

an introduction to the split step fourier method using matlab

Published: September 19, 2026 | Index ID: #-

An Introduction to the Split Step Fourier Method Using MATLAB

When studying wave propagation in optical fibers, quantum systems, or fluid dynamics, one often encounters nonlinear partial differential equations that are difficult or impossible to solve analytically. The split step Fourier method (SSFM) is a powerful numerical technique that turns this challenge into a manageable computation. This article offers a comprehensive, step by step introduction to SSFM, focusing on its implementation in MATLAB. By the end, you will understand the theory, know how to code the algorithm, and be ready to apply it to real world problems.

Why the Split Step Fourier Method?

Many physical systems are governed by equations that couple linear dispersion or diffraction with nonlinear interactions. The nonlinear Schrödinger equation (NLSE) is a classic example:

$$i \frac{\partial \psi}{\partial z} + \frac{1}{2k_0} \frac{\partial^2 \psi}{\partial t^2} + \gamma |\psi|^2 \psi = 0$$

Here, ψ represents the complex envelope of a wave, z is the propagation distance, t is the retarded time, and γ characterizes the nonlinearity. Analytic solutions exist only for special cases; otherwise, we must rely on numerical methods.

Traditional finite difference or finite element approaches discretize both space and time simultaneously, which can lead to stability issues or excessive computational cost, especially when high frequency components are present. The split step Fourier method overcomes these limitations by separating the linear and nonlinear parts of the equation and treating each part exactly over a small step.

Fundamental Idea of the Split Step Fourier Method

The SSFM is based on the Trotter–Suzuki decomposition. For a small propagation step Δz the evolution operator can be approximated as a product of two exponentials:

$$U(\Delta z) \approx \exp(i \Delta z L) \exp(i \Delta z N)$$

where L represents the linear operator (dispersion, diffraction) and N represents the nonlinear operator (self phase modulation, Kerr effect). Because each exponential can be evaluated exactly in its natural domain—linear in Fourier space, nonlinear in real space—the method achieves high accuracy with relatively large step sizes.

Linear and Nonlinear Operators in the NLSE

Let's break down the operators for the standard NLSE:

- Linear operator (L): $\frac{1}{2k_0} \frac{\partial^2}{\partial t^2}$. In the frequency domain, this becomes multiplication by $-\frac{1}{2k_0} \omega^2$. ω is the angular frequency.
- Nonlinear operator (N): $\gamma |\psi|^2 \psi$. This acts locally in the time domain and simply multiplies the field by a phase factor.

Algorithmic Steps of the SSFM

1. Initialize the field: Define the input pulse $\psi(0, z)$ over a time grid.
2. Compute Fourier transform: Apply FFT to $\psi(0, z)$ to get $\tilde{\psi}(0, \omega)$.
3. Linear propagation in Fourier space: Multiply by the linear propagator $\exp(-i(1/2)\beta_2 \omega^2 z)$.
4. Inverse Fourier transform: Convert back to the time domain.
5. Nonlinear phase shift: Multiply by $\exp(i\gamma |\psi|^2 z)$.
6. Repeat for the desired number of steps.

Each loop step is called a “split step” because it splits the evolution into a linear part and a nonlinear part. The order of operations can be swapped (nonlinear first, then linear) for a slightly different variant known as the symmetric or Strang splitting, which reduces error.

Strang Splitting for Improved Accuracy

In Strang splitting, the sequence is:

1. Half linear step: $\psi \leftarrow \tilde{\psi} \cdot \exp(-i(1/2)\beta_2 \omega^2 z/2)$
2. Full nonlinear step: $\psi \leftarrow \psi \cdot \exp(i\gamma |\psi|^2 z)$
3. Half linear step: $\psi \leftarrow \tilde{\psi} \cdot \exp(-i(1/2)\beta_2 \omega^2 z/2)$

This symmetric arrangement cancels out some leading error terms, providing a more accurate solution for the same step size.

Implementing SSFM in MATLAB

MATLAB's vectorized operations and built in FFT functions make it ideal for implementing SSFM. Below is a complete, commented MATLAB script that simulates pulse propagation in a single mode fiber using the NLSE.

MATLAB Code:

```
t = linspace(-50, 50, 2^12); % Time grid in ps
dt = t(2)-t(1);

w = 2*pi*fftshift(2*pi*(0:(length(t)-1))/(dt*length(t))); % Angular frequency grid
dz = 0.01; % Propagation step in km
Nsteps = 1000; % Number of steps
beta2 = -20; % GVD parameter in ps^2/km
gamma = 1.3; % Nonlinearity in 1/(W·km)
A0 = sqrt(1).*exp(-t.^2/2); % Input Gaussian pulse (peak power 1 W)
A = A0;

for n = 1:Nsteps
    % Linear half step (Strang)
    A_fft = fftshift(fft(A));
    A_fft = A_fft .* exp(-1i*beta2*w.^2*dz/4);
    A = ifft(ifftshift(A_fft));
```

```

% Nonlinear full step
A = A .* exp(1i*gamma*abs(A).^2*dz);

% Linear half step
A_fft = fftshift(fft(A));
A_fft = A_fft .* exp(-1i*beta2*w.^2*dz/4);
A = ifft(ifftshift(A_fft));

end

plot(t, abs(A).^2); xlabel('Time (ps)'); ylabel('Power (W)'); title('Pulse after propagation'); grid on;

```

This script follows the Strang splitting scheme. The **fftshift** and **ifftshift** functions center the zero frequency component, ensuring that the linear phase factor aligns correctly with the frequency grid.

