

[Help](#)

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#else

#include "
href../../../../common/math/ap_kou_model/common_h_src.pdfcommon.h"
#include <cmath>
#include <iostream>
#include <cstdlib>

#ifndef _MSC_VER
#include "config.h"
#endif

extern "C" {
#include "pnl/pnl_mathtools.h"
}
using namespace std;

//
//fonction Hh-1
static long double Hh(const long double &x)
{
    return expl(-x * x / 2.);
}
//gaussian pdf
long double dnorm(long double x)
{
    return (expl(-x * x / 2.) / (sqrtl(2.*M_PI)));
}

long double MAXLOGL = 1.1356523406294143949491931077970764891253E4L;
long double SQRTHL = 7.0710678118654752440084436210484903928484E-1L;

#if (defined _WIN32 || defined _MSC_VER || !defined HAVE_ERFL)
/* erfc(x + 0.25) = erfc(0.25) + x R(x)
   0 <= x < 0.125
   Peak relative error 1.4e-35 */
#define NRNR13 8
```

```

static long double RNr13[NRnr13 + 1] =
{
    -2.353707097641280550282633036456457014829E3L,
    3.871159656228743599994116143079870279866E2L,
    -3.888105134258266192210485617504098426679E2L,
    -2.129998539120061668038806696199343094971E1L,
    -8.125462263594034672468446317145384108734E1L,
    8.151549093983505810118308635926270319660E0L,
    -5.033362032729207310462422357772568553670E0L,
    -4.253956621135136090295893547735851168471E-2L,
    -8.098602878463854789780108161581050357814E-2L
};
#define NRDr13 7
static long double RDr13[NRDr13 + 1] =
{
    2.220448796306693503549505450626652881752E3L,
    1.899133258779578688791041599040951431383E2L,
    1.061906712284961110196427571557149268454E3L,
    7.497086072306967965180978101974566760042E1L,
    2.146796115662672795876463568170441327274E2L,
    1.120156008362573736664338015952284925592E1L,
    2.211014952075052616409845051695042741074E1L,
    6.469655675326150785692908453094054988938E-1L
    /* 1.0E0 */
};
/* erfc(0.25) = C13a + C13b to extra precision. */
static long double C13a = 0.723663330078125L;
static long double C13b = 1.0279753638067014931732235184287934646022E-5L;

/* erfc(x + 0.375) = erfc(0.375) + x R(x)
   0 <= x < 0.125
   Peak relative error 1.2e-35 */
#define NRNr14 8
static long double RNr14[NRnr14 + 1] =
{
    -2.446164016404426277577283038988918202456E3L,
    6.718753324496563913392217011618096698140E2L,
    -4.581631138049836157425391886957389240794E2L,
    -2.382844088987092233033215402335026078208E1L,
    -7.119237852400600507927038680970936336458E1L,
    1.313609646108420136332418282286454287146E1L,

```

```

-6.188608702082264389155862490056401365834E0L,
-2.787116601106678287277373011101132659279E-2L,
-2.230395570574153963203348263549700967918E-2L
};
#define NRDr14 7
static long double RDr14[NRDr14 + 1] =
{
    2.495187439241869732696223349840963702875E3L,
    2.503549449872925580011284635695738412162E2L,
    1.159033560988895481698051531263861842461E3L,
    9.493751466542304491261487998684383688622E1L,
    2.276214929562354328261422263078480321204E2L,
    1.367697521219069280358984081407807931847E1L,
    2.276988395995528495055594829206582732682E1L,
    7.647745753648996559837591812375456641163E-1L
    /* 1.0E0 */
};
/* erfc(0.375) = C14a + C14b to extra precision. */
static long double C14a = 0.5958709716796875L;
static long double C14b = 1.2118885490201676174914080878232469565953E-5L;

/* erfc(x + 0.5) = erfc(0.5) + x R(x)
   0 <= x < 0.125
   Peak relative error 4.7e-36 */
#define NRNr15 8
static long double RNr15[NRNr15 + 1] =
{
    -2.624212418011181487924855581955853461925E3L,
    8.473828904647825181073831556439301342756E2L,
    -5.286207458628380765099405359607331669027E2L,
    -3.895781234155315729088407259045269652318E1L,
    -6.200857908065163618041240848728398496256E1L,
    1.469324610346924001393137895116129204737E1L,
    -6.961356525370658572800674953305625578903E0L,
    5.145724386641163809595512876629030548495E-3L,
    1.990253655948179713415957791776180406812E-2L
};
#define NRDr15 7
static long double RDr15[NRDr15 + 1] =
{
    2.986190760847974943034021764693341524962E3L,

```

