0 Introduction

```
> MethodeNaive:=proc(n,a)
  evalf(sum((-1)^k*a(k),k=0..n));
  end;
```

proc(n, a) evalf(sum((-1)^k*a(k), k = 0 .. n)) end proc

Calcul de Log(2) en utilisant la somme des (-1)^n / (n+1).

```
> MethodeNaive(100,k->1/(k+1));
```

0.69807316940920510423474489259447416494827129284001

```
> MethodeNaive(1000,k->1/(k+1));
```

0.69364643155882130866823005996492748070286435631103

```
> evalf(log(2));
```

0.6931471806

1 Calcul des Pn

Pn(X) est une variante du polynôme de Chebyshev Tn(X) définie par
Pn(sin t) = cos(2nt), si t est un réel. (Pn(X) = Tn(1 - 2X)).
P0(X) = 1,
P1(X) = 1 - 2X,
Pn+2(X) = 2(1 - 2X)Pn+1(X) - Pn(X).

```
> Pn:=proc(n) option remember;
  if n=0 then
       1;
  else
       if n=1 then
            1-2*X;
       else
            expand(2*(1-2*X)*Pn(n-1)-Pn(n-2));
       fi;
  fi;
  end;
```

proc(n)
option remember;
if n = 0 then 1
else if n = 1 then 1 − 2∗X else expand(2∗(1 − 2∗X)∗Pn(n − 1) − Pn(n − 2)) end if
end if
end proc

```
> Pn(0);
```

1

```
> Pn(1);
```

$$1 - 2\,X$$

```
> Pn(5);
```

$$1 - 50\,X + 400\,X^2 - 1120\,X^3 + 1280\,X^4 - 512\,X^5$$

```
> with(orthopoly);
```

[G, H, L, P, T, U]

Une variante pour définir la suite de polynômes Pn
en utilisant la suite Tn déjà définie dans Maple dans
package orthopoly et la relation Pn(X) = Tn(1 - 2X)

```
> Pn2:=proc(n)
  expand(T(n,1-2*X));
  end;
```

proc(n) expand(T(n, 1 − 2∗X)) end proc

```
> Pn2(5);
```

$$1 - 50\,X + 400\,X^2 - 1120\,X^3 + 1280\,X^4 - 512\,X^5$$

Une dernière définition de Pn(X) via la relation (2) du 4ème paragraphe.

```
> Pn3:=proc(n)
  sum((-1)^k*n/(n+k)*binomial(n+k,2*k)*2^(2*k)*X^k,k=0..n);
  end;
```

proc(n) sum((-1)^k∗n∗binomial(n + k, 2∗k)∗2^(2∗k)∗X^k / (n + k), k = 0 .. n) end proc

```
> Pn3(5);
```

$$1 - 50\,X + 400\,X^2 - 1120\,X^3 + 1280\,X^4 - 512\,X^5$$

```
> Pn4:=proc(n)
  expand(sum((-1)^k*binomial(2*n,2*k)*(1-X)^(n-k)*X^k,k=0..n));
  end;
```

proc(n) expand(sum((-1)^k∗binomial(2∗n, 2∗k)∗(1 − X)^(n − k)∗X^k, k = 0 .. n)) end proc

```
> Pn4(5);
```

$$1 - 50\,X + 400\,X^2 - 1120\,X^3 + 1280\,X^4 - 512\,X^5$$

```
> f:=unapply(Pn(6),X);
```

$$X \to 1 - 72\,X + 840\,X^2 - 3584\,X^3 + 6912\,X^4 - 6144\,X^5 + 2048\,X^6$$

```
> plot(f(x),x=0..1);
```

```
> with(plots):
```

2 Mise en oeuvre

Calcul de dn via la relation 2*dn = (3+2*2^(1/2))^n+(3-2*2^(1/2))^n

```
> dn:=proc(n)
  simplify(((3+sqrt(8))^n+(3-sqrt(8))^n)/2);
  end;
```

proc(n) simplify(1 / 2*(3 + sqrt(8))^n + 1 / 2*(3 − sqrt(8))^n) end proc

Calcul direct de dn:=Pn(-1)

```
> dn2:=proc(n)
  local f;
  f:=unapply(Pn(n),X);
  f(-1);
  end;
```

proc(n) local f; f := unapply(Pn(n), X); f(-1) end proc

```
> dn2(5);
```

3363

```
> dn(5);
```

3363

Calcul des coefficients de Pn

```
> pk:=proc(n,k)
  coeff(Pn(n),X,k);
  end;
```

