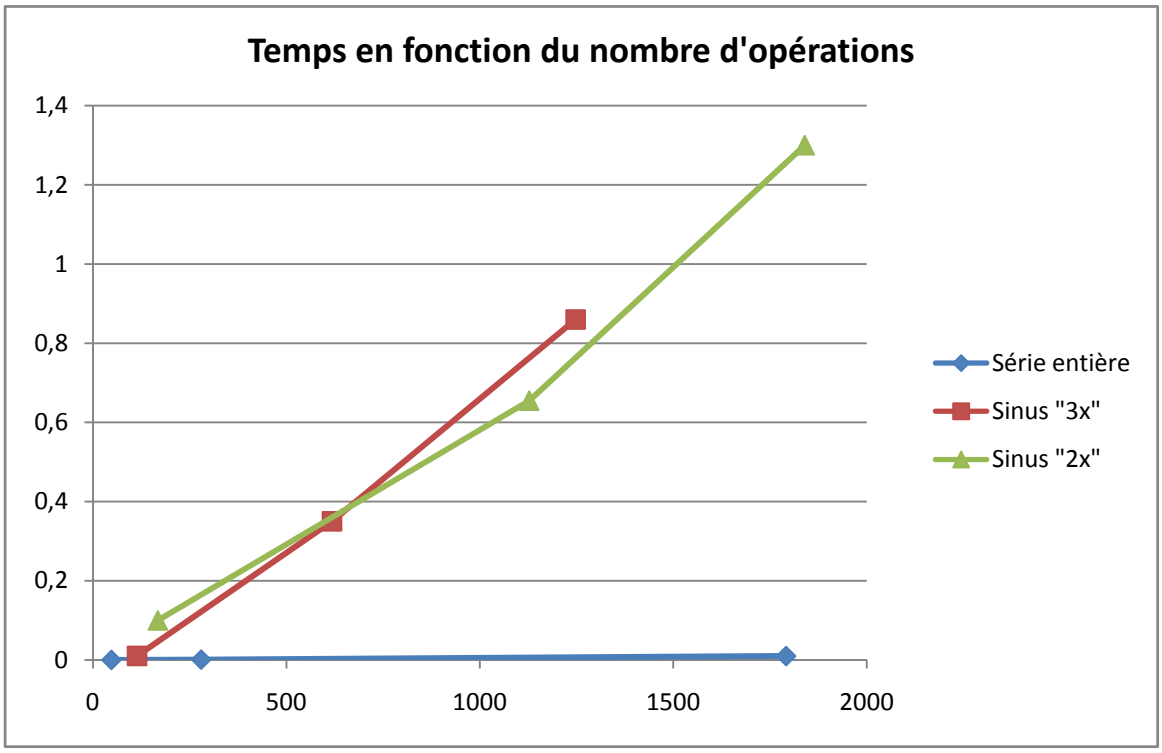
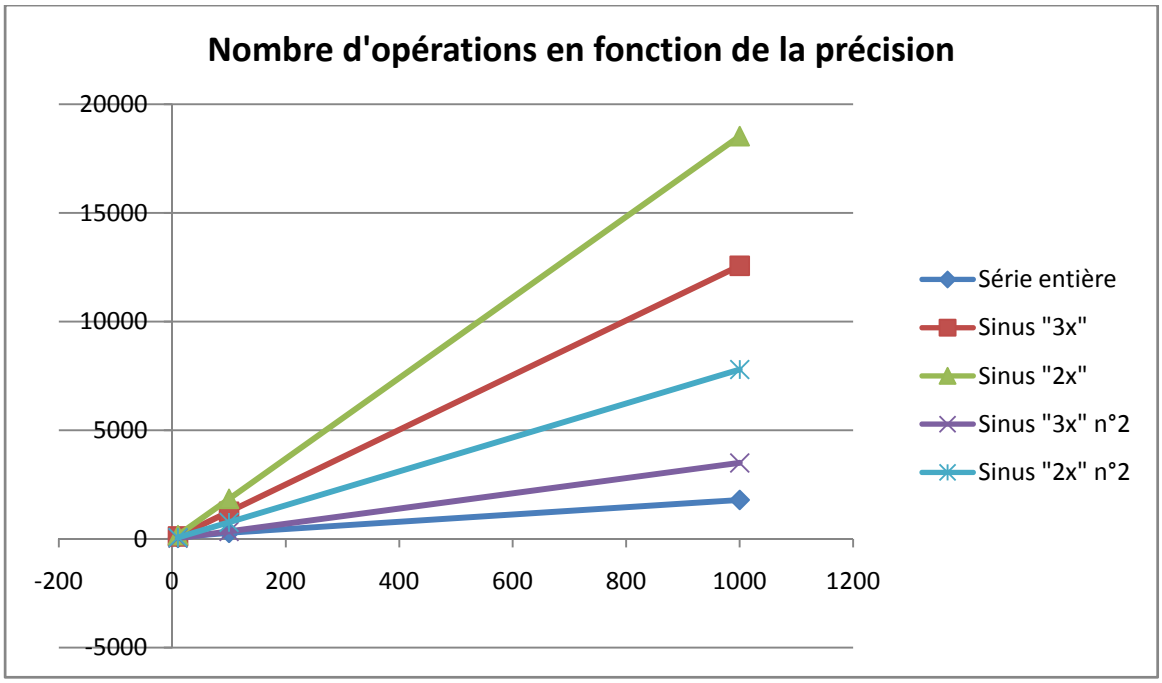
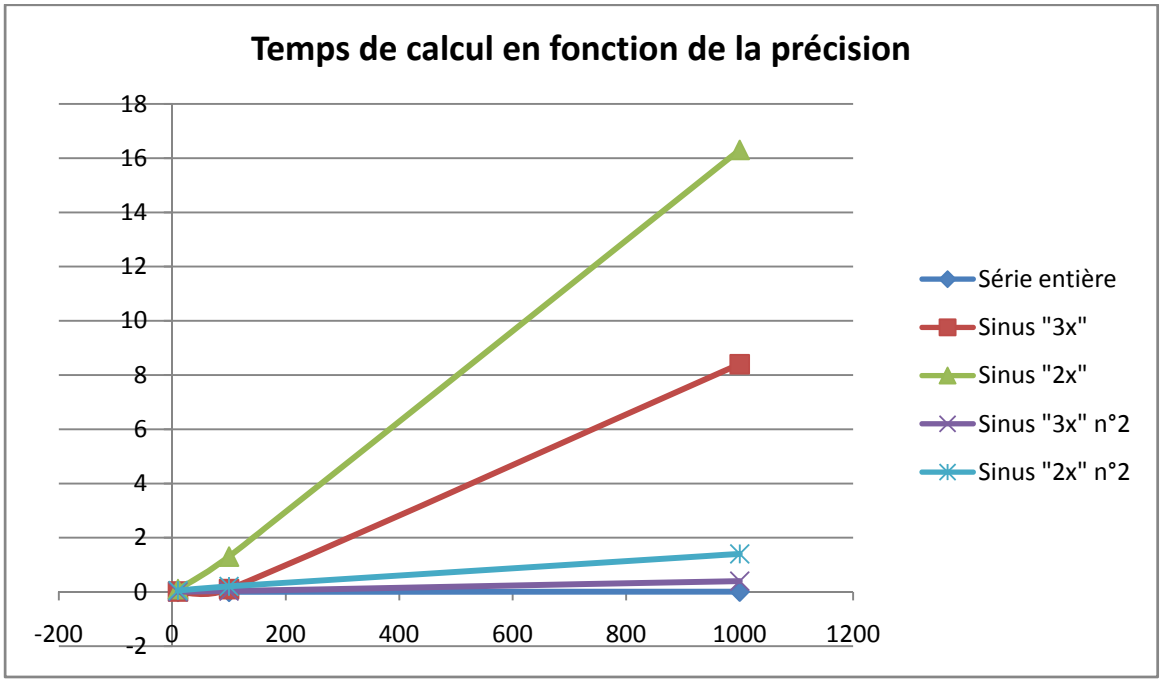
	Série entière			Sinus « 3x »			Sinus « 2x »		
Précision	10^{-10}	10^{-100}	10^{-1000}	10^{-10}	10^{-100}	10^{-1000}	10^{-10}	10^{-100}	10^{-1000}
Nb. It.	6	35	224	19	208	2094	21	230	2316
Nb. Op.	48	280	1792	114	1248	12564	168	1840	18528
Temps	$10^{-4}s$	$10^{-3}s$	$10^{-2}s$	$10^{-2}s$	$10^{-1}s$	8,4s	$10^{-1}s$	$1{,}3s$	$16{,}3s$

