Fortran 77… declarations… Complex variables!

```
IMPLICIT NONE
INTEGER j,nt,jmax,ntmax
PARAMETER(jmax=1000,ntmax=1000)
DOUBLE COMPLEX a(0:jmax-1),b(0:jmax-1),c(0:jmax-1)
DOUBLE COMPLEX chi(0:jmax-1),psi(0:jmax-1)
DOUBLE COMPLEX ar,ai
DOUBLE PRECISION L,sigma,x0,k0,dx,dt,alpha
DOUBLE PRECISION x,sspi,pi
PARAMETER(L=200.0d0,sigma=3.0d0)
PARAMETER(x0=L/6.0d0,k0=1.414213562d0) ! central x,k of
PARAMETER(dx=L/jmax) ! spatial step
PARAMETER(dt=0.1d0) ! time step
PARAMETER(alpha=dt/dx**2)
```

Fortran 90… a little more concise approach for arrays

```
IMPLICIT NONE
INTEGER, PARAMETER :: Prec14=SELECTED_REAL_KIND(14)
INTEGER :: j,nt
INTEGER, PARAMETER :: jmax=1000, ntmax=1000
COMPLEX(KIND=Prec14), DIMENSION(0:jmax-1) :: a,b,c,chi
COMPLEX(KIND=Prec14), DIMENSION(0:jmax-1) :: psi
COMPLEX(KIND=Prec14) :: ar,ai
REAL(KIND=Prec14), PARAMETER :: L=200.0d0,sigma=3.0d0
REAL(KIND=Prec14), PARAMETER :: x0=L/6.0d0,k0=1.414213562
REAL(KIND=Prec14), PARAMETER :: dx=L/jmax
REAL(KIND=Prec14), PARAMETER :: dt=0.1d0
REAL(KIND=Prec14), PARAMETER :: alpha=dt/dx**2
REAL(KIND=Prec14) :: x,sspi,pi
```

For convenience, define ar=1, ai=i…

```
ai=(0.0d0,1.0d0) ! i=sqrt(-1) (imaginary
ar=(1.0d0,0.0d0) ! 1 (real)
```

Then, to get complex arrays populated, we can take the real part * ar, and the imaginary part * ai

Some useful things then… if psi(j) is complex

```
psir = real(psi(j))            Real part of psi(j)
psii = imag(psi(j))            Imaginary part of psi(j)
mag = conjg(psi(j))*psi(j)   conjg give complex conjugate
```

Here, psir, psii are real variables, and mag is complex. However mag should in fact have zero imaginary component

Tridag… solves the linear equations… does all the hard work!

call tridag(a,b,c,psi,chi,jmax)

a,b,c,psi are the inputs, chi is what is returned by tridag after solving equations

jmax is the length of the input arrays

SUBROUTINE TRIDAG(A,B,C,R,U,N)

From numerical recipes, but modified for complex numbers linear equations

Tridag.f subroutine

Solves the matrix equation:

$Au=r$

Here A is a tridiagonal matrix, and u and r are column vectors.

We assume we can find:

$A=LU$

Here L is lower triangular and U is upper triangular.

So we solve $L(Uu)=r$, or $Ly=r$ where $y=Uu$

Good description online in Numerical Recipes

$$\mathbf{A} \cdot \mathbf{x} = (\mathbf{L} \cdot \mathbf{U}) \cdot \mathbf{x} = \mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b}$$

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

$$y_1 = \frac{b_1}{\alpha_{11}}$$

$$y_i = \frac{1}{\alpha_{ii}} \left[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j \right] \qquad i = 2, 3, \ldots, N$$

Forward substitution

$$x_N = \frac{y_N}{\beta_{NN}}$$

$$x_i = \frac{1}{\beta_{ii}} \left[y_i - \sum_{j=i+1}^{N} \beta_{ij} x_j \right] \qquad i = N-1, N-2, \ldots, 1$$

Backward substitution

Assumes $\beta_{ii}=1$

A final note on declarations… precision, etc.

