Department of Physics and Astronomy

PHYSICS ON PARALLEL COMPUTERS
Project Report

Scattering of Quantum Wave Packets

Peter Stefan Ruszczynski

Mai 1996

Abstract

The Time Dependent Schrödinger Equation is solved numerically in order to investigate the scattering of Quantum Wave Packets from various time independent potentials. In general such problems are not solvable analytically.

Supervisor: Dr.D.Richards

Contents

1 Introduction

The classical assumption of a particle as a point at a certain location is not longer valid in Quantum Mechanics. As some experiments at the change of this century had shown, there is rather a coexistence of a wave and a point particle, both describing the same physical object. This duality can not been understood using Newtons Laws, but by the assumption of the Quantum Mechanics.
The Quantum Mechanics assumes a wave function

$$\Psi(\vec{x}, t)$$

which is not the particle itself but an abstract object, the so called state function. This mathematical state function can be used to describe the probability density $P(\vec{x}, t)$ for the particle in $\vec{x}$ at time t:

$$P(\vec{x}, t) = \Psi^*(\vec{x}, t)\Psi(\vec{x}, t).$$

A result of the quantum mechanical formalism is the "Heisenberg principle of uncertainty", which means that either the position x or the momentum p can be known exactly only if we renounce to know anything about the other of these values. Furthermore it gives a relation for the uncertainties Δx and Δp.

$$\Delta x \Delta p \geq \frac{\hbar}{2}.$$

That is a principle, caused by the assumptions of Quantum Mechanics and not a result of our unaccurate techniques to measure anything.
Considering this spreading particle, one can assume an uncertain particle as a Gaussian distributed wave packet, which in two dimensions looks as follows:

$$\Psi(\vec{x}, 0) = \left(2\pi\sigma^2\right)^{-\frac{1}{2}} \exp\left(i\vec{k_0}.\vec{x} - \frac{(\vec{x} - \vec{x_0})^2}{4\sigma^2}\right), \tag{1}$$

with $\vec{k_0}$ as the initial wave vector. This distribution contains an infinite number of possible positions, each with an exact value of its probability. The Gaussian distribution is symmetric around its mean value $\vec{x_0}$. σ is the width of the distribution, which determines the range where the "biggest" part of the distribution is localised.One can calculate the mean value $\vec{x_0}$ a by taking

$$\langle\vec{x}\rangle_0 = \langle\Psi(\vec{x}, 0)|\vec{x}|\Psi(\vec{x}, 0)\rangle = \vec{x_0}.$$

Similar one can determine the mean value for the velocity and so proof the validity of the "Heisenberg principle of uncertainty" for the Gaussian distribution.

The Time Dependent Schrödinger Equation is the fundamental equation which describes the time evolution of a quantum mechanical system. It is an operator equation for the energy and takes the form:

$$\hat{E}\Psi(\vec{x}, t) = \hat{H}\Psi(\vec{x}, t). \tag{2}$$

Since the energy operator is

$$\hat{E} = i\hbar\frac{\partial}{\partial t}$$

and the Hamilton operator is

$$\hat{H} = \hat{T} + \hat{V} = -\frac{\hbar^2}{2m}\nabla^2 + \hat{V}(\vec{x})$$

the Schrödinger Equation takes the form

$$i\hbar\frac{\partial}{\partial t}\Psi(\vec{x},t) = \left(-\frac{\hbar^2}{2m}\nabla^2 + \hat{V}(\vec{x})\right)\Psi(\vec{x},t) \tag{3}$$

2 Analytical Solution: Zero Potential

For the zero potential, i.e. the free particle, one can solve the The Time Dependent Schrödinger Equation analytically for the Gaussian wave packet as the initial wave packet and hence determine the time evolution of the interesting values, like mean values and probabilities. To do that it is advisable to transform the equation into the Fourier space where the term of the kinetic energy does not longer contains second derivatives with respect to the position, but just a term containing k which has now only to be multiplied:

$$-\frac{\hbar^2}{2m}\nabla^2 \rightarrow -i\frac{\hbar k^2}{2m}.$$

The Schrödinger equation becomes the form:

$$\frac{\partial}{\partial t}\tilde{\Psi}(\vec{k},t) = -i\frac{\hbar k^2}{2m}\tilde{\Psi}(\vec{k},t).$$

Separating $\tilde{\Psi}$ in its time dependent and time independent part and integrating the time derivations gives the solution in Fourier space as

$$\tilde{\Psi}(\vec{k},t) = \exp\left(-i\frac{\hbar k^2}{2m}t\right)\tilde{\Psi}(\vec{k},0) + C,$$

where $\Psi(\vec{k},0)$ is the Fourier transformed Gaussian wave packet. By taking the inverse Fourier transform one gets the solution in ordinary space for two dimensions as :

$$\begin{aligned}\Psi(\vec{x},t) &= \frac{\sigma}{\sqrt{2\pi}\left(\frac{i\hbar t}{2m}+\sigma^2\right)} \\ &\quad \exp\left(\frac{1}{i\frac{\hbar t}{2m\sigma^2}+1}\left(i\vec{k_0}\vec{x} - \frac{(\vec{x}-\vec{x_0})^2}{4\sigma^2} - \frac{\hbar t}{2m\sigma^2}\vec{k_0}\vec{x_0} - i\frac{\hbar t}{2m\sigma^2}\vec{k_0}^2\sigma^2\right)\right)\end{aligned} \tag{4}$$

This analytic solution is exact and therefore the best opportunity to check the reliability of the computer simulation program. A lot of other solutions of the Time Dependent Schrödinger Equation , especially those including a potential, can be found only numerically.
For the above analytical result one can now compute the probability density P in order to get another reliability check of the program:

$$\begin{aligned}P(\vec{x},t) &= \Psi^*(\vec{x},t)\Psi(\vec{x},t) \\ &= \left(2\pi\sigma^2\left(1+\left(\frac{\hbar t}{2m\sigma^2}\right)^2\right)\right)^{-1}\exp\left(-\frac{\left(\vec{x}-\left(\vec{x_0}+\frac{\hbar}{m}\vec{k_0}t\right)\right)^2}{2\sigma^2\left(1+\left(\frac{\hbar t}{2m\sigma^2}\right)^2\right)}\right).\end{aligned} \tag{5}$$

This is again a Gaussian distribution, with the width

$$\sigma\left(1+(\frac{\hbar t}{2m\sigma^2})^2\right)^{\frac{1}{2}}, \tag{6}$$

which obviously increases in time.
Another interesting value is the mean value of x and its time evolution.

$$\langle\vec{x}\rangle_t = \langle\Psi(\vec{x},t)|\vec{x}|\Psi(\vec{x},t)\rangle = \vec{x_0} + \frac{\hbar}{m}\vec{k_0}t. \tag{7}$$

Later on, these values will be compared with the simulated ones.

