Report of Numerical Computation
Project1-Bessel Function

Ziyan Li

1 Model

When you want to finish a project, you must first figure out how it works. To solve Bessel fuction $J_n(x)$ which satisfies the recurrence relation,we can finish it step by step.

1.1 Recurrence

The Bessel Function $J_n(x)$ satisfy

$$J_{n+1}(x) - \frac{2n}{x} J_n(x) + J_{n-1}(x)$$

It also satisfy the sum identity

$$[J_0(X)]^2 + 2\sum_{n}^{\infty}[J_n(X)]^2 = 1$$

Then we set $J_M(x) = 0, J_{M-1}(x) = 1$. Apply the recurrence then we can get the lower order of Bessel fuction.

1.2 Normalization

From 1.1 we can get $J_n(x)$ separately. Then we can sum them up by

$$sum = [J_0(X)]^2 + 2\sum_{n}^{\infty}[J_n(X)]^2$$

Then we can get the factor we use to normalize: $e = \sqrt{1/sum}$, then we just multiply J with e to complete the normalization

$$J = e * J$$

.

1.3 Draw

If we want to find $J_n x$, we must find the corresponding value.Then we fix n and change x so that we can draw the picture of $J_n(x)$

2 Calculation Results

2.1 Plot Bessel fuction Jn(x)

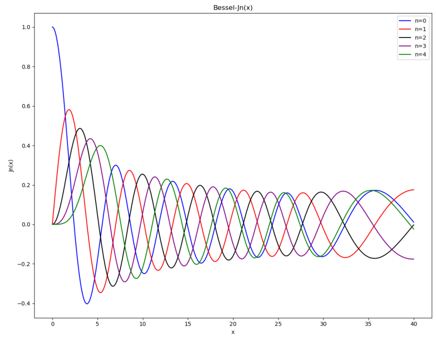


FIG 1: Bessel function with M=30

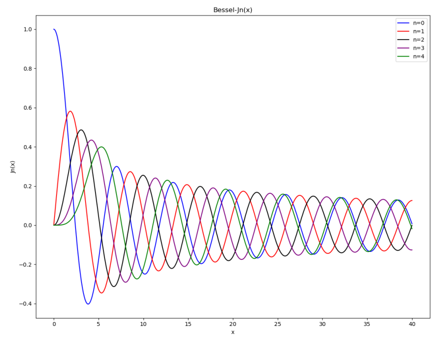


FIG 2: Bessel function with M=50

2.2 Compared with Standard datas

```
 Enter the length of Array J:
40
 Enter the value of x:
1
 J(            0 )=  0.765197686557967
 J(            1 )=  0.440050585744934
 J(            2 )=  0.114903484931900
 J(            3 )=  1.956335398266841E-002
 J(            4 )=  2.476638964109955E-003
 J(            5 )=  2.497577302112344E-004
 J(            6 )=  2.093833800238927E-005
 J(            7 )=  1.502325817436808E-006
 J(            8 )=  9.422344172604502E-008
 J(            9 )=  5.249250179911876E-009
 J(           10 )=  2.630615123687453E-010
 J(           11 )=  1.198006746303137E-011
 J(           12 )=  4.999718179448405E-013
 J(           13 )=  1.925616764480173E-014
```

(a) Produced by Code

#-		x= 1.
#-	n	Jn(x)
	0	0.7651976866E+00
	1	0.4400505857E+00
	2	0.1149034849E+00
	3	0.1956335398E-01
	4	0.2476638964E-02
	5	0.2497577302E-03
	6	0.2093833800E-04
	7	0.1502325817E-05
	8	0.9422344173E-07
	9	0.5249250180E-08
	10	0.2630615124E-09
	11	0.1198006746E-10
	12	0.4999718179E-12
	13	0.1925616764E-13

