Computing the cumulative bivariate normal distribution using recursions
Axel Vogt, Jan 2005

about

Numerical procedures to compute the standard bivariate normal distribution N2(x,y,rho) through recursions. Exactness should be of the size of the used onedimensional cumulative normal distribution, hence set needed Digits.

For 'small' correlation a bivariate application of the univariate series due to Marsaglia is used and for 'large' correlation an approach of Vasicek is applied (i had problems with his recursion, so i build my own).

For that the standard BVN is transformed to the univariate case (cf Vasicek or Abramowitz & Stegun). That integrals have series for which the coefficients can be computed by recursion (based on the incomplete Gamma function or looking at the integrals by partial integration).

The main functions are

reduce_N2(x,y,rho) which reduces (symbolical) a bivariate normal integral, (cf Abramowitz & Stegun, § 26.3 [especially Eq 26.3.20], p 936/937) to the cases N2(x,0,rho_new) where one variable is 0 (by changing the correlation) and then expresses it as a one-dimensional integral N2_simplified(x,0,rho_new). It is enclosed for completeness only and not needed, but may serve for testing.

N2_as_sum(x,y,rho,nSteps::posint) which does the actual numerical computation: it first reduces to the case just described (and usually splits in two integrals) and then it calls N2_reduced_as_sum (having one variable less). This procedure cares for signs and calls N2_series. The later decides whether N2_Marsaglia_series or N2_Vasicek_series will be used, so these are the actual core.

So use N2_as_sum(1.0, 2.0, 0.8, 200) to compute the bivariate normal

$$\int_{-\infty}^{x}\int_{-\infty}^{y}\frac{e^{\left(-\frac{1}{2}\frac{(\xi^2-2\rho\xi\eta+\eta^2)}{(1-\rho^2)}\right)}}{\sqrt{1-\rho^2}\,\pi\,2}\,d\eta\,d\xi$$

for x= 1.0, y=2.0 and rho=0.8 with at most 200 recursions.

nSteps is the maximum number of iterations for the recursions to be used whithin the core solutions. Since both report when they quit (according to exactness used) that number gards against 'endless' loops. In the above example it will stop after around 2*20 steps for 12 digits with .839454198052 as result.

Proofs are not included. And i have not (yet?) done the error estimations for cutting series to sums (depending on the used recursion steps, Vasicek has it) ...

And for the 'Marsaglia series' one has probably not to reduce to the 'simplified' case, but can recurse directly (for correlation not too large).

At maple-new http://groups.yahoo.com/group/maple-new/message/212 i asked for help having trouble with a crash. The two case are discussed as extreme examples, and have been my motivation to write this sheet from snippets laying around.

Finally the limiting case |rho| = 1 is treated.

As i do not have Fortran and high precision integrators ... i hope i am not heavily wrong ...
any comments and hints are welcome.

References:

George Marsaglia, Evaluating the Normal Distribution, Journal of Statistical Software (2004),
http://www.jstatsoft.org/counter.php?id=100&url=v11/i04/v11i04.pdf&ct=1
for the univariate case

Oldrich Alfons Vasicek (1998), Moody's KMV,
http://www.moodyskmv.com/research/whitepaper/A_Series_Expansion_for_the_Bivariate_Normal_Integral.pdf
as series in 1-rho^2 (which is like Drezner & Wesolowsky, compare Genz below)

Abramowitz & Stegun ...

Related:

Alan Miller 'for classical' Fortran Code refering to Genz
http://users.bigpond.net.au/amiller/

Alan Genz, Numerical Computation of Rectangular Bivariate ...
http://www.sci.wsu.edu/math/faculty/genz/homepage

This code works, it is fast and good for ~ 14 - 15 decimal places.

```
> restart; kernelopts(version);
```

Maple 9.52, IBM INTEL NT, Dec 17 2004 Build ID 175529

```
> #signum(0); signum(0.); signum(-0.0); evalf(%);
  #eval(_Envsignum0); anames('environment');
  #_Envsignum0 := 0:
  Digits:= 14;
```

$$\text{Digits} := 14$$

Notations and conventions

```
> pdfN:= x -> exp(-x^2/2)/sqrt(2*Pi);
  N:= x -> (1+erf(x/sqrt(2)))/2;
```

$$\text{pdfN} := x \to \frac{e^{(-1/2\,x^2)}}{\sqrt{2\,\pi}}$$

$$N := x \to \frac{1}{2} + \frac{1}{2}\,\text{erf}\left(\frac{x}{\sqrt{2}}\right)$$

```
> pdfN2:= (x,y,rho) ->
  1/sqrt(1-rho^2)/(2*Pi)*exp(-(x^2-2*rho*x*y+y^2)/(2*(1-rho^2)));
  ``;
  N2:= (x,y,rho) ->

  1/sqrt(1-rho^2)/2/Pi*Int(Int(exp(-1/2*(xi^2-2*rho*xi*eta+eta^2)/((1-rho^2)
  )),   eta=-infinity..y),xi=-infinity..x);
```

$$\text{pdfN2} := (x, y, \rho) \to \frac{1}{2}\,\frac{e^{\left(-\frac{x^2 - 2\,\rho\,x\,y + y^2}{2 - 2\,\rho^2}\right)}}{\sqrt{1 - \rho^2}\,\pi}$$

$$N2 := (x, y, \rho) \to \frac{1}{2}\left(\frac{1}{\sqrt{1-\rho^2}\,\pi}\int_{-\infty}^{x}\int_{-\infty}^{y} e^{\left(-1/2\,\frac{\xi^2-2\rho\xi\eta+\eta^2}{1-\rho^2}\right)}\,d\eta\,d\xi\right)$$

```
> N2_simplified:= (x,y,rho) ->
  Int(pdfN(xi)*N((y-rho*xi)/sqrt(1-rho^2)),xi=-infinity..x);
  #``;
  #N2_reduced:= (x,rho) -> N2_simplified(x,0,rho);
```

$$N2\_simplified := (x, y, \rho) \to \int_{-\infty}^{x} pdfN(\xi)\, N\left(\frac{y-\rho\,\xi}{\sqrt{1-\rho^2}}\right) d\xi$$

Note that $N2(x, y, \rho) = N2\_simplified(x, y, \rho)$ (reference?)

```
> assume(-1<rho);additionally(rho<1);
  getassumptions(rho);
```

$$\{\rho::RealRange(Open(-1), Open(1))\}$$

```
>
```

code

```
> reduce_N2:=proc(x,y,rho)
  local r,rho_x,rho_y,result;

  description "reduce cdfN2 to N2_simplified(x,0,rho_new) by A&S 26.3.20";

  #if 1<abs(rho) then
  #  return N2(x,y,rho)
  #else
  #end if;

  if signum(rho)=0 then
    return N(x)*N(y);
  elif (signum(x)=0 and signum(y)=0) then
    return 1/4*(Pi+2*arctan(rho/(1-rho^2)^(1/2)))/Pi;
  else
  end if;

  if signum(x)=0 then
    return N2_simplified(y,0,rho);
  elif signum(y)=0 then
    return N2_simplified(x,0,rho);
  else
  end if;

  r:= ( 1/sqrt(x^2-2*rho*x*y+y^2));
  rho_x:=( (rho*x-y)*r*signum(x) );
  rho_y:=( (rho*y-x)*r*signum(y) );

  if signum(x*y) = -1 then
    result:=N2_simplified(x,0,rho_x)+
      N2_simplified(y,0,rho_y) - 1/2;
  #print(rho_x, rho_y);
  elif signum(x*y) =  1 then
    result:=N2_simplified(x,0,rho_x)+
      N2_simplified(y,0,rho_y);
  #print(rho_x, rho_y);
```

```
  else
    result:=N2(x,y,rho);
  end if;
  return (result);
  end proc; maplemint(%); ``;
```