```

5.288262758961073066335410218650047725985E2L,
1.363649178071006978355113026427856008978E3L,
1.921707975649915894241864988942255320833E2L,
2.588651100651029023069013885900085533226E2L,
2.628752920321455606558942309396855629459E1L,
2.455649035885114308978333741080991380610E1L,
1.378826653595128464383127836412100939126E0L
/* 1.0E0 */
};
/* erfc(0.5) = C15a + C15b to extra precision. */
static long double C15a = 0.4794921875L;
static long double C15b = 7.9346869534623172533461080354712635484242E-6L;

/* erfc(x + 0.625) = erfc(0.625) + x R(x)
   0 <= x < 0.125
   Peak relative error 5.1e-36 */
#define NRNr16 8
static long double RNr16[NRNr16 + 1] =
{
    -2.347887943200680563784690094002722906820E3L,
    8.008590660692105004780722726421020136482E2L,
    -5.257363310384119728760181252132311447963E2L,
    -4.471737717857801230450290232600243795637E1L,
    -4.849540386452573306708795324759300320304E1L,
    1.140885264677134679275986782978655952843E1L,
    -6.731591085460269447926746876983786152300E0L,
    1.370831653033047440345050025876085121231E-1L,
    2.022958279982138755020825717073966576670E-2L,
};
#define NRDr16 7
static long double RDr16[NRDr16 + 1] =
{
    3.075166170024837215399323264868308087281E3L,
    8.730468942160798031608053127270430036627E2L,
    1.458472799166340479742581949088453244767E3L,
    3.230423687568019709453130785873540386217E2L,
    2.804009872719893612081109617983169474655E2L,
    4.465334221323222943418085830026979293091E1L,
    2.612723259683205928103787842214809134746E1L,
    2.341526751185244109722204018543276124997E0L,
/* 1.0E0 */

```

```

};
/* erfc(0.625) = C16a + C16b to extra precision. */
static long double C16a = 0.3767547607421875L;
static long double C16b = 4.3570693945275513594941232097252997287766E-6L;

/* erfc(x + 0.75) = erfc(0.75) + x R(x)
   0 <= x < 0.125
   Peak relative error 1.7e-35 */
#define NRNr17 8
static long double RNr17[NRnr17 + 1] =
{
    -1.767068734220277728233364375724380366826E3L,
    6.693746645665242832426891888805363898707E2L,
    -4.746224241837275958126060307406616817753E2L,
    -2.274160637728782675145666064841883803196E1L,
    -3.541232266140939050094370552538987982637E1L,
    6.988950514747052676394491563585179503865E0L,
    -5.807687216836540830881352383529281215100E0L,
    3.631915988567346438830283503729569443642E-1L,
    -1.488945487149634820537348176770282391202E-2L
};
#define NRDr17 7
static long double RDr17[NRDr17 + 1] =
{
    2.748457523498150741964464942246913394647E3L,
    1.020213390713477686776037331757871252652E3L,
    1.388857635935432621972601695296561952738E3L,
    3.903363681143817750895999579637315491087E2L,
    2.784568344378139499217928969529219886578E2L,
    5.555800830216764702779238020065345401144E1L,
    2.646215470959050279430447295801291168941E1L,
    2.984905282103517497081766758550112011265E0L,
    /* 1.0E0 */
};
/* erfc(0.75) = C17a + C17b to extra precision. */
static long double C17a = 0.2888336181640625L;
static long double C17b = 1.0748182422368401062165408589222625794046E-5L;

