature of the problem or application. For example, we may choose to additionally enforce preservation of $\mathcal{H}$ or other integrals of motion. The machine learning approach is very flexible in this way.

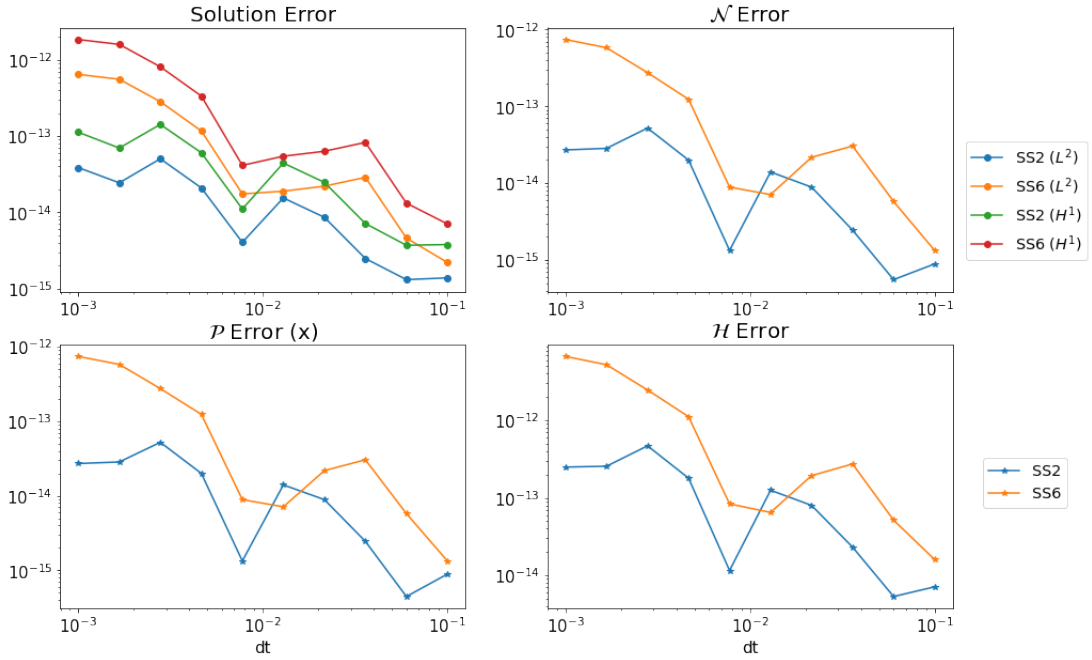

Fig. 3.6: Errors in computed versus exact Stokes solution at final time using $n = 2^2$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times. Momentum in the y direction omitted as it is similar to the x direction.

4.1. 1D NLS. Now we apply this idea to the 1D NLS and derive the MSE. Consider the 1D NLS initial value problem given by

$$
\begin{aligned}
&i\Phi_t + \alpha\Phi_{xx} + \gamma|\Phi|^2\Phi = 0,\ x \in [-L/2, L/2],\ t \in [0, T] \\
&\Phi(0, x) = \Phi_0(x) \\
&\Phi(t, -L/2) = \Phi(t, L/2) \\
&\Phi_x(t, -L/2) = \Phi_x(L/2).
\end{aligned}
$$

Define $f(x, t)$ to be the LHS of the PDE,

$$
f = i\Phi_t + \alpha\Phi_{xx} + \gamma|\Phi|^2\Phi. \tag{4.2}
$$

Given that we have boundary and initial data, we will need to define two MSEs. One for the boundary data, and the other for the initial data. So altogether we have three MSEs which are