numerical routines:

```
> N2_as_sum:=proc(x,y,rho,nSteps::posint)
  # decompose to get the reduced case y=0 by changing rho
  local r,rho_x,rho_y,result;

  description "reduce cdfN2 to N2_simplified(x,0,rho_new) by A&S 26.3.20";

  if 1 < abs(rho) then
  #  return FAIL;
  elif abs(rho) = 1 then
  #  return evalf( N2_absRhoEqualOne(x,y,rho) );
  else
  end if;

  if signum(rho)=0 then
    return evalf(N(x)*N(y));
  elif (signum(x)=0 and signum(y)=0) then
    return evalf(1/4*(Pi+2*arctan(rho/(1-rho^2)^(1/2)))/Pi);
  else
  end if;

  if signum(x)=0 then
    return evalf(N2_reduced_as_sum(y,rho,nSteps));
  elif signum(y)=0 then
    return evalf(N2_reduced_as_sum(x,rho,nSteps));
  else
  end if;

  r:= ( 1/sqrt(x^2-2*rho*x*y+y^2));
  rho_x:=( (rho*x-y)*r*signum(x) );
  rho_y:=( (rho*y-x)*r*signum(y) );

  if signum(x*y) = -1 then
    result:=N2_reduced_as_sum(x,rho_x,nSteps)+
      N2_reduced_as_sum(y,rho_y,nSteps) - 1/2;
  #print(rho_x, rho_y);
  elif signum(x*y) =  1 then
    result:=N2_reduced_as_sum(x,rho_x,nSteps)+
      N2_reduced_as_sum(y,rho_y,nSteps);
  #print(rho_x, rho_y);
  else
    result:=reduce_N2(x,y,rho);
  end if;
  return evalf(result);
  end proc; maplemint(%); ``;
```

```
> N2_reduced_as_sum:=proc(x,r,nSteps::posint)
  # reduce to positive variables (needed only for Vasicek)
  # assumed real variables

  description "prepare to call series computation with positive arguments";

  if type(x,numeric) and type(r,numeric) then
    # ok
```

```
  else
    # exit with a formal integral
    return N2_simplified(x,0,r);
  end if;

  if 1<=abs(r) then
    return N2_simplified(x,0,r);
  end if;

  # trivial cases first, signum(0)=0 assumed
  if signum(x) = 0 then
    return evalf( 1/4+1/2*1/Pi*arctan(r/(1-r^2)^(1/2)) );
  end if;
  if signum(r) = 0 then
    return evalf( N(x)/2 );
  end if;

  if (0<x) and 0<r then
    return N2_series(x,r,nSteps);
  elif (0<x) and 0<-r then
    return evalf( N(x) ) - N2_reduced_as_sum(x,-r,nSteps);
  elif 0<-x and 0<r then
    return evalf( 1/2 - N(-x) ) + N2_reduced_as_sum(-x,r,nSteps);
  elif 0<-x and 0<-r then
    return 0.5 - N2_reduced_as_sum(-x,-r,nSteps);
  else # something strange happend ...
    return N2_simplified(x,0,r);
  end if;
  end proc; maplemint(%); ``;
```

```
> N2_series:=proc(x,r,nSteps::posint)
  # now everthing is a float and strictly positive

  description "decide which series is used (with positive arguments)";

  if abs(r) <= evalf(1/sqrt(2)) then
    return N2_Marsaglia_series(x,-r,nSteps); # sign changes !
  elif abs(r) < 1.0 then
    return N2_Vasicek_series(x,r,nSteps);
  else
    return N2_simplified(x,0,r);
  end if;
  end proc; maplemint(%); ``;
```

```
> N2_Vasicek_series:=proc(x,rho,nSteps::posint)
  local beta0,A,B,k,sumA;

  description "cdfN2(x,0,rho) through int(pdfN2(x,0,r,r=rho..1) by recursion
  using nSteps";

  Digits:=Digits+3;

  beta0 := (1-rho^2)^(1/2)*signum(x)*exp(1/2/(rho+1)/(rho-1)*x^2);

  sumA:=0.0;
  A:= evalf(- (x*2*sqrt(2*Pi)*N(-abs(x)/(1-rho^2)^(1/2)) -2*beta0)); # A(0)
  #A:= evalf(- ((x)*2*sqrt(2*Pi)*N(-(x)/(1-rho^2)^(1/2)) -2*beta0)); # A(0)
  sumA:=evalf(sumA+A);

  B:= evalf(- 2*beta0*(1-rho^2)); # B(1)
  A:= evalf(-1/6*x^2*A-1/6*B); # A(1)
  sumA:=evalf(sumA+A);
```

```
  for k from 2 to nSteps do
    B:= evalf(-1/2*(-1+rho^2)*(2*k-1)/(k-1)*B); #
    A:= evalf((x^2*A*(-2*k+1)-B)*1/2/k/(1+2*k)); #
    sumA:=evalf(sumA+A);
    if sumA = sumA - A then break; end if; # sum does not change anymore
  end do;

  #return evalf(sumA/(4*Pi));
  print('iterations_Vasicek'=k);
  return 0.5 - evalf(sumA/(4*Pi));
  end proc; maplemint(%); ``;
```

```
> N2_Marsaglia_series:=proc(x::numeric,rho::numeric,nSteps::posint)
  local a::numeric, i::integer,A::numeric,B::numeric,sumA::numeric;

  description "int(pdfN(xi)*cdfN(rho*xi),xi=-infinity..x) by recursion using
  nSteps";

  Digits:=Digits+3;

  a:=evalf(rho/sqrt(1-rho^2)); # sign changes !

  A:=0; B:=0; sumA:=0;

  A:=evalf(-1/2*1/(1+a^2)*exp(-1/2*(1+a^2)*x^2)/Pi*a); # A(1)
  B:=evalf( (a*x)^2 * A); # B(1)

  sumA:=evalf(A);
  for i from 1 to nSteps-1 do
  #print(A);
    A:=evalf( (2*i*a^2/(1+a^2)*A + B)/(1+2*i)); # A(n+1)
    B:=evalf( (a*x)^2 * B /(1+2*i)); # B(n+1)
    sumA:=evalf(sumA + A);
    if sumA = sumA - A then break; end if; # sum does not change anymore
  end do;

  print('iterations_Marsaglia'=i);
  return sumA + 0.5*evalf(N(x));
  end proc;: maplemint(%); ``;
```

```
>
```

```
>
```