/* erfc(x + 0.875) = erfc(0.875) + x R(x)
   0 <= x < 0.125

```

```

    Peak relative error 2.2e-35  */
#define NRNr18 8
static long double RNr18[NRNr18 + 1] =
{
    -1.342044899087593397419622771847219619588E3L,
    6.127221294229172997509252330961641850598E2L,
    -4.519821356522291185621206350470820610727E2L,
    1.223275177825128732497510264197915160235E1L,
    -2.730789571382971355625020710543532867692E1L,
    4.045181204921538886880171727755445395862E0L,
    -4.925146477876592723401384464691452700539E0L,
    5.933878036611279244654299924101068088582E-1L,
    -5.557645435858916025452563379795159124753E-2L
};
#define NRDr18 7
static long double RDr18[NRDr18 + 1] =
{
    2.557518000661700588758505116291983092951E3L,
    1.070171433382888994954602511991940418588E3L,
    1.344842834423493081054489613250688918709E3L,
    4.161144478449381901208660598266288188426E2L,
    2.763670252219855198052378138756906980422E2L,
    5.998153487868943708236273854747564557632E1L,
    2.657695108438628847733050476209037025318E1L,
    3.252140524394421868923289114410336976512E0L,
    /* 1.0E0 */
};
/* erfc(0.875) = C18a + C18b to extra precision.  */
static long double C18a = 0.215911865234375L;
static long double C18b = 1.3073705765341685464282101150637224028267E-5L;

/* erfc(x + 1.0) = erfc(1.0) + x R(x)
   0 <= x < 0.125
   Peak relative error 1.6e-35  */
#define NRNr19 8
static long double RNr19[NRNr19 + 1] =
{
    -1.139180936454157193495882956565663294826E3L,
    6.134903129086899737514712477207945973616E2L,
    -4.628909024715329562325555164720732868263E2L,
    4.165702387210732352564932347500364010833E1L,

```

```

-2.286979913515229747204101330405771801610E1L,
1.870695256449872743066783202326943667722E0L,
-4.177486601273105752879868187237000032364E0L,
7.533980372789646140112424811291782526263E-1L,
-8.629945436917752003058064731308767664446E-2L
};
#define NRDr19 7
static long double RDr19[NRDr19 + 1] =
{
2.744303447981132701432716278363418643778E3L,
1.266396359526187065222528050591302171471E3L,
1.466739461422073351497972255511919814273E3L,
4.868710570759693955597496520298058147162E2L,
2.993694301559756046478189634131722579643E2L,
6.868976819510254139741559102693828237440E1L,
2.801505816247677193480190483913753613630E1L,
3.604439909194350263552750347742663954481E0L,
/* 1.0E0 */
};
/* erfc(1.0) = C19a + C19b to extra precision. */
static long double C19a = 0.15728759765625L;
static long double C19b = 1.1609394035130658779364917390740703933002E-5L;

/* erfc(x + 1.125) = erfc(1.125) + x R(x)
0 <= x < 0.125
Peak relative error 3.6e-36 */
#define NRNr20 8
static long double RNr20[NRNr20 + 1] =
{
-9.652706916457973956366721379612508047640E2L,
5.577066396050932776683469951773643880634E2L,
-4.406335508848496713572223098693575485978E2L,
5.202893466490242733570232680736966655434E1L,
-1.931311847665757913322495948705563937159E1L,
-9.364318268748287664267341457164918090611E-2L,
-3.306390351286352764891355375882586201069E0L,
7.573806045289044647727613003096916516475E-1L,
-9.611744011489092894027478899545635991213E-2L
};
#define NRDr20 7
static long double RDr20[NRDr20 + 1] =

```

```

{
    3.032829629520142564106649167182428189014E3L,
    1.659648470721967719961167083684972196891E3L,
    1.703545128657284619402511356932569292535E3L,
    6.393465677731598872500200253155257708763E2L,
    3.489131397281030947405287112726059221934E2L,
    8.848641738570783406484348434387611713070E1L,
    3.132269062552392974833215844236160958502E1L,
    4.430131663290563523933419966185230513168E0L
    /* 1.0E0 */
};
/* erfc(1.125) = C20a + C20b to extra precision. */
static long double C20a = 0.111602783203125L;
static long double C20b = 8.9850951672359304215530728365232161564636E-6L;

#endif /* (defined _WIN32 || defined _MSC_VER || !defined HAVE_ERFL) */

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
   7/8 <= 1/x < 1
   Peak relative error 1.4e-35 */
#define NRNr8 9
static long double RNr8[NRnr8 + 1] =
{
    3.587451489255356250759834295199296936784E1L,
    5.406249749087340431871378009874875889602E2L,
    2.931301290625250886238822286506381194157E3L,
    7.359254185241795584113047248898753470923E3L,
    9.201031849810636104112101947312492532314E3L,
    5.749697096193191467751650366613289284777E3L,
    1.710415234419860825710780802678697889231E3L,
    2.150753982543378580859546706243022719599E2L,
    8.740953582272147335100537849981160931197E0L,
    4.876422978828717219629814794707963640913E-2L
};
#define NRDr8 8
static long double RDr8[NRDr8 + 1] =
{
    6.358593134096908350929496535931630140282E1L,
    9.900253816552450073757174323424051765523E2L,
    5.642928777856801020545245437089490805186E3L,
    1.524195375199570868195152698617273739609E4L,

```