Optimizing Performance

- Vectorization: All operations are performed on entire arrays, avoiding explicit loops.
- Pre computation: Store the linear propagator $\exp(-i\beta_2 \omega^2 dz/4)$ before the loop.
- Memory Management: Use single precision if the dynamic range allows, reducing memory usage.
- Parallel Computing Toolbox: For very long propagation distances, distribute the steps across multiple cores using `parfor`.

Common Pitfalls and How to Avoid Them

1. Aliasing: The time window must be large enough to contain the pulse throughout propagation. If the pulse broadens, the window should be increased or a zero padding strategy applied.
2. Step Size Selection: Too large a step size leads to phase errors; too small increases computation time. A rule of thumb is to keep $\Delta z \ll \lambda^2 / |\beta_2|$, where λ is the pulse width.
3. Boundary Conditions: The FFT assumes periodic boundaries. If the pulse does not vanish at the edges, wrap around artifacts appear. Padding with zeros mitigates this.
4. Nonlinearity Saturation: For very high peak powers, the simple Kerr model may break down. Incorporate higher order terms or saturable nonlinearity if needed.

Extending SSFM Beyond the NLSE

The split step Fourier method is versatile and can be adapted to many other equations:

- Higher order NLSE – include third order dispersion or Raman scattering terms.
- Coupled equations – model dual polarization or wavelength division multiplexed systems.
- Non paraxial wave equations – treat beam propagation in free space.
- Hydrodynamic equations – simulate surface gravity waves in shallow water.

In each case, identify the linear operator that can be diagonalized in Fourier space, and treat the remaining terms in real space. The MATLAB implementation remains essentially the same; only the propagators and nonlinear phase factors change.

Practical Applications

- Optical Fiber Communications: Design dispersion managed links, evaluate soliton dynamics, and study

supercontinuum generation.

- Nonlinear Photonics: Predict the behavior of microresonators, photonic crystal fibers, and waveguides.
- Quantum Optics: Model the evolution of squeezed states and entangled photon pairs in nonlinear media.
- Acoustics: Simulate nonlinear sound propagation in underwater environments.

Benchmarking and Validation

Before applying SSFM to a new problem, validate the implementation:

1. Analytical Test Cases: Compare SSFM results with known soliton solutions or linear dispersion solutions.
2. Convergence Study: Reduce Δt and Δx until the solution converges to a stable result.
3. Energy Conservation: For conservative systems, the total power should remain constant; monitor this quantity to detect numerical drift.

Resources for Further Learning

- Agrawal, G. P. Nonlinear Fiber Optics – foundational text covering NLSE and SSFM.
- Numerical Recipes – sections on FFTs and solving differential equations.
- MATLAB Central File Exchange – community submitted SSFM codes for various equations.
- Online tutorials from university labs – many provide step by step MATLAB examples.

Conclusion

The split step Fourier method is a cornerstone of modern computational physics and engineering. By cleverly exploiting the diagonal nature of linear operators in Fourier space and the locality of nonlinear effects in real space, SSFM delivers accurate, efficient simulations of complex wave phenomena. MATLAB's powerful vectorized operations and FFT routines make it straightforward to implement, test, and extend the algorithm for a wide range of applications.

Armed with the theory, algorithmic steps, and a ready to run MATLAB script, you are now equipped to tackle nonlinear wave propagation problems with confidence. Whether you are modeling optical solitons, designing fiber optic communication links, or exploring exotic quantum states, the split step Fourier method will be an indispensable tool in your computational arsenal.

Baca Juga (Artikel Terkait)

- [algebra and trigonometry stewart](http://www.aussizzlegal.com/algebra-and-trigonometry-stewart.pdf)

URL: <http://www.aussizzlegal.com/algebra-and-trigonometry-stewart.pdf>

- [algebra and trigonometry lial miller solution](http://www.aussizzlegal.com/algebra-and-trigonometry-lial-miller-solution.pdf)

URL: <http://www.aussizzlegal.com/algebra-and-trigonometry-lial-miller-solution.pdf>

- [algebra and trigonometry larson hostetler 7th edition online](http://www.aussizzlegal.com/algebra-and-trigonometry-larson-hostetler-7th-edition-online.pdf)

URL: <http://www.aussizzlegal.com/algebra-and-trigonometry-larson-hostetler-7th-edition-online.pdf>

- [algebra and trigonometry larson hostetler 6th edition answers](http://www.aussizzlegal.com/algebra-and-trigonometry-larson-hostetler-6th-edition-answers.pdf)

URL: <http://www.aussizzlegal.com/algebra-and-trigonometry-larson-hostetler-6th-edition-answers.pdf>

Tags: #introduction #split #step #fourier #method #using #matlab