3 The Numerical Algorithm

The algorithm for integrating the Schrödinger Equation (3) is taken from John Richardson [2] and is called "Space Splitting Algorithm". It is an example for an "Operator Splitting Algorithm", which is in general an often used method. This algorithm satisfies the conditions to be stable (unitary) and reasonable for computational time. A short explanation for the "Space Splitting Algorithm" is given by the following:
In general the solution of the Schrödinger Equation (3) is:

$$\Psi(\vec{x},t) = \exp\left(-i\frac{t}{\hbar}\hat{H}\right)\Psi_0(\vec{x}),$$

where the time evolution operator can be written as:

$$\exp\left(-i\frac{\Delta t}{\hbar}\hat{H}\right) = \exp\left(-i\frac{\Delta t}{\hbar}\hat{T}\right)\exp\left(-i\frac{\Delta t}{\hbar}\hat{V}\right) + \mathcal{O}\left(\left[\frac{\Delta t}{\hbar}\hat{H}\right]^2\right). \tag{8}$$

The terms of $\mathcal{O}\left(\left[\frac{\Delta t}{\hbar}\hat{H}\right]^2\right)$ vanish only if $\hat{T}$ and $\hat{V}$ commute: $\left[\hat{T},\hat{V}\right]=0$. This can be seen by expanding $e^{\hat{T}+\hat{V}}$ and $e^{\hat{T}}e^{\hat{V}}$:

$$\begin{aligned} e^{\hat{T}+\hat{V}} &= 1+\hat{T}+\hat{V}+\frac{\hat{T}^2}{2}+\frac{\hat{T}\hat{V}}{2}+\frac{\hat{V}\hat{T}}{2}+\frac{\hat{V}^2}{2}+\cdots \\ e^{\hat{T}}e^{\hat{V}} &= 1+\hat{T}+\hat{V}+\frac{\hat{T}^2}{2}+\hat{T}\hat{V}+\frac{\hat{V}^2}{2}. \end{aligned} \tag{9}$$

Considering the Laplacian operator as the "stencil", the kinetic energy in one dimension can be written in matrix form:

$$\hat{T} = \frac{\hbar^2}{2m}\left(\frac{1}{\Delta x}\right)^2\begin{pmatrix} 2 & -1 & 0 & . & . & 0 & -1 \\ -1 & 2 & -1 & . & . & 0 & 0 \\ 0 & -1 & 2 & -1 & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ -1 & 0 & . & . & 0 & -1 & 2 \end{pmatrix},$$

where the number of the rows and columns are the site index. Richardson's idea is to express $\hat{T}$ in terms of two matrices which remain sparse on exponentiation: $\hat{T} = \hat{T}^{\text{even}} + \hat{T}^{odd}$, where

$$\hat{T}^{\text{even}} = \frac{\hbar^2}{2m}\left(\frac{1}{\Delta x}\right)^2 \begin{pmatrix} 1 & -1 & 0 & . & . & 0 & 0 \\ -1 & 1 & 0 & . & . & 0 & 0 \\ 0 & 0 & 1 & -1 & . & 0 & 0 \\ . & . & -1 & 1 & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & 0 & 1 & -1 \\ 0 & 0 & . & . & 0 & -1 & 1 \end{pmatrix}$$

and

$$\hat{T}^{\text{odd}} = \frac{\hbar^2}{2m}\left(\frac{1}{\Delta x}\right)^2 \begin{pmatrix} 1 & 0 & 0 & . & . & 0 & -1 \\ 0 & 1 & -1 & . & . & 0 & 0 \\ 0 & -1 & 1 & 0 & . & 0 & 0 \\ . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & 1 & -1 & 0 \\ 0 & 0 & . & . & -1 & 1 & 0 \\ -1 & 0 & . & . & . & 0 & 1 \end{pmatrix}$$

These matrices can now be identified as direct sums of

$$\hat{M} = \frac{\hbar^2}{2m}\left(\frac{1}{\Delta x}\right)^2 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

for which

$$\exp\left(-i\frac{\Delta t}{\hbar}\hat{M}\right) = \begin{pmatrix} \alpha & \beta \\ \beta & \alpha \end{pmatrix}$$

where

$$\alpha = \frac{1}{2}(1 + e^{-i\epsilon})$$

$$\beta = \frac{1}{2}(1 - e^{-i\epsilon})$$

with

$$\epsilon = \Delta t\, \hbar / m\, \Delta x^2.$$

Hence one can write the kinetic energy terms for equation (8) as

$$\begin{aligned} &(\exp(-i\frac{\Delta t}{\hbar}\hat{T}^{\text{even}})\psi)(n) = \\ &\quad \alpha\psi(n) + \beta\frac{1-(-1)^n}{2}\psi(n-1) + \beta\frac{1+(-1)^n}{2}\psi(n+1) \\ &(\exp(-i\frac{\Delta t}{\hbar}\hat{T}^{\text{odd}})\psi)(n) = \\ &\quad \alpha\psi(n) + \beta\frac{1+(-1)^n}{2}\psi(n-1) + \beta\frac{1-(-1)^n}{2}\psi(n+1). \end{aligned}$$

This formalism can be extended to two dimensions by considering the x- and y-directions separately:

$$\Psi(\vec{x}, t + \Delta t) = \exp\left(-i\frac{\Delta t}{\hbar}\hat{T}_x^{\text{even}}\right)\exp\left(-i\frac{\Delta t}{\hbar}\hat{T}_x^{\text{odd}}\right)\exp\left(-i\frac{\Delta t}{\hbar}\hat{T}_y^{\text{even}}\right)\exp\left(-i\frac{\Delta t}{\hbar}\hat{T}_y^{\text{odd}}\right)\exp\left(-i\frac{\Delta t}{\hbar}\hat{V}\right)\Psi(\vec{x}, t),.$$

4 The Simulation

4.1 General Aspects

The above algorithm is used by a Computer program written in CM Fortran. The CM Fortran language is an implementation of Fortran 77 supplemented with array-processing extensions from the ANSI and ISO standard Fortran 90. These array-processing features map naturally onto the data parallel architecture of the Connection Machine (CM) system, which is designed for computations on large data sets. In particular a CM-200 parallel supercomputer of "Thinking Machines Corporation" was used for the simulation.
The two dimensional simulation runs on a 256 x 256 array, using periodic boundary conditions. Periodic boundary conditions are useful, because a wave packet travelling along the x-direction can be observed over a greater distance as 256 grid points, assumed that no part of the wave packet overlap with the rest, which would cause unphysical interferences. In order to avoid unphysical interferences one has to choose the width σ of the initial wave packet not too large and consider the fact that this width spreads out in time, according equation (6).