proc(n, k) coeff(Pn(n), X, k) end proc

```
> pk(5,5);
```

-512

Construction conjointe de pk et de ck via les relations de récurrence
$p_0 = 1$, $p_{k+1} = p_k * (k+n)*(k-n) / ((k+1/2)*(k+1))$
$c_0 = dn - p0$, $c_k = - p_k - c_{k-1}$
Le programme renvoit le couple $[p_k,c_k]$
qui dépend de n et de k.

```
> pkck:=proc(n,k)
  local p,c,d,i;
  d:=dn(n);
  p:=1;
  c:=d-p;
  for i from 0 to k-1 do
      p:=p*(i+n)*(i-n)/((i+1/2)*(i+1));
      c:=-p-c;
  od;
  [p,c];
  end;
```

proc(n, k)
local p, c, d, i;
d := dn(n);
p := 1;
c := d − p;

```
    for i from 0 to k − 1 do p := p*(i + n)*(i − n) / ((i + 1 / 2)*(i + 1)); c := −p − c end do;
    [p, c]
end proc
> pkck(2,2);
```

[8, 0]

SerieAlt prend en argument la suite (Ak) et renvoit
une valeur approchée de la somme des (-1)^k * Ak.

```
> SerieAlt:=proc(n,a)
  local S,k,p,c,d;
  d:=dn(n);
  p:=1;
  c:=d-p;
  S:=c*a(0);
  for k from 1 to n-1 do
      p:=p*(k-1+n)*(k-1-n)/((k-1/2)*(k));
      c:=-p-c;
      S:=S+c*a(k);
  od;
  evalf(S/d);
  end;
```

```
proc(n, a)
local S, k, p, c, d;
    d := dn(n);
    p := 1;
    c := d − p;
    S := c*a(0);
    for k to n − 1 do p := p*(k − 1 + n)*(k − 1 − n) / ((k − 1 / 2)*k); c := −p − c; S := S + c*a(k)
    end do;
    evalf(S / d)
end proc
```

3 Exemples de somme de séries alternées

```
> Digits:=50;
```

50

Méthode naive de calcule des sommes partielles des séries alternées.

```
> MethodeNaive:=proc(n,a)
  evalf(sum((-1)^k*a(k),k=0..n));
  end;
```

proc(n, a) evalf(sum((-1)^k*a(k), k = 0 .. n)) end proc

Log(2)

Calcul de Log(2) en utilisant la somme des (-1)^n / (n+1).

```
> ln2:=proc(n)
  SerieAlt(n,k->1/(k+1));
  end;
```

proc(n) SerieAlt(n, k → 1 / (k + 1)) end proc

```
> ln2(100);
```

0.693147180559945309417232121458176568075500134360 26

```
> evalf(log(2));
```

0.693147180559945309417232121458176568075500134360 26

```
> MethodeNaive(10,k->1/(k+1));
```

0.736544011544011544011544011544011544011544011544 01

Pi

Calcul d' une valeur approchée du nombre Pi en utilisant la somme des (-1)^n / (2n+1).

```
> SuitePi:=proc(n)
  4*SerieAlt(n,k->1/(2*k+1));
  end;
```

proc(n) 4∗SerieAlt(n, k → 1 / (2∗k + 1)) end proc

```
> SuitePi(25);
```

3.14159265358979323846252724156984661245560737630 93

```
> evalf(Pi);
```

3.14159265358979323846264338327950288419716939937 51

```
> 4*MethodeNaive(25,k->1/(2*k+1));
```

3.10314531288601160745816887627797705891242294746 11

Constante d'Euler

Calcul de la constante d'Euler.

```
> Gamma:=proc(n)
  local l;
  l:=ln2(n);
  SerieAlt(n,k->-log(k+1)/(k+1))/l+l/2;
  end;
```

proc(n) local l; l := ln2(n); SerieAlt(n, k → –log(k + 1) / (k + 1)) / l + 1 / 2∗l end proc

```
> Gamma(10);
```

0.577215664998221383161049485360464815197688572847 40

```
> evalf(gamma);
```

0.577215664901532860606512090082402431042159335939 92

4 la fonction Zeta

La fonction zeta de Riemann et définie par f(s):=somme n>=O 1/(n+1)^s

```
> zeta:=proc(n,s)
  1/(1-2^(1-s))*SerieAlt(n,k->1/(k+1)^s);
  end;
```

proc(n, s) SerieAlt(n, k → 1 / (k + 1)^s) / (1 − 2^(1 − s)) end proc

```
> s:=2;
  zeta(50,2);
```

