Numerical

```
! This version of the program models a gaussian pulse travelling along the
string
program wave
    real :: string_length, time_end, x_step, t_step, phase_vel ,lambda
    real :: init_pos, width
    !Here string_length is the ending point of string i.e. the length of
string
    !time_end is the final time
    !x_step is the step size of x
    !t_step is the step size of t
    !phase_vel is the constant in the 1D wave equation
    integer :: nt, nx
    !nt and nx are the number of steps in t and x respectively
    real, dimension(:,:), allocatable :: y
    !y will be containing the the displacement of the x at time t

    integer :: i, j
    !just two variable to move through the matrix

    write(*, '("The following program is to calculate the position of a
wave genrated by a streched string is space-time")')
    write(*, '("Using 1-D Wave Equation", /, 15X, "u_tt = phase_vel^2
u_xx", /, /)')
    write(*, '("Enter the length of the string : ")', advance = "no")
    read *, string_length
    write(*, '("Enter the final time t : ")', advance = "no")
    read *, time_end
    write(*, '("Enter the step size in x : ")', advance = "no")
    read *,x_step
    write(*, '("Enter the step size in t : ")', advance = "no")
    read *, t_step
    write(*, '("Enter the constant A : ")', advance = "no")
    read *, phase_vel
    ! Calculate the number of time and distance points
    nt = time_end/t_step + 0.5
    nx = string_length/x_step + 0.5
    print *, "nx = ", nx, ", nt = ", nt
    !Make the sure the value is 1 or less than 1 otherwise it will shoot
to infinity
    lambda = phase_vel*phase_vel*t_step*t_step/(x_step*x_step)
    print *,"Lambda = ", lambda
    ! Now the numerical calculation
    allocate(y(0:nx, 0:nt))
    ! The ends have to be at 0 or all times
    do j = 0, nt
        y(0, j) = 0
        y(nx, j) = 0
    end do
    ! Set the 1st initial values: y(x,0) = exp(-(x - x0)^2/a^2)
    ! We will start with x0 = L/2 i.e. halfway along the string
    init_pos = string_length/2
    width = string_length/5
    do i = 1, nx - 1
```

```
        y(i,0) = exp(-(i*x_step - init_pos)**2/width**2)
    end do
    ! For the second initial values the pulse has moved a distance v dt 
along the string
    init_pos = init_pos + phase_vel*t_step
    do i = 1, nx-1
        y(i,1) = exp(-(i*x_step - init_pos)**2/width**2)
    end do
    ! Now do the calculation
    do j = 1, nt-1
        do i = 1, nx-1
          y(i,j+1) = 2*y(i,j) +  lambda*(y(i+1,j) - 2*y(i,j) + y(i-1,j)) - 
y(i,j-1)
        end do
    end do
    open(19, file = "wavepulse.txt")
    do j = 0, nt
        do i = 0, nx
            write(19,'(F0.4, 1X, F0.4, 1x, F0.4)') x_step*i, y(i,j)
        end do
        write(19, '(/)')
    end do
    ! All finished so free the arrays we allocated
    deallocate(y)
end program wave
```

Analytical

```
program ana
    implicit none
    real :: string_length, time_end, x_step, t_step, phase_vel, 
lambda, initial_pos, width, g
    common initial_pos, width, phase_vel, string_length
    integer :: nt, nx, i, j
    real, dimension(:,:), allocatable :: ya

    write(*, '("The following program is to calculate the position 
of a wave genrated by a streched string is space-time")')
    write(*, '("Using 1-D Wave Equation", /, 15X, "u_tt = phase_vel^
2 u_xx", /, /)')
    write(*, '("Enter the length of the string : ")', advance = 
"no")
    read *, string_length
    write(*, '("Enter the final time t : ")', advance = "no")
    read *, time_end
    write(*, '("Enter the step size in x : ")', advance = "no")
    read *,x_step
    write(*, '("Enter the step size in t : ")', advance = "no")
    read *, t_step
    write(*, '("Enter the constant A : ")', advance = "no")
    read *, phase_vel

    initial_pos = string_length/2
    width = string_length/5
```

```
    nt = time_end / t_step + 0.5
    nx = string_length / x_step + 0.5
    lambda = phase_vel*phase_vel*t_step*t_step/(x_step*x_step)
    allocate(ya(0:nx, 0:nt))
    open(19, file = "wavepulse_a.txt")
    !Analytical Methos
    do j = 0, nt
        write(19,'(a,f0.4)') "# Time: ", j*t_step
        do i = 0, nx
            ya(i,j) = g(i*x_step, j*t_step)
            write(19, '(F0.4, 2X, F0.4)') i*x_step, ya(i,j)
        end do
        write(19,'(/)')
    end do
    close(19)

    deallocate(ya)
end program ana
real function g(x,t)
    implicit none
    real :: x, t
    real :: x2
    real :: initial_pos, width, phase_vel, string_length
    common initial_pos, width, phase_vel, string_length
    g = 0
    ! This is the right moving pulse
    x2 = -x + t*phase_vel + initial_pos
    ! x2 needs to be in the range -string_length < x2
< +string_length
    ! to simulate an infinite train of pulses
    x2 = mod(x2 + string_length, 2*string_length) - string_length
    g = g + exp(-(x2/width)**2)
    ! And we subtract the left moving pulse
    x2 = x + t*phase_vel - initial_pos - string_length
    ! x2 needs to be in the range -string_length < x2
< +string_length
    ! to simulate an infinite train of pulses
    x2 = mod(x2 + string_length, 2*string_length) - string_length
    g = g - exp(-(x2/width)**2)
    return
end function g
```