4.2 Dimensionless Variables

For reasons of computational effort the variables can transformed into dimensionless variables:

$$\begin{aligned} t &= t_{dimless}\ T \\ x &= x_{dimless}\ X \\ m &= m_{dimless}\ M \end{aligned} \tag{10}$$

In this simulation I have chosen $\hbar = 1.0$ and $m = 1.0$. One can get the SI - units of values like energy, etc by considering T, X and M as the new units in the transformation (10). That means that one can deal with dimensionless variables and determine the dimension later on. Because I have chosen in this simulation the spacing between the grid points $\delta x = 1.0$, all the following value for positions, etc. are given in grid points.

4.3 Visualization of the Wave Packet

The routines for visualization of the simulated system are taken again from Richardson [2] and included with the file "pict_subs.h".
The routines transform the amplitude of the wave into the intensity of a certain

colour, which is determined by the phase. Furthermore, zero amplitudes appear black and infinite amplitudes would appear white. A purely real wave is presented red.

4.4 Program Features

The program is written to simulate the scattering of a wave packet from the following potentials:

1. **Zero Potential:** Corresponds to a free particle.
2. **Sharp cylindrical potential:** The radius, position and extreme value (positive or negative) are to be chosen.
3. **Potential Barrier:** The Barrier is rectangular to the x-direction, the height (positive or negative) can be chosen as well as the width and the position.
4. **Soft Potential Barrier:** This potential is similar to the Potential Barrier from above, but before and behind the barrier is a linear gradient over 20 grid points from zero to the maximal potential value.
5. **Harmonical potential:** This potential has a parabolical shape. Figure (5.5) shows the version for positive potential values. The mirrored case can also be chosen. The shape is determined by the radius and the extreme value. The potential can be placed anywhere inside the grid.
6. **Slots:** This is almost a barrier, but interrupted by slots. The number of slots, their width, and the thickness of the remaining parts of the barrier are to be chosen.

The parameters of the initial wave have to be entered as well. These are the width σ, the initial position $\vec{x_0}$ and the initial wave vector $\vec{k_0}$.
In order to get some quantitative values, either to control the reliability of the program or to get some interpretable results, the following values can be printed to files:

- expected values for x and y over iteration steps: $\langle\Psi|x|\Psi\rangle$ and $\langle\Psi|y|\Psi\rangle$
- expected values for the kinetic and potential energy over steps: $\left\langle\Psi|\hat{T}|\Psi\right\rangle$ and $\left\langle\Psi|\hat{V}|\Psi\right\rangle$
- expected value for the total energy over steps: $\left\langle\Psi|\hat{E}|\Psi\right\rangle$
- the probability to find Ψ inside the whole grid over steps: $\langle\Psi|\Psi\rangle$

If required, these values are printed into files every 10th steps. The output to the screen consists of a print of some control values and a display of the wave packet (if desired with potential). The graphs can then be saved into files by performing a X Window Dump. The values for the potential along $y = 128$ (half the grid) can be printed to a file as well, in order to get a control over the right shape of the potential at the start of the run. The source text of the program is given in appendix B.

5 Results & Discussion

5.1 Reliability of the Program: Zero Potential

The analytical results of section (2.2) give an opportunity to show, if the simulation really reflects the physical nature.

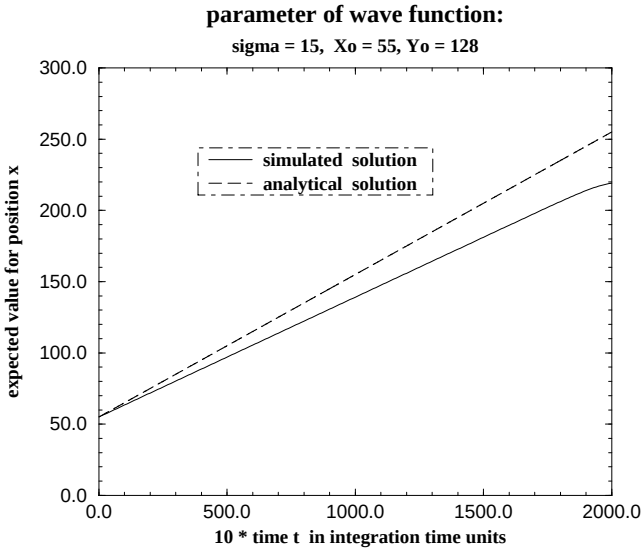


Figure 1: The expectation value for x shows a systematical error in the simulation.

- **Spreading of the Wave Packet:** One can observe a spreading of the wave packet. That means that the width σ of the Gaussian packet increases in time, which corresponds to equation (6).

- **Expectation value for x:** Figure (1) shows the simulated values, compared with the analytical ones. The first is produced, using the corresponding function of the program , whereas the latter is determined by equation (7). The reason for the systematical deviation is explained in the next item.

- **Expectation value for the total energy:** Assuming the applicability of the energy equation for a single wave ,the expectation value for the total energy should be $k_{x0}^2/2 = 0.5$ for $\vec{k_0} = (1.0, 0.0)$, because the potential is zero everywhere and $m = \hbar = 1.0$.
  Figure (2) shows a value $E_{tot} = 0.4608 \pm 0.0004$, which is obviously too small. The reason is, that waves with a wave length $\lambda < \delta x$ can not be considered by the calculation for the energy, since they lie "between" two neighbouring grid points. Hence the wave packet has a lower velocity, what explains the deviation in figure (1). The symmetric deviation of the energy around its mean value over a range of $1\ {}^{\circ}/_{\circ\circ}$ in figure (2) is also supposed to be caused by the discretisation of the wave on the grid.
  Since the total energy is reasonable constant, the chosen parameter can be used for the rest of the simulation. But one has always to remind the lest energy.