(b) Standard Datas

FIG 3: x=1

Compare Jn(1) produced by codes with standard datas.

```
Jn( 0 )= -0.24593576445134835
Jn( 1 )= 0.04347274616886152
Jn( 2 )= 0.2546303136851207
Jn( 3 )= 0.05837937930518675
Jn( 4 )= -0.21960268610200862
Jn( 5 )= -0.23406152818679365
Jn( 6 )= -0.014458842084785047
Jn( 7 )= 0.2167109176850516
Jn( 8 )= 0.3178541268438573
Jn( 9 )= 0.29185568526512007
Jn( 10 )= 0.20748610663335884
Jn( 11 )= 0.12311652800159764
Jn( 12 )= 0.06337025497015598
Jn( 13 )= 0.028972083926776755
Jn( 14 )= 0.011957163239463576
Jn( 15 )= 0.004507973143721252
Jn( 16 )= 0.0015667561917001801
Jn( 17 )= 0.0005056466697193248
Jn( 18 )= 0.00015244248534552427
Jn( 19 )= 4.3146277524562545e-05
Jn( 20 )= 1.1513369247813394e-05
Jn( 21 )= 2.9071994666910335e-06
Jn( 22 )= 6.968685122889468e-07
Jn( 23 )= 1.5902198738033276e-07
Jn( 24 )= 3.463262966058384e-08
Jn( 25 )= 7.214634990469657e-09
Jn( 26 )= 1.4405452917644454e-09
Jn( 27 )= 2.7620052670546067e-10
Jn( 28 )= 5.0937552445042223e-11
Jn( 29 )= 9.049766986775809e-12
Jn( 30 )= 1.551096078257466e-12
Jn( 31 )= 2.568094827689877e-13
Jn( 32 )= 4.1122714910257646e-14
```

(a) Produced by Code

:-		x= 10.
:-	n	Jn(x)
	0	-0.2459357645E+00
	1	0.4347274617E-01
	2	0.2546303137E+00
	3	0.5837937931E-01
	4	-0.2196026861E+00
	5	-0.2340615282E+00
	6	-0.1445884208E-01
	7	0.2167109177E+00
	8	0.3178541268E+00
	9	0.2918556853E+00
	10	0.2074861066E+00
	11	0.1231165280E+00
	12	0.6337025497E-01
	13	0.2897208393E-01
	14	0.1195716324E-01
	15	0.4507973144E-02
	16	0.1566756192E-02
	17	0.5056466697E-03
	18	0.1524424853E-03
	19	0.4314627752E-04
	20	0.1151336925E-04
	21	0.2907199467E-05
	22	0.6968685123E-06
	23	0.1590219874E-06
	24	0.3463262966E-07
	25	0.7214634990E-08
	26	0.1440545292E-08
	27	0.2762005267E-09
	28	0.5093755245E-10
	29	0.9049766987E-11
	30	0.1551096078E-11
	31	0.2568094828E-12
	32	0.4112271491E-13
	33	0.6375892657E-14
	34	0.9581766237E-15

(b) Standard Datas

FIG 4: x=10

Compare Jn(10) produced by codes with standard datas.

Project2-Numerical Integration

1 Model

1.1 Trapezoidal Rule

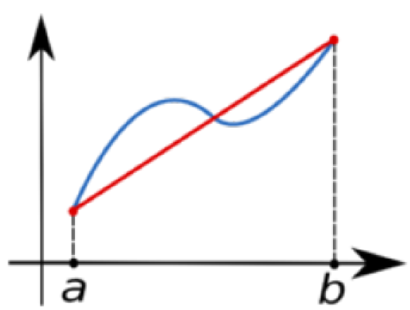


FIG 1: Theory of Trapezoidal Rule

Numerical integration is indeed to approximate the integration by sum all the infinite small areas. For the trapezoidal rule, we can use the approximation equation:

$$\int_a^b f(x)dx \approx (b-a)[\frac{f(a)+f(b)}{2}]$$

1.2 Simpson's rule

The same theory as trapezoidal rule. The difference is the equation convert to :

$$\int_a^b f(x)dx \approx \frac{(b-a)}{6}[f(a)+4f(\frac{a+b}{2})+f(b)]$$

So we can calculate the integration :

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + ... + 4f(x_{n-1}) + f(x_n)]$$

1.3 Monte Carlo

Use Monte Carlo Rule to calculate π:

$$\int_{-1}^{1}\int_{-1}^{1} H(x,y)dxdy = \pi$$

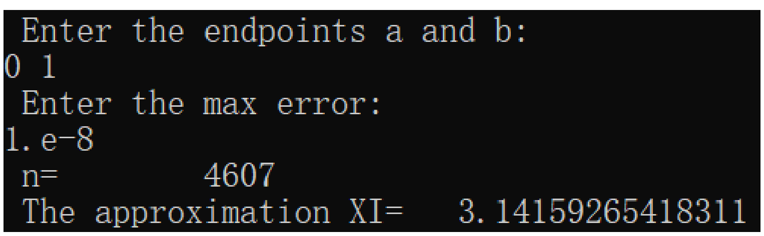
So we can find that

$$\pi \approx I_N = \frac{4}{N}\sum_{i=1}^{N} H(x_i, y_i).$$

2 Calculation Results of Π

2.1 The Trapezoidal Rule

Calculate numerically the value of π by $I = \int_0^1 \frac{4}{1+x^2} = \pi$ We set the errors then we can get the grid numbers n.