```

2.113829644500006749947332935305800887345E4L,
1.526438562626465706267943737310282977138E4L,
5.561370922149241457131421914140039411782E3L,
9.394035530179705051609070428036834496942E2L,
6.147019596150394577984175188032707343615E1L
/* 1.0E0 */
};

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
3/4 <= 1/x < 7/8
Peak relative error 1.7e-36 */
#define NRNr7 9
static long double RNr7[NRNr7 + 1] =
{
1.293515364043117601705535485785956592493E2L,
2.474534371269189867053251150063459055230E3L,
1.756537563959875738809491329250457486510E4L,
5.977479535376639763773153344676726091607E4L,
1.054313816139671870123172936972055385049E5L,
9.754699773487726957401038094714603033904E4L,
4.579131242577671038339922925213209214880E4L,
1.000710322164510887997115157797717324370E4L,
8.496863250712471449526805271633794700452E2L,
1.797349831386892396933210199236530557333E1L
};

#define NRDr7 9
static long double RDr7[NRDr7 + 1] =
{
2.292696320307033494820399866075534515002E2L,
4.500632608295626968062258401895610053116E3L,
3.321218723485498111535866988511716659339E4L,
1.196084512221845156596781258490840961462E5L,
2.287033883912529843927983406878910939930E5L,
2.370223495794642027268482075021298394425E5L,
1.305173734022437154610938308944995159199E5L,
3.589386258485887630236490009835928559621E4L,
4.339996864041074149726360516336773136101E3L,
1.753135522665469574605384979152863899099E2L
/* 1.0E0 */
};

```

```

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
   5/8 <= 1/x < 3/4
   Peak relative error 1.6e-35 */
#define NRNr6 9
static long double RNr6[NRNr6 + 1] =
{
    1.423313561367394923305025174137639124533E1L,
    3.244462503273003837514629113846075327206E2L,
    2.784937282403293364911673341412846781934E3L,
    1.163060685810874867196849890286455473382E4L,
    2.554141394931962276102205517358731053756E4L,
    2.982733782500729530503336931258698708782E4L,
    1.789683564523810605328169719436374742840E4L,
    5.056032142227470121262177112822018882754E3L,
    5.605349942234782054561269306895707034586E2L,
    1.561652599080729507274832243665726064881E1L
};
#define NRDr6 9
static long double RDr6[NRDr6 + 1] =
{
    2.522757606611479946069351519410222913326E1L,
    5.876797910931896554014229647006604017806E2L,
    5.211092128250480712011248211246144751074E3L,
    2.282679910855404599271496827409168580797E4L,
    5.371245819205596609986320599133109262447E4L,
    6.926186102106400355114925675028888924445E4L,
    4.794366033363621432575096487724913414473E4L,
    1.673190682734065914573814938835674963896E4L,
    2.589544846151313120096957014256536236242E3L,
    1.349438432583208276883323156200117027433E2L
/* 1.00000000000000000000000000000000000000000000000000000E0 */
};

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
   1/2 <= 1/x < 5/8
   Peak relative error 4.3e-36 */
#define NRNr5 10
static long double RNr5[NRNr5 + 1] =
{
    6.743447478279267339131211137241149796763E-2L,
    2.031339015932962998168472743355874796350E0L,

```

```

2.369234815713196480221800940201367484379E1L,
1.387819614016107433603101545594790875922E2L,
4.435600256936515839937720907171966121786E2L,
7.881577949936817507981170892417739733047E2L,
7.615749099291545976179905281851765734680E2L,
3.752484528663442467089606663006771157777E2L,
8.279644286027145214308303292537009564726E1L,
6.201462983413738162709722770960040042647E0L,
6.649631608720062333043506249503378282697E-2L
};
#define NRDr5 9
static long double RDr5[NRDr5 + 1] =
{
1.195244945161889822018178270706903972343E-1L,
3.660216908153253021384862427197665991311E0L,
4.373405883243078019655721779021995159854E1L,
2.653305963056235008916733402786877121865E2L,
8.921329790491152046318422124415895506335E2L,
1.705552231555600759729260640562363304312E3L,
1.832711109606893446763174603477244625325E3L,
1.056823953275835649973998168744261083316E3L,
2.975561981792909722126456997074344895584E2L,
3.393149095158232521894537008472203487436E1L
/* 1.0E0 */
};