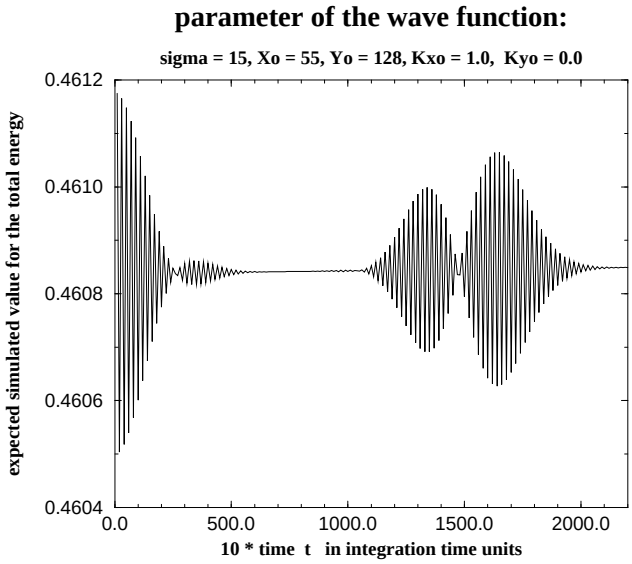


Figure 2: The expectation value for E_{tot} shows a systematical deviation from the analytical value 0.5 .

5.1.1 Initial Values

If not other mentioned, the following initial values and parameters are chosen in the rest of the following discussion.

$$\vec{x_0} = (55, 128) \; ; \quad \vec{k_0} = (1.0, 0.0) \; ; \quad \sigma = 15 \; ; \quad \delta x = 1.0 \; ; \quad \delta t = 0.1 \; .$$

5.2 Sharp Cylindrical Potential

As a first application of the simulation, a sharp, cylindrical potential is considered:

$$V(\vec{x}) = \begin{cases} V_0 & : |\vec{x} - \vec{x_0}| < r \\ 0 & : |\vec{x} - \vec{x_0}| > r \end{cases} \quad .$$

$\vec{x_0}$ is the position of the centre of the cylinder and r the radius. In particular I have chosen a radius $r = 10$ grid points and a maximal value of $V_0 = 10$. The resulting time evolution of the wave function can be seen as a sequence of pictures in figure (5).
As the wave moves along the x-direction with an initial kinetic energy $\hat{T} = 0.5$ it gets in contact with the cylinder and scatters. Some phenomena can be observed here:
In picture (5.c) one can see interferences shortly before the potential. The wave packet gets reflected here and interferes with itself. The reflection continues till the whole wave packet moves away from the potential.
In general one can draw the conclusion that the wave packet shuns the potential in future, spreading out as a ring around the potential.
Important to note is the region in front of the packet, where the probability density is higher then anywhere else. This region is separated from the rest of the ring by

negative interferences.
The bigger the radius of the cylinder the bigger the amount of the wave packet that gets reflected. In case of the kinetic energy being greater than the potential, the most part of the packet passes the potential easily. No full ring is performed, as can be seen in figure (6).
Figure (7) presents the case of an attractive potential. The absolute value of the potential is less then the kinetic energy of the packet. Basically, the probability density is distributed as in the case before, but remarkable is that still a part of the packet gets reflected. I will come back to this point later on.

5.2.1 Born Approximation

The first Born Approximation arises from the first term in the perturbation series and is valid only for a weak interaction potential. It gives a formula for the cross section:

$$\left(\frac{d\sigma}{d\Omega}\right) = const \quad \frac{1}{q^2}, \tag{11}$$

where

$$q = \sin(\Theta/2).$$

The angle Θ gives the deviation from the direction of the incoming motion. Note that σ here is not the width of the wave packet!
From equation (11) follows, that at an angle of $\Theta = 180°$, i.e. opposite to the direction of the incoming motion, the cross section has its minima. That can clearly be seen in picture (6.e). This holds not for the case in figure (5), because the interaction potential is not anymore weak. A Partial Wave Analysis should be done in that case.

5.3 Potential Barrier

Figure (8) shows the reflection of a wave packet by a potential barrier of height $V_0 = 1.0$. After the packet is squeezed to the half and interferes with itself, it gets reflected and looks identical to its state before.
If the barrier is less then the kinetic energy $(V_0 < T)$ it can pass the barrier, exactly as one would expect. A part of the packet gets reflected when the packet enters the potential region, but again some part gets reflected when the packet leaves this region, i.e. by a negative potential step.
An example that shows clearly the reflection by a negative potential step is given in figure (9). Although the potential barrier has a value $V = -1.0$, a part of the wave packet gets reflected. This is not understandable classically. A way to understand this, is to calculate the reflection coefficient R for a plane wave at a potential step through the Schröedinger equation. This has been done in [1]:

$$R = \left(\frac{k_0 - k}{k_0 + k}\right)^2, \tag{12}$$

where $k = \frac{2m}{\hbar^2}(E - V_0)$.
It is obvious that the sign of k does not matter, i.e. the reflection coefficient is the same for a wave, coming from the higher level of the potential step, as one coming

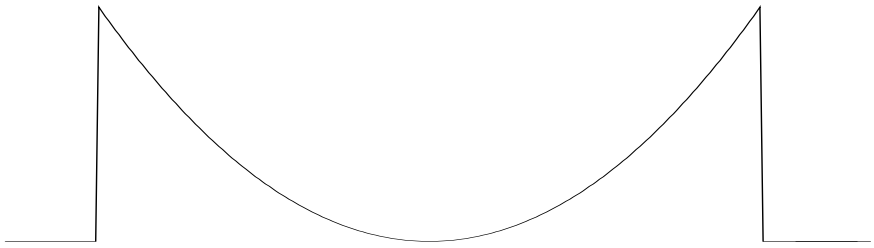
from the lower level. Since a wave packet can be understood as an ensemble of waves, the qualitative behaviour should be analogous.
The reflection inside a negative potential barrier can be seen as infinite, since a part of every reflected wave gets reflected again by the other side of the barrier. Figure (9) gives a rough idea of it.
Another interesting phenomena is the so called "Tunnelling effect". This means basically that a wave with an energy E can cross a potential barrier with $V_0 > E$. Figure (10) shows such a situation. This effect is bigger for thin walls and for small differences $V_0 - E$. Clearly to verify is the separation of the wave packet into two parts of nearly the same size, caused by the tunneling effect.

5.4 Soft Potential Barrier

A barrier as above can be called "hard barrier", because the potential jumps up from zero to V_0. Figure (11) shows what is different, when the gradient of the potential is rather smooth. Such barriers are called "soft". The soft barrier, used here, is basically a hard barrier, where a linear slope is added on both sides. As figure (11) indicates, the reflection is decreased and the wave packet seems to pass the barrier completely. The locations of the potential barriers are the same as for example in figure (8); both pictures are taken at time $t = 200$.