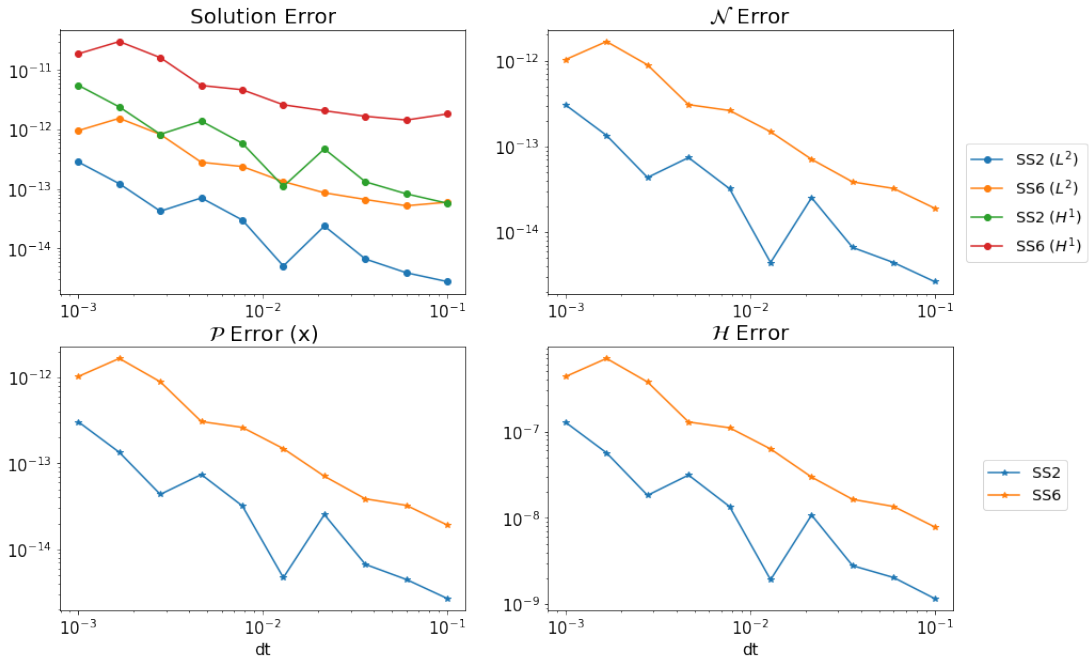

Fig. 3.7: Errors in computed versus exact Stokes solution at final time using $n = 2^6$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times. Momentum in the y direction omitted as it is similar to the x direction.

given by

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left|\Phi(0,x_0^i) - \Phi_0^i\right|^2$$

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(\left|\Phi^i(t_b^i,-L/2) - \Phi^i(t_b^i,L/2)\right|^2 + \left|\Phi_x^i(t_b^i,-L/2) - \Phi_x^i(t_b^i,L/2)\right|^2\right)$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f(t_f^i,x_f^i)\right|^2.$$

Thus, the total MSE for the 1D NLS is

$$MSE = MSE_0 + MSE_b + MSE_f. \tag{4.3}$$

Similar to the general case at the beginning of this section, MSE_f enforces the structure of (4.2), while MSE_0 and MSE_b gives the loss of the initial and boundary data respectively.

Now we need to address how to set up the neural network for training. We can write 1D NLS

