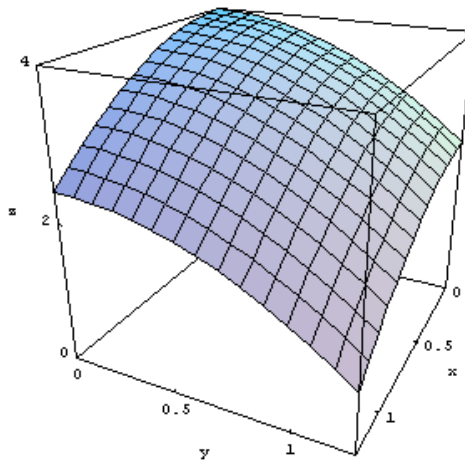
$$I=\int_0^{\frac{5}{4}}\int_0^{\frac{5}{4}}(4-x^2-y^2)dxdy=\left[\frac{835x}{192}-\frac{5x^3}{12}\right]_{x=0}^{x=\frac{5}{4}}$$

$$I=\int_0^{\frac{5}{4}}\int_0^{\frac{5}{4}}(4-x^2-y^2)dxdy=\frac{1775}{384}$$

$$I=\int_0^{\frac{5}{4}}\int_0^{\frac{5}{4}}(4-x^2-y^2)dxdy=4.622395833$$



[1] – figure issue du site
http://math.fullerton.edu/mathews/n2003/montecarlo/MonteCarloMod/Links/MonteCarloMod_lnk_3.html

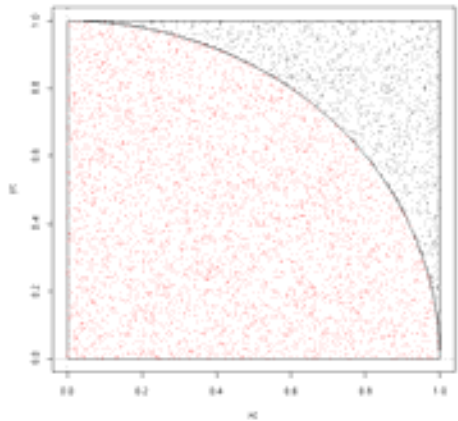
2.6.2 Implémentation

```
(***** fonction *****)
let f x y = 4.-.(x**2.)-.(y**2.);;

(******* corps du calcul ******)
let integration_double xa xb ya yb dx dy  =
        let rec fx hx sum = match hx with
                                |hx when hx>xb->sum*.dx*.dy
                                |_-> let rec fy hy sum =
```

un. La probabilité d'obtenir d'avoir un point à l'intérieur de l'aire du disque est égale au rapport des aires du quart du disque de rayon un et du carré de côté un. Elle est donc de π/4.

Nous allons considérer un grand nombre N de paires de nombres (r1,r2), tirés au hasard avec une distribution uniforme entre 0 et 1. Chaque paire est représentée en coordonnées cartésiennes par un point à l'intérieur du carré de côté unitaire.



Répartition uniforme des points dans le carré [0,1]x[0,1]

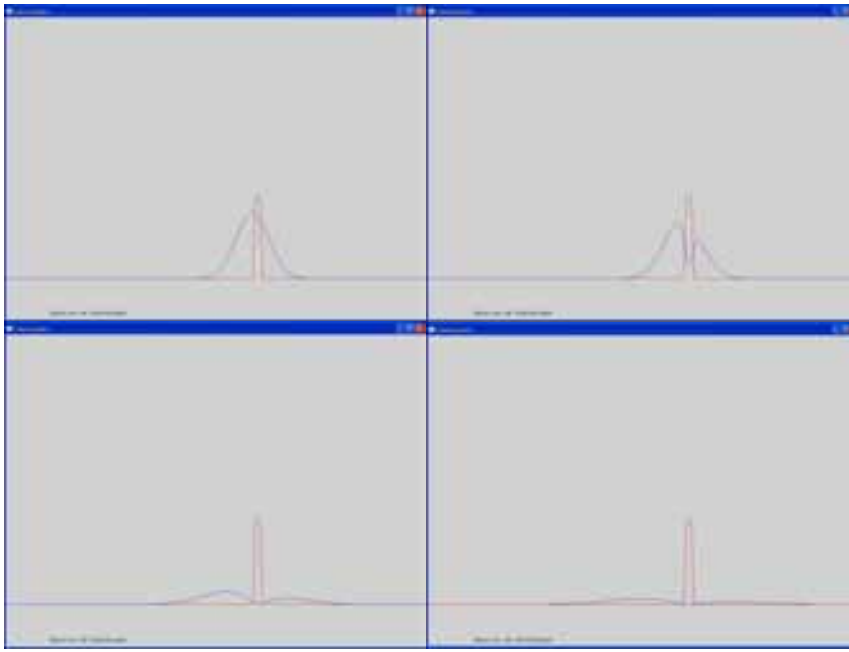
3.1 Code Ocaml

(* Calcul de pi par la méthode de Monte-Carlo *)

```
let pi_monte_carlo nb_tirage =
    let rec calcul_pi som n =match n with
        |0-> 4. *. (float_of_int som) /. (float_of_int nb_tirage)
        (* Tirage aléatoire pour x et y *)
        |_-> let x = Random.float 1. and y = Random.float 1.
             in let pt_interieur = if x*.x +. y*.y < 1. then 1 else 0
                in calcul_pi (som+pt_interieur) (n-1)
    in calcul_pi 0 nb_tirage;;

print_string "valeur approximée ";;
Printf.printf "%f\n"  (pi_monte_carlo 1000000);;
(***** Valeur exacte *****)
print_string "valeur exacte :";;
print_float (4.*.(atan 1.));;
```

```
Graphics.draw_string " ";
Graphics.moveto 100 50;
Graphics.draw_string " ";
Graphics.moveto 300 380;
Graphics.draw_string " ";
Graphics.moveto 300 300;
Graphics.draw_string " ";;
```



Comme on peut le constater, la fonction d'onde est ici stationnaire, afin d'étudier son déplacement et la collision sur la barrière de potentiel, il faut introduire l'équation de transport (Hamiltonien) qui est caractérisée par l' EDP *suivante :*

$$\begin{cases} \dfrac{\partial u}{\partial t}+\alpha\dfrac{\partial u}{\partial x}=0 \quad \forall x\in R,\ t>0 \\ \\ u(x,0)=u_0(x) \quad \forall x\in R \end{cases} \qquad u(x,t)=u_0(x-\alpha t)$$

Afin de construire une approximation de cette EDP*, on peut utiliser différents schémas numériques :*

Schéma centré : $u_j^{n+1}=u_j^n-\lambda\dfrac{u_{j+1}^n-u_{j-1}^n}{2}$

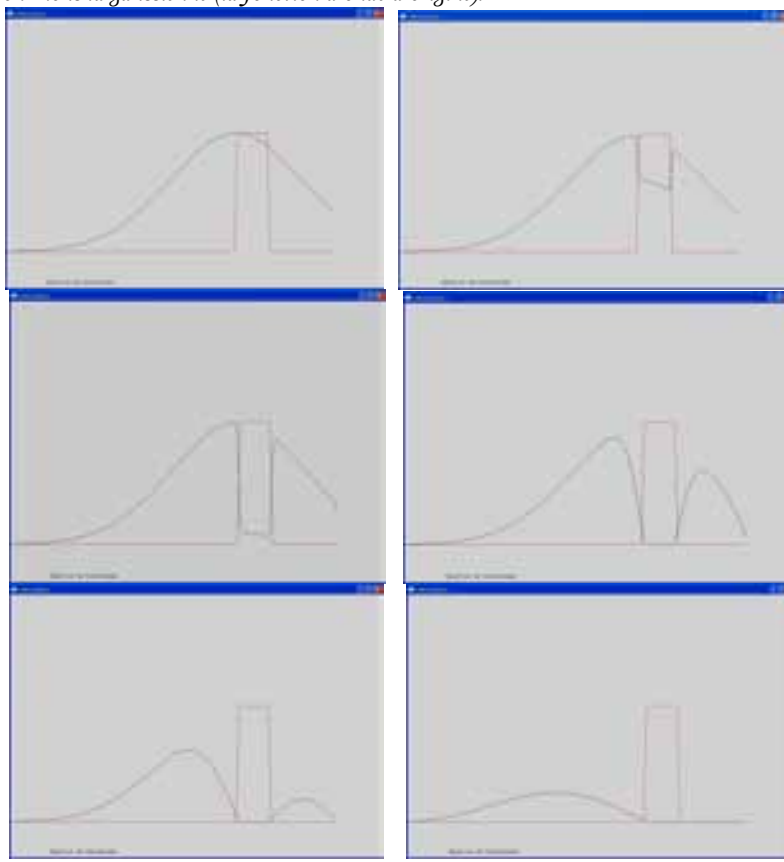
Avec $\lambda=\alpha\dfrac{\Delta t}{\Delta x}\le 1$ *ce schéma est d'ordre 1 et instable*