5.5 Harmonical Potential

A circular symmetrical potential as the following has been considered:



The centre of the potential is located at $\vec{x} = (128, 128)$ and its radius is 100. A wave packet was placed approximately half way up $\vec{x} = (65, 128)$ and had no initial velocity, i.e. $\vec{k} = 0$. The initial potential energy causes the packet to oscillate, during the packet itself starts to spread. At the start of the oscillation, the distribution of the packet was rather localised by quite a small $\sigma = 5$. Figure (3) shows the expectation value for x over the time.
As time proceeds the distribution spreads more and more in the direction of the oscillation and the expectation value tends to the bottom of the potential at $\vec{x} = (128, 128)$. This is caused by the symmetry of the potential and the spreading of the packet itself and not because the packet came to rest.

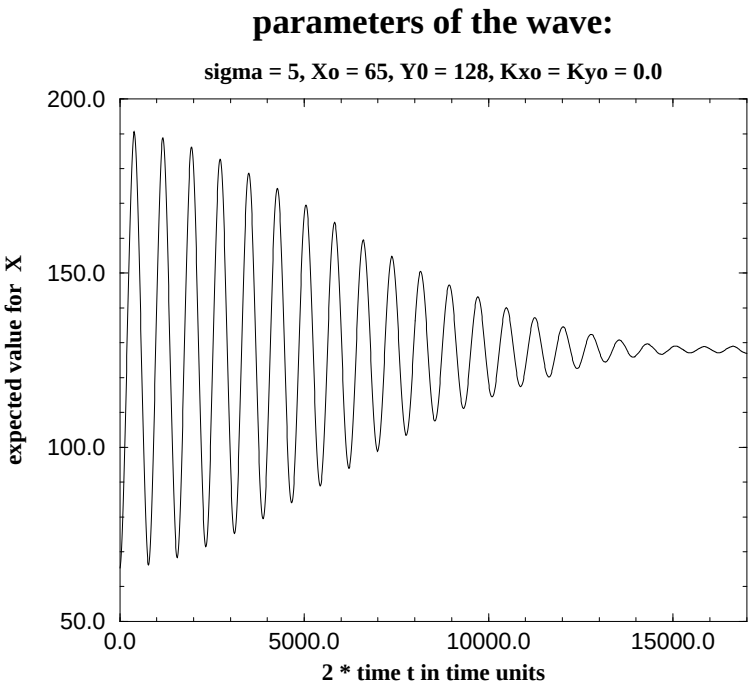


Figure 3: harm expect x

5.6 Slots

This potential model consists of vertical stripes, which are arranged in equal distances in the y-direction. If the wave packet is scattered by such a potential, some pieces of the wave packet interfere with each other. The closer the pieces are to the direction of the incoming motion, the more they will interfere with each other. See figure (12).
The probability of passing the potential depends on the width between the stripes. As figure (4) indicates, there is a critical value for the "slot width", at which the probability for a particle to be reflected is the same as for one to be transmitted.

6 Conclusions

Different Phenomena have been observed:

- Reflection occurs for all kinds of potentials, although they are negative.
- Interference can be obtained if scattered wave parts overlap each other.
- A Soft Barrier can be easier passed by a wave packet then a hard one.
- The Tunneling effect can be obtained if the potential is higher than the energy of the packet.

Phenomena like the Tunneling Effect and the reflection by a negative potential steps are purely quantum mechanical effects.

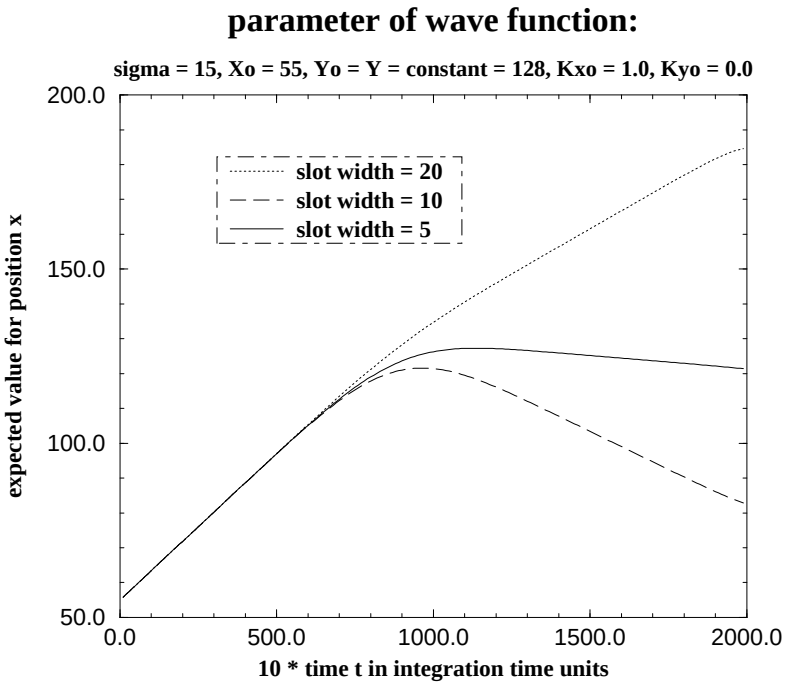


Figure 4: slotexp

Even though an analytical solution is not always possible, a quantum mechanical system can be studied numerically. Since it is possible to get quantitative results, it is worth to make computational experiments for theoretical research as well as to reconsider experimental results. This is in accordance to the actual development of the third "main stream" in physics, the **Computational Physics**.
Experiments, which are inefficient or difficult for any reason, might be substituted by computational experiments.
The benefit for the Theoretical Physics can consist of verifications of theories as well as finding phenomena, which are still unknown. A basic premise to do that are efficient algorithms like the space splitting algorithm and fast computers. The latter can be reached not only by fast processors, but also by the architecture of the machines, such as **parallelism**.

7 References

[1] Nolting, W., Grundkurs: Theoretische Physik, 5.1/2. Quantenmechanik, Zimmermann-Neufang Verlag, Ulmen, (1994)

[2] Richardson, J.L., Visualising quantum scattering on the CM-2 supercomputer, Computer Physics Communication **63**, page 84-94, (1991)

A Graphics

The following two pages contain sequences of picture for different cases. The time at which the pictures are taken are always the same (except figure(**??**):
time t: a) $\rightarrow 0$; b) $\rightarrow 50$; c) $\rightarrow 100$; d) $\rightarrow 150$; e) $\rightarrow 200$.
The shape and values of the potentials are given in each figure. For the initial values see section 5.1.1 .