	Sinus « 3x » n°2			Sinus « 2x » n°2		
Précision	10^{-10}	10^{-100}	10^{-1000}	10^{-10}	10^{-100}	10^{-1000}
Nb. It.	5	57	374	7	85	866
Nb. Op.	30	342	2244	49	765	7794
Temps	$10^{-2}s$	$0{,}03s$	$0{,}4s$	$0{,}04s$	$0{,}15s$	$1{,}4s$




Nombre d'opérations en fonction de la précision
20000
15000
10000
5000
0
-5000
-200
0
200
400
600
800
1000
1200
Série entière
Sinus "3x"
Sinus "2x"
Sinus "3x" n°2
Sinus "2x" n°2


Temps de calcul en fonction de la précision
18
16
14
12
10
8
6
4
2
0
-2
-200
0
200
400
600
800
1000
1200
Série entière
Sinus "3x"
Sinus "2x"
Sinus "3x" n°2
Sinus "2x" n°2

Sinus “3x”:

```
> rang:=proc(x,eps)
  local n:
  n:=evalf((3*ln(x)-ln(6)-ln(eps))/(3*ln(3)-ln(4.7))):
  n:=ceil(n):
end:

> sinus3_aux := proc (x,alpha)
  local n,b,a:
  b:=evalf(alpha):
  if x <= b then RETURN(x):
  else
    a:=sinus3_aux (x/3,alpha):
    RETURN(evalf(-4*a^3+3*a)):
  fi
end:

sinus3:=proc(x,eps)
  local n:
  n:=rang(x,eps):
  RETURN(sinus3_aux(x,x/3^n)):
end:
```

Sinus “3x” n°2:

```
> rang:=proc(x,eps)

  local n:
  n:=evalf((5*ln(x)-ln(120)-ln(eps))/(5*ln(3)-ln(4.7))):
  n:=ceil(n):
end:

> sinus3_aux := proc (x,alpha)
  local n,b,a:
  b:=evalf(alpha):
  if x <= b then RETURN(x-x^3/6):
  else
    a:=sinus3_aux (x/3,alpha):
    RETURN(evalf(-4*a^3+3*a)):
  fi
end:

sinus3:=proc(x,eps)
  local n:
  n:=rang(x,eps):
  RETURN(sinus3_aux(x,x/3^n)):
end:
```

Sinus “2x”:

```
> rang:=proc(x,eps)
  local n:
  n:=evalf((3*ln(x)-ln(eps)-ln(6))/(3*ln(2)-ln(9/4))):
  n:=ceil(n):
end:

> sinus2_aux := proc (x,alpha)
  local n,b,a:
  b:=evalf(alpha):
  if x <= b then RETURN(x):
  else
    a:= sinus2_aux (x/2,alpha):
    RETURN(evalf(2*a*sqrt(1-a*a))):
  fi
end:

sinus2:=proc(x,eps)
  local n:
  n:=rang(x,eps):
  RETURN(sinus2_aux(x,x/2^n)):
end:
```