Ou encore

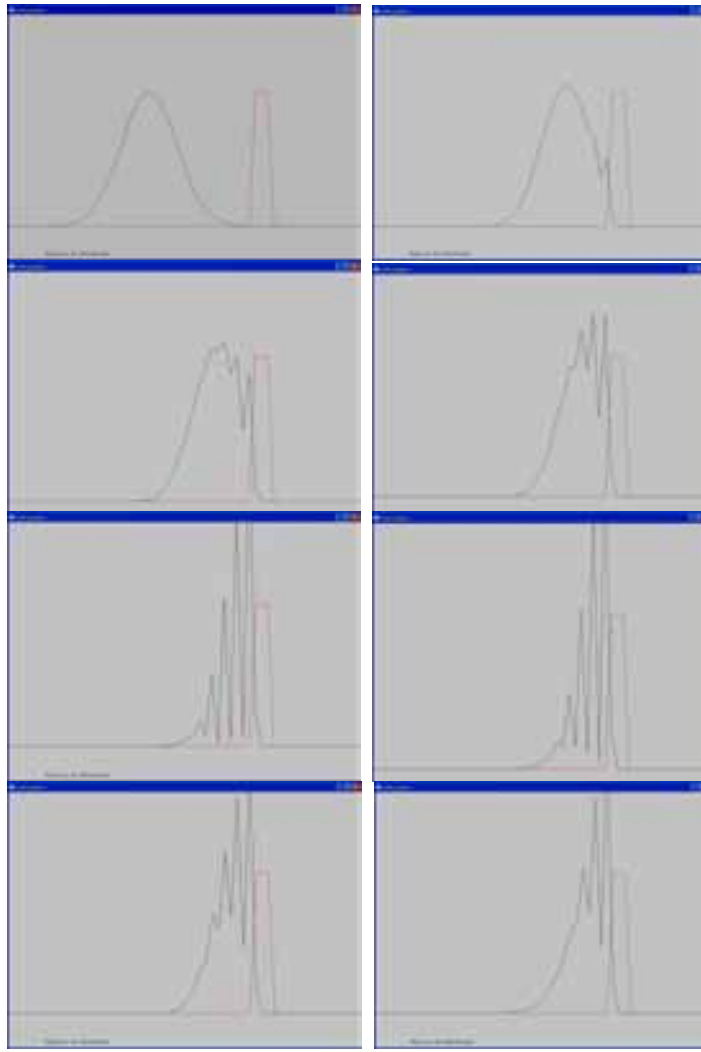
$$\begin{bmatrix} a1 & b1 & & & & \\ b2 & a2 & b2 & & & \\ & b3 & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{n-2} & a_{n-2} & b_{n-2} \\ & & & & b_{n-1} & a_{n-1} \end{bmatrix} \begin{bmatrix} \Psi_1^{k+1} \\ \Psi_2^{k+1} \\ \Psi_3^{k+1} \\ \vdots \\ \Psi_{n-2}^{k+1} \\ \Psi_{n-1}^{k+1} \end{bmatrix} = \begin{bmatrix} u_1 - b1.\Psi_0^{k+1} \\ u2 \\ u3 \\ \vdots \\ u_{n-2} \\ u_{n-1} - b_{n-1}.\Psi_n^{k+1} \end{bmatrix}$$

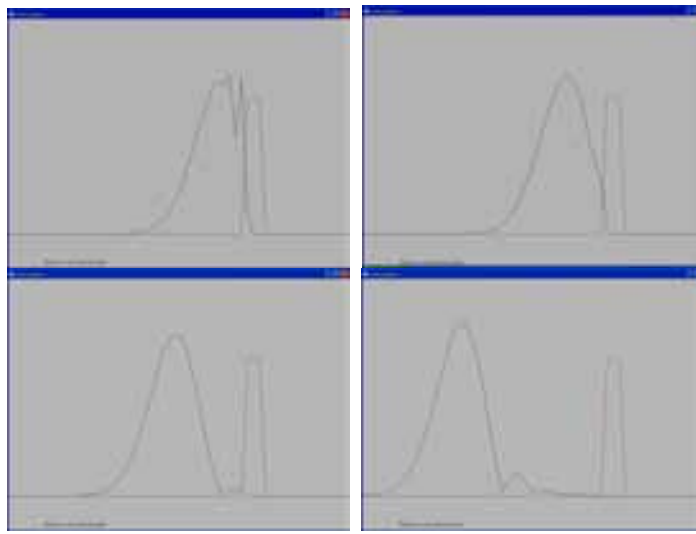
Avec $a_i = 1 + \frac{i.\Delta t.\hbar}{2.m.\Delta x^2} + \frac{i.\Delta t}{2.\hbar}.V_i$ *et* $b_i = -\frac{i.\Delta t.\hbar}{4.m.\Delta x^2}$

En posant $\Psi_0^{k+1} = \Psi_n^{k+1} = 0$ *le membre de droite dépend uniquement des* u_i *représentant pour nous la gaussienne (la fonction d'onde d'origine).*



2.3.2 Équation implicite - Rencontre de la fonction d'onde sur une barrière de potentiel sans effet tunnel :





```
(* Résolution de l'équation de Schrödinger dépendante du temps *)
#use "C:/gradient bi_conjugué.ml";;

#directory "+threads";;
# load "unix.cma";;
# load "threads.cma";;
#load "graphics.cma";;
open Graphics;;
open_graph "1024x768";;

(*****************************************************************)
(********************** définition des paramètres ***************)
(*****************************************************)*********)
let pi=acos(-.1.);;

(******* constante de Planck réduite ********)
let hb=1.05459e-34;;

(********** masse de la particule ***********)
let m=9.109*.10e-31;;

(******** Borne de l'espace: [-X;X] *********)
let _X=5.;;

(************ le pas d'espace ***************)
let dx=0.09;;

(****** nombre de points d'intégration ******)
let n=2.*.(_X/.dx)+.1.;;

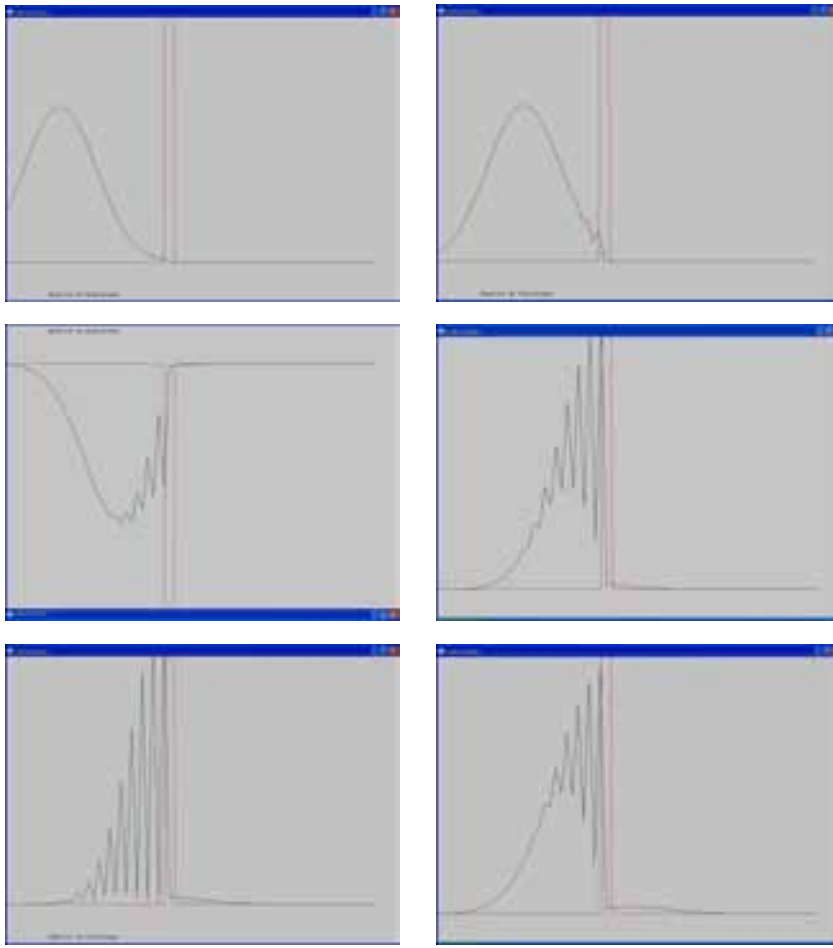
(************ pas temporel **********)
let dt=10e-7;;

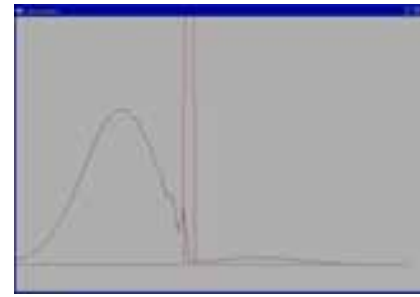
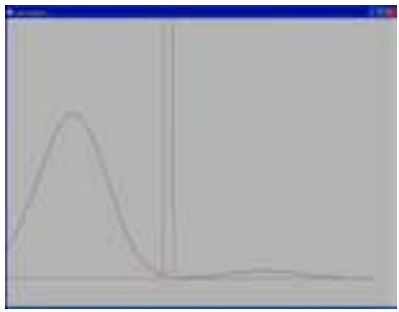
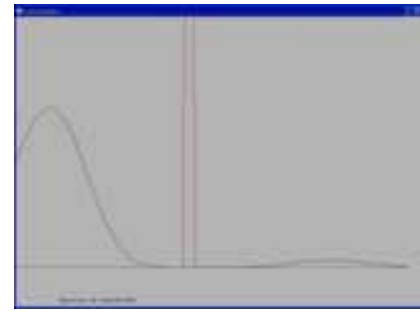
(***** vitesse du paquet d'onde *****)
let xi=5000.;;
```

```
        in
          trace v red;
          trace (List.flatten efface) white;
          trace  (List.flatten u1) black;
          solution uk v (i+1) xi u1;;

solution u _V 1 xi u ;;
```