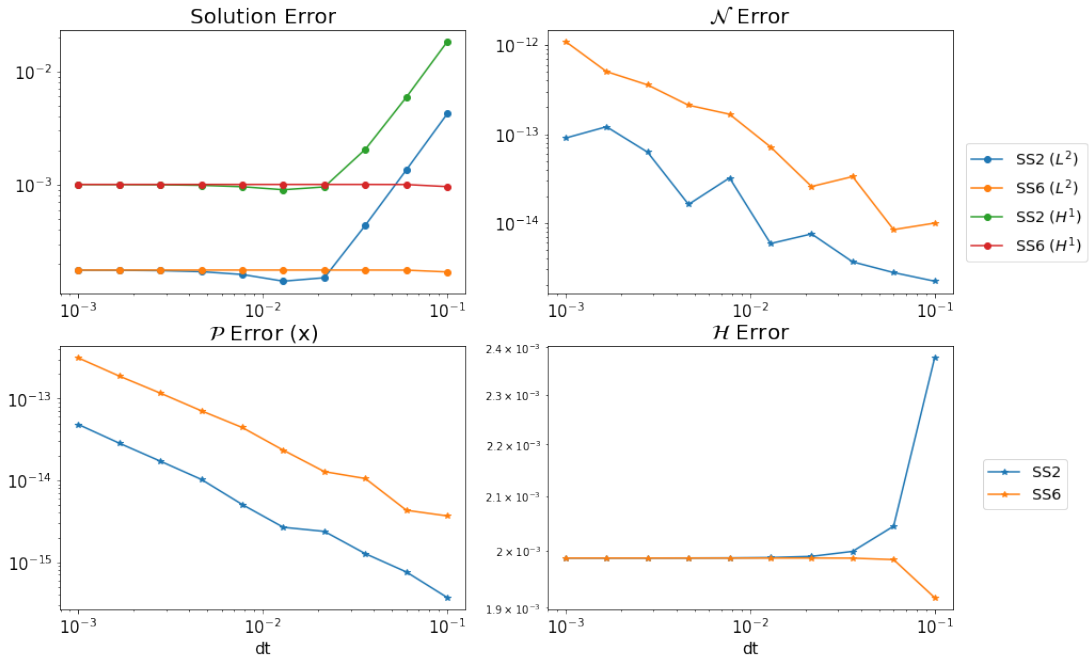

Fig. 3.8: Errors in computed versus exact 2D Jacobi elliptic cnoidal solution at final time using $n = 2^5$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times. Momentum in the y direction omitted as it is similar to the x direction.

dt	SS2 error rate (L^2)	SS2 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	2.135	2.135
3.59×10^{-2}	1.961	1.961
2.15×10^{-2}	2.013	2.013
dt	SS6 error rate (L^2)	SS6 error rate (H^1)
10^{-1}	NA	NA
5.99×10^{-2}	8.694	8.707
3.59×10^{-2}	5.727	5.626
2.15×10^{-2}	6.292	6.421

Table 3.6: Error convergence rates in computed using SS2. (Top) and SS6 (Bottom) versus exact 2D Jacobi elliptic cnoidal solution at final time using $n = 2^7$ for the first few successive time step sizes.

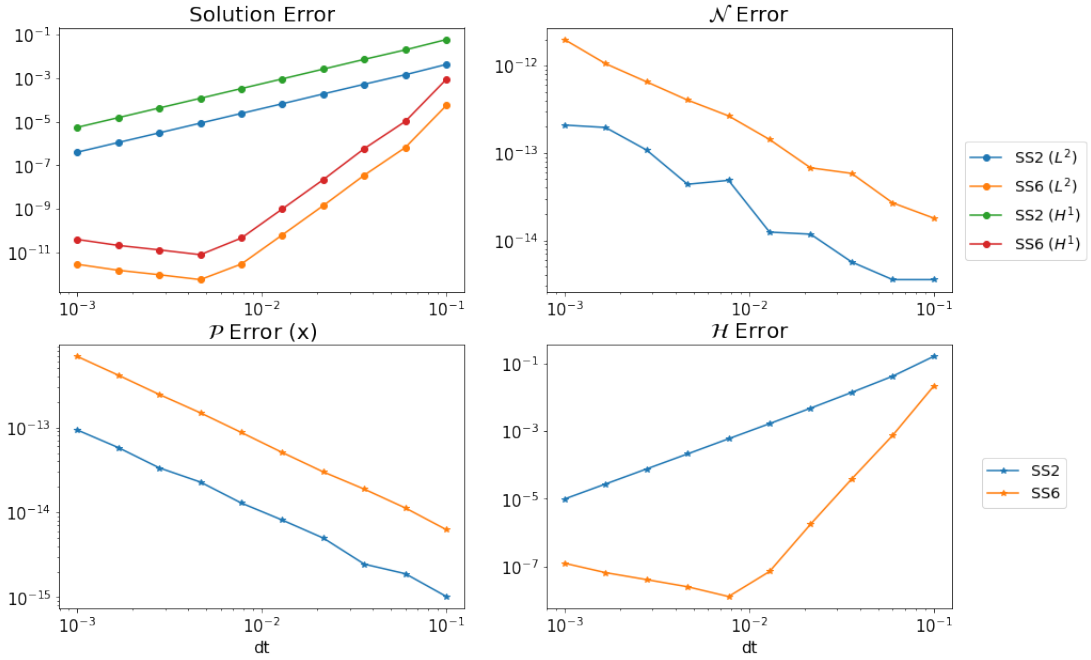

Fig. 3.9: Errors in computed versus exact 2D Jacobi elliptic cnoidal solution at final time using $n = 2^7$. Top left shows both L^2 and H^1 grid norms and the other plots show difference in conserved quantities at initial and final times. Momentum in the y direction omitted as it is similar to the x direction.