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
3/8 <= 1/x < 1/2
Peak relative error 1.8e-36 */
#define NRNr4 10
static long double RNr4[NRNr4 + 1] =
{
3.558685919236420073872459554885612994007E-2L,
1.460223642496950651561817195253277924528E0L,
2.379856746555189546876720308841066577268E1L,
2.005205521246554860334064698817220160117E2L,
9.533374928664989955629120027419490517596E2L,
2.623024576994438336130421711314560425373E3L,
4.126446434603735586340585027628851620886E3L,
3.540675861596687801829655387867654030013E3L,
1.506037084891064572653273761987617394259E3L,

```

```

2.630715699182706745867272452228891752353E2L,
1.202476629652900619635409242749750364878E1L
};
#define NRDr4 10
static long double RDr4[NRDr4 + 1] =
{
6.307606561714590590399683184410336583739E-2L,
2.619717051134271249293056836082721776665E0L,
4.344441402681725017630451522968410844608E1L,
3.752891116408399440953195184301023399176E2L,
1.849305988804654653921972804388006355502E3L,
5.358505261991675891835885654499883449403E3L,
9.091890995405251314631428721090705475825E3L,
8.731418313949291797856351745278287516416E3L,
4.420211285043270337492325400764271868740E3L,
1.031487363021856106882306509107923200832E3L,
8.387036084846046121805145056040429461783E1L
/* 1.0E0 */
};

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
1/4 <= 1/x < 3/8
Peak relative error 8.1e-37 */
#define NRNr3 11
static long double RNr3[NRNr3 + 1] =
{
4.584481542956275354582319313040418316755E-5L,
2.674923158288848442110883948437930349128E-3L,
6.344232532055212248017211243740466847311E-2L,
7.985145965992002744933550450451513513963E-1L,
5.845078061888281665064746347663123946270E0L,
2.566625318816866587482759497608029522596E1L,
6.736225182343446605268837827950856640948E1L,
1.021796460139598089409347761712730512053E2L,
8.344336615515430530929955615400706619764E1L,
3.207749011528249356283897356277376306967E1L,
4.386185123863412086856423971695142026036E0L,
8.971576448581208351826868348023528863856E-2L
};
#define NRDr3 10
static long double RDr3[NRDr3 + 1] =

```

```

{
    8.125781965218112303281657065320409661370E-5L,
    4.781806762611504685247817818428945295520E-3L,
    1.147785538413798317790357996845767614561E-1L,
    1.469285552007088106614218996464752307606E0L,
    1.101712261349880339221039938999124077650E1L,
    5.008507527095093413713171655268276861071E1L,
    1.383058691613468970486425146336829447433E2L,
    2.264114250278912520501010108736339599752E2L,
    2.081377197698598680576330179979996940039E2L,
    9.724438129690013609440151781601781137944E1L,
    1.907905050771832372089975877589291760121E1L
    /* 1.0E0 */
};

/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
   1/8 <= 1/x < 1/4
   Peak relative error 1.5e-36 */
#define NRNr2 11
static long double RNr2[NRNr2 + 1] =
{
    6.928615158005256885698045840589513728399E-7L,
    5.616245938942075826026382337922413007879E-5L,
    1.871624980715261794832358438894219696113E-3L,
    3.349922063795792371642023765253747563009E-2L,
    3.531865233974349943956345502463135695834E-1L,
    2.264714157625072773976468825160906342360E0L,
    8.810720294489253776747319730638214883026E0L,
    2.014056685571655833019183248931442888437E1L,
    2.524586947657190747039554310814128743320E1L,
    1.520656940937208886246188940244581671609E1L,
    3.334145500790963675035841482334493680498E0L,
    1.122108380007109245896534245151140632457E-1L
};

#define NRDr2 10
static long double RDr2[NRDr2 + 1] =
{
    1.228065061824874795984937092427781089256E-6L,
    1.001593999520159167559129042893802235969E-4L,
    3.366527555699367241421450749821030974446E-3L,
    6.098626947195865254152265585991861150369E-2L,

```