2.3.3 Équation implicite - Rencontre de la fonction d'onde sur une barrière de potentiel avec effet tunnel :



```
(* Résolution de l'équation de Schrödinger dépendante du temps *)
#use "C:/analyse numerique/gradient bi_conjugué.ml";;

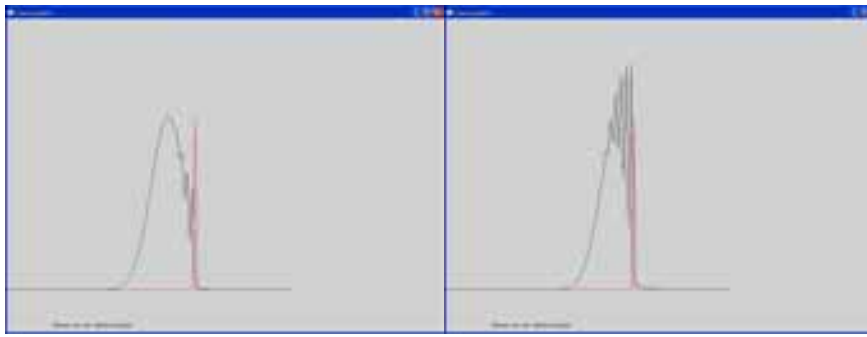
#directory "+threads";;
# load "unix.cma";;
# load "threads.cma";;
#load "graphics.cma";;
open Graphics;;
open_graph "1024x768";;

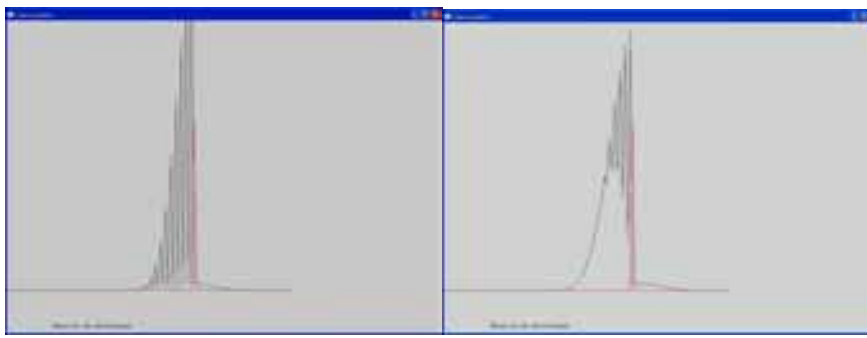
(*********************************************************)
(***************** définition des paramètres *************)
(*********************************************************)
let pi=acos(-.1.);;
(******* constante de Planck réduite ********)
let hb=1.05459e-34;;
(********** masse de la particule ***********)
let m=9.109*.10e-31;;

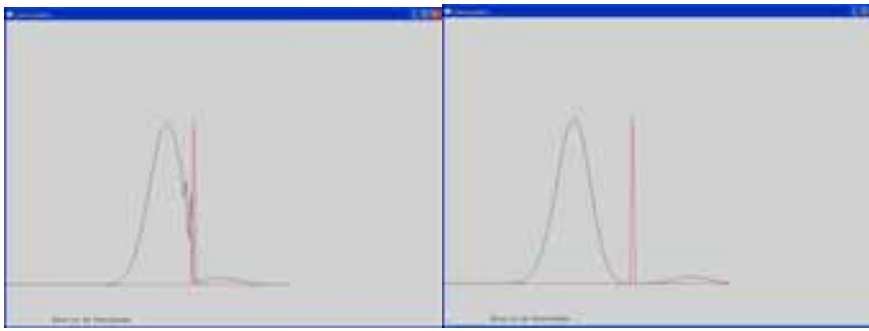
(******** Borne de l'espace: [-X;X] *********)
let _X=5.;;
(************ le pas d'espace ***************)
let dx=0.09;;
(****** nombre de points d'intégration ******)
let n=2.*.(_X/.dx)+.1.;;
(************ pas temporel **********)
let dt=10e-7;;
(***** vitesse du paquet d'onde *****)
let xi=1000.;;
```

```
(*****************************************************************)
(******************** Calcul de u au pas de temps suivant *******)
(*****************************************************************)
let rec solution  u v i xi cond efface = match i with
                  |i when i==13000 -> failwith "fin"
                  |_-> let u0= (2.*.dt*.xi/.dx) **. ((uk_moins_1 u)
                               *-. (uk_plus_1 u))
                       in
                        let _u= gradient_bi_conjug _H u
                        in
                          let _u= u0 *+. _u
                          in
          let u1= List.map (fun x->[abs_float (List.hd x)])  _u
         in
         trace v red;
         trace (List.flatten efface) white;
         trace  (List.flatten u1) black;
         solution _u v (i+1) xi cond u1;;

solution u _V 1 xi  "dirichlet" u ;;
```







On remarque l'effet tunnel- l'énergie générée par la barrière augmente la fréquence en sortie

2.3.5. Équation explicite :

$$\Psi(x,t+\frac{\Delta t}{2})=\left[1-\frac{i.\Delta t.H}{2\hbar}\right]\Psi(x,t) \qquad \textit{(équa n°2)}$$

On peut facilement résoudre numériquement cette équation en discrétisant l'espace des x en intervalles Δx, cette équation peut alors s'écrire :

$$u_i = \Psi_{i,j}+\frac{i.\Delta t.\hbar}{4.m.\Delta x^2}\Psi_{i-1,j}-\frac{i.\Delta t.\hbar}{2.m.\Delta x^2}\Psi_{i,j}+\frac{i.\Delta t.\hbar}{4.m.\Delta x^2}\Psi_{i+1,j}-\frac{i.\Delta t.}{2.\hbar.}V_i.\Psi_{i,j}$$

Avec les quantités indicées par i, j correspondent aux valeurs en $x=i.\Delta x$ et $t=j.\Delta t$

Cette équation étant explicite, elle permet de calculer directement les u_i correspondant à $\psi(i.\Delta x,t+\Delta t/2)$ pour un j donné.

En arrangeant sous une forme plus sympathique cette équation :

$$u_i = \Psi_{i,j}(1-\frac{i.\Delta t.\hbar}{2.m.\Delta x^2})-\frac{i.\Delta t.}{2.\hbar.}V_i.\Psi_{i,j}+\frac{i.\Delta t.\hbar}{4.m.\Delta x^2}(\Psi_{i-1,j}+\Psi_{i+1,j})$$

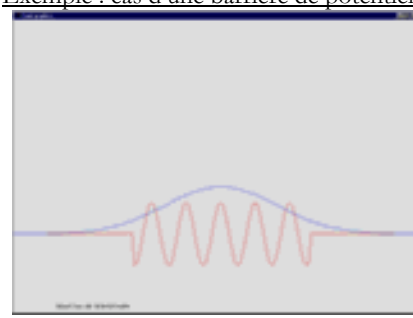
En posant $k=\frac{\hbar}{4.m}$ on obtient :

$$u_i = \Psi_{i,j}(1-2.i.k\frac{\Delta t.}{\Delta x^2})-i\frac{\Delta t.}{2.\hbar.}V_i.\Psi_{i,j}+i.k\frac{\Delta t.}{\Delta x^2}(\Psi_{i-1,j}+\Psi_{i+1,j})$$

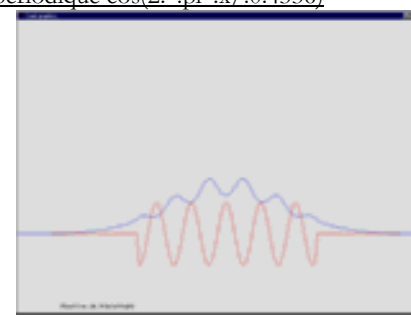
```
(*****************************************************)
(************* Calcul de u au pas de temps suivant *****)
(*****************************************************)
let rec solution  u v i dx dt cond efface = match i with
                  |i when i==13000 -> failwith "fin"
                  |_-> let u1 = scal ((float_of_int i)*.k*.dt/.dx**2.)
                              (List.map2 (+.) (uk_moins_1 u) (uk_plus_1 u))
       in let u2 = scal (1.-.2.*.k*.(float_of_int i)*.dt/.(dx**2.)) u
   in let u3 = scal (-.(float_of_int i)*.dt/.(2.*.hb))
              (List.map2 ( *.) v u)
      in
      let uk =List.map2 (+.) u3 (List.map2 (+.) u1 u2 )
      in (*** conditions aux choix *******)
       let u_n=if cond="dirichlet" then
               (** condition de Dirichlet: u[1]=0 et u[k]=0 **)
                  dirichlet uk
               else
               (* condition de Neuman: u[1]=u[2] et u[k]=u[k-1]*)
                  neuman uk
       in
         let u_=List.map abs_float u_n
         in
         trace _V red;
         trace efface white;
         trace uk blue;
         solution u_n v (i+1)  dx dt cond uk;;

(*** point d'entrée ***)
solution u _V 1  dx dt "dirichlet" _V ;;
```