as a system of PDEs

$$u_t = -\alpha v_{xx} - (u^2 + v^2)v = \mathcal{N}_1(u, v, u_x, v_x, u_{xx}, v_{xx})$$
$$v_t = -\alpha u_{xx} + (u^2 + v^2)u = \mathcal{N}_2(u, v, u_x, v_x, u_{xx}, v_{xx})$$

where $|\Phi| = \sqrt{u^2 + v^2}$. The goal is to learn $\mathcal{N}_1$ and $\mathcal{N}_2$. To do this, we'll need to train 4 networks to learn u, v, $\mathcal{N}_1$, and $\mathcal{N}_2$. The reason is because the networks for u and v are for learning the solution and $\mathcal{N}_1$ and $\mathcal{N}_2$ are for learning the structure the of PDE. The networks for training u and v are 5 layer deep neural networks with 50 neurons per hidden layer. The networks for $\mathcal{N}_1$ and $\mathcal{N}_2$ are 2 layer neural networks with 100 neurons per hidden layer. The four networks are learned by minimizing (4.3).

To implement this we used code already written by Raissi in Python, which was provided in [6]. We will also note that after training the data we did not use the trained neural network for other initial conditions. So we re-trained the neural network for different initial conditions. This was because we didn't figure out how to use the trained neural network with different initial conditions without getting large errors, i.e. about $\mathcal{O}(10^{-1})$. Also, we specify the period in the algorithm so we would need to use a new example with the same period even if we had performed this test. Note that we are learning the model and so we could not expect to be able to change α or γ in a new

data set and use the previously trained network as a model. This is a limitation of the machine learning approach.

4.2. 2D NLS. For 2D, we follow the same process as we did for the 1D NLS. Consider the following 2D NLS initial value problem

$$
\begin{aligned}
&i\Phi_t + \alpha_1\Phi_{xx} + \alpha_2\Phi_{yy} + \gamma|\Phi|^2\Phi = 0, x \in [-L_x/2, L_x/2], y \in [-L_y/2, L_y/2], t > 0\\
&\Phi(0, x, y) = \Phi_0(x, y)\\
&\Phi(t, -L_x/2, y) = \Phi(t, L_x/2, y)\\
&\Phi_x(t, -L_x/2, y) = \Phi_x(t, L_x/2, y)\\
&\Phi(t, x, -L_y/2) = \Phi(t, x, L_y/2)\\
&\Phi_y(t, x, -L_y/2) = \Phi_y(t, x, L_y/2)
\end{aligned}
$$

and let

$$
f = i\Phi_t + \alpha_1\Phi_{xx} + \alpha_2\Phi_{yy} + \gamma|\Phi|^2\Phi \tag{4.4}
$$

The mean squared errors are given by

$$
\begin{aligned}
MSE_0 =& \frac{1}{N_0}\sum_{i=1}^{N_0} |\Phi(0, x_0^i, y_0^i) - \Phi_0^i|^2\\
MSE_{b_x} =& \frac{1}{N_{b_x}}\sum_{i=1}^{N_{b_x}} \left(\left|\Phi^i\left(t_{b_x}^i, -L_x/2, y_{b_x}^i\right) - \Phi^i\left(t_{b_x}^i, L_x/2, y_{b_x}^i\right)\right|^2 + \left|\Phi_x^i\left(t_{b_x}^i, -L_x/2, y_{b_x}^i\right) - \Phi_x^i\left(t_{b_x}^i, L_x/2, y_{b_x}^i\right)\right|^2\right)\\
MSE_{b_y} =& \frac{1}{N_{b_y}}\sum_{i=1}^{N_{b_y}} \left(\left|\Phi^i\left(t_{b_y}^i, x_{b_y}^i, -L_y/2\right) - \Phi^i\left(t_{b_y}^i, x_b^i, L_y/2\right)\right|^2 + \left|\Phi_y^i\left(t_{b_y}^i, x_{b_y}^i, -L_y/2\right) - \Phi_y^i\left(t_{b_y}^i, x_{b_y}^i, L_y/2\right)\right|^2\right)\\
MSE_f =& \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i, y_f^i)|^2
\end{aligned}
$$