Examples

Set y=0 for first tests and call the 'core' directly

```
> xTst:=-0.20; rhoTst:=0.5; nTst:=30;
  ``;
  'N2_Marsaglia_series(xTst,-rhoTst,nTst)': '%'=%; # sign changes !
  'N2_reduced(xTst,rhoTst)': '%'= evalf(%);
  'eval(N2(xTst,0,rhoTst),infinity=80)': '%'= evalf(%);
```

$$xTst := -0.20$$

$$rhoTst := 0.5$$

$$nTst := 30$$

$$iterations\_Marsaglia = 28$$

$$N2\_Marsaglia\_series(xTst, -rhoTst, nTst) = 0.29188598360845697$$

$$N2\_reduced(xTst, rhoTst) = N2\_reduced(-0.20, 0.5)$$

$$N2(xTst, 0, rhoTst)\Big|_{\infty = 80} = 0.29188598360844$$

Do the same, but with high correlation

```
> xTst:=-3.20; rhoTst:=0.9; nTst:=30;
  ``;
  'N2_Marsaglia_series(xTst,-rhoTst,nTst)': '%'=%;
  'N2_reduced(xTst,rhoTst)': '%'= evalf(%);
  'eval(N2(xTst,0,rhoTst),infinity=80)': '%'= evalf(%);
```

$$xTst := -3.20$$

$$rhoTst := 0.9$$

$$nTst := 30$$

$$iterations\_Marsaglia = 30$$

$$N2\_Marsaglia\_series(xTst, -rhoTst, nTst) = 0.00060904187843913935$$

$$N2\_reduced(xTst, rhoTst) = N2\_reduced(-3.20, 0.9)$$

$$N2(xTst, 0, rhoTst)\Big|_{\infty = 80} = 0.00068713793791305$$

So increase number of steps and compare the two methods

```
> xTst:=-3.20; yTst:=0; rhoTst:=0.9; nTst:=300;
  ``;
  'N2_Marsaglia_series(xTst,-rhoTst,nTst)': '%'=%;
  'N2_Vasicek_series(xTst,rhoTst,nTst)': '%'=%;
  'N2_as_sum(xTst,yTst,rhoTst,nTst)': '%'=%;
```

$$xTst := -3.20$$

$$yTst := 0$$

$$rhoTst := 0.9$$

$$nTst := 300$$

$$iterations\_Marsaglia = 196$$

$$N2\_Marsaglia\_series(xTst, -rhoTst, nTst) = 0.00068713793791317066$$

$$iterations\_Vasicek = 32$$

$$N2\_Vasicek\_series(xTst, rhoTst, nTst) = 0.50000000000000268$$

$$iterations\_Vasicek = 32$$

$$N2\_as\_sum(xTst, yTst, rhoTst, nTst) = 0.00068713793792$$

which shows that

- Marsaglia will converge after more iterations (196, but have patients for very high rho),
- it is neccessary to care for signs (if calling directly the 'Vasicek' routine),
- and combining is better

And just that is, was the procedures around the two methods do.

```
> xTst:=-1.20; yTst:=1.7; rhoTst:=0.9; nTst:=30;
  ``;
  'N2_as_sum(xTst,yTst,rhoTst,nTst)': '%'=%;
  'reduce_N2(xTst,yTst,rhoTst)': '%'= evalf(%);
```

$$xTst := -1.20$$

$$yTst := 1.7$$

$$rhoTst := 0.9$$

$$nTst := 30$$

$$iterations\_Vasicek = 13$$

$$iterations\_Vasicek = 16$$

$$N2\_as\_sum(xTst, yTst, rhoTst, nTst) = 0.11506967022053$$

$$reduce\_N2(xTst, yTst, rhoTst) = 0.11506967022055$$

Now take extreme variables at an average correlation:

```
> xTst:=0.001; yTst:=5; rhoTst:=0.5; nTst:=30;
  ``;
  'N2_as_sum(xTst,yTst,rhoTst,nTst)': '%'=%;
  'reduce_N2(xTst,yTst,rhoTst)': '%'= evalf(%);
  'eval(N2(xTst,yTst,rhoTst),infinity=80)': '%'= evalf(%);
```

$$xTst := 0.001$$

$$yTst := 5$$

$$rhoTst := 0.5$$

$$nTst := 30$$

$$iterations\_Vasicek = 3$$

$$iterations\_Marsaglia = 30$$

$$N2\_as\_sum(xTst, yTst, rhoTst, nTst) = 0.50039894180000$$

$$reduce\_N2(xTst, yTst, rhoTst) = 0.49999999958606$$

$$N2(xTst, yTst, rhoTst)\Big|_{\infty = 80} = 0.50039894180001$$

This is somewhat strange: greater 1/2 or not?

Actually it is above 1/2 i think, but Maple needs too much time to compute that.

And it may be worth to think whether it is really a good to reduce the integral allways in that form ...

Run through (some) combinations [and generating by random would be better]:

```
> remDigits:=Digits: Digits:=14:
  yTst:=2; nTst:=40;
  for xTst in {1.0} do #{0.01, 0.1, 1.0, 3.0} do #
  for rTst in {0.5} do #{0.7, 0.8, 0.9, 0.95, 0.99} do #

  for sx in {-1,+1} do
    for sr in {-1,+1} do
      print(sx,sr);

  `error`=N2_as_sum(sx*xTst,yTst,sr*rhoTst,nTst)-evalf(reduce_N2(sx*xTst,yTs
  t,sr*rhoTst));
      print(%);
    end do; # sr
  end do; # sx
  sx:='sx': sr:='sr':

  end do; # rTst
  end do; # xTst
  Digits:=remDigits:
```

$$yTst := 2$$

$$nTst := 40$$

$$-1, -1$$

$$iterations\_Vasicek = 25$$

$$error = -0.1\ 10^{-13}$$

$$-1, 1$$

$$iterations\_Vasicek = 17$$

$$iterations\_Vasicek = 40$$

$$error = -0.1\ 10^{-13}$$

1, -1

iterations_Vasicek = 17

iterations_Vasicek = 40

error = $-0.2\ 10^{-13}$

1, 1

iterations_Vasicek = 25

error = $-0.2\ 10^{-13}$

For extreme correlation use high numbers of steps and look, what happens close below the the needed steps:

```
> remDigits:=Digits: Digits:=100:
  'N2_as_sum(1,2,0.000000001,1000)': '%'=%;
  'N2_as_sum(1,2,0.000000001,100)': '%'=%;
  Digits:=remDigits:
```

iterations_Vasicek = 143

iterations_Marsaglia = 148

$$N2\_as\_sum(1, 2, 0.1\ 10^{-8}, 1000) = 0.8222040420946405005141472283874342904237633784909642507\backslash 1\backslash$$
$$038206529377466966806377097591880369315827 71$$

iterations_Vasicek = 101

iterations_Marsaglia = 100

$$N2\_as\_sum(1, 2, 0.1\ 10^{-8}, 100) = 0.8222040420946405005141472283874342904237633784909642507\backslash 10\backslash$$
$$38206529377467055757985312114712509128399 30$$

Compare reported steps and use some more steps (of course it does not change):

```
> remDigits:=Digits: Digits:=100:
  'N2_as_sum(1,2,0.999999999,1000)': '%'=%;
  'N2_as_sum(1,2,0.999999999,100)': '%'=%;
  Digits:=remDigits:
```

iterations_Vasicek = 61

iterations_Vasicek = 84

$$N2\_as\_sum(1, 2, 0.999999999, 1000) = 0.8413447460685429485852325456320379224779129667266 04\backslash$$
$$3909873944502429914419872048295008849184056393276$$

iterations_Vasicek = 61

iterations_Vasicek = 84

$$N2\_as\_sum(1, 2, 0.999999999, 100) = 0.8413447460685429485852325456320379224779129667266 043\backslash$$
$$9098739445024299144198720482950088491840563932 76$$

```
> 
```

```
> 
```

extreme example

Now attack my questions posted at maple-new on 29 Dec 2004,
http://groups.yahoo.com/group/maple-new/message/212
where i got answers from Robert Israel and Chris 'CW'.

That was to compute Int(pdfN(x)*N((b-rho*x)/sqrt(1-rho^2)),x=-infinity..a)
for a=b=2 and rho = 0.999999999 or rho= - 0.999999999 (9 times the 9).

Using the notation above this is N2_simplified(a,b,rho) and restricting to
14 digits Maple computes it (via the NAG library?) but has difficulties with
higher exactness.