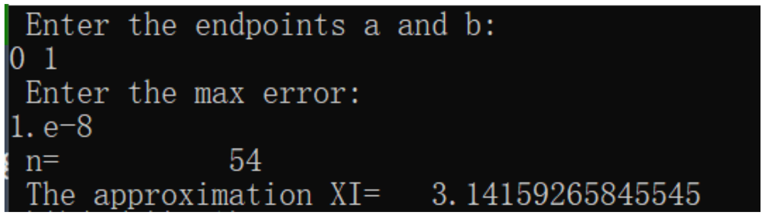
```
 Enter the endpoints a and b:
0 1
 Enter the max error:
1.e-8
 n=         4607
 The approximation XI=   3.14159265418311
```

FIG 2: Result of the Trapezoidal Rule

2.2 The Simplson Rule

Use Simpson' rule to calculate numerically the value of π by $I = \int_0^1 \frac{4}{1+x^2} = \pi$. We set the errors then we can get the grid numbers n.

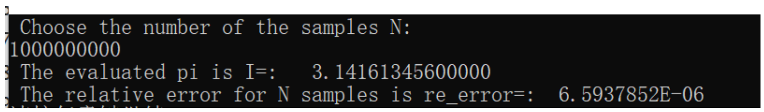
We can find that for than same error, the Simplson rule uses less recurrences and is more efficient.



```
 Enter the endpoints a and b:
0 1
 Enter the max error:
1.e-8
 n=           54
 The approximation XI=    3.14159265845545
```

FIG 3: Result of Simplson

2.3 Monte Carlo Calculation



```
 Choose the number of the samples N:
1000000000
 The evaluated pi is I=:    3.14161345600000
 The relative error for N samples is re_error=:   6.5937852E-06
```

图 4: Use Monte Carlo to Calculate Π

By applying the Monte Carlo method, I calculate the constant value Π within relative error smaller than 1×10^{-5}.

Project3-Fast Fouier Transformation

1 Model

1.1 about the Fast Fourier Transformation(FFT)

When we talk about the Fast Fourier Transformation, we are referring to a special kind of fourier transformation. For example,

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j \\ &= \sum_{j=0}^{N/2-1} e^{2\pi i(2j)k/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi i(2j+1)k/N} f_{2j+1} \\ &= \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi ijk/(N/2)} f_{2j+1} = F_k^e + W^k F_k^o \end{aligned} \tag{1}$$

And we can find that F_k^e and F_k^o are similar to F_k except for the number N. So recurrence can be applied here.

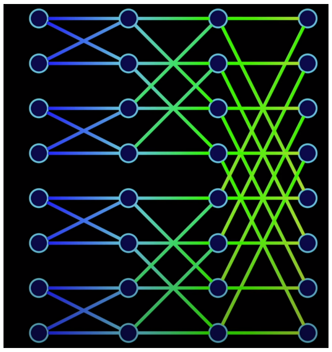


FIG 1: Recurrence of FFT

1.2 Why FFT?

When we use a complex algorithm rather than a much more complex one, we must view the benefit of the former one. As for FFT, it reduce our runtime.