```

6.541547922508613985813189387198804660235E-1L,
4.301130233305371976727117480925676583204E0L,
1.737155892350891711527711121692994762909E1L,
4.206892112110558214680649401236873828801E1L,
5.787487996025016843403524261574779631219E1L,
4.094047148590822715163181507813774861621E1L,
1.230603930056944875836549716515643997094E1L
/* 1.0E0 */
};
/* erfc(x) = exp(-x^2) 1/x R(1/x^2) / S(1/x^2)
1/128 <= 1/x < 1/8
Peak relative error 2.2e-36 */
#define NRNr1 9
static long double RNr1[NRNr1 + 1] =
{
1.293111801138199795250035229383033322539E-6L,
9.785224720880980456171438759161161816706E-5L,
2.932474396233212166056331430621176065943E-3L,
4.496429595719847083917435337780697436921E-2L,
3.805989855927720844877478869846718877846E-1L,
1.789745532222426292126781724570152590071E0L,
4.465737379634389318903237306594171764628E0L,
5.268535822258082278401240171488850433767E0L,
2.258357766807433839494276681092713991651E0L,
1.504459334078750002966538036652860809497E-1L
};
#define NRDr1 8
static long double RDr1[NRDr1 + 1] =
{
2.291980991578770070179177302906728371406E-6L,
1.745845828808028552029674694534934620384E-4L,
5.283248841982102317072923869576785278019E-3L,
8.221212297078141470944454807434634848018E-2L,
7.120500674861902950423510939060230945621E-1L,
3.475435367560809622183983439133664598155E0L,
9.243253391989233533874386043611304387113E0L,
1.227894792475280941511758877318903197188E1L,
6.789361410398841316638617624392719077724E0L
/* 1.0E0 */
};

```

```

/* erf(z+1) = erf_const + P(z)/Q(z)
   -.125 <= z <= 0
   Peak relative error 7.3e-36 */

#if (defined _WIN32 || defined _MSC_VER || !defined HAVE_ERFL)

static long double erf_const = 0.845062911510467529296875L;
#define NTN2 8
static long double TN2[NTN2 + 1] =
{
    -4.088889697077485301010486931817357000235E1L,
    7.157046430681808553842307502826960051036E3L,
    -2.191561912574409865550015485451373731780E3L,
    2.180174916555316874988981177654057337219E3L,
    2.848578658049670668231333682379720943455E2L,
    1.630362490952512836762810462174798925274E2L,
    6.317712353961866974143739396865293596895E0L,
    2.450441034183492434655586496522857578066E1L,
    5.127662277706787664956025545897050896203E-1L
};

#define NTD2 8
static long double TD2[NTD2 + 1] =
{
    1.731026445926834008273768924015161048885E4L,
    1.209682239007990370796112604286048173750E4L,
    1.160950290217993641320602282462976163857E4L,
    5.394294645127126577825507169061355698157E3L,
    2.791239340533632669442158497532521776093E3L,
    8.989365571337319032943005387378993827684E2L,
    2.974016493766349409725385710897298069677E2L,
    6.148192754590376378740261072533527271947E1L,
    1.178502892490738445655468927408440847480E1L
    /* 1.0E0 */
};

/* erf(x) = x + x P(x^2)/Q(x^2)
   0 <= x <= 7/8
   Peak relative error 1.8e-35 */
#define NTN1 8
static long double TN1[NTN1 + 1] =

```

```

{
    -3.858252324254637124543172907442106422373E10L,
    9.580319248590464682316366876952214879858E10L,
    1.302170519734879977595901236693040544854E10L,
    2.922956950426397417800321486727032845006E9L,
    1.764317520783319397868923218385468729799E8L,
    1.573436014601118630105796794840834145120E7L,
    4.028077380105721388745632295157816229289E5L,
    1.644056806467289066852135096352853491530E4L,
    3.390868480059991640235675479463287886081E1L
};