To make a single precision real number, we declare

```
REAL :: X, Y, Z
```

These are accurate to 7 significant digits, usually stored in 32 bits. Single precision is the default

If we want double precision,

```
REAL*8 :: X, Y, Z
DOUBLE PRECISION :: A, B, C
INTEGER, PARAMETER :: Prec14 = SELECTED_REAL_KIND(14)
REAL(KIND = Prec14) :: D, E, F
```

Each of these gives 14 digit precision in 64 bits

What physics to check?

We should have that the magnitude squared of the wave function Integrated over are space = 1 at all times… probability conservatior

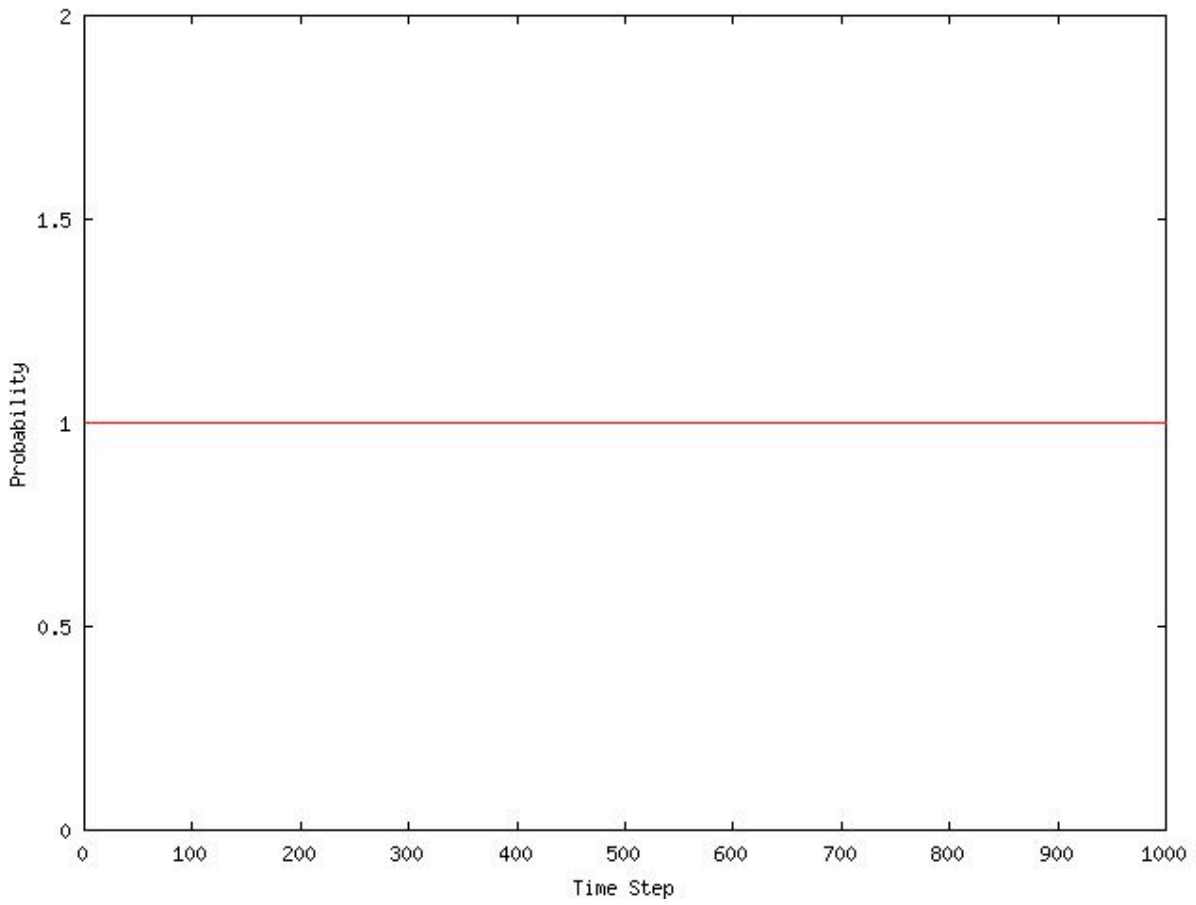
We can check this numerically!