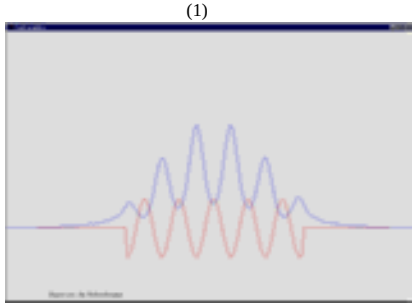
Exemple : cas d'une barrière de potentiel périodique cos(2.*.pi*.x/.0.4336)



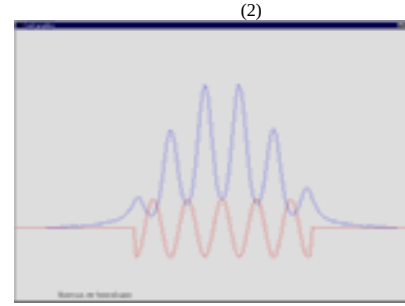
(1)



(2)



(3)

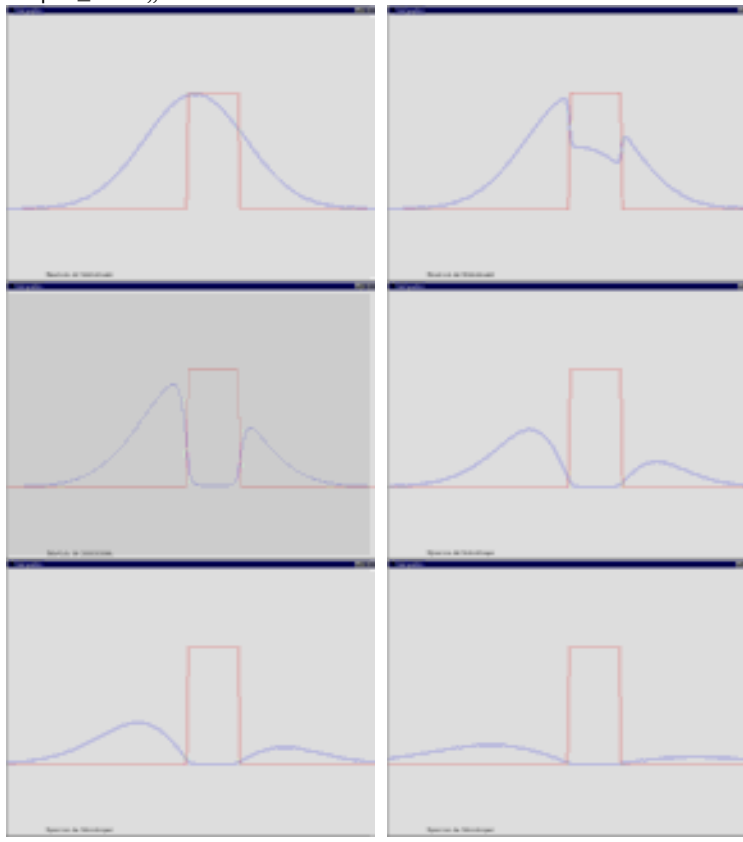


(4)

Exemple cas d'une barrière de potentiel

```
(**** Une Gaussienne ****)
let decalage_psi= 1.31;;
let largeur_psi=0.35;;
let u=
    let rec pot_aux i=
        let e=(float_of_int i-.(n/.2.))*._X/.n-.decalage_psi
        in
        match i with
 |i when i>=(int_of_float n)+1 -> []
 |_ ->800.*.(exp(-.(e**2.)/.(2.*.(largeur_psi**2.)))) :: pot_aux (i+1)
    in pot_aux 1;;

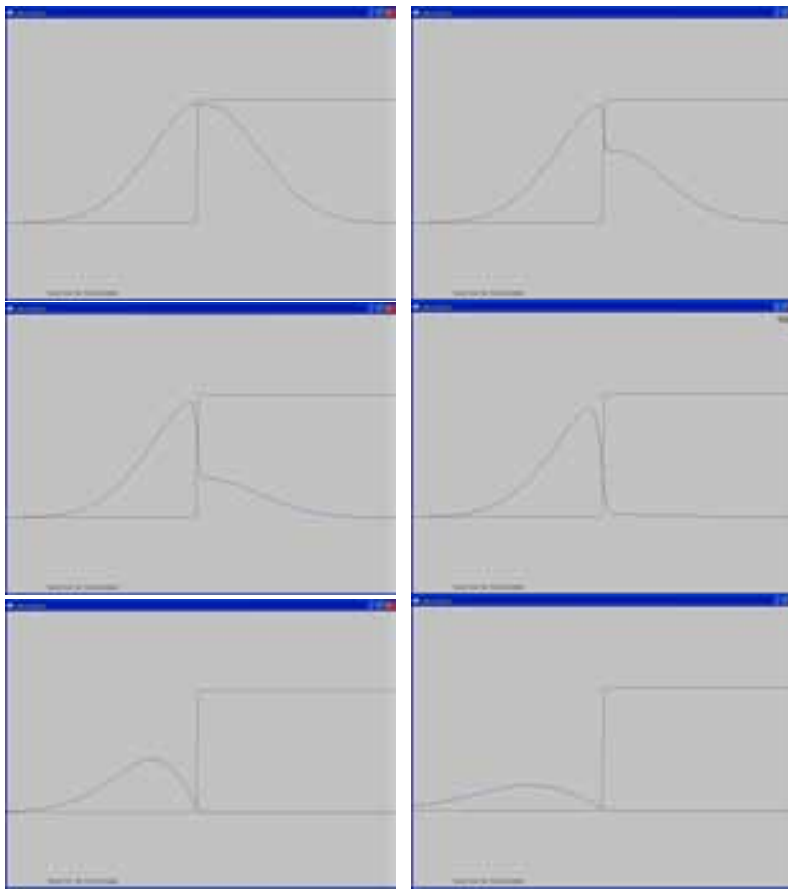
let predicat x= (x>= 1.5 && x<=3.73)
let potentiel d p=
    let rec pot_aux dx=
        match dx with
          |[]-> []
           (*** amplitude du potentiel ***)
          |x::r-> if (p x) then 800.:: pot_aux r
                  else 0.::pot_aux r
      in pot_aux x;;
```



Exemple cas d'une marche de potentiel

```
let predicat x= (x>= 2.5 )
let potentiel d p=
    let rec pot_aux dx=
        match dx with
          |[]-> []
           (*** amplitude du potentiel ***)
          |x::r-> if (p x) then 800.:: pot_aux r
                  else 0.::pot_aux r
    in
       pot_aux x;;

let _V=potentiel x predicat;;
```



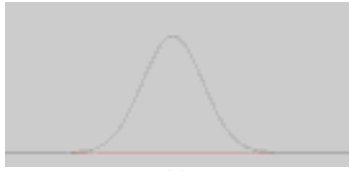
Dans le cas d'un potentiel nul, on constate l'étalement de la gaussienne :

```
(*********************************************************************)
(*****************   calcul de la gaussienne psi0  *******************)
(*********************************************************************)
let decalage_psi= 1.255;;
let largeur_psi=0.2;;

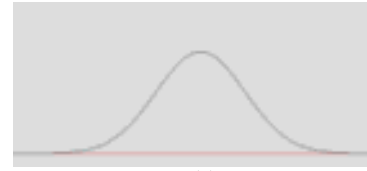
let u=
    let rec pot_aux i=
        let e=(float_of_int i-.(n/.2.))*._X/.n-.decalage_psi
        in
       match i with
       |i when i>=(int_of_float n)+1 -> []
       |_ ->800.*.(exp(-.(e**2.)/.(2.*.(largeur_psi**2.))))
                                                    :: pot_aux (i+1)
    in
      pot_aux 1;;

(*****************************************************)
(************* définition du potentiel V *************)
(*****************************************************)
(************* étalement de la gaussienne *************)
(****************  aucun potentiel ********************)
let potentiel d =
    let rec pot_aux dx=
        match dx with
          |[]-> []
        (*** étalement de la gaussienne -> aucun potentiel ***)
          |x::r-> 0.::pot_aux r
    in
       pot_aux x;;

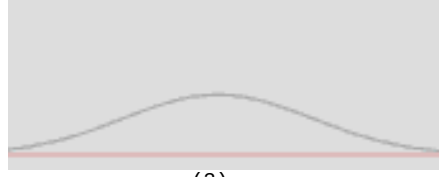
let _V=potentiel x ;;
```



(1)

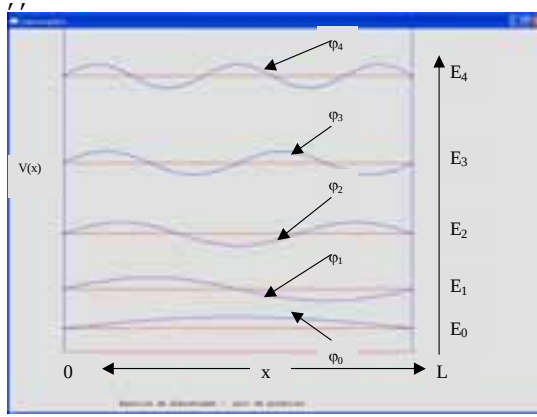


(2)



(3)

```
(*****************************************************************)
(******************** Calcul de phi n ****************************)
(*****************************************************************)
let rec solution  i  = match i with
                  |i when i==niv+1 -> failwith "fin"
                  |_-> trace (u i) i blue;
                                        trace _V i red;
                       solution (i+1) ;;
(** point d'entrée **)
solution 0 ;;
```