Chris CW reports (with 128 digits):

$$.9772489047859669278612243461924684438407637586050528388420816755126968058670756710957424 2\backslash$$

39571193227656325154730715568922105458 4
and N2_as_sum would be the same (128 digits and after 2*15 steps)
.977248904785966927861224346192468443840763758605052838842081675512696805867075671095742 42\
39571193227656325154730715568922105458 4

Chris CW (with 96 digits, not sure that i understood his answer right):
.954494886386402429208688647347934284261823176219001417827270552206855420921731296582001 08\
5575261
Robert Israel (with 35 digits):
.954499736103641585599434725666933125056447552596643132032667999739047419294448503303461 69\
and N2_as_sum would be (96 digits, after 2*82 steps, so using almost no time)
.954499736103641585599434725666933125056447552596643132032667999739047419294448503303461 69\
5848420

In the following i want to know, which one is 'correct'.

```
> 'N2_simplified(a,b,rho)'=
  'Int(pdfN(x)*N((b-rho*x)/sqrt(1-rho^2)),x=-infinity..a)';
  subs(x=xi,%): is(%);
```

$$\text{N2\_simplified}(a, b, \rho) = \int_{-\infty}^{a} \text{pdfN}(x)\, N\left(\frac{b - \rho\, x}{\sqrt{1 - \rho^2}}\right) dx$$

true

The above numerical procedures can compute it for both rho:

```
> remDigits:=Digits: Digits:=100:
  'N2_as_sum(2,2,+0.999999999,100)'; evalf(%,96);
  ``;
  'N2_as_sum(2,2,-0.999999999,100)'; evalf(%,96);
  Digits:=remDigits:
```

N2_as_sum(2, 2, 0.999999999, 100)

iterations_Marsaglia = 11

iterations_Marsaglia = 11

0.977248904785966927861224346192468443840763758605052838842081675512696805867075671095742423957118

N2_as_sum(2, 2, -0.999999999, 100)

iterations_Vasicek = 82

iterations_Vasicek = 82

0.954499736103641585599434725666933125056447552596643132032667999739047419294448503303461695848420

The second result differs from the value reported by Chris, which was

```
> 0.954494886386402429208688647347934284261823176219001417827270552206855420
  92173129658200108557526 1;
```

0.954494886386402429208688647347934284261823176219001417827270552206855420921731296582001085575261

For the rest let us take a closer look at this.

First compare, what a direct use of numerical integration for the standard (double integral)
and reduced version would give for that data at 14 digits

```
> rTst:=-0.9;
  'N2_simplified(2,2,rTst)': '%'=evalf(%);
```

```
'reduce_N2(2,2,rTst)': '%'=evalf(%);
'N2(2,2,rTst)': '%'= evalf(subs(infinity=40,%));
```

$$rTst := -0.9$$

$$N2\_simplified(2, 2, rTst) = 0.95449973610366$$

$$reduce\_N2(2, 2, rTst) = 0.95449973610366$$

$$N2(2, 2, rTst) = 0.95449973610364$$

```
> rTst:=-0.999999999;
```

$$rTst := -0.999999999$$

```
> 'N2_simplified(2,2,rTst)': '%'=evalf(%);
'reduce_N2(2,2,rTst)': '%'=evalf(%);
'N2(2,2,rTst)': '%'= evalf(%);
```

$$N2\_simplified(2, 2, rTst) = 0.95449973610366$$

$$reduce\_N2(2, 2, rTst) = 0.95449973610366$$

$$N2(2, 2, rTst) = 0.$$

So for usual exactness that is ok (of course), if one does not work with double integrals, but with the reduced form (but may be not without troubles, cf above).

Estimate, where to cut off integration 'certainly':

```
> 'reduce_N2(2,2,rho)': '%'= simplify(%);

  remDigits:=Digits: Digits:=100:
  myProblem:='reduce_N2(2,2,rTst)';:
  Digits:=14:
  `myProblem at 14 digits`=evalf(myProblem);
  Digits:=remDigits:
```

$$reduce\_N2(2, 2, \rho) = 2 \int_{-\infty}^{2} \frac{1}{4} \frac{e^{\left(-\frac{\xi^2}{2}\right)} \sqrt{2} \left(1 + \operatorname{erf}\left(\frac{\xi \sqrt{1-\rho}}{\sqrt{2\rho+2}}\right)\right)}{\sqrt{\pi}} d\xi$$

$$myProblem := reduce\_N2(2, 2, rTst)$$

$$myProblem \text{ at } 14 \text{ digits} = 0.95449973610366$$

```
> remDigits:=Digits: Digits:=80:
  #'integrand(myProblem/2)'; %;
  #plot(%,xi=-3..2);
  'fsolve(integrand(myProblem/2)=1e-1000, xi)': '%'=%;
  rhs(%):
  cutoff:=max(70,%);
  myProblem_cutted_off:='(subs(infinity=cutoff,myProblem))';
  Digits:=remDigits:
```

$$fsolve\left(integrand\left(\frac{myProblem}{2}\right) = 0.1\ 10^{-999}, \xi\right) =$$

$$67.847861491145629346578759672681429299943104365344897748801139045423650639180761$$

$$cutoff := 70$$

$$myProblem\_cutted\_off := subs(\infty = cutoff, myProblem)$$

and check, what the integral would be with the cut off used (at 45 digits):

```
> remDigits:=Digits: Digits:=35:
  'N2_as_sum(2,2,rTst,100)'; %;
  ``;
  myProblem_cutted_off; evalf(%);
  Digits:=remDigits:
```

$$N2\_as\_sum(2, 2, rTst, 100)$$

$$iterations\_Vasicek = 41$$

$$\text{iterations\_Vasicek} = 41$$

$$0.954499736103641585599434725666933 12$$

$$2\int_{-70}^{2}\frac{1}{2}\,\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(\frac{1}{2}+\frac{1}{2}\,\mathrm{erf}(22360.679769407727019643490308719260\,\xi\sqrt{2})\right)}{\sqrt{\pi}}\,d\xi$$

$$0.954499736103641585599434725666933 12$$

Hence for 35 digits it is 'ok' and for more i lost patients with Maple.