Sinus “2x” n°2:

```
> rang:=proc(x,eps)

  local n:
  n:=evalf((3*ln(x)-ln(eps)-ln(120))/(5*ln(2)-ln(9/4))):
  n:=ceil(n):
end:

> sinus2_aux := proc (x,alpha)
  local n,b,a:
  b:=evalf(alpha):
  if x <= b then RETURN(x-x^3/6):
  else
    a:= sinus2_aux (x/2,alpha):
    RETURN(evalf(2*a*sqrt(1-a*a))):
  fi
end:

sinus2:=proc(x,eps)
  local n:
  n:=rang(x,eps):
  RETURN(sinus2_aux(x,x/2^n)):
end:
```

Logarithme népérien:

```
> rel_fonct_ln := proc(x,eps)
  local k,a:
  if evalf(x) < 1 then RETURN(-1*rel_fonct_ln(1/x,eps)) fi:
  if evalf(x-1) <= evalf(eps) then
    RETURN(evalf(x-1)):
  else
    a:=rel_fonct_ln(sqrt(x),eps):
    RETURN(2*a):
  fi:
end:
```

Série entière du sinus:

```
> rang:=proc(x,eps)
  local n,S,A,B:
  S:=evalf(ln(eps)):
  n:=0:
  A:=x:
  B:=A:
  while (S < B) do
    n:=n+1:
    S:=S+evalf(ln(2*n+1)+ln(2*n+2)):
    B:=A*(2*n+1):
  od:
  n:
end:
```

```
> calcul_se:=proc(x,eps)
  local k,S,n:
  S:=0:
  n:=rang(0.8,eps):
  for k from n to 1 by -1 do
    S:=(x^2/(2*k*(2*k+1)))*(x-S):
  od:
  S:=x-S:
  evalf(S):
end:
```

Algorithme du Cordic :

```
> cordic:=proc(a)
  local angle,x,y,z,r,i,b:
  angle:=NULL:
  angle:={.785398163,.099668652,.009999667,.001,.0001,10^(-5),10^(-6),10^(-7),10^(-8)}:
  b:=a:
  x:=1:
  y:=0:
  r:=1:
  for i from 0 to 8 do
    while (b>=angle[i+1]) do
      z:=x-y/10^i:
      y:=y+x/10^i:
      x:=z:
      r:=r*sqrt(1+10^(-2*i)):
      b:=b-angle[i+1]:
    od:
  od:
{evalf(x/r),evalf(y/r)}:
end:
```

Série entière de l’exponentielle:

```
> rang:=proc(x,eps)

  local n,S,A:
  A:=evalf(ln(eps)):
  n:=1:
  S:=evalf(n*ln(x)-sum(ln(k),k=1..n)):
  while (S > A) do
    n:=n+1:
    S:=evalf(n*ln(x)-sum(ln(k),k=1..n)):
  od:
  n:
end:

> calcul_se:=proc(x,eps)
  local n,k,S:
  S:=0:
  n:=rang(x,eps):
  for k from n to 1 by -1 do
    S:=x/k*(S+1):
  od:
  S:=S+1:
  evalf(S):
end:
```

Série entière du logarithme népérien:

```
> ln2:=proc(n)

  local i,S,U:
  S:=0:
  U:=1/2:
  for i from 1 to n do
    S:=S+U/i:
    U:=U/2:
  od:
  RETURN(evalf(S)):
end:

> se_ln:=proc(x)
  local i,S,U:
  S:=0:
  U:=x:
  for i from 1 to 3306 do
    S:=S+U/i:
    U:=-U*x:
  od:
  RETURN(evalf(S)):
end:

> ln_2:=proc(ln2,x)
  local U:
  U:=evalf(x):
  if (U>3/2) then
    RETURN(ln_2(ln2,U/2) + ln2):
  else
  if (U<1/2) then
    RETURN(-ln_2(ln2,1/U)):
  else
    RETURN(se_ln(U-1)):
  fi:
  fi:
end:
```