2

1.64493406684822643647241516664602518921898524376 39

```
> evalf(Zeta(2));
```

1.64493406684822643647241516664602518921894990120 68

```
> evalf(Pi^2/6);
```

1.64493406684822643647241516664602518921894990120 68

```
> s:=4;
  zeta(10,4);
```

4

1.08232323478922393686793898326094021784404327206 10

```
> evalf(Zeta(4));
```

1.08232323371113819151600369654116790277475095191 87

```
> evalf(Pi^4/90);
```

1.08232323371113819151600369654116790277475095191 87

```
> s:=0;
  zeta(10,0);
> evalf(Zeta(0));
```

0

-0.49999997789521509657779467369292306911498674796040

-0.50000000000000000000000000000000000000000000000000

5 Séries à termes positifs

Methode de sommation de séries à termes positifs

Methode naive pour évaluer la somme des séries à termes positifs.

```
> Sn:=proc(n,a)
  evalf(sum(1/k^2,k=1..n));
  end;
```

proc(n, a) evalf(sum(1 / k^2, k = 1 .. n)) end proc

Methode pour évaluer la somme des séries à termes positifs utilisant la méthode naive pour

évaluer les séries alternées.

```
> SnBis:=proc(n,a) option remember;
  evalf(sum((-1)^(m-1)*sum(2^k*a(2^k*m),k=0..n),m=1..n));
  end;
```

proc(n, a)
option remember;
    evalf(sum((-1)^(m − 1)*sum(2^k*a(2^k*m), k = 0 .. n), m = 1 .. n))
end proc

```
> bm:=proc(m,n,a)
  evalf(sum(2^k*a(2^k*m),k=0..n));
  end;
```

proc(m, n, a) evalf(sum(2^k*a(2^k*m), k = 0 .. n)) end proc

Methode pour évaluer la somme des séries à termes positifs utilisant la nouvelle méthode pour évaluer les séries alternées.

```
> SeriePos:=proc(n,a)
  SerieAlt(n,m->bm(m+1,n,a));
  end;
```

proc(n, a) SerieAlt(n, m → bm(m + 1, n, a)) end proc

Somme des 1/k

Somme des 1/k

```
> n:=10;
  a:=k->1/k^2;
  SeriePos(n,a);
```

$$\sum_{k=1}^{10} \frac{1}{k^2}$$

1.644130878694505187599725425903868775747945637989 9

```
> Sn(10,k->1/k^2);
```

1.549767731166540690350214159737969261778785588309 4

```
> SnBis(10,k->1/k^2);
```

1.635125560034850166918619299571680524061476442428 8

```
> evalf(Pi^2/6);
```

1.644934066848226436472415166646025189218949901206 8

Somme des (-1)^(n+1) / n

```
> SerieAlt(25,k->1/(k+1)^2);
```

0.82246703342411321823702126071233997094380066647475

```
> evalf(Pi^2/12);
```

0.82246703342411321823620758332301259460947495060340

Somme des 1/(2n+1)

```
> a:=k->1/(2*k-1)^2;
  n:=10;
  SeriePos(n,a);
```

$$k \to \frac{1}{(2\,k-1)^2}$$

$$10$$

1.23349967775253117175411061817660799642824051744 03

```
> evalf(Pi^2/8);
```

1.23370055013616982735431137498451889191421242590 51

Typesetting:-mrow(Typesetting:-mn(
"1.23370055013616982735431137498451889191421242590 51", mathvariant = "normal"))

Somme des 1/k^4

```
> n:=15;
  a:=k->1/k^4;
  SeriePos(n,a);
```

$$15$$

$$k \to \frac{1}{k^4}$$

1.08232323371136772404632998783276343353178906243 62

```
> s:=4;
  zeta(15,4);
```

$$4$$

1.08232323371137156923088727781399602956325646446 45

```
> evalf(Pi^4/90);
```

1.08232323371113819151600369654116790277475095191 87

6 Exemples avec Pn(X) = (1-X)^n

On prend Pn(X)= (1-X)^n,
Pn(X)= sum((-1)^k*C(n,k)*X^k,k=0..n)
d'où pour n fixé, pk:= (-1)^k C(n,k)
avec C(n,k) le coefficent binomial k parmi n.
dn= Pn(-1) = 2^n.

```
> SerieAlt2:=proc(n,a)
  local S,k,p,c,d;
```

```
  d:=2^n;
  p:=1;
  c:=d-p;
  S:=c*a(0);
  for k from 0 to n-1 do
      p:=(-1)^(k+1)*binomial(n,k+1);
      c:=-p-c;
      S:=S+c*a(k+1);
  od;
  evalf(S/d);
  end;
```

```
proc(n, a)
local S, k, p, c, d;
    d := 2^n;
    p := 1;
    c := d – p;
    S := c*a(0);
    for k from 0 to n – 1 do
        p := (-1)^(k + 1)*binomial(n, k + 1); c := –p – c; S := S + c*a(k + 1)
    end do;
    evalf(S / d)
end proc
```

```
> ln2bis:=proc(n)
  SerieAlt2(n,k->1/(k+1));
  end;
```