$$\int_0^L \psi^*(x,t)\psi(x,t)dx = 1$$

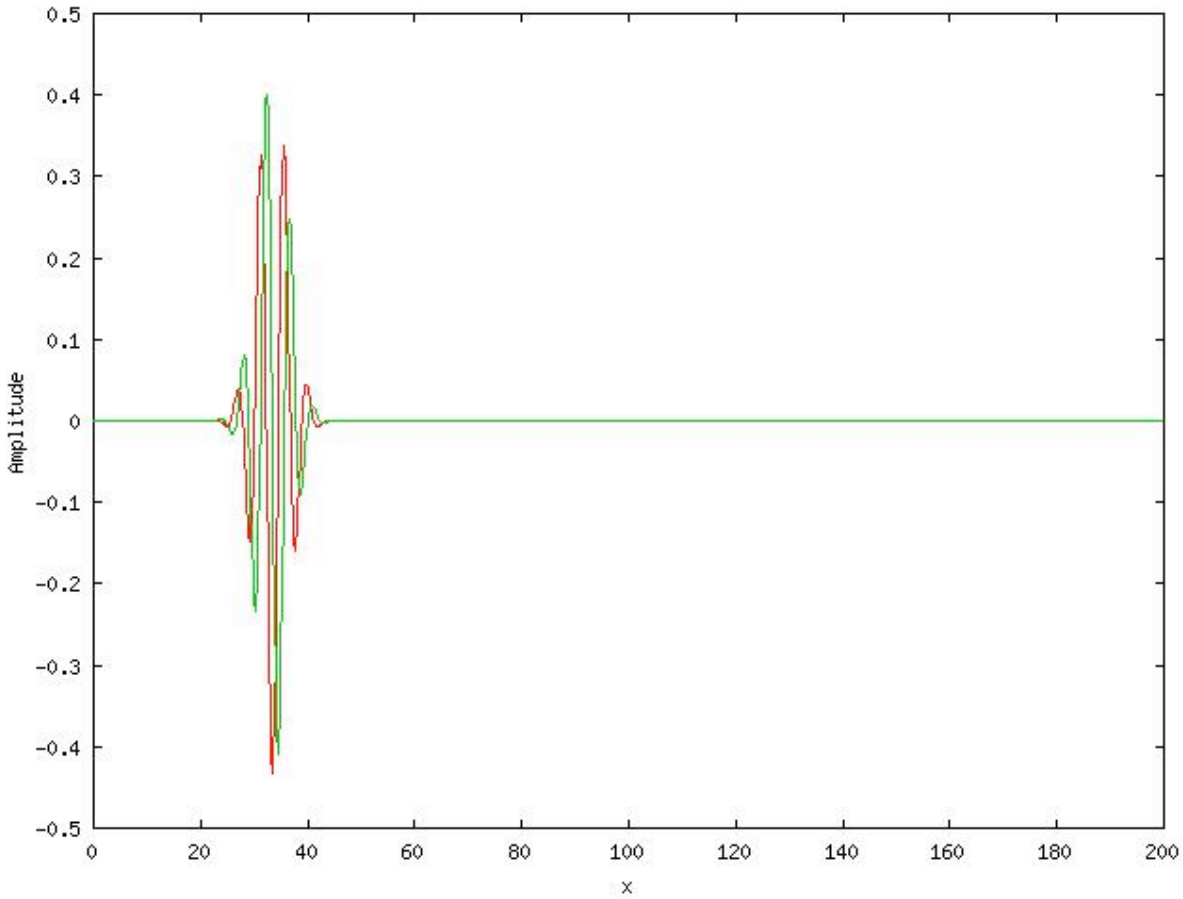
Also, we can estimate the *group velocity* of the wave packet and compare with our calculation… next time…

Remember, it is critical that we check the physics that we know must apply (e.g. conservation laws) to verify that our code is obeying at least the minimal physical requirements!

The probability is conserved!