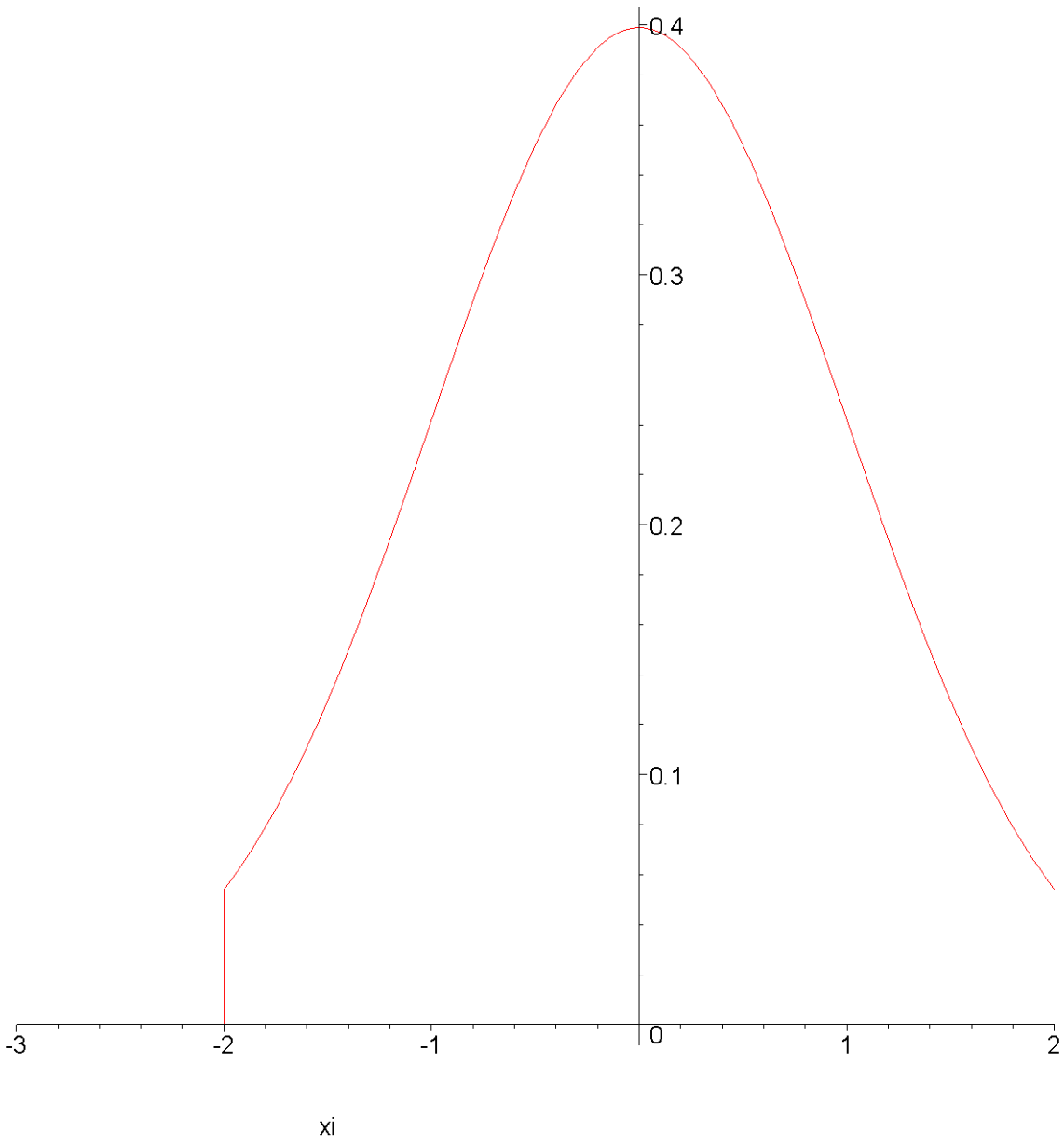
But then i wanted to know it ...

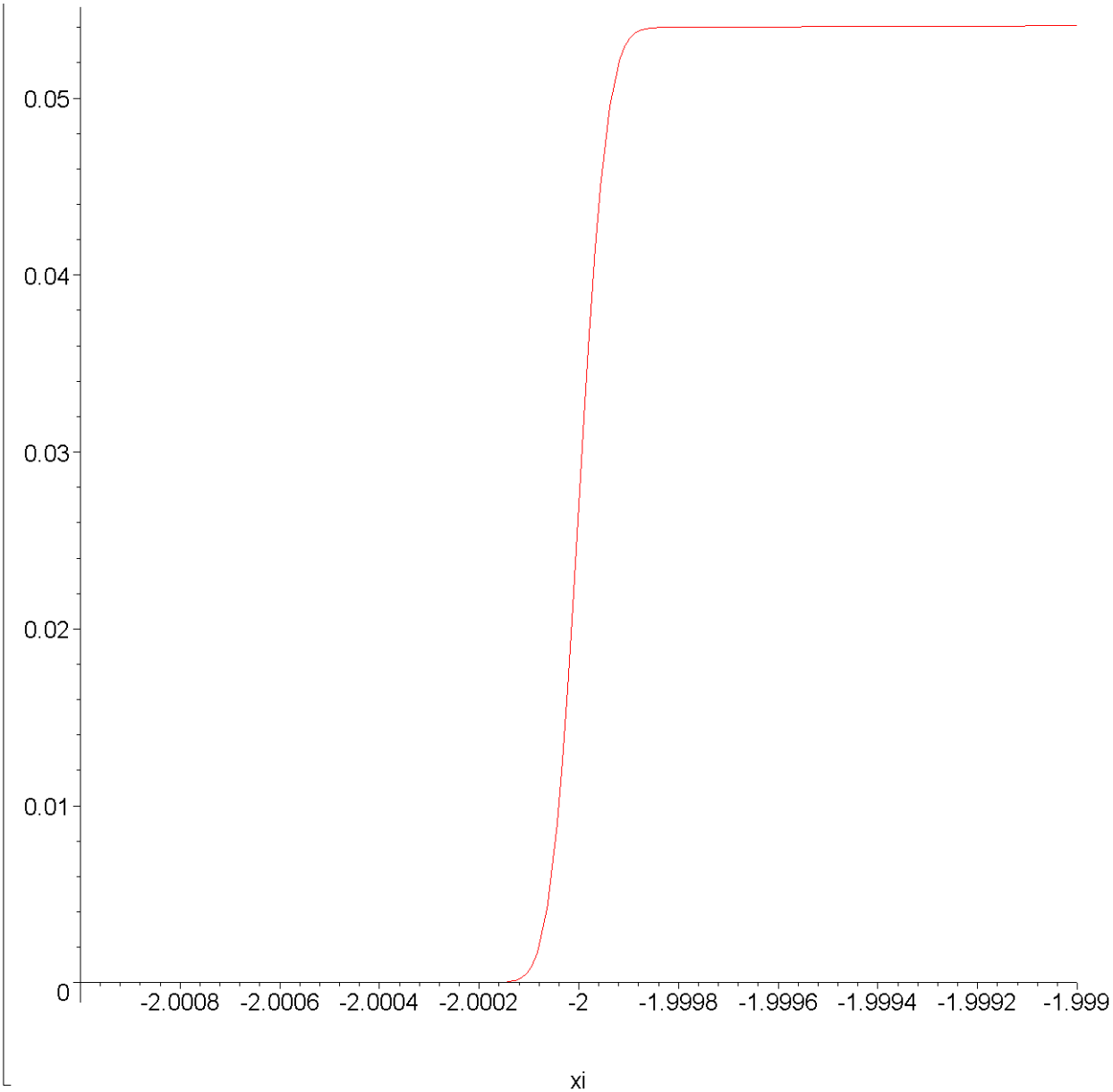
Let us see, why Maple has troubles with the chosen parameter:

```
> 'rTst': '%'=%;
  'eval(integrand(N2_simplified(2,2,rho)), rho=rTst)';
  plot(%,xi=-3..2);
  plot(%%,xi=-2-1e-3..-2+1e-3);
```

$$\text{rTst} = -0.999999999$$

$$\text{integrand}(\text{N2\_simplified}(2, 2, \rho))\Big|_{\rho = \text{rTst}}$$