B Program Source Text

```
C-------------------------------------------------------------
C    QUANTUM WAVE PACKET SCATTERING
C
C    APRIL 96,  EDINBURGH, SCOTLAND
C
C    PHYSICS ON PARALLEL COMPUTERS - PROJECT
C
C    PETER RUSZCZYNSKI
C
C-------------------------------------------------------------



      implicit none

      include 'pict_subs.h'

      real, parameter:: pi = 3.1415927
      real, parameter:: mass = 1.0
      real, parameter:: h_bar= 1.0
      integer, parameter:: bits = 8
      integer, parameter:: length=256          ! latice length
      integer i, ii, j, step
      character decide
      real Xo, Yo, Ko_x, Ko_y                  ! initial wave values
      real sigma,t, t_step,  x_unit            ! values for wave packet
      real  kin_energy,pot_energy, psisquare   !  functions
      real expected_x, expected_y              !  functions
      complex  alpha, beta
      logical A, B, C, D, E, F, G, H
      complex, dimension(length,length):: psi, pot, e_pot   ! wave & pot.
      complex, dimension(length,length):: betaeven, betaodd
      complex, dimension(length,length):: visual           ! display array


!_for potential_____________
      character choice
      real pot_value
      integer  help1, help2, help3, help4, help5
      real help6
      logical, dimension(length, length):: zyl
!_for plot__________________
      integer  rnx, rny, nxp, nyp, display


C_____ALL ARRAYS ARE LOCATED ON PARALLEL PROCESSORS_________________
```

```
CMF$    layout PSI(:news, :news)
CMF$   layout POT(:news, :news)
CMF$    layout betaeven(:news, :news)
CMF$    layout betaodd(:news, :news)
CMF$    layout e_pot(:news, :news)
CMF$    layout visual(:news, :news)



C___INITIALISATION: PARAMETERS FOR INTEGRATION, POTENTIAL AND WAVE____

 8    H = .false.
      A = .false.
      B = .false.
      C = .false.
      D = .false.
      E = .false.
      F = .false.
      G = .false.

      write(*,*) ' '
      write(*,*)'_____________________________________________________'
      write(*,*)'TIME STEP dt: '
      read(*,*) t_step
      write(*,*)'SPACE UNIT dx: '
      read(*,*) x_unit

      write(*,*)'A QUANTUM WAVE PACKET WILL BE SCATTERED AT A'
      write(*,*)'POTENTIAL OF YOUR CHOISE. ENTER YOUR LETTER:'
 16   write(*,*)' '
      write(*,*)'  <Z>YLINDRICAL POTENTIAL'
      write(*,*)'               <B>ARRIER '
      write(*,*)'  SO<F>T         BARRIER '
      write(*,*)'  <S>LOT       POTENTIAL '
      write(*,*)'  <C>IRCLE       BARRIER '
      write(*,*)'  <H>ARMONIC    POTENTIAL'
      write(*,*)'  <N>O          POTENTIAL'

      read(*,*) choice

      if   (choice.ne.'N'.and.choice.ne.'S'.and.choice.ne.'B'
     & .and.choice.ne.'Z'.and.choice.ne.'H'.and.choice.ne.'F'
     & .and.choice.ne.'C')         then
      write(*,*)'!!!  C A P I T A L    L E T T E R S  !!!'
      goto 16
      end if
```

```
!-------------------------------------------------------------------
!----------CONFIGURATION OF THE POTENTIAL---------------------------


      pot = (0.0,0.0)

      if (choice.ne.'N') then
         write(*,*)'ENTER EXTREME VALUE FOR THE POTENTIAL (! sign !):'
         read(*,*) pot_value
      else
         goto 32
      end if


C------HARMONIC POTENTIAL-----------------------------

      if (choice.eq.'H') then
       write(*,*)'ENTER POTENTIAL CENTRE:'
       read(*,*) help1, help2                    ! centre_x, centre_y
       write(*,*)'ENTER RADIUS:'
       read(*,*) help6                           !   radius

       forall(i=1:length,j=1:length)
     &  pot(i,j) = ((sqrt(abs(pot_value))/help6 * (i-help1))**2. +
     &              (sqrt(abs(pot_value))/help6 * (j-help2))**2. -
     &              abs(pot_value)) * pot_value/abs(pot_value)*(-1)

       if (pot_value.gt.0)   then
         where (real(pot).lt.0)   pot = 0.0
       end if
       if (pot_value.lt.0)  then
         where(real(pot).gt.0)     pot = 0.0
        end if

       where (real(pot).ne.0.0) pot = pot - pot_value

       where (real(pot).ne.0.0)
         zyl = .true.
       elsewhere
         zyl = .false.
       end where

      end if

C-------SHARP ZYLINDRICAL POTENTIAL------------------------

      if (choice.eq.'Z') then
        write(*,*)'ENTER POTENTIAL CENTRE :'
```

```
        read(*,*) help1, help2                        !  centre_x, centre_y
        write(*,*)'ENTER RADIUS:'
        read(*,*) help3                               !  radius

       forall(i=1:length,j=1:length)
     &  zyl(i,j) = (((i-help1)**2. + (j-help2)**2.).le.help3**2.0)

        where (zyl)  pot = pot_value

       end if


C_______BARRIER PARALLEL Y-AXIS__________________________

       if (choice.eq.'B') then
 24     write(*,*)'ENTER X_START AND X_END VALUES FOR BARRIER :'
        read(*,*) help1, help2       ! barrier_start, barrier_end
        if ((help1.ge.help2).or.(max(help1,help2).gt.length)) goto 24

        forall(i=help1:help2)        POT(i,:) = pot_value

       end if

C______SOFT BARRIER PARALLEL Y-AXIS_____________________

       if (choice.eq.'F') then
 28     write(*,*)'ENTER X_START AND X_END VALUES FOR BARRIER :'
        write(*,*)'     !!!! (X_END - X_START) > 3 !!!!'
        read(*,*) help1, help2       ! barrier_start, barrier_end
        if ((help1.ge.help2).or.(max(help1,help2).gt.length)) goto 28

        forall(i=help1:help2)        POT(i,:) = pot_value

        forall(i=help1-10:help1+10)
     &           pot(i,:) = (1.-(help1+10-i)*0.05)*pot_value
        forall(i=help2-10:help2+10)
     &           pot(i,:) = (1.-(i-(help2-10))*0.05)*pot_value


       end if

C_______SLOTS_____________________________________

       if (choice.eq.'S')  then
         write(*,*)'ENTER NUMBER OF SLOTS :'
         read(*,*) help1              !  number of slots
         write(*,*)'ENTER WIDTH OF SLOT (ALL ARE THE SAME)'
         read(*,*) help2
```

```
         write(*,*)'ENTER X_START AND X_END OF ''WALL'':'