4 RENCONTRE D'UN PAQUET D'ONDE GAUSSIEN VERS UNE MARCHE DE POTENTIEL - RÉSOLUTION DE : $-\frac{\partial \Psi(x,t)}{\partial t} = -\frac{\partial^2 \Psi(x)}{\partial x^2} + V(x)\Psi(x)$

On choisit pour définir l'intervalle de discrétisation en espace, la valeur de x *et le pas* dx *:*

[-x, -x+dx, -x+2dx, ……., -x+(n-2)dx, x=-x+(n-1)dx]

Avec dx *le pas d'espace et* n= 2.(x/dx)+1 *le nombre de points de discrétisation.*

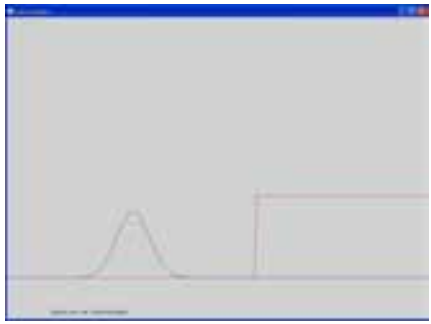
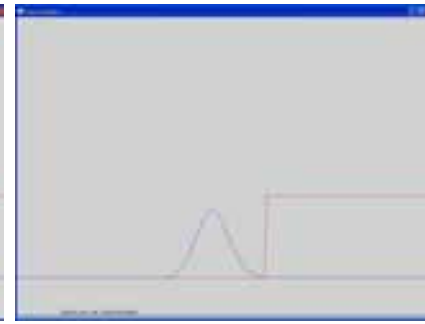
```
(* Borne de l'espace: [-X;X] *)
let _X=40.;;
(* pas d'espace *)
let dx=0.03;;
(* nombre de points d'intégration *)
let n=2.*.(_X/.dx)+.1.;;

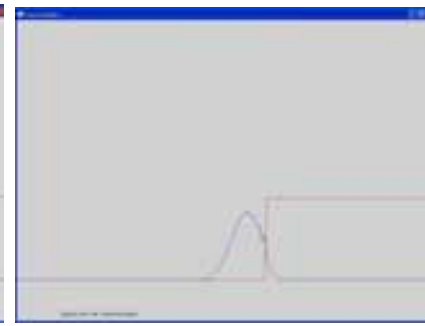
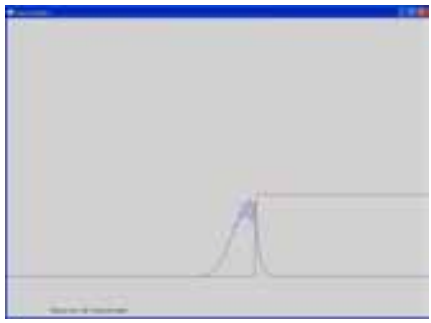
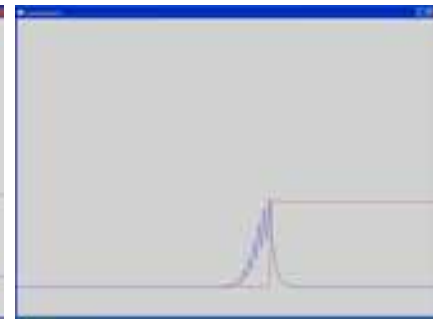
(******************** discrétisation spatiale *****************)
(****** création du vecteur contenant les abscisses ***********)
(************  progression linéaire entre -X et X *************)
let discr_spat x dx =
    let rec discr_aux x y=
        match x with
          |x when x>y -> []
          | _ -> x:: discr_aux (x+.dx) y
    in
       discr_aux (-.x) x;;
(*** espace discrétisé ***)
let x=discr_spat _X dx;;
```

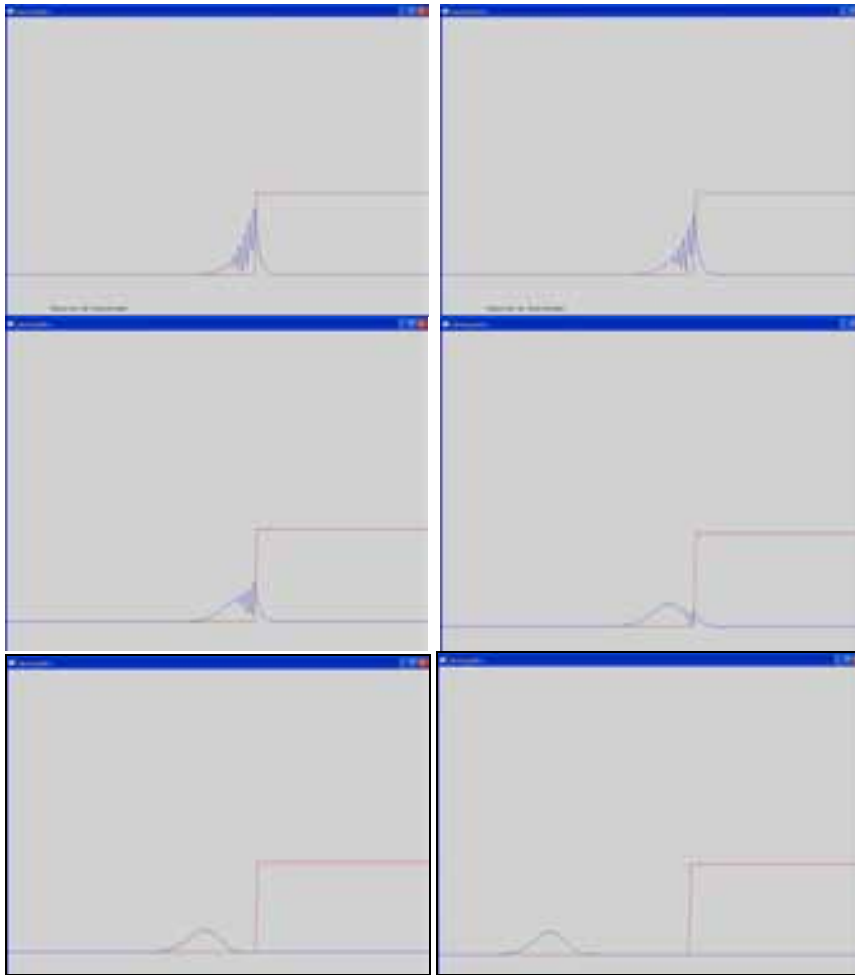
```
        trace _V red;
        trace efface white;
        trace u_ blue;
        solution u_n v (i+1) xi dx dt cond u_;;

(*** application – point d'entrée de l'application ***)
solution u _V 1 vit dx dt "dirichlet" _V;;
```

Dans le cas de la propagation d'un paquet d'onde Gaussien, lorsqu'il franchit une marche de potentiel, où la hauteur de la marche est supérieure à l'énergie de la particule, on observe une réflexion totale, ainsi que les interférences entre l'onde réfléchie et l'onde incidente lorsqu'elles se superposent.

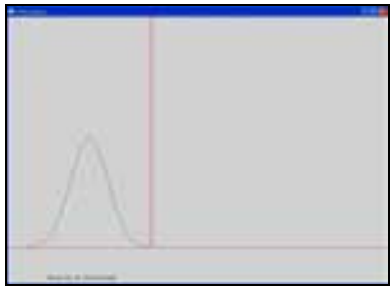



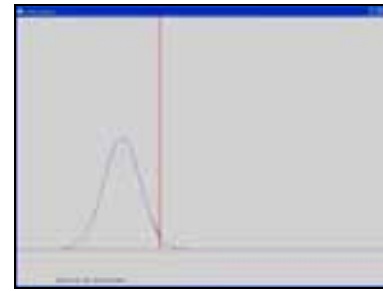




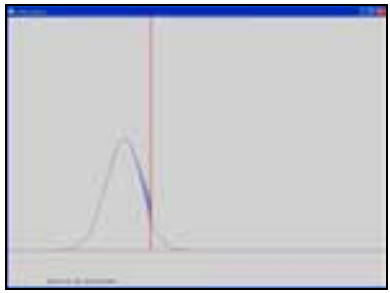


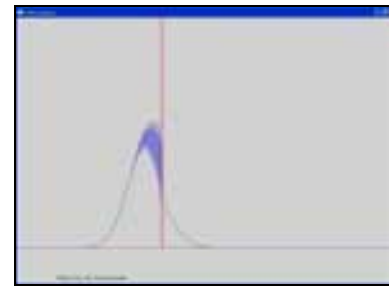
Étalement du paquet d'ondes gaussien

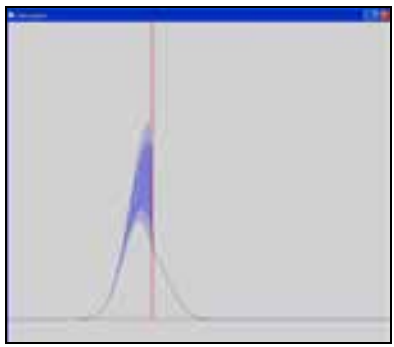
Dans le cas où la particule est partiellement transmise, sa vitesse est fortement atténuée.
Le paquet d'onde s'étale, la largeur à mi-hauteur augmente au cours du temps. Le paquet gaussien conserve sa forme en s'étalant; ceci est spécifique au cas du paquet gaussien.
« On observe l'étalement du paquet d'ondes qui est une conséquence directe du principe d'incertitude de Heisenberg : si le paquet est bien localisé dans l'espace des `x`, *il possède nécessairement une dispersion importante dans l'espace des probabilités. Les parties du paquet d'ondes associées aux plus grandes impulsions se propagent plus vite, d'où l'étalement. Ceci se vérifie directement en observant non pas la probabilité de présence mais la partie réelle de la fonction d'onde. Celle-ci fait apparaître la longueur d'onde de De Broglie, d'autant plus petite que l'impulsion est grande. On constate bien que les plus petites longueurs d'ondes prennent de l'avance sur les plus grandes, contribuant ainsi à l'étalement du paquet d'ondes. »*[3].