Hence, the total MSE is

$$
MSE = MSE_0 + MSE_{b_x} + MSE_{b_y} + MSE_f. \tag{4.5}
$$

We can express the nonlinear parts as the following system of PDEs

$$
\begin{aligned}
u_t =& -\alpha_1 v_{xx} - \alpha_2 v_{yy} - (u^2 + v^2)v = \mathcal{N}_1(u, v, u_x, v_x, u_y, v_y, u_{xx}, v_{xx}, u_{yy}, v_{yy})\\
v_t =& -\alpha_1 v_{xx} - \alpha_2 v_{yy} + (u^2 + v^2)u = \mathcal{N}_2(u, v, u_x, v_x, u_y, v_y, u_{xx}, v_{xx}, u_{yy}, v_{yy})
\end{aligned}
$$

Just like in the 1D case, we want to learn $\mathcal{N}_1$ and $\mathcal{N}_2$ by training 4 networks to learn u, v, $\mathcal{N}_1$ and $\mathcal{N}_2$.

For the 2D code, we made changes to the code for the 1D NLS from [6]. Unfortunately, our code for the 2D case did not work as expected. The code ran without compilation issues but the loss errors were too high. There are likely indexing issues causing the problem. We would also like to note that for the 2D case the neural network architecture may need to be adjusted. We increased the number of nodes per hidden layer in attempt to account for higher data dimensionality. We still provide the draft code for training the 2D NLS and will perhaps revisit this in the future.

# Training points	Φ error	$\mathcal{N}$ error	$\mathcal{P}$ error	$\mathcal{H}$ error
5000	3.0290×10^{-2}	3.8280×10^{-2}	5.5650×10^{-2}	3.2160
10000	4.0282×10^{-2}	1.12227×10^{-3}	1.6316×10^{-10}	5.2359993×10^{-1}

Table 4.1: Errors in the solution and conserved quantities for the hyperbolic secant initial condition.

4.3. Error Analysis in 1D. In this section we will implement the machine learning code from [6] for the 1D NLS for two different examples. For our "exact solution", we used the split-step method with 512 spatial points but the domain length will vary slightly between the examples. We produced data up to $t_f = \frac{\pi}{2}$ with 501 time values but the time step is smaller than $t_f/501$ to ensure stability and a high level of accuracy. From this set, we randomly selected 10000 training points and 20000 points for MSE_f. We also considered training the neural network with 5000 training data and 10000 points for MSE_f. For MSE_0 and MSE_b we will use all of the data points from the exact solution, but we can adjust this to use fewer points.

We report the error of the predicted solution and the conserved quantities. But as previously mentioned, the conservation of the Hamiltonian, $\mathcal{H}$, is not enforced. We hold no expectations on its ability to preserve conserved quantities. For solution error we do not report the H^1 grid norm errors as we did for the split-step because we wanted to limit the code and table entries. The H^1 errors would be expected to have a similar relationship to the L^2 errors as they did for the split-step method. Raissi [6] does not monitor conserved quantities so this is an addition we made to the code.

First, we consider the hyperbolic secant initial data given by

$$\Phi(x, t = 0) = 2\,\mathrm{sech}\,(x) \tag{4.6}$$

with $\alpha = 0.5$ and $\gamma = 1$. We impose the spatial period $L = 10$. We solve up to time $t_f = \pi/2$. We use the sixth-order split-step code to construct a high resolution data set with 501 different time values along the run which we feed into the machine learning algorithm as our "exact solution". Note this is not an exact solution of NLS so we do not have preservation of $\mathcal{H}$. We take $n = 512$ and $dt = \frac{0.5}{\pi}(10/n)^2$.

Figure 4.1 shows contour plots of the "exact" dynamics and the resulting learned dynamics. In the eyeball norm, there is no apparent difference. Table 4.1 shows the corresponding errors for the two different numbers of training points. To obtain the conserved quantity error, we take the absolute value of difference between the starting and ending values. We observed that the error for Φ is slightly lower when we use 5000 training points which could be indicative of over-fitting or simply an artifact of the randomness in selecting training points. However, the errors for the conserved quantities do decrease as we use more training points. There is no clear rate even with the inclusion of more data points. The split-step method was able to preserve $\mathcal{N}$ and $\mathcal{P}$ almost to machine precision but the neural network approach does not have this feature.