         read(*,*) help3, help4

         forall(i=help3:help4,j=1:length)  pot(i,j) = pot_value

         do ii=1,help1
           forall(i=help3:help4,j=ii*((length-help1*help2)/(help1+1))
     &           + (ii-1)*help2 + 1 : ii*(((length-help1*help2)/
     &             (help1+1))+help2))                pot(i,j) = 0.0
         end do
        end if

C_______CIRCLE BARRIER___________________________________

       if (choice.eq.'C')    then
         write(*,*)'ENTER CENTRE OF CIRCLE'
         read(*,*) help1, help2
         write(*,*)'ENTER INNER AND OUTER RADIUS'
         read(*,*) help3, help4

        forall(i=1:length,j=1:length)
     &  zyl(i,j) = (((i-help1)**2. + (j-help2)**2.).le.help4**2.0)

        where (zyl)  pot = pot_value

        forall(i=1:length,j=1:length)
     &    zyl(i,j) = (((i-help1)**2. + (j-help2)**2.).le.help3**2.0)
        where (zyl)  pot = 0.0

       end if
C_____________________________________________________________________


C______PLOT POTENTIAL DATA TO FILE FOR EXTERNAL VISUALISATION_______

         write(*,*)'DO YOU WHISH EXTERNAL PLOT OF POTENTIAL <n> ?'
         read(*,*) decide
         if (decide.ne.'n')  then
            H = .true.
            open(unit=10,file='pot.dat',status='unknown')
            do i=1,length
                write(10,*) i, real(pot(i,128))
            end do
         end if

c______PLOT SOME POTENTIAL DATA FOR SCREEN CONTROL___________________

      write(*,*)'          X    potential  '
```

```
      do i=1,length,10
        write(*,*) i, real(pot(i,128))
      end do

        write(*,*)'IS EVERYTHING SATISFYING YOU , <Y> OR <n> ???'
        read(*,*) decide
        if (decide.eq.'n')  goto 72

C______SHOW DISPLAY WINDOW_____________________________________________

 32     call init_display(length, length, bits)

C_______VALUES FOR INITIAL WAVE_____________________________________

        write(*,*)' '
        write(*,*) 'YOU WILL USE A GAUSSIAN WAVE PACKET !!!'
write(*,*) 'ENTER STANDARD DEVIATION (SIGMA):'
read (*,*) sigma
           sigma = x_unit * sigma
write(*,*) 'ENTER INITIAL CENTRE OF WAVE PACKET (Xo, Yo)'
read (*,*) Xo, Yo
            Xo = x_unit * Xo
            Yo = x_unit * Yo
write(*,*) 'ENTER START WAVE VECTOR(K0_x,K0_y)'
read (*,*) Ko_X, Ko_Y


C_____SET INITIAL WAVE_________________________________________________

forall(i=1:length,j=1:length)  psi(i,j) =
     &      1.0 / (sqrt( 2.0 * pi )* sigma)
     &  * exp( (0.,1.) * x_unit * (Ko_X * i + Ko_Y * j) )
     &  * exp( -(((x_unit * i - Xo)**2.0
     &  + (x_unit * j - Yo)**2.0) / (4.0*sigma**2.0)))

            !!! INITIAL WAVE IS SCALED WITH X_UNIT !!!


C_______DISPLAY INITIAL WAVE (PSI) AND POTENTIAL ____________________

        visual = psi
        where (pot.ne.0.0)  visual = maxval(abs(psi))

call display_psi(visual/maxval(abs(psi)), length, length)

write(*,*) 'psi square = ', psisquare(length,psi,x_unit)

        write(*,*)'IS EVERYTHING SATISFYING YOU , <Y> OR <n> ???'
```

```
        read(*,*) decide
        if (decide.eq.'n')  goto 72

C_____CALCULATE ALPHA AND BETA FOR "RICHARDSON ALGORITHMUS"__________

  alpha = 0.5 * (1+exp( (0.0,-1.0) * t_step * h_bar /
     &       (mass * x_unit**2.0)) )
  beta  = 0.5 * (1-exp( (0.0,-1.0) * t_step * h_bar /
     &       (mass * x_unit**2.0)) )


C_____BETA MASKS___________________________

        forall(i=1:length,j=1:length)
     &      betaeven(i,j) = beta * (1.-(-1.)**(i+j))/2.
        forall(i=1:length,j=1:length)
     &      betaodd(i,j) = beta * (1.+(-1.)**(i+j))/2.


C_____HELP VALUE FOR SUBROUTINE_________

      e_pot =  exp( (0.0,-1.0) * t_step * pot / h_bar)

        step = 0

C_____SEVERAL VALUES CAN BE PRINTED TO FILES___________________________

        write(*,*)' '
        a = .true.
        write(*,*)'wish data printed to file  <n> ?'
        read(*,*) decide
        if (decide.ne.'n')   then
           write(*,*)' expected x over step <n> ???'
           read(*,*) decide
           if (decide.ne.'n') A = .true.
           write(*,*)' expected y over step <n> ???'
           read(*,*) decide
           if (decide.ne.'n') B = .true.
            write(*,*)' kin. Energy over step <n> ???'
           read(*,*) decide
           if (decide.ne.'n') C = .true.
           write(*,*)' pot Energy over step <n> ???'
           read(*,*) decide
           if (decide.ne.'n') D = .true.
           write(*,*)' total Energy over step <n> ???'
           read(*,*) decide
           if (decide.ne.'n') E = .true.
           write(*,*)' psi**2 over step <n> ???'
```

```
          read(*,*) decide
          if (decide.ne.'n') F = .true.
          write(*,*)' for circle only: psi**2 inside circle over step'
          read(*,*) decide
          if (decide.ne.'n') G = .true.
         end if
         if (A) open(unit=20,file='expect_x.dat',status='unknown')
         if (B) open(unit=30,file='expect_y.dat',status='unknown')
         if (C) open(unit=40,file='ekin.dat',status='unknown')
         if (D) open(unit=50,file='epot.dat',status='unknown')
         if (E) open(unit=60,file='etot.dat',status='unknown')
         if (F) open(unit=70,file='abs.dat',status='unknown')
         if (G) open(unit=80,file='decay.dat',status='unknown')
        write(*,*) ' '
        write(*,*)'<S T A R T >'
        read(*,*)

!-----------------------------------------------------------------------
C________M A I N P A R T_____________________________________________

        do while(1.le.5)

          step = step + 1
          t = t + t_step

  call psi_update(length,alpha,betaeven,betaodd,psi,e_pot)



C______RUN MANIPULATION_________________

          if (mod(step,500).eq.0) then

 64        write(*,*)'<s>how with potential ? ; continue the computation
     &      , <y>es or <n>o   ???'
           read(*,*) decide

             if (decide.eq.'s')    then