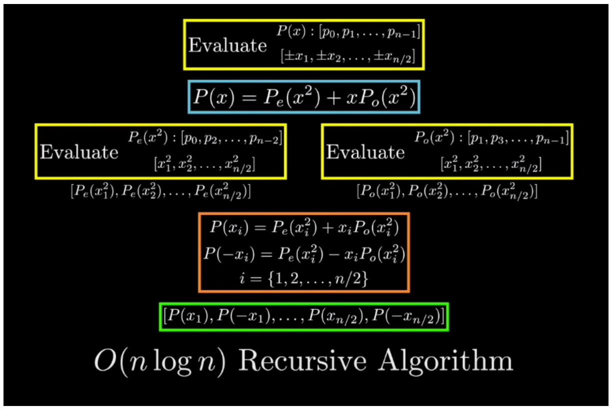


FIG 2: Recurrence of FFT

If you don't use FFT and calculate $F_k = \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j$. The run time will be $O(n^2)$ for calculation of all F_k from 0 to N. However, if using FFT, the run time is just $O(nlogn)$

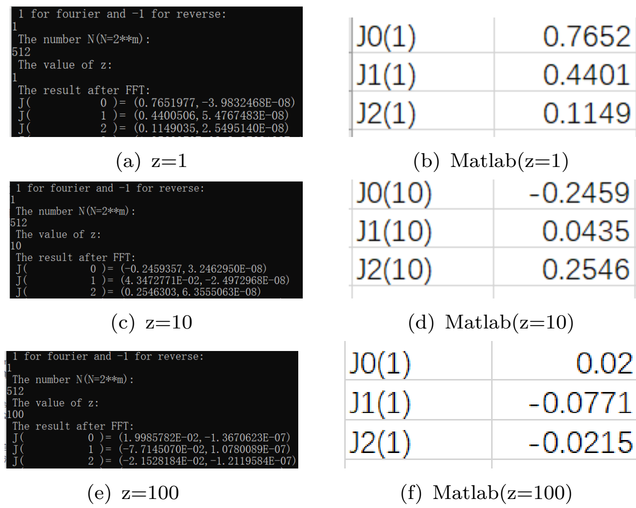
2 Question

Use the program written using Fortran to calculate the Bessel fuction by using the following integral

$$J_n(z) = \frac{i^{-n}}{2\pi} \int_0^{2\pi} e^{izcos(\theta)} e^{in\theta} d\theta$$

Calculate $J_n(z)$ with z =1, 10 and 100, and compare them with those using Matlab function Besselj(n,z).

3 Calculation Results



(a) z=1 (b) Matlab(z=1)

(c) z=10 (d) Matlab(z=10)

(e) z=100 (f) Matlab(z=100)

FIG 3: FFT for Bessel Fuction

We compare the results from both fortran program written by me and Matlab. Easy to find that they are nearly equal within finite error allowed.

Final Porject: Time Dependent Maxwell/Schrödinger Equation

1 Model

1.1 Time Dependent Maxwell Equation(TDME)

There are several ways to solve TDME problems. Yee time integration and Lie-Trotter-Suzuki time integration are two ways I used.

1.1.1 Yee time Integration

Actually it's a algorithm based on iteration. We can easily find that the time derivative of $f^n(i,j,k)$ can be written as

$$\frac{\partial}{\partial t} f^n(i,j,k) = \frac{f^{n+1/2}(i,j,k) - f^{n-1/2}(i,j,k)}{\Delta t} + O((\Delta t)^2).$$

According to maxwell equation we can get E_x component. Combine the spatial and time derivatives, we can update equation:

$$E_x^{n+1}(i,j,k) = E_x^n(i,j,k) + \frac{\Delta t}{\epsilon(i,j,k)}[\frac{H_z^{n+1/2}(i,j+1,k) - H_z^{n+1/2}(i,j-1,k)}{\Delta y} - \frac{H_z^{n+1/2}(i,j,k+1) - H_z^{n+1/2}(i,j,k-1)}{\Delta z}]$$

We can also derive $H_z^{n+1/2}(i,j+1,k)$ and $H_y^{n+1/2}(i,j,k+1)$ using the similar way. So we update E_z^{n+1} and other quantum again and again until we can find all of (i,j,k).

1.1.2 Lie-Trotter-Suzuki

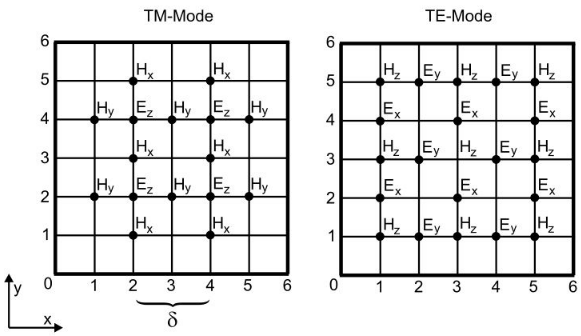


Figure 2-3: Positions of the TM and TE-mode EM field components on the two-dimensional grid for $n_x = 5$ and $n_y = 5$.