Next, we try out the in-exact Jacobi elliptic cnoidal initial data

$$\Phi(x, t = 0) = \sqrt{\frac{\alpha}{\gamma}}\, 0.8^2\, \mathrm{cn}(bx, 0.8) \tag{4.7}$$

with $\alpha = 1$ and $\gamma = 1$. We have spatial period $4\,\kappa(0.8) \approx 9$ using (2.3). We solve up to time $t_f = \pi/2$. In the same manner, we use the sixth-order split-step code to construct a high resolution

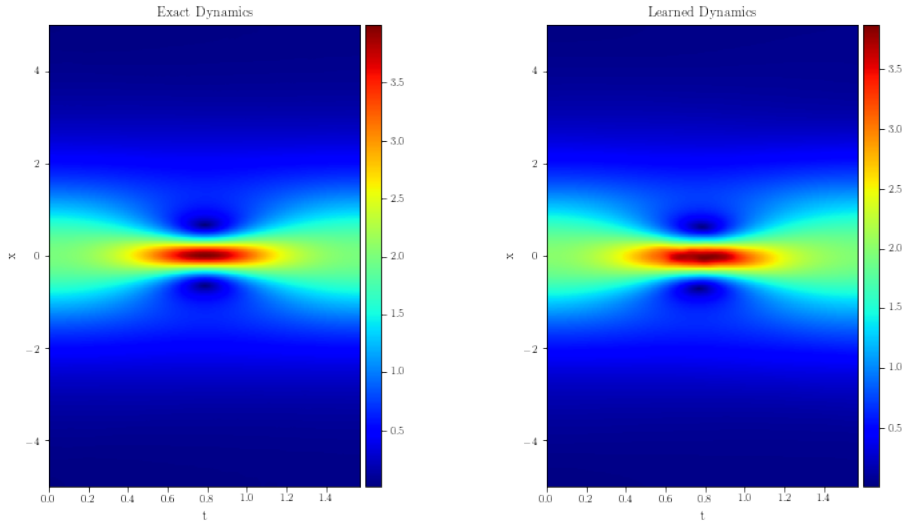

Fig. 4.1: Modulus of the the hyperbolic secant solution using SS6 data (Left) and the learned solution (Right).

# Training points	Φ error	$\mathcal{N}$ error	$\mathcal{P}$ error	$\mathcal{H}$ error
5000	3.0072×10^{-1}	6.9225×10^{-2}	2.5090×10^{-1}	1.3704
10000	2.4513×10^{-3}	2.0176×10^{-3}	7.5241×10^{-4}	1.2992

Table 4.2: Errors in the solution and conserved quantities for the in-exact Jacobi cnoidal initial condition.

data set with 501 different time values along the run which we feed into the machine learning algorithm as our "exact solution". We take $n = 512$ and $dt = \frac{0.5}{\pi}(L/n)^2$. Again, note this is not an exact solution of NLS so we do not have preservation of $\mathcal{H}$.

Figure 4.2 shows contour plots of the "exact" dynamics and the resulting learned dynamics. These plots demonstrate the machine learning algorithm's success at picking up on the model from the data even for a solution which is not exact. Table 4.2 shows the corresponding errors for the two different numbers of training points. As before, we obtain the conserved quantity error by taking the absolute value of difference between the starting and ending values. Here, we observe a significant decrease in all errors when we take more training points. Recall, the Hamiltonian is not conserved in this example. By taking a larger selection of training point numbers we do not see any clear rates so we only include two rows in the table.

We also tried out an exact Jacobi elliptic cnoidal solution but excluded the results from the report as they were very similar to those of the in-exact one above. The only difference is that we do have preservation of the Hamiltonian for the exact solution and the machine learning method captures this to a similar order of accuracy as the solution itself.