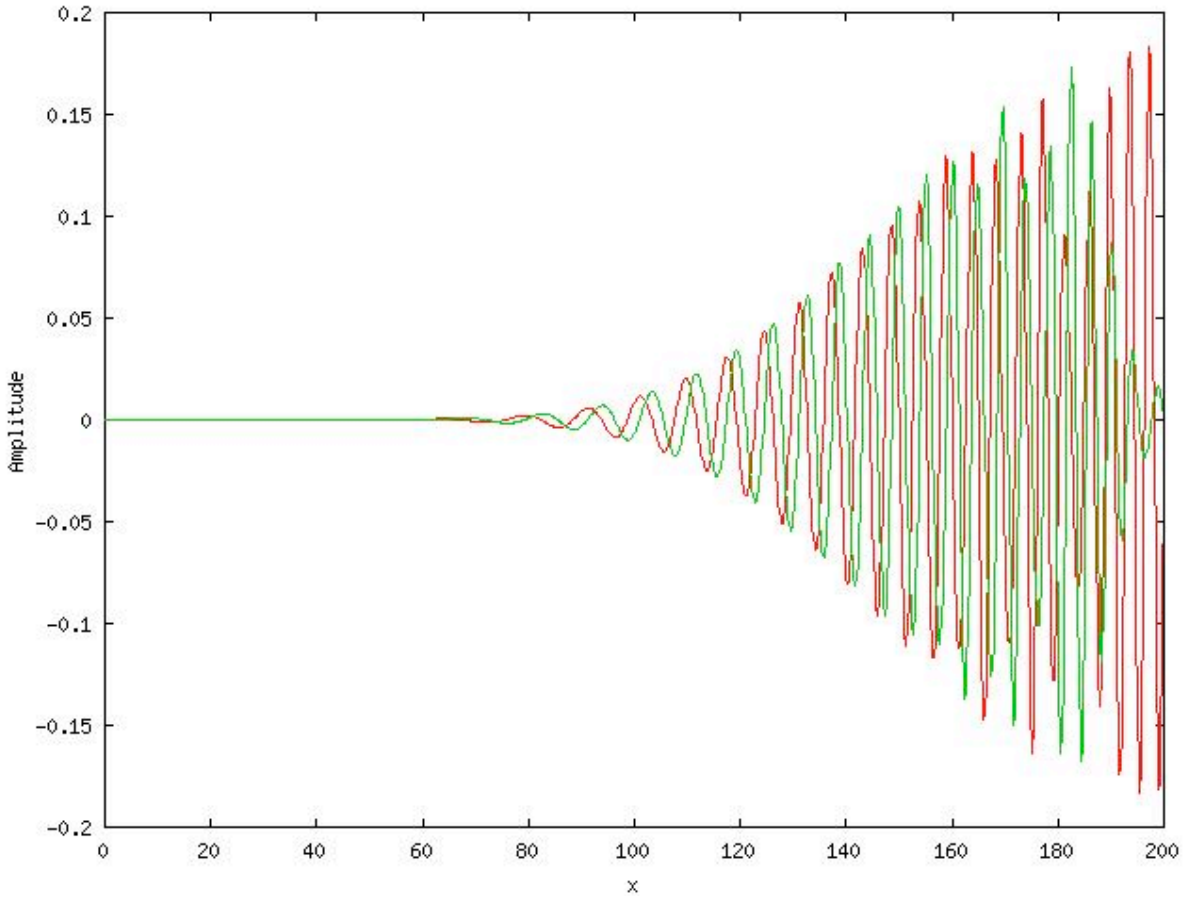


Spreading/reflecting of packet



Initial wave packet, real and imaginary parts

Spreading/reflecting of packet



Final packet, real and imaginary parts while reflecting, 1000 steps with dt=0.1… what is group velocity?

Group velocity..

I used k_0= sqrt(2.0).. Different from project...

In units we are using, s= k_0. So in 100 time units, the center of the wave packet should move 100*sqrt(2.0), or 141. Just guessing from figure, it starts and L/6 and is at about L, with L=200 in final frame, so this is about s=167… I could be more accurate for shorter times, packet spreading and reflecting makes it hard to be accurate.

Final packet, real and imaginary parts while reflecting, 1000 steps with dt=0.1… what is group velocity?