0.4
0.3
0.2
0.1
0
-3
-2
-1
1
2
xi



As i asserted (and use, but did not prove explicitely): N2(2,2,rho) = reduce_N2(2,2,rho)
and i want to use the later one, calling the integrand A_r:

```
> ''reduce_N2(2,2,rho)/2'': %= simplify(eval(%));
    rhs(%):
    A_r:= integrand(%);
```

$$\frac{1}{2}\,\mathrm{reduce\_N2}(2,\,2,\,\rho) = \int_{-\infty}^{2} \frac{1}{4}\,\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(1+\mathrm{erf}\left(\frac{\xi\sqrt{1-\rho}}{\sqrt{2\rho+2}}\right)\right)}{\sqrt{\pi}}\,d\xi$$

$$A\_r := \frac{1}{4}\,\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(1+\mathrm{erf}\left(\frac{\xi\sqrt{1-\rho}}{\sqrt{2\rho+2}}\right)\right)}{\sqrt{\pi}}$$

but this itself is a N2_simplified

```
> ''N2_simplified(2,0,-beta)'': %= simplify(eval(%));
    rhs(%):
    A_s:= integrand(%);
```

$$\mathrm{N2\_simplified}(2,\,0,\,-\beta) = \int_{-\infty}^{2} \frac{1}{4}\,\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(1+\mathrm{erf}\left(\frac{\beta\,\xi\,\sqrt{2}}{2\sqrt{1-\beta^2}}\right)\right)}{\sqrt{\pi}}\,d\xi$$

$$A\_s := \frac{1}{4}\,\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(1+\mathrm{erf}\left(\frac{\beta\,\xi\,\sqrt{2}}{2\sqrt{1-\beta^2}}\right)\right)}{\sqrt{\pi}}$$

and calling the integrand A_s one determines the beta so they become equal:

```
> #A_r/A_s; simplify(%);
  (xi*(1-rho)^(1/2)/(2*rho+2)^(1/2))=(1/2*beta*xi/(1-beta^2)^(1/2)*2^(1/2));

  %/xi: [solve(%,beta)];
  beta1:=%[1];
```

$$\frac{\xi\,\sqrt{1-\rho}}{\sqrt{2\,\rho+2}} = \frac{\beta\,\xi\,\sqrt{2}}{2\sqrt{1-\beta^2}}$$

$$\left[\frac{\sqrt{-2\,\rho+2}}{2},\,-\frac{\sqrt{-2\,\rho+2}}{2}\right]$$

$$\beta 1 := \frac{\sqrt{-2\,\rho+2}}{2}$$

check, that with $\beta 1$ the desired indentity holds:

```
> 'reduce_N2(2,2,rho)'-'2*N2_simplified(2,0,-beta1)'=0;
  simplify(%): is(%);
```

$$\mathrm{reduce\_N2}(2,\,2,\,\rho) - 2\,\mathrm{N2\_simplified}(2,\,0,\,-\beta 1) = 0$$