              visual = psi
              where(pot.ne.0.0)   visual = maxval(abs(psi))
              call display_psi(visual/maxval(abs(psi)), length, length)
              write(*,*)'<s>how without potential ? ; continue the
     &     computation  , <y>es  or <n>o   ???'
              read(*,*) decide
              if (decide.eq.'s')  then
                call display_psi(psi/maxval(abs(psi)), length, length)
                goto 64
              end if
```

```
              end if
              if (decide.eq.'n') goto 72

            end if


C__________DISPLAY PSI_____________________

if (mod(step,50).eq.0) then
  call display_psi(psi/maxval(abs(psi)), length, length)
        end if

C_________SCREEN AND FILE OUTPUT OF DIFFERENT VALUES_____

        if (mod(step,10).eq.0) then
          write(*,*) '*'
          write(*,*)'step       = ',step
          write(*,*)'psisquare  = ',psisquare(length, psi,x_unit)
          write(*,*)'E_kin      = ', kin_energy(length,psi, pot)
          write(*,*)'E_pot      = ',pot_energy(length, x_unit, psi, pot),
          write(*,*)'E_total    = ', kin_energy(length,psi, pot) +
     &                               pot_energy(length,x_unit,psi,pot)
          write(*,*)'expected x = ',expected_x(psi, x_unit, length)
          write(*,*)'expected y = ',expected_y(psi, x_unit, length)

          if (A) write(20,*) step, expected_x(psi,x_unit,length)
          if (B) write(30,*) step, expected_y(psi,x_unit,length)
          if (C) write(40,*) step, kin_energy(length,psi,pot)
          if (D) write(50,*) step, pot_energy(length,x_unit,psi,pot)
          if (E) write(60,*) step, kin_energy(length,psi, pot) +
     &                             pot_energy(length,x_unit,psi,pot)
          if (F) write(70,*) step, psisquare(length, psi, pot)
          if (G) write(80,*) step, sum((abs(psi)**2),mask=zyl)*x_unit**2
      end if

       end do     !   END MAIN PART

 72   write(*,*)'!!! new try, new luck !!!'

       if (H) close(10)
       if (A) close(20)
       if (B) close(30)
       if (C) close(40)
       if (D) close(50)
       if (E) close(60)
       if (F) close(70)
       if (G) close(80)
```

```
C______LAST SCREEN OUTPUT_____________________________________________

      write(*,*)' '
      write(*,*)'dt= ',t_step,' x_unit= ',x_unit,' sigma= ',sigma
      write(*,*)'potential type was :  ',choice
      write(*,*)' '
 80   write(*,*)'<a>gain or <l>eave ?'
      read(*,*) decide
      if (decide.eq.'a')  goto 8
      if (decide.ne.'a'.and.decide.ne.'l')   goto 80

      stop
      end



C_________S U B R O U T I N E _________________________________
C_________RICHARDSON ALGORITHM:________________________

      subroutine psi_update(length,alpha,betaeven,betaodd,psi,e_pot)

      implicit none

      integer length
      complex alpha
      complex, dimension(length,length):: psi, e_pot
      complex, dimension(length,length):: betaeven, betaodd

cmf$ layout psi(:news, :news)
cmf$ layout e_pot(:news, :news)
cmf$ layout betaeven(:news,:news)
cmf$ layout betaodd(:news,:news)

        psi = psi * e_pot

        psi = alpha * psi + betaeven * cshift(psi,dim=1,shift=-1)
     &               + betaodd * cshift(psi,dim=1,shift=+1)

        psi = alpha * psi + betaeven * cshift(psi,dim=1,shift=+1)
     &               + betaodd * cshift(psi,dim=1,shift=-1)

        psi = alpha * psi + betaeven * cshift(psi,dim=2,shift=-1)
     &               + betaodd * cshift(psi,dim=2,shift=+1)

        psi = alpha * psi + betaeven * cshift(psi,dim=2,shift=+1)
     &               + betaodd * cshift(psi,dim=2,shift=-1)

      return
```

```
      end


C__________ F U N C T I O N S _______________________________

C_______NORM OF THE WAVE PACKET_________

real function psisquare(length, array, x_unit)

implicit none
integer length
        real x_unit
complex, dimension(length,length):: array
cmf$   layout array(:news, :news)

psisquare = sum(abs(array)**2) * x_unit**2

return
end


C_________KINETIC ENERGY__________________

real function kin_energy(length,psi)

implicit none

integer length
        real, parameter:: h_bar = 1.0
real, parameter::  mass = 1.0
complex, dimension(length,length)::psi

cmf$ layout psi(:news, :news)

       kin_energy =
     & - h_bar**2/(2*mass)*sum(conjg(psi)*
     &             (  cshift(psi, dim=1, shift=+1) +
     &                cshift(psi, dim=1, shift=-1) +
     &                cshift(psi, dim=2, shift=+1) +
     &                cshift(psi, dim=2, shift=-1) - 4*psi) )


return
end

C___________POTENTIAL ENERGY_________________

real function pot_energy(length,x_unit,psi, pot)
```

```
implicit none
        real x_unit
integer length
complex, dimension(length,length)::psi
real, dimension(length,length):: pot

cmf$ layout psi(:news, :news)
cmf$ layout pot(:news,:news)

       pot_energy = sum(abs(psi)**2 * pot ) * x_unit**2

return
end

C_________EXPECTED X-VALUE___________________

      real function expected_x(psi,x_unit,length)

      implicit none
      integer i, j, length
      real    x_unit
      complex, dimension(length,length):: psi,xpsisquare

CMF$  LAYOUT psi(:news,:news)
CMF$  LAYOUT xpsisquare(:news,:news)

      forall(i=1:length,j=1:length)
     &        xpsisquare(i,j) = i * x_unit * abs(psi(i,j))**2

      expected_x = sum(xpsisquare) * x_unit**2


      return
      end

C_________EXPECTED Y-VALUE___________________

      real function expected_y(psi,x_unit,length)

      implicit none
      integer i, j, length
      real    x_unit
      complex, dimension(length,length):: psi,ypsisquare

CMF$  LAYOUT psi(:news,:news)
CMF$  LAYOUT ypsisquare(:news,:news)
```

```
      forall(i=1:length,j=1:length)
     &        ypsisquare(i,j) = j * x_unit * abs(psi(i,j))**2

      expected_y = sum(ypsisquare) * x_unit**2

      return
      end

C_____________________________________________________________________
```