FIG 1: Discretization into Lattice

Firstly we discretize the E_z , H_x and H_y. We set

$$\Psi_{TM}(i,j,t) = \begin{cases} X_x(i,j,y) = \sqrt{\mu_{i,j}} H_x(i,j,t) \\ X_y(i,j,y) = \sqrt{\mu_{i,j}} H_y(i,j,t) \\ Y_z(i,j,y) = \sqrt{\epsilon_{i,j}} E_z(i,j,t) \end{cases} \tag{1}$$

So it follows on the lattice. We have:

$$\frac{\partial}{\partial t}\Psi_{TM}(t) = H\Psi_{TM}(t) = \sum_{i=1}^{n_x-2}{}' \sum_{j=1}^{n_y-2}{}' [H^{(x)}(i,j) + H^{(y)}(i,j)]\Psi_{TM}(t)$$

Where

$$H^{(x)}(i,j) = +\frac{e_{i,j+1}e^T_{i+1,j+1} - e_{i+1,j+1}e^T_{i,j+1}}{\delta\sqrt{\epsilon_{i+1,j+1}\mu_{i,j+1}}} + \frac{e_{i+1,j+1}e^T_{i+2,j+1} - e_{i+2,j+1}e^T_{i+1,j+1}}{\delta\sqrt{\epsilon_{i+1,j+1}\mu_{i+2,j+1}}}$$

$$H^{(y)}(i,j) = -\frac{e_{i+1,j}e^T_{i+1,j+1} - e_{i+1,j+1}e^T_{i+1,j}}{\delta\sqrt{\epsilon_{i+1,j+1}\mu_{i+1,j}}} - \frac{e_{i+1,j+1}e^T_{i+1,j+2} - e_{i+1,j+2}e^T_{i+1,j+1}}{\delta\sqrt{\epsilon_{i+1,j+2}\mu_{i+1,j+1}}}$$

And we all know that $e^{\tau H_{x1}}$ and $e^{\tau H_{x2}}$ is given by

$$exp[\alpha \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}] = \begin{pmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{pmatrix}$$

Therefore we can calculate $Y_z(i,j,t)$ and $Y_z(i+1,j,t)$ one by one.($Y_z(i+1,j,t)$ and $Y_z(i+2,j,t)$ the same)

1.2 Time Dependent Schrodinger Equation(TDSE)

From the Schrodinger equation

$$i\frac{\partial}{\partial t}|\phi(t)>= H|\phi(t)>$$

Therefore, we can get $|\phi(m\tau)>= e^{-im\tau H}|\phi(t=0)>$, as we do as TDME. We can discretize them into grid.

$$\begin{aligned}
H\Phi_{l,k}(t) = \frac{1}{48\pi^2\delta^2}\Bigg\{ \; & \Big[1 - i\delta\big(A_{l,k} + A_{l+2,k}\big)\Big]\Phi_{l+2,k}(t) \\
& + \Big[1 + i\delta\big(A_{l-2,k} + A_{l,k}\big)\Big]\Phi_{l-2,k}(t) \\
& - 16\Big[1 - \frac{i\delta}{2}\big(A_{l,k} + A_{l+1,k}\big)\Big]\Phi_{l+1,k}(t) \\
& - 16\Big[1 + \frac{i\delta}{2}\big(A_{l-1,k} + A_{l,k}\big)\Big]\Phi_{l-1,k}(t) \\
& + \Phi_{l,k+2} + \Phi_{l,k-2} - 16\Phi_{l,k+1} - 16\Phi_{l,k-1}(t) \\
& + \Big[60 + 12\delta^2 A_{l,k}^2 + 48\pi^2\delta^2 V_{l,k}\Big]\Phi_{l,k}(t)\Bigg\} + \mathcal{O}(\delta^5) \quad , \quad (37)
\end{aligned}$$

FIG 2: Theory of TDSE

So we can divide the calculation into nine parts, which makes the algorithm more clear and easy.