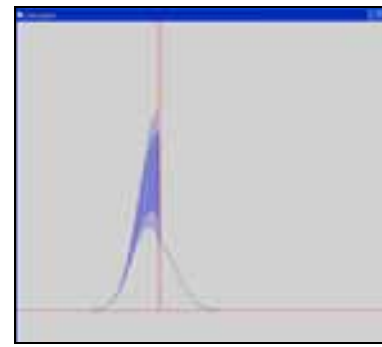


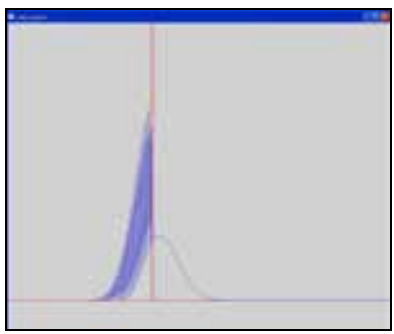


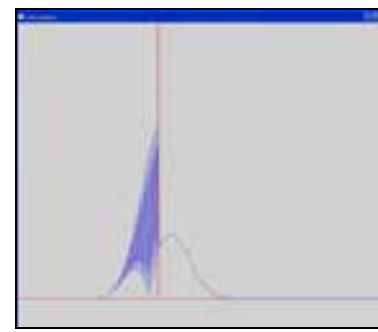


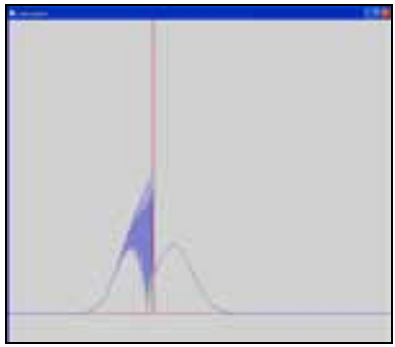


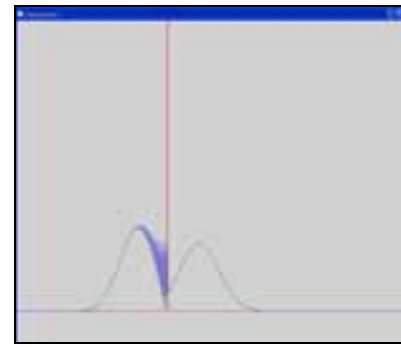


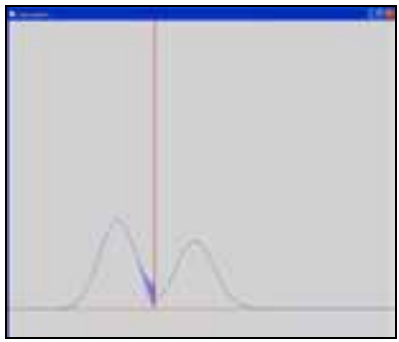


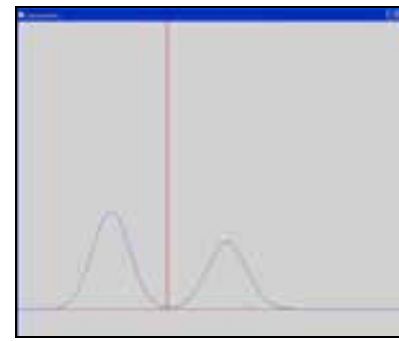


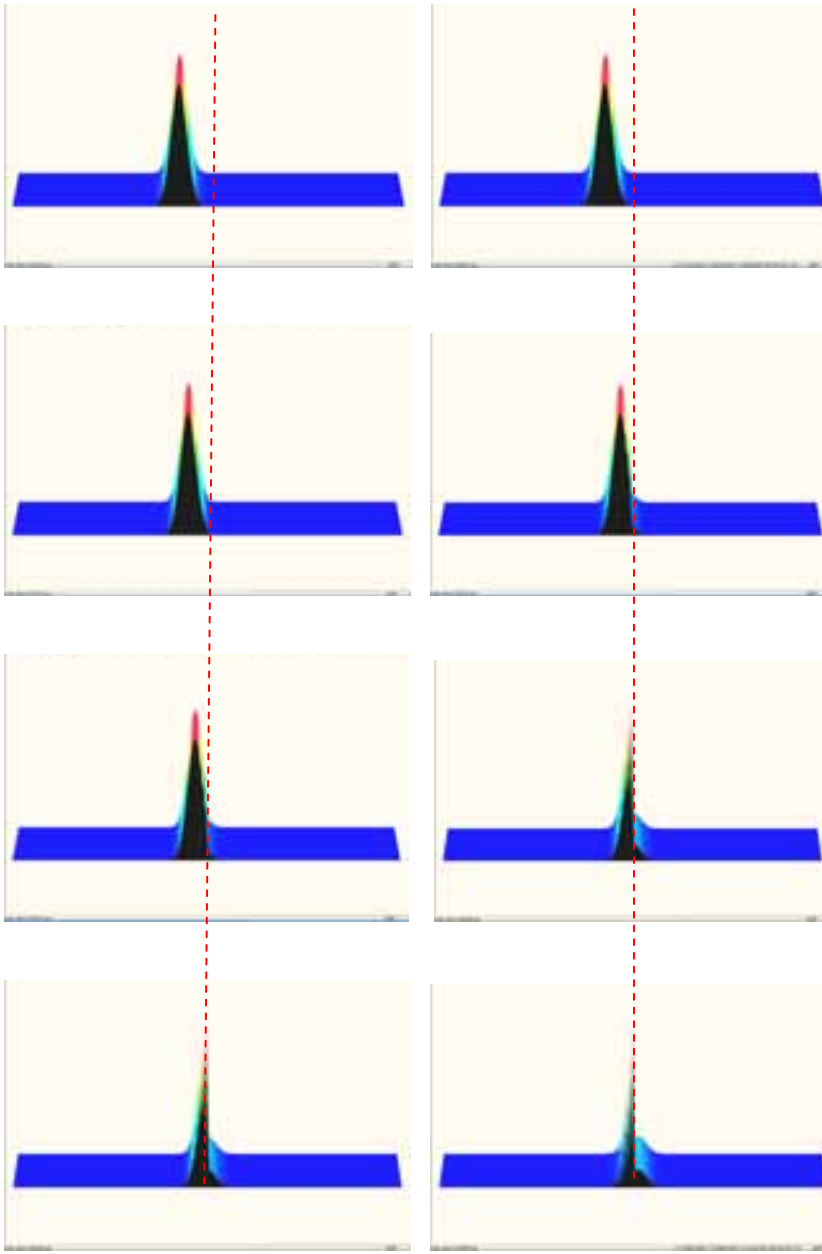


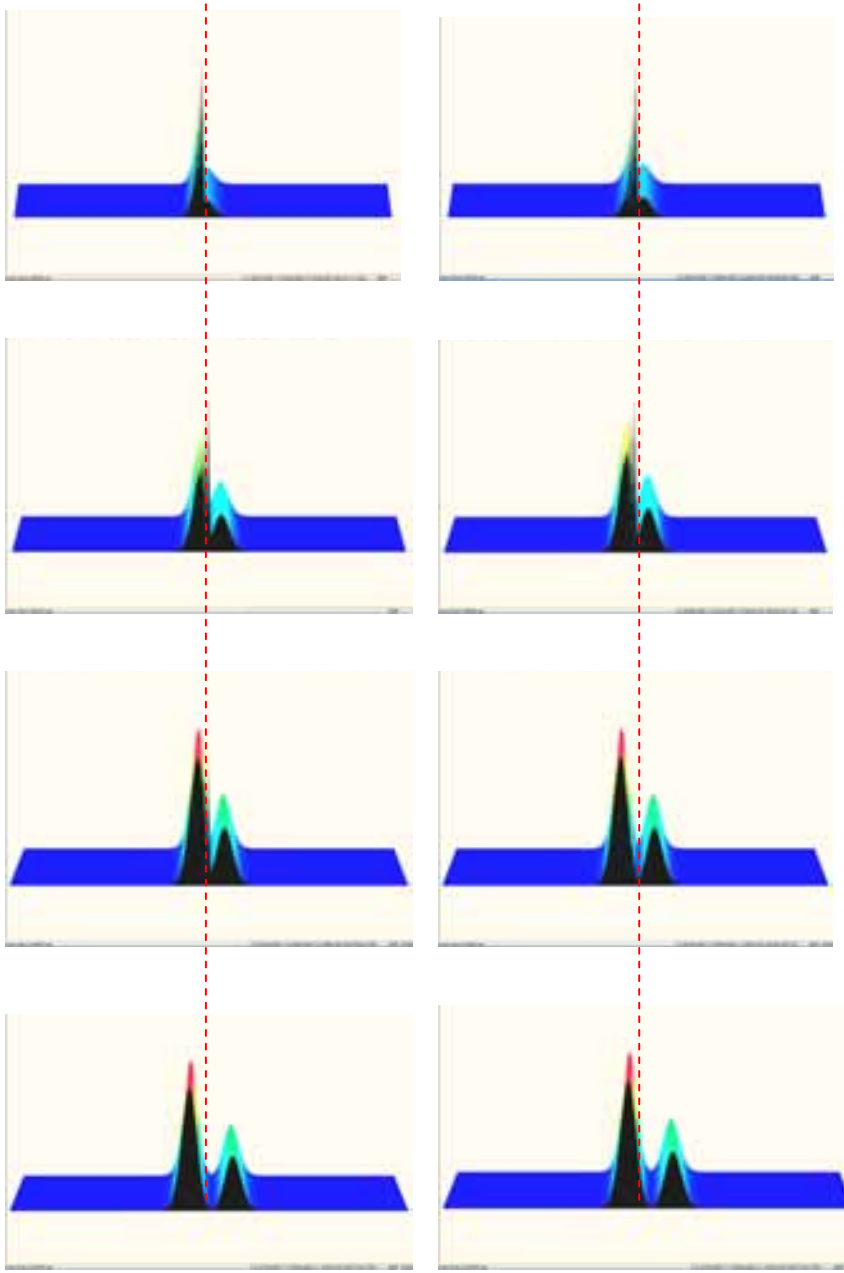






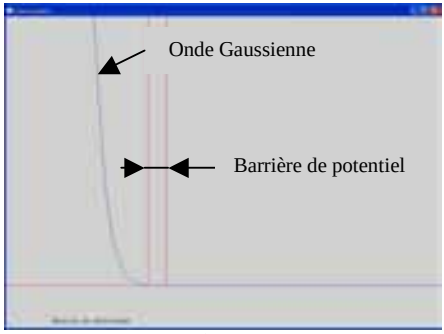




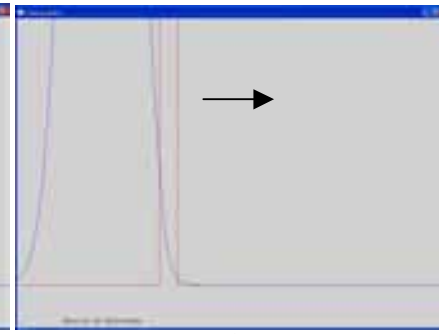


```
let rec solution  u v i xi dx dt cond efface= match i with
                  |i when i==13000 -> failwith "fin"
                  |_->
     let u1 = scal (2.*.dt*.vitesse/.dx) (List.map2 (-.)
           (right u) (left u))
 in let u2 = scal (dt/.(dx**2.)) (List.map2 (-.) (scal 2. u)
                     (List.map2 (+.) (left u) (right u)))
 in let u3 = scal ((float_of_int i)*.dt)(List.map2 ( *.) v u)
 in let uk = List.map2 (-.) (List.map2 (+.) u
           (List.map2 (+.) u1 u2 )) u3
   in (*** conditions aux choix *******)
       let u_n=if cond="dirichlet" then
     (** condition de Dirichlet: u[1]=0 et u[k]=0 **)