true

```
> A_t:='eval(A_s,beta=beta1)'; simplify(%);
```

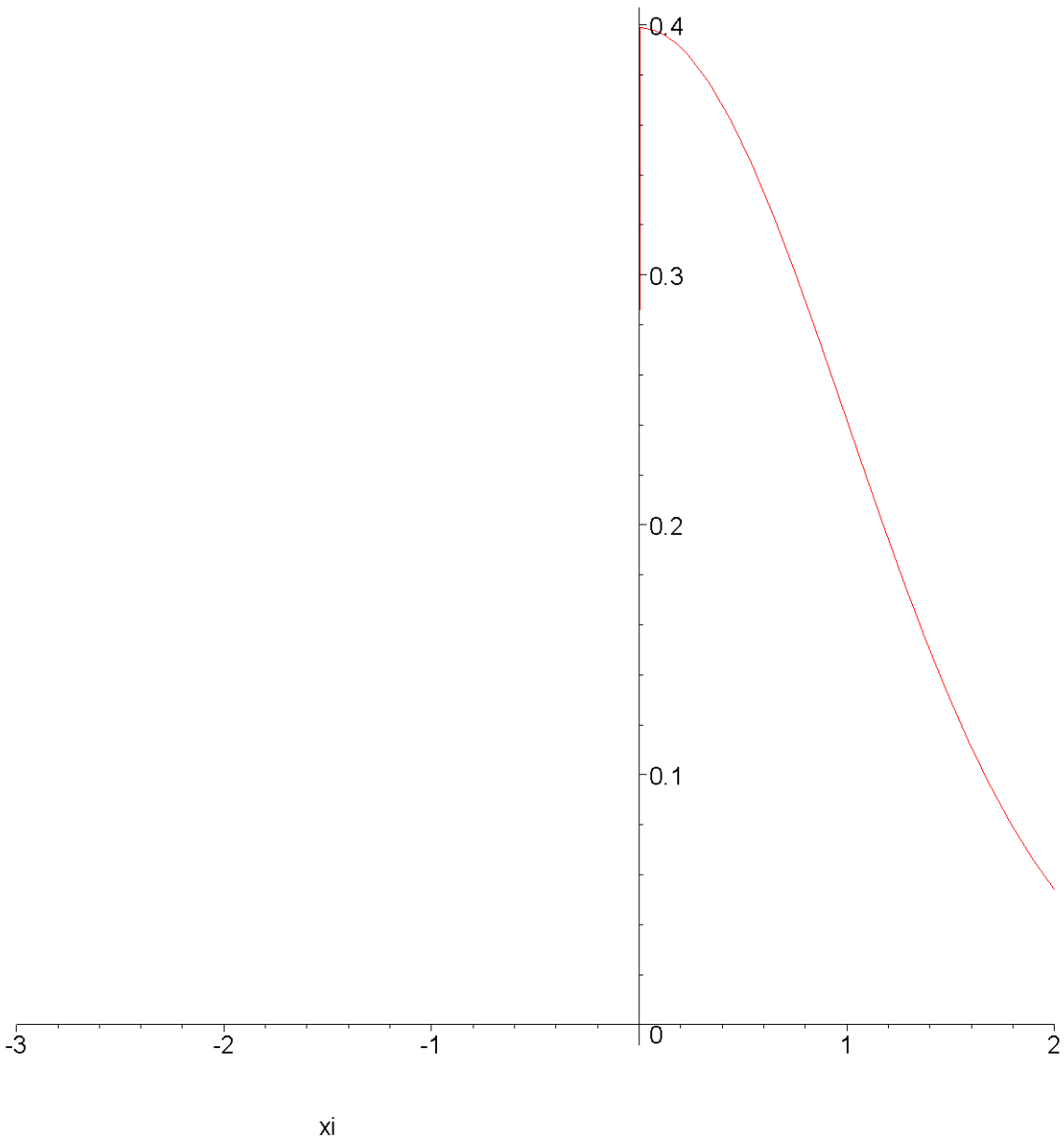
$$A\_t := A\_s\Big|_{\beta=\beta 1}$$

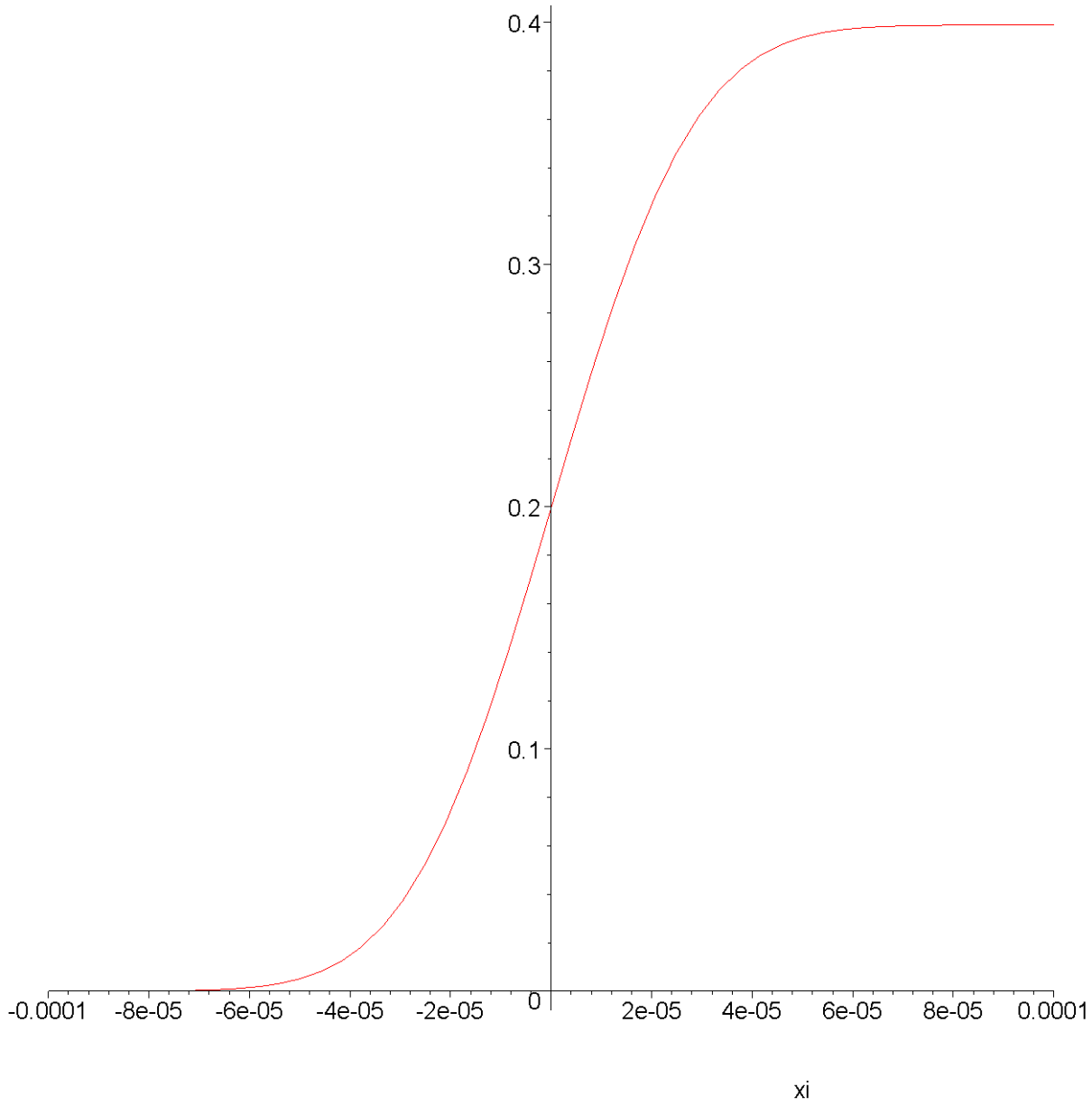
$$\frac{1}{4}\,\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(1+\mathrm{erf}\left(\frac{\sqrt{-2\,\rho+2}\,\xi}{2\sqrt{\rho+1}}\right)\right)}{\sqrt{\pi}}$$

Let us see, where Maple may have troubles for the translated task:

```
> 'eval(A_r,[rho=rTst])';
  plot(%,xi=-3..2);
  plot(%%,xi=-1e-4..1e-4);
```

$$A\_r\Big|_{\rho=\mathrm{rTst}}$$


0.4
0.3
0.2
0.1
0
-3
-2
-1
1
2
xi



So i split integration at 0

```
> 2*(N2_simplified(0,0,-beta) +
  Int(1/2*exp(-1/2*xi^2)*2^(1/2)/Pi^(1/2)*(1/2+1/2*erf(1/2*beta*xi/(1-beta^2
  )^(1/2)*2^(1/2))),xi = 0 .. 2));
```

$$2\left(\int_{-\infty}^{0}\frac{1}{2}\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(\frac{1}{2}+\frac{1}{2}\operatorname{erf}\left(\frac{\beta\,\xi\sqrt{2}}{2\sqrt{1-\beta^2}}\right)\right)}{\sqrt{\pi}}d\xi+\int_{0}^{2}\frac{1}{2}\frac{e^{\left(-\frac{\xi^2}{2}\right)}\sqrt{2}\left(\frac{1}{2}+\frac{1}{2}\operatorname{erf}\left(\frac{\beta\,\xi\sqrt{2}}{2\sqrt{1-\beta^2}}\right)\right)}{\sqrt{\pi}}d\xi\right)$$

The first integral $N2\_simplified(0, 0, -\beta)$ writes as

```
> reduce_N2(0,0,-beta);
  eval(%,beta=beta1):
  lowerPart:=simplify(%);;
```

$$\frac{1}{4}\frac{\pi-2\arctan\left(\frac{\beta}{\sqrt{1-\beta^2}}\right)}{\pi}$$

$$\text{lowerPart} := \frac{1}{4}\,\frac{\pi - 2\arctan\left(\frac{\sqrt{1-\rho}}{\sqrt{\rho+1}}\right)}{\pi}$$

while the second integral now is evaluated by Maple:

```
> 'eval(Int(1/2*exp(-1/2*xi^2)*2^(1/2)/Pi^(1/2)*(1/2+1/2*erf(1/2*beta*xi/(1-
  beta^2)^(1/2)*2^(1/2))),xi = 0 .. 2), beta=beta1)':
  remainingPart:=simplify(%);
```

$$\text{remainingPart} := \frac{1}{4}\left(\frac{\sqrt{2}}{\sqrt{\pi}}\int_0^2 \mathbf{e}^{\left(-\frac{\xi^2}{2}\right)}\left(1+\operatorname{erf}\left(\frac{\sqrt{-2\rho+2}\,\xi}{2\sqrt{\rho+1}}\right)\right)d\xi\right)$$

hoping, that Maple can do it easily - which is not quite the case,
but splitting again helps:

```
> remDigits:=Digits: Digits:=96:
  'eval(1/4*2^(1/2)/Pi^(1/2)*Int(exp(-1/2*xi^2)*(1+erf(1/2*(-2*rho+2)^(1/2)*
  xi/(rho+1)^(1/2))),xi = 0 .. 1e-2),rho=rTst)';
  remaining1:=evalf(%);
  Digits:=remDigits:
```

$$\left(\frac{1}{4}\left(\frac{\sqrt{2}}{\sqrt{\pi}}\int_0^{0.01} \mathbf{e}^{\left(-\frac{\xi^2}{2}\right)}\left(1+\operatorname{erf}\left(\frac{\sqrt{-2\rho+2}\,\xi}{2\sqrt{\rho+1}}\right)\right)d\xi\right)\right)\Bigg|_{\rho=\text{rTst}}$$

$$\text{remaining1} := 0.00398579750191422124877112354356871456747998154425421675626724642053861892334409066037349764867722$$

calling Maple for the below with that much digits crashes with an error in "LIBGMP-3.DLL"

```
> remDigits:=Digits: Digits:=96:
  'eval(1/4*2^(1/2)/Pi^(1/2)*Int(exp(-1/2*xi^2)*(1+erf(1/2*(-2*rho+2)^(1/2)*
  xi/(rho+1)^(1/2))),xi =  1e-2..2),rho=rTst)';
  %;:
  #remaining2:=evalf(%);
  #erf(31622.776593778099168*0.01);
  Digits:=remDigits:
```

$$\left(\frac{1}{4}\left(\frac{\sqrt{2}}{\sqrt{\pi}}\int_{0.01}^2 \mathbf{e}^{\left(-\frac{\xi^2}{2}\right)}\left(1+\operatorname{erf}\left(\frac{\sqrt{-2\rho+2}\,\xi}{2\sqrt{\rho+1}}\right)\right)d\xi\right)\right)\Bigg|_{\rho=\text{rTst}}$$

$$\frac{1}{4}\left(\frac{\sqrt{2}}{\sqrt{\pi}}\int_{0.01}^2 \mathbf{e}^{\left(-\frac{\xi^2}{2}\right)}\left(1+\operatorname{erf}\left(31622.776593778099168579775345280279992586242146540916331217380475797860041687964219994179873814 6\,\xi\right)\right)d\xi\right)$$

but the argument in erf approaches 1 with increasing xi and even for 10000 digits it 'is' already 1 for xi=0.01. So i set it to 1. And only need an integral, which is the difference of error functions:

```
> 1/4*2^(1/2)/Pi^(1/2)*Int(exp(-1/2*xi^2)*(1+1),xi =  1e-2..2);
  remaining2:=evalf(%,96);
```

$$\frac{1}{4}\left(\frac{\sqrt{2}}{\sqrt{\pi}}\int_{0.01}^{2} 2\,\mathbf{e}^{\left(-\frac{\xi^2}{2}\right)}\,d\xi\right)$$

remaining2 := 0.473260511737189189097927364925002998407841577225718959244925631746702679\
81450969388409908496038 8

