Allure des orbitales, en coupe, par une méthode de Monte-Carlo

```
restart
TypO := ("s1", "s2", "p2x", "p2z", "s3", "p3x", "p3z", "d3z2", "d3zx") : # orbitales autorisées ici

amplitude := proc(x :: float, z :: float, typo :: string) :: float;
 local
  r, th, aa :: float;
  r := sqrt(x^2 + z^2);
  if (r = 0.00) then th := 0.00;
  else th := arcsin(abs(x)/r);
  end if;
  if typo = "s1" then aa := a0^(-3/2)/sqrt(Pi)*exp(-r/a0);
  elif typo = "s2" then aa := (2*a0)^(-3/2)/sqrt(Pi)*(1-r/(2*a0))*exp(-r/(2*a0));
  elif typo = "p2x" then aa := (2*a0)^(-3/2)/sqrt(Pi)*(r/(2*a0))*exp(-r/(2*a0))*sin(th);
  elif typo = "p2z" then aa := (2*a0)^(-3/2)/sqrt(Pi)*(r/(2*a0))*exp(-r/(2*a0))*cos(th);
  elif typo = "s3" then aa := (3*a0)^(-3/2)/sqrt(Pi)*(1-2*r/(3*a0)+2/3*(r/(3*a0))^2)*exp(-r/(3*a0));
  elif typo = "p3x" then aa := (3*a0)^(-3/2)/sqrt(3/2*Pi)*(r/(3*a0))*(2-r/(3*a0))*exp(-r/(3*a0))*sin(th);
  elif typo = "p3z" then aa := (3*a0)^(-3/2)/sqrt(3/2*Pi)*(r/(3*a0))*(2-r/(3*a0))*exp(-r/(3*a0))*cos(th);
  elif typo = "d3z2" then aa := (3*a0)^(-3/2)/sqrt(6*Pi)*(r/(3*a0))^2*exp(-r/(3*a0))*(3*cos(th)^2-1);
  elif typo = "d3zx" then aa := (3*a0)^(-3/2)/sqrt(6*Pi)*(r/(3*a0))^2*exp(-r/(3*a0))*sin(2*th);
  else aa := 0.00;
 end if;
  aa := evalf(aa);
```

```
  return(aa);
 end proc: #amplitude

AmpliMax := proc(typo :: string) :: float;
 local
  aa :: float;
  if typo = "s1" then aa := a0^(-3/2)/sqrt(π);
  elif typo = "s2" then aa := (2·a0)^(-3/2)/sqrt(π);
  elif typo = "p2x" then aa := (2·a0)^(-3/2)/sqrt(π)·exp(-1);
  elif typo = "p2z" then aa := (2·a0)^(-3/2)/sqrt(π)·exp(-1);
  elif typo = "s3" then aa := (3·a0)^(-3/2)/sqrt(π);
  elif typo = "p3x" then aa := (3·a0)^(-3/2)/sqrt(3/2·π)·(2-sqrt(2))·exp(-2+sqrt(2));
  elif typo = "p3z" then aa := (3·a0)^(-3/2)/sqrt(3/2·π)·(2-sqrt(2))·exp(-2+sqrt(2));
  elif typo = "d3z2" then aa := (3·a0)^(-3/2)/sqrt(6·π)·8·exp(-2);
  elif typo = "d3zx" then aa := (3·a0)^(-3/2)/sqrt(6·π)·4·exp(-2);
  else aa := 0.00;
 end if;
  aa := evalf(aa);
  return(aa);
 end proc: #AmpliMax


a0 := 1.00;
```

1.00 **(1)**

```
orbitale := "d3zx"
```

"d3zx" **(2)**

```
pmax := (AmpliMax(orbitale))^2;
```

$$0.0005758073019 \tag{3}$$

```
xx := Array(1..5000, datatype = float, fill = 0.);
```

$$\begin{bmatrix} 1 .. 5000\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix} \tag{4}$$

```
zz := Array(1..5000, datatype = float, fill = 0.);
```

$$\begin{bmatrix} 1 .. 5000\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix} \tag{5}$$

```
ii := 0;
```

$$0 \tag{6}$$

```
limite := 10000000000
```

$$10000000000 \tag{7}$$

```
roll := rand(0..limite);
```

```
proc()
```

(8)