#define NTD1 8
static long double TD1[NTD1 + 1] =
{
    -3.005357030696532927149885530689529032152E11L,
    -1.342602283126282827411658673839982164042E11L,
    -2.777153893355340961288511024443668743399E10L,
    -3.483826391033531996955620074072768276974E9L,
    -2.906321047071299585682722511260895227921E8L,
    -1.653347985722154162439387878512427542691E7L,
    -6.245520581562848778466500301865173123136E5L,
    -1.402124304177498828590239373389110545142E4L,
    -1.209368072473510674493129989468348633579E2L
    /* 1.00000000000000000000000000000000000000000000000E0 */
};

static long double INFINITYL = 1000.0;
#endif /* #if (defined _WIN32 || defined _MSC_VER || !defined HAVE_ERFL) */

static long double neval(long double x, long double *p, int n)
{
    long double y;

    p += n;
    y = *p--;
    do
    {
        y = y * x + *p--;
    }
    while (--n > 0);
    return y;
}

```

```

}

static long double deval(long double x, long double *p, int n)
{
    long double y;

    p += n;
    y = x + *p--;
    do
    {
        y = y * x + *p--;
    }
    while (--n > 0);
    return y;
}

#if (defined _WIN32 || defined _MSC_VER || !defined HAVE_ERFL)
long double erfl(long double x);

static long double erfccl(long double x);
long double erfcl(long double a)
{
    long double x, y = 0., z;
    int i;

    if (a == INFINITYL)
        return (0.0L);
    if (a == -INFINITYL)
        return (2.0L);

    if (a < 0.0L)
        x = -a;
    else
        x = a;

    if (x < 0.25L)
        return (1.0L - erfl(a));

    if (x < 1.25L)
    {
        i = int(8.0 * x);

```

```

switch (i)
{
case 2:
    z = x - 0.25L;
    y = C13b + z * neval(z, RNr13, NRNr13) / deval(z, RDr13, NRDr13);
    y += C13a;
    break;
case 3:
    z = x - 0.375L;
    y = C14b + z * neval(z, RNr14, NRNr14) / deval(z, RDr14, NRDr14);
    y += C14a;
    break;
case 4:
    z = x - 0.5L;
    y = C15b + z * neval(z, RNr15, NRNr15) / deval(z, RDr15, NRDr15);
    y += C15a;
    break;
case 5:
    z = x - 0.625L;
    y = C16b + z * neval(z, RNr16, NRNr16) / deval(z, RDr16, NRDr16);
    y += C16a;
    break;
case 6:
    z = x - 0.75L;
    y = C17b + z * neval(z, RNr17, NRNr17) / deval(z, RDr17, NRDr17);
    y += C17a;
    break;
case 7:
    z = x - 0.875L;
    y = C18b + z * neval(z, RNr18, NRNr18) / deval(z, RDr18, NRDr18);
    y += C18a;
    break;
case 8:
    z = x - 1.0L;
    y = C19b + z * neval(z, RNr19, NRNr19) / deval(z, RDr19, NRDr19);
    y += C19a;
    break;
case 9:
    z = x - 1.125L;
    y = C20b + z * neval(z, RNr20, NRNr20) / deval(z, RDr20, NRDr20);
    y += C20a;

```

```

        break;
    }
    if (a < 0.0L)
        y = 2.0L - y;
    return y;
}

z = -a * a;

if (z < -MAXLOGL)
{
    if (a < 0)
        return (2.0L);
    else
        return (0.0L);
}

y = exp(-a * a) * erfc(x);

if (a < 0.0L)
    y = 2.0L - y;

if (y == 0.0L)
{
    if (a < 0)
        return (2.0L);
    else
        return (0.0L);
}
return y;
}

long double erfl(long double x)
{
    long double a, y, z;

    if (x == 0.0L)
        return (x);

    if (x == -INFINITYL)

```

```

    return (-1.0L);
if (x == INFINITYL)
    return (1.0L);

a = abs(x);
if (a > 1.0L)
    return (1.0L - erfc1(x));

z = x * x;
if (a < 0.875)
{
    y = a + a * neval(z, TN1, NTN1) / deval(z, TD1, NTD1);
}
else
{
    a = a - 1.0L;
    y = erf_const + neval(a, TN2, NTN2) / deval(a, TD2, NTD2);
}

if (x < 0)
    y = -y;
return (y);
}
#endif /* (defined _WIN32 || defined _MSC_VER || !defined HAVE_ERFL) */

static long double erfc1(long double x)
{
    long double p, y, z;
    int i;

    z = 1.0L / (x * x);

    i = int(8.0 / x);
    switch (i)
    {
        default:
        case 0:
            p = neval(z, RNr1, NRNr1) / deval(z, RDr1, NRDr1);
            break;
        case 1:
            p = neval(z, RNr2, NRNr2) / deval(z, RDr2, NRDr2);

```