Now add the things and compare with the recursive procedure

```
> remDigits:=Digits: Digits:=96:
  '2*(eval(lowerPart,rho=rTst)+remaining1+remaining2)'; evalf(%);
  ``;
  'N2_as_sum(2,2,rTst,100)'; evalf(%,96);
  Digits:=remDigits:
```

$$2\,\text{lowerPart}\Big|_{\rho=\text{rTst}} + 2\,\text{remaining1} + 2\,\text{remaining2}$$

0.954499736103641585599434725666933125056447552596643132032667999739047419294448503303\
461695848419

$$\text{N2\_as\_sum}(2,\,2,\,\text{rTst},\,100)$$

iterations_Vasicek = 82

iterations_Vasicek = 82

0.954499736103641585599434725666933125056447552596643132032667999739047419294448503303\
461695848420

which makes me happy :-)

At least almost ... taking -1 instead of -0.999999999 and using A&S 26.3.16 the result should be N(-2) - N(2) = - erf(sqrt(2)), which is the above - but we have opposite sign:

```
> eval(N(-x)-N(y), [x=2,y=2]); evalf(%,96); #evalf(-%); identify(%);
```

$$-\text{erf}(\sqrt{2})$$

-0.954499736103641585599434725666933125056447552596643132032667999739047419294448503303\
461695848421

```
>
```

```
>
```

appendix: abs(rho) = 1

I think the sign in A&S 26.3.16 actually is false, it has to be (-1) times their result, the 3 other cases are correct (but my proofs are too ugly to include them):

```
> N2_absRhoEqualOne:=proc(x,y,rho)
  # limiting case abs(rho) = 1 for N2(x,y,rho)

  if rho = 1 then
    if x <= y then
      return N(x)
    elif y <= x then
      return N(y)
    else
    end if;
  elif rho = - 1 then
    if -y <= x then
      return N(x)+N(y)-1
    elif x <= -y then
      return 0
    else
```

```
    end if;
  else
  end if;

  end proc; maplemint(%); ``;
```

Abramowitz & Stegun is as follows:

```
> N2_absRhoEqualOne_AS:=proc(x,y,rho)
  # A&S 26.3.14 ff, N2(x,y,rho) = L(-x,-y,rho), P(t) = N(t), Q(t) = N(-t)

  if rho = -1 then
    if -x-y >= 0 then       #   26.3.16
      return 0
    else                    #   26.3.16
      return N(-x)-N(-(-y)) # = P(-x) - Q(-y)
    end if;
  elif rho = 1 then
    if -y <= -x then        #   26.3.17
      return N(-(-x))       # = Q(-x)
    else                    #   26.3.18
      return N(-(y))        # = Q(-y)
    end if;
  else
  end if;

  end proc; maplemint(%); ``;
```

compare that procedures using some test data

```
> xTst:=1; yTst:='xTst'; rTst:=1-(1e-9); nStepsTst:=100:
  ``;
  remDigits:=Digits: Digits:=20:
  'N2_as_sum(xTst,yTst,rTst, nStepsTst)'; %;
  'N2_absRhoEqualOne(xTst,yTst,signum(rTst))'; evalf(%); ``;
  'N2_absRhoEqualOne_AS(xTst,yTst,signum(rTst))'; evalf(%);
  Digits:=remDigits:
```

$$xTst := 1$$

$$yTst := xTst$$

$$rTst := 0.999999999$$

$$N2\_as\_sum(xTst, yTst, rTst, nStepsTst)$$

$$iterations\_Marsaglia = 3$$

$$iterations\_Marsaglia = 3$$

$$0.84134042901049277296$$

$$N2\_absRhoEqualOne(xTst, yTst, signum(rTst))$$

$$0.84134474606854294858$$

$$N2\_absRhoEqualOne\_AS(xTst, yTst, signum(rTst))$$

$$0.84134474606854294858$$

```
> xTst:=-1; yTst:='xTst'; rTst:=1-(1e-9); nStepsTst:=100:
  ``;
  remDigits:=Digits: Digits:=20:
  'N2_as_sum(xTst,yTst,rTst, nStepsTst)'; %;
  'N2_absRhoEqualOne(xTst,yTst,signum(rTst))'; evalf(%); ``;
  'N2_absRhoEqualOne_AS(xTst,yTst,signum(rTst))'; evalf(%);
  Digits:=remDigits:
```

$$xTst := -1$$

$$yTst := xTst$$

rTst := 0.999999999

N2_as_sum(xTst, yTst, rTst, nStepsTst)

iterations_Marsaglia = 3

iterations_Marsaglia = 3

0.15865093687340687580

N2_absRhoEqualOne(xTst, yTst, signum(rTst))

0.15865525393145705142

N2_absRhoEqualOne_AS(xTst, yTst, signum(rTst))

0.15865525393145705142

```
> xTst:=-1; yTst:='xTst'; rTst:=-(1-(1e-9)); nStepsTst:=100:
  ``;
  remDigits:=Digits: Digits:=20:
  'N2_as_sum(xTst,yTst,rTst, nStepsTst)'; %;
  'N2_absRhoEqualOne(xTst,yTst,signum(rTst))'; evalf(%); ``;
  'N2_absRhoEqualOne_AS(xTst,yTst,signum(rTst))'; evalf(%);
  Digits:=remDigits:
```

xTst := -1

yTst := xTst

rTst := -0.999999999

N2_as_sum(xTst, yTst, rTst, nStepsTst)

iterations_Vasicek = 19

iterations_Vasicek = 19

0.

N2_absRhoEqualOne(xTst, yTst, signum(rTst))

0.

N2_absRhoEqualOne_AS(xTst, yTst, signum(rTst))

0.

```
> xTst:=1; yTst:='xTst'; rTst:=-(1-(1e-2)); nStepsTst:=100:
  ``;
  remDigits:=Digits: Digits:=20:
  'N2_as_sum(xTst,yTst,rTst, nStepsTst)'; %; ``;
  'reduce_N2(xTst,yTst,rTst)'; evalf(%); ``;
  'N2_absRhoEqualOne(xTst,yTst,signum(rTst))'; evalf(%); ``;
  'N2_absRhoEqualOne_AS(xTst,yTst,signum(rTst))'; evalf(%);
  Digits:=remDigits:
```

xTst := 1

yTst := xTst

rTst := -0.99

N2_as_sum(xTst, yTst, rTst, nStepsTst)

iterations_Vasicek = 19

iterations_Vasicek = 19

0.68268949213708589716

reduce_N2(xTst, yTst, rTst)

0.68268949213708589718

N2_absRhoEqualOne(xTst, yTst, signum(rTst))

0.68268949213708589717

N2_absRhoEqualOne_AS(xTst, yTst, signum(rTst))

-0.68268949213708589717

[>

[>