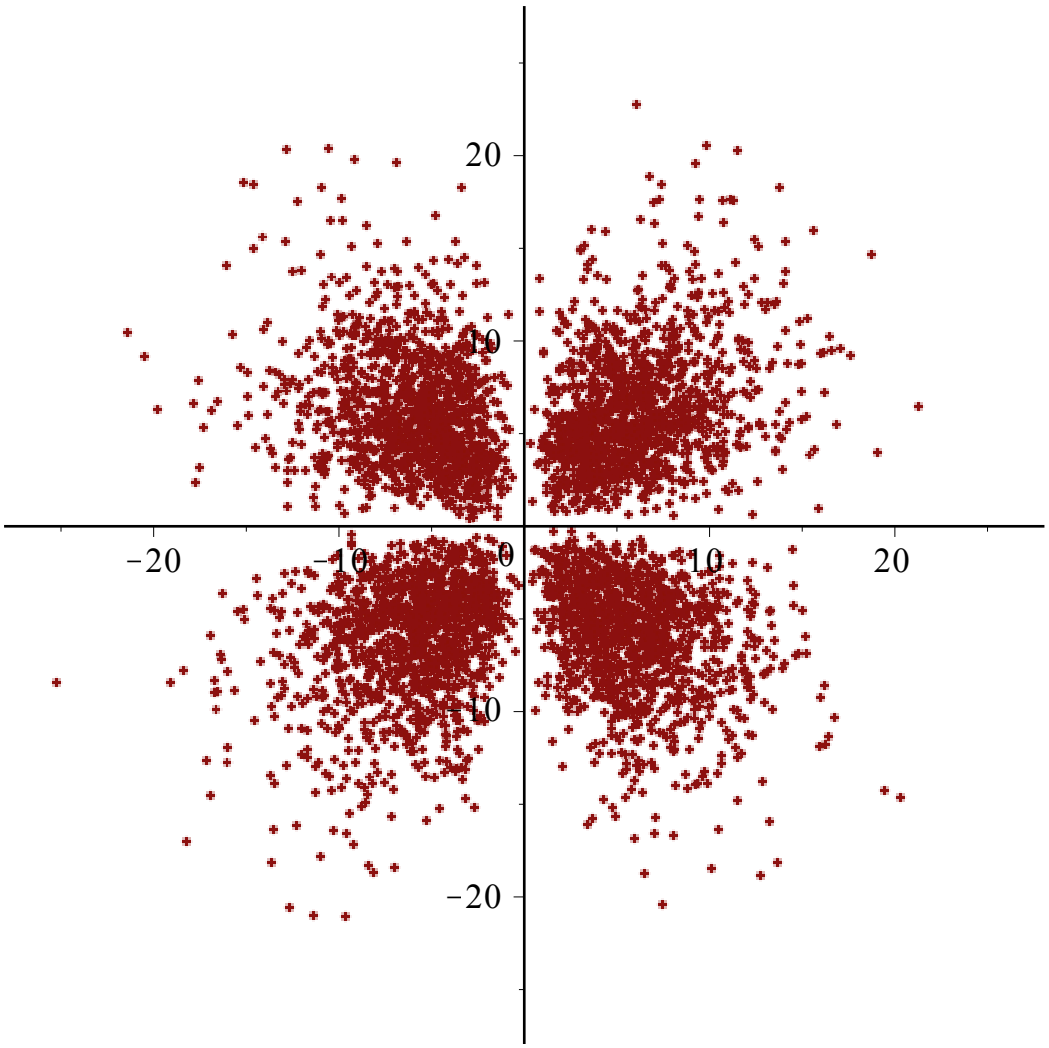
```
    proc() option builtin = RandNumberInterface; end proc(6, 10000000001, 34)
end proc
```

```
borne := 28.;
```

$$28. \tag{9}$$

```
for i from 1 while ii < 5000
 do
 x := evalf(borne·((2·roll())/limite − 1));
 z := evalf(borne·((2·roll())/limite − 1));
 psi := amplitude(x, z, orbitale);
 p := psi^2;
 test := evalf(roll()/limite);
 if (test < p/pmax) then
  ii := ii + 1;
  xx[ii] := x;
  zz[ii] := z;
 end if;
end do:

plot(xx, zz, view = [−borne..borne, −borne..borne], style = point, symbolsize = 5);
```


20
10
−20
−10
0
10
20
−10
−20