Reduced Order Model for non-linear Schrödinger Equation

Mohamad Nabizadeh

3/7/2022

Problem definition

The objective of this report is to present a numerical solution for the non-linear Schrödinger equation and to showcase the applicability of reduced order model on such computationally exhaustive problems. Schrödinger equation can be written as follows:

$$iu_t + \frac{1}{2}u_{xx} + |u|^2u = 0,$$

Where u is the physical state of the system, u_t is the time derivative, u_{xx} is the second spatial derivative, and $|u|$ is a scalar representing the absolute size of u. To solve this ODE, first, let's keep u_t on the left hand side of the equation and take all the other terms to the right hand side, and multiply both sides by $-i$, we get:

$$u_t = \frac{i}{2}u_{xx} + i|u|^2u$$

Numerical solution

We can now use Fourier basis algorithm to computationally solve this equation, taking Fourier transform of both sides gives the following:

$$\hat{u_t} = -\frac{i}{2}k^2\hat{u} + iFT(|u|^2u)$$

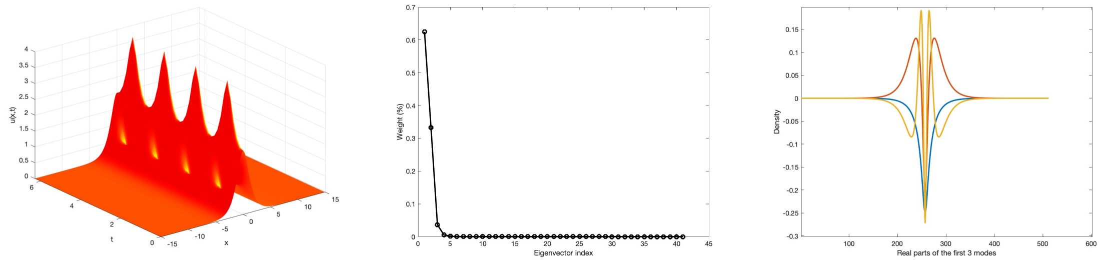
Now we can use this first order ODE, and solve it in a time-stepper loop. A sample code is imlemented in Matlab.

```
clear all; close all; clc

L = 30; n = 512;
x2 = linspace(-L/2, L/2, n+1); x = x2(1:n);
t = linspace(0, 2*pi, 41);

k = (2*pi/L) * [0:n/2-1 -n/2:-1].';

u = 2*sech(x);
ut = fft(u); % initial condition

[t, utsol] = ode45('nls_rhs',t,ut,[],k);

for j=1:length(t)
    usol(j,:) = ifft(utsol(j,:));
end
```

where the **nls_rhs** is defined as:

```
function rhs = nls_rhs(t, ut, dummy, k)
    u = ifft(ut);
    rhs = -i/2 * k.^2 .* ut + i .* fft((abs(u).^2) .* u);
```

Results



(a) Surface plot of the numeric solution, (b). weights of different modes in percentage, and (c). Top three modes.

Reduced order modeling

First-order model

Despite the high dimensional nature of the problem, we see -both visually and quantitatively- that 3 orthogonal components can explain more than 99% of the standard deviation in the answer. Therefore, we're planning to use only those components to build the solution. First, let's consider only one component. Let's use separation of variables:

$$u = a(t)\psi(t),$$

where,

$$\psi(t) = [\psi_1]$$

$$ia'\psi = \frac{1}{2}a\psi_x x + |a|^2 a|\psi^2|\psi$$

there is only one mode, and we know that the inner product of each component by itself is:

$$\psi.\psi = 1$$

Plug that in:

$$ia' = \frac{1}{2}a(\psi_{xx}.\psi) + |a|^2 a(|\psi|^2\psi.\psi)$$

Therefore, in the case of reducing the order to 1, we get to the following ODE:

$$ia_t + \frac{\alpha}{2}a + \beta|a|^2 a = 0,$$

which in fact has the following analytical solution.

$$a(t) = a(0)e^{[i\frac{\alpha}{2}+\beta|a(0)|^2 t]}$$

But, we know that existence of an analytical solution is highly unlikely for higher orders and we'll need numerical techniques to aquire a solution.

Third-order model

Let’s use the first three orthogonal components now, rewrite the ODE first:

$$u_t = \frac{i}{2} u_x x + i|u|^2 u,$$

From seperation of variables, we have:

$$u = \phi(x) a(t)$$

Also, we have from eigenvalue decomposition the following:

$$X = U \Sigma V^*,$$

where,

$$U = \phi_r,$$

Let’s plug that into our ODE:

$$a' = \frac{i}{2} \phi_r^T \phi_{xx} + i \phi_r^T |\phi_r a|^2 \phi_r a$$

Which is our approximation of the solution for this 2D system. Apart from the equation, we also need initial conditions, where $sech(x)$ is proposed as an appropriate initial condition for $u(x, 0)$. Therefore:

$$u(0) = 2sech(x) = \phi_r a(0),$$

$$a(0) = \phi_r^T \times (2sech(x))$$

Having established the reduced order model, and the initial conditions, we can now proceed to the implementation of the model, but before that, let’s quickly remind about how we calculate a second derivative by the means of Fourier transform.

$$\phi_{xx}^i = IFFT(-k^2.FFT(\phi_r^i))$$

Let’s now take a look at our implementation in Matlab:

Implementation

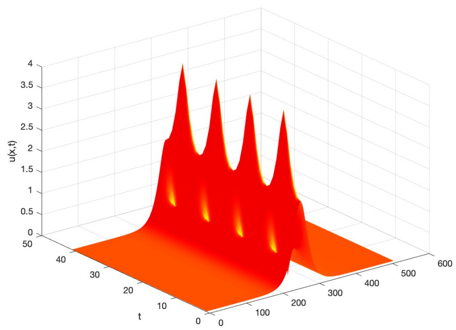
```
%% reduced order model
phi = u(:,1:3);

mode1_xx = ifft(-(k.^2).*fft(u(:,1)));
mode2_xx = ifft(-(k.^2).*fft(u(:,2)));
mode3_xx = ifft(-(k.^2).*fft(u(:,3)));


phixx = [mode1_xx mode2_xx mode3_xx];

a0 = phi' * 2 *sech(x).';
[t, asol] = ode45('a_rhs', t, a0, [], phi, phixx);


for j = 1:length(t)
    u(j,:) = asol(1) * phi(:,1) + asol(2) * phi(:,2)+ asol(3) * phi(:,3);
end
```

where,

```
function rhs = a_rhs(t, a, dummy, phi, phixx)
 rhs = i/2 .* (phi.') * phixx *a + i*(phi.') * (sqrt(sum(abs(phi * a).^2)) * phi * a);
```

Results

We can see that using the top three vectors, we can build a model that can reproduce a solution that closely resembles that of the original problem. This is a great example on how reduced order modeling can provide much more efficient solutions with minor loss of information.



A special shout-out to Nathan Kutz for his great content on the subject.