                    dirichlet uk
                else
     (** condition de Neuman: u[1]=u[2] et u[k]=u[k-1] **)
                         neuman uk
              in
               let u_=List.map abs_float u_n
               in
                trace potentiel red;
                trace efface white;
                trace u_ blue;
                solution u_n v (i+1) xi dx dt cond u_;;

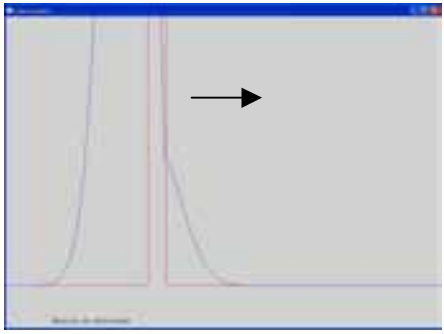
(** point d'entrée – Application **)
solution psi0 potentiel 1 vitesse dx dt "dirichlet" potentiel;;
```



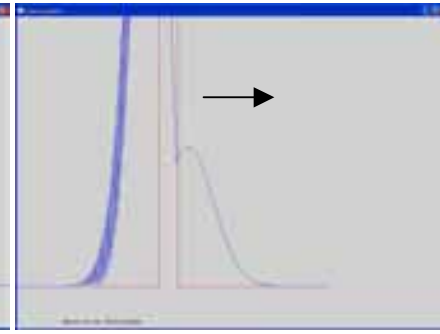
- 1 -



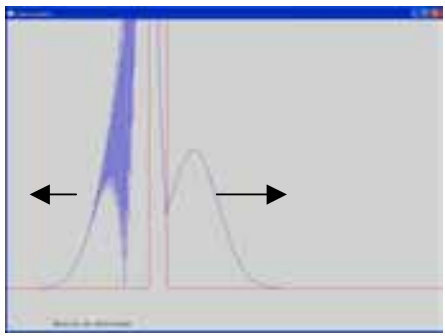
- 2 –



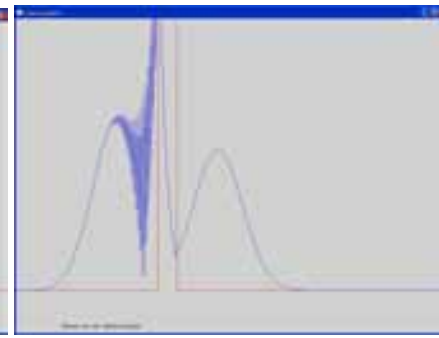
- 3 -



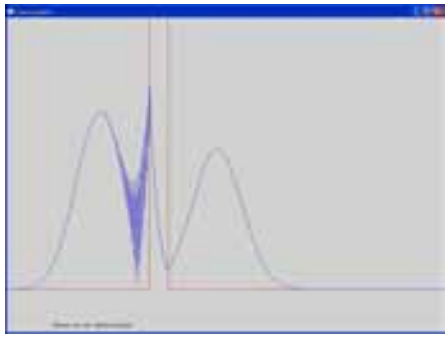
- 4 –



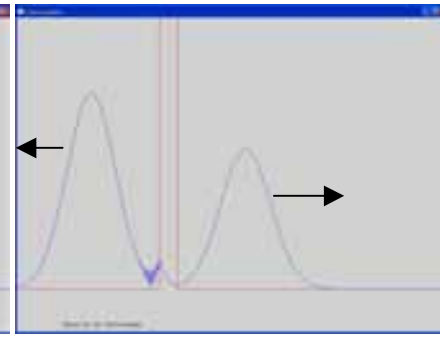
- 5 -



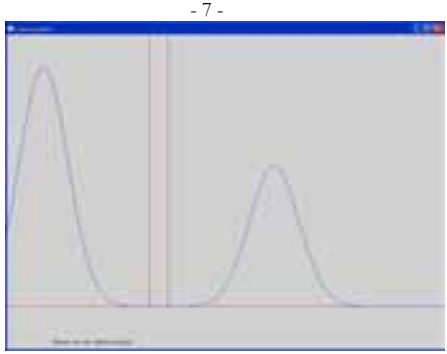
- 6 -



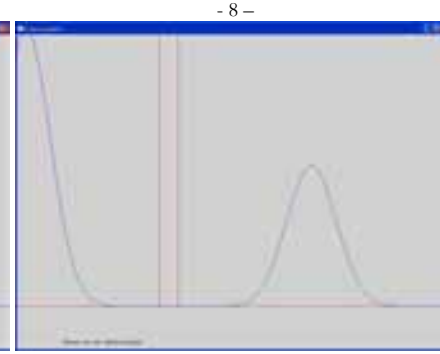
- 7 -



- 8 –



- 9 -

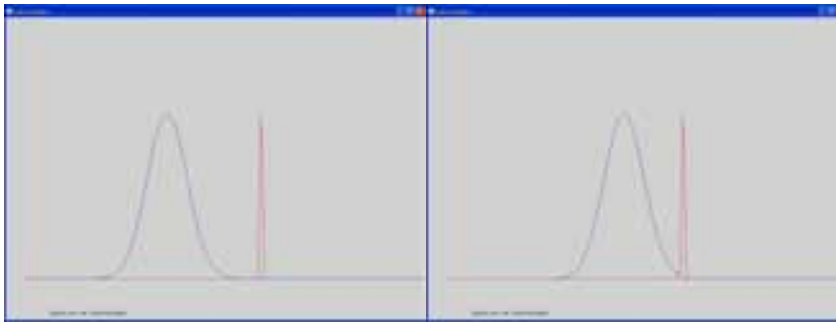


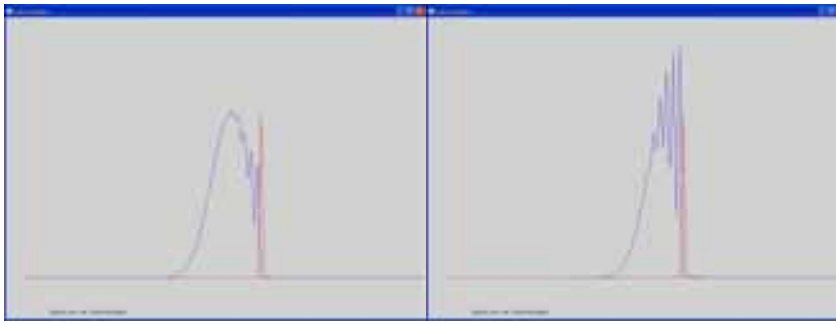
- 10 -

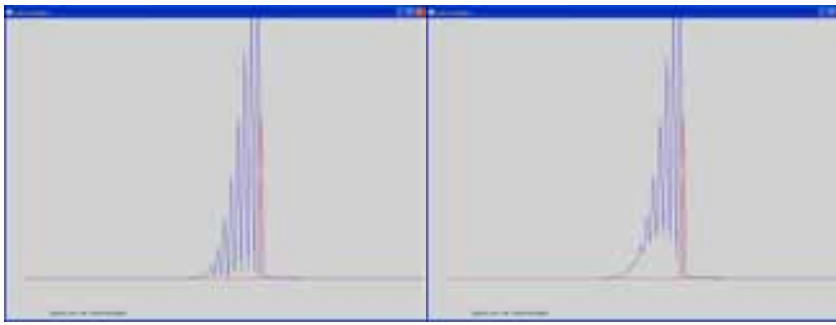
5.4. Essai en variables réelles $\hbar$ *et* m