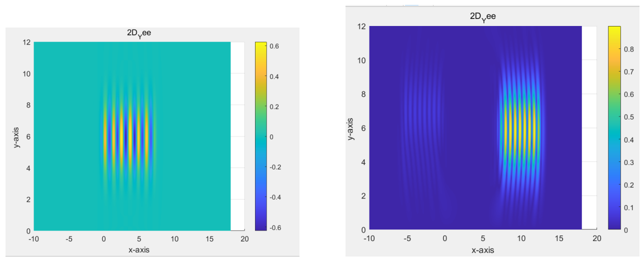
Take one part of it as an example.

$$\mathrm{exp}(\mathrm{i}\tau\alpha \mathrm{c}^{+}_{\mathrm{l,k}}\mathrm{c}_{\mathrm{l',k'}}+\mathrm{i}\tau\alpha^{*}\mathrm{c}^{+}_{\mathrm{l',k'}}\mathrm{c}_{\mathrm{l,k}}) = (\mathrm{c}^{+}_{\mathrm{l,k}}\mathrm{c}_{\mathrm{l,k}}+\mathrm{c}^{+}_{\mathrm{l',k'}}\mathrm{c}_{\mathrm{l',k'}})\mathrm{cos}\tau|\alpha|+\mathrm{i}(\alpha^{*-1}\mathrm{c}^{+}_{\mathrm{l,k}}\mathrm{c}_{\mathrm{l',k'}}+\alpha^{-1}\mathrm{c}^{+}_{\mathrm{l',k'}}\mathrm{c}_{\mathrm{l,k}})\mathrm{sin}\tau|\alpha|$$

2 Calculation Results

2.1 TDME

2.1.1 Yee

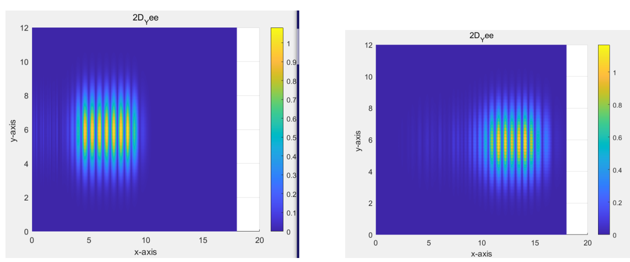


(a) t=0 (b) t=10

FIG 3: 2D TDME from t=0 to t=10(Yee)

We can find it's not all as we expect because there are two wave patterns in two side(indeed we hope the wave could move to the right). However, I have check my algorithm again and again, but I fail to solve it though I change it several times. It's a pity anyway.

2.1.2 Lie-Tro-Suzuki

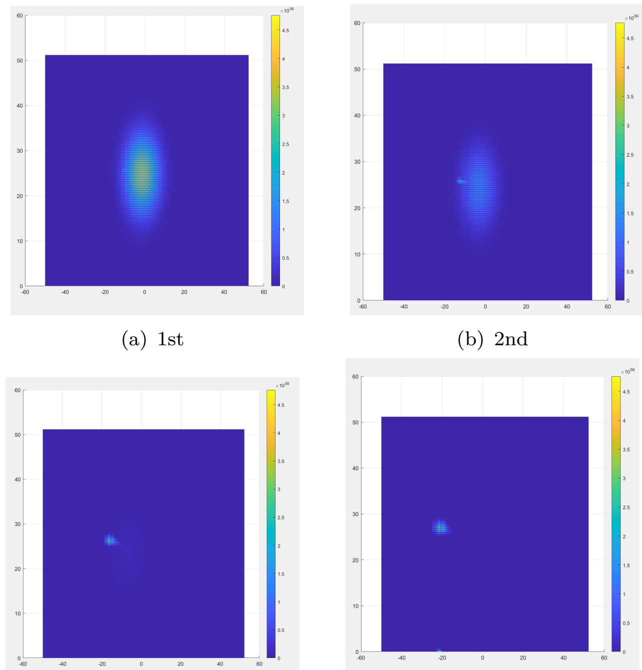


(a) t=0 (b) t=10

FIG 4: 2D TDME from t=0 to t=10(Tro)

Different from the results of former Yee way. We can find that wave can move to the right. However, it doesn't curve. It's still not perfect.

2.2 TDSE



(a) 1st

(b) 2nd

(c) 3rd

(d) 4th

FIG 5: The Movement of 2D TDSE

We can find that the movement of the 2D TDSE when it encounters a wave guide. And the wave propagating pattern is as we expect: shrink then swell a little.