proc(n) SerieAlt2(n, k → 1 / (k + 1)) end proc

```
> ln2bis(100);
```

0.693147180559945309417232121458168832699520819900 27

```
> evalf(log(2));
```

0.693147180559945309417232121458176568075500134360 26

Pi

```
> SuitePi2:=proc(n)
  4*SerieAlt2(n,k->1/(2*k+1));
  end;
```

proc(n) 4*SerieAlt2(n, k → 1 / (2*k + 1)) end proc

```
> SuitePi2(25);
```

3.141592633144036001590786986260601709899132809382 4

```
> evalf(Pi);
```

3.141592653589793238462643383279502884197169399375 1

7 Exemples avec Pn(X)=X^q (1-X)^q, 3n = 3q

On prend Pn(X)=X^q (1-X)^q, 3n = 3q

dn = (-1)^q*2^(2q)

```
> SerieAlt3:=proc(q,a)
  local S,k,p,c,d,n;
  n:=3*q;
  d:=(-1)^q*2^(2*q);
  p:=0;
  c:=d;
  S:=c*a(0);
  for k from 1 to n-1 do
      if k=q then
          p:=1;
      else
          if k>q then
               p:=p*(k-1-n)/(k-q);
          fi;
      fi;
      c:=-p-c;
      S:=S+c*a(k);
  od;
  evalf(S/d);
  end;
```

```
proc(q, a)
local S, k, p, c, d, n;
     n := 3*q;
     d := (-1)^q*2^(2*q);
     p := 0;
     c := d;
     S := c*a(0);
     for k to n - 1 do
          if k = q then p := 1 else if q < k then p := p*(k - 1 - n) / (k - q) end if end if;
          c := -p - c;
          S := S + c*a(k)
     end do;
     evalf(S / d)
end proc
```

```
> ln2ter:=proc(n)
  SerieAlt3(n,k->1/(k+1));
  end;
```

proc(n) SerieAlt3(n, k → 1 / (k + 1)) end proc

```
> ln2ter(30);
```

0.693147180559945309417232121458176568075500123754 95

```
> evalf(log(2));
```

0.693147180559945309417232121458176568075500134360 26

Pi

```
> SuitePi3:=proc(n)
```

```
  4*SerieAlt3(n,k->1/(2*k+1));
  end;
```

proc(n) 4∗SerieAlt3(n, k → 1 / (2∗k + 1)) end proc

```
> SuitePi3(10);
```

3.14159265358979054678429413069523841946555765835 20

```
> evalf(Pi);
```

3.14159265358979323846264338327950288419716939937 51

8 Comparaison des méthodes

La méthode de Simpson est une méthode rapide de calcul de valeur approchée d'intégrales.
On peut l'applique pour calculer l'intégrale de 0 à 1 de 1/ 1 + x qui vaut Pi/4.
La méthode présentée dans le texte donne une meilleur approximation de Pi.

```
> Simpson:=proc(n,f,a,b)
  local k,S,j;
  j:=(b-a)/n;
  S:=0;
  for k from 0 to n-1 do
  S:=S+(f(a+k*j)+f(a+(k+1)*j)+4*f(a+(k+1/2)*j))/6;
  od;
  evalf(j*S);
  end;
```

```
proc(n, f, a, b)
local k, S, j;
    j := (b − a) / n;
    S := 0;
    for k from 0 to n − 1 do
        S := S + 1 / 6∗f(a + k∗j) + 1 / 6∗f(a + (k + 1)∗j) + 2 / 3∗f(a + (k + 1 / 2)∗j)
    end do;
    evalf(j∗S)
end proc
```

```
> f:=x->1/(1+x^2);
```

$$x \to \frac{1}{1+x^2}$$

```
> 4*Simpson(1000,f,0,1);
```

3.14159265358979323846202334359696351608101905579 44

```
> SuitePi(50);
```

3.14159265358979323846264338327950288419717150996 86

```
> evalf(Pi);
```

3.14159265358979323846264338327950288419716939937 51

⌊ [> ?