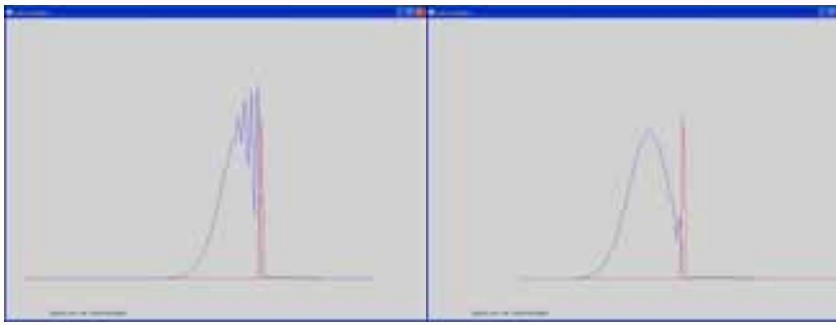
La difficulté en variables réelles c'est de travailler avec des exposants très petits. La constante de Planck est de l'ordre de 10e-35 et la masse de l'électron de 10e-30, sans oublier la valeur d'une barrière de potentiel exprimée en joules et définie par l'électron-volt sachant que eV=1.6e-19 Joule, le temps est également très faible de l'ordre de 1e-18 à 1e-22 seconde

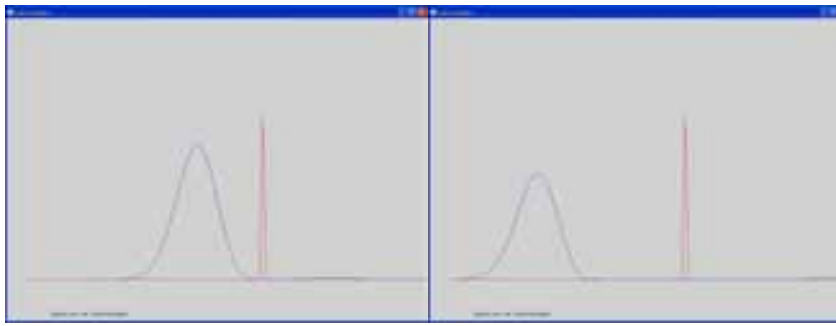
Manipuler des nombres si petits à l'aide d'un langage de programmation réside donc de l'exploit.

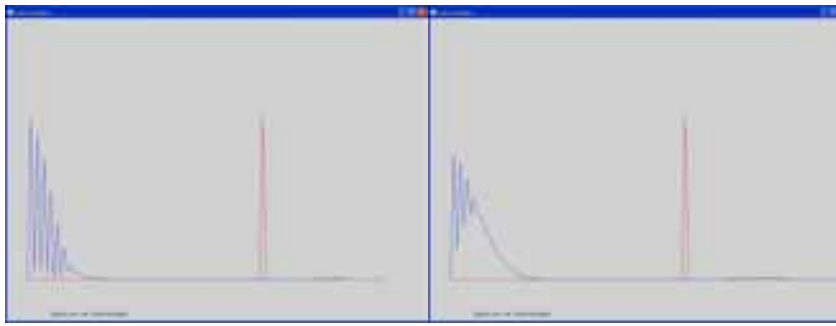


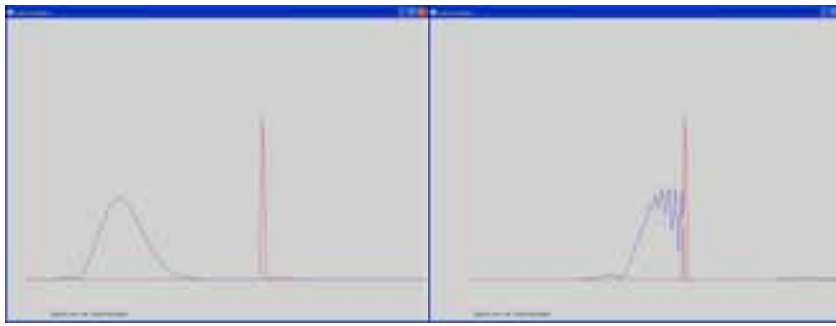


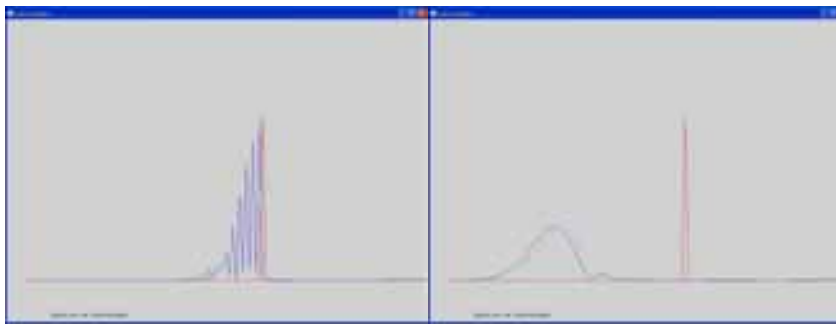


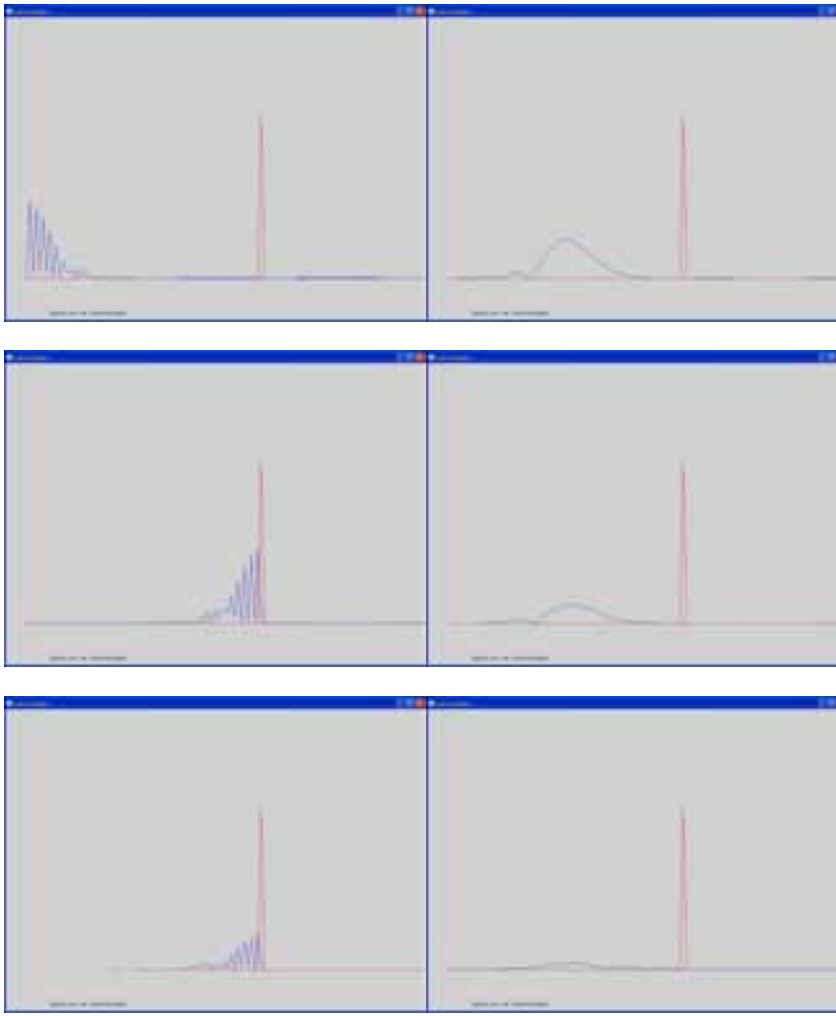












(* Résolution de l'équation de Schrödinger dépendante du temps *)

```
#directory "+threads";;
# load "unix.cma";;
# load "threads.cma";;
#load "graphics.cma";;

open Graphics;;
open_graph "1024x768";;

#use "D:/primitives_matricielles.ml";;

type complexe = {re :float; im :float} ;;
```