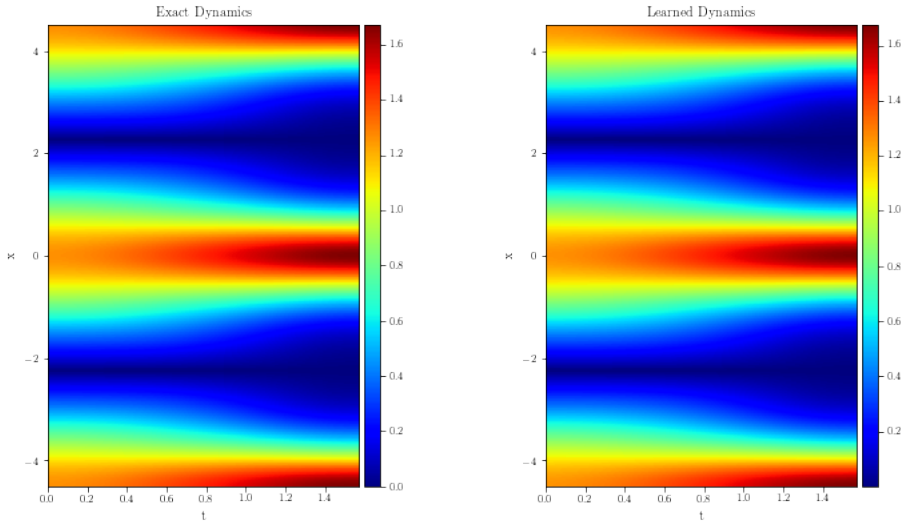

Fig. 4.2: Modulus of the the in-exact Jacobi elliptic cnoidal solution using SS6 data (Left) and the learned solution (Right).

5. Conclusion. Based on our experience with both methods we prefer the pseudo-spectral method over the machine learning method. Running the psuedo-spectral method took at most 10 minutes even for the sixth-order scheme whereas training a neural network took about 1 hour and 30 minutes when we used 10000 training points. Additionally, the pseudo-spectral method had much lower errors than the machine learning approach. We recognize that the neural network approach offers something the split-step method does not which is the ability to detect the model from a data set. Also, the neural network approach is very flexible in that we can customize what errors to minimize and choose infinitely many combinations of hidden layers and nodes within each of the layers. While this is of value, it is also largely why we cannot put as much trust into the machine learning method as we can for the split-step methods where we can analyze the accuracy of the scheme more easily.

REFERENCES

[1] Anastassiya Semenovaa, Sergey A. Dyachenko, Alexander O. Korotkevicha, Pavel M. Lushnikova, *Comparison of Split-Step and Hamiltonian Integration Methods for Simulation of the Nonlinear Schrödinger Equation.*, arXiv preprint submitted to Journal of Computational Physics, (2020).

[2] C. Sulem, P. Sulem, *The nonlinear schrödinger equation: Self-focusing and wave collapse*, Applied Mathematical Sciences, Springer, New York, (1999).

[3] J. Carter, *Stability and existence of traveling-wave solutions of the two-dimensional nonlinear schrodinger equation and its higher-order generalizations.*, 2001.

[4] K. P. Leisman, D. Zhou, J. W. Banks, G. Kovačič, and D. Cai, *Effective dispersion in the focusing nonlinear schrödinger equation*, Physical review. E, 100 (2019).

[5] H. Potgieter, J. D. Carter, and D. M. Henderson, *Modeling the second harmonic in surface water waves using generalizations of nls*, 2021, https://arxiv.org/abs/2008.09715.

[6] M. Raissi, P. Perdikaris, and G. Karniadakis, *Physics-informed neural networks: A deep learning frame-*

work for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378 (2019), pp. 686–707, https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045, https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[7] J. A. C. Weideman and B. M. Herbst, *Split-step methods for the solution of the nonlinear schrodinger equation*, SIAM Journal on Numerical Analysis, 23 (1986), pp. 485–507.

[8] J. A. C. Weideman and B. M. Herbst, *Split-step methods for the solution of the nonlinear schrödinger equation*, SIAM Journal on Numerical Analysis, 23 (1986), pp. 485–507, https://doi.org/10.1137/0723033.

[9] H. Yoshida, *Construction of higher order symplectic integrators.*, Physics Letters A, (1990), pp. 150:262–268.