```

        break;
    case 2:
        p = neval(z, RNr3, NRNr3) / deval(z, RDr3, NRDr3);
        break;
    case 3:
        p = neval(z, RNr4, NRNr4) / deval(z, RDr4, NRDr4);
        break;
    case 4:
        p = neval(z, RNr5, NRNr5) / deval(z, RDr5, NRDr5);
        break;
    case 5:
        p = neval(z, RNr6, NRNr6) / deval(z, RDr6, NRDr6);
        break;
    case 6:
        p = neval(z, RNr7, NRNr7) / deval(z, RDr7, NRDr7);
        break;
    case 7:
        p = neval(z, RNr8, NRNr8) / deval(z, RDr8, NRDr8);
        break;
    }
    y = p / x;
    return (y);
}

```

```

static long double cumnorm(long double a)
{
    long double x, y, z;

    x = a * SQRTHL;
    z = abs(x);

    if (z < 0.25L)
        y = 0.5L + 0.5L * erfl(x);
    else if (z < 1.25)
    {
        y = 0.5L * erfc1(z);
        if (x > 0)
            y = 1.0L - y;
    }
    else

```

```

    {
        y = 0.5L * (erfc1(z));
        z = expl(-a * a);
        y = y * sqrt1(z);
        if (x > 0.0L)
            y = 1.0L - y;
    }

    return (y);
}

//fonction Hh0
long double Hh0(const long double &x)
{
    return sqrt1(2.0L * M_PI) * (cumnorm(-x));
}

//fonctions qui retourne le vecteur des n premiers fonctions Hhn>
static std::vector<long double> Hhn(const long double &x, const int &n)
{
    std::vector<long double> Hhv(n);
    long double H1, H2;
    int i;
    if (n >= 1)
    {
        Hhv[0] = Hh0(x);
        H1 = Hh(x);
        H2 = Hhv[0];
        for (i = 1; i < n; i++)
        {
            Hhv[i] = (H1 - x * H2) / (double)i;
            H1 = H2;
            H2 = Hhv[i];
            if (Hhv[i] < 0)
            {
                Hhv[i] = 0;
                H1 = 0;
                H2 = 0;
            }
        }
    }

    return Hhv;
}

```

```

else
{
    //printf("Fonction long double *Hhn : l'argument 2 doit être > 0 \ n");
    exit(0);
}
}
//fonction I-1
static long double I(const long double &c, const long double &alpha, const long
{
    long double cst = 0.;
    if (alpha != 0 && beta != 0)
        cst = expl(alpha * delta / beta + alpha * alpha / (2 * beta * beta)) / beta;
    if (beta > 0 && alpha != 0)
        return cst * Hh0(beta * c - delta - alpha / beta);
    if (beta < 0 && alpha < 0)
        return -cst * Hh0(-beta * c + delta + alpha / beta);
    if (beta > 0 && alpha == 0)
        return Hh0(-delta + beta * c) / beta;

    printf("Fonction long double I, parametres incorrects \ n");
    exit(0);
}
//fonction IO
long double IO(const long double &c, const long double &alpha, const long double
{
    long double x, y, h, cst = 0.0, cst3 = 0.0;

    h = beta / alpha;
    x = beta * c - delta;
    y = x - 1 / h;
    if (alpha != 0 && beta != 0)
    {
        cst = expl(alpha * delta / beta + alpha * alpha / (2 * beta * beta)) / bet
        cst3 = -expl(alpha * c) / alpha;
    }
    if (beta > 0 && alpha != 0)
    {
        return cst3 * Hh0(x) + h * cst * Hh0(y);
    }
    if (beta < 0 && alpha < 0)
    {

```

```

        return cst3 * Hh0(x) - h * cst * Hh0(-y);
    }
    if (beta > 0 && alpha == 0)
    {
        return (Hh(x) - x * Hh0(x)) / beta;
    }
    printf("Fonction long double I0, parametres incorrects \n");
    exit(0);
}
//fonction qui retourne le vecteur des n premiers fonctions In
std::vector<long double> In(const long double &c, const long double &alpha, const
                           long double &beta, const long double &delta, const i
{
    long double x, y, h, cst = 0.0, cst3 = 0.0;
    std::vector<long double> Hhv, Iv;
    int n;
    h = beta / alpha;
    x = beta * c - delta;
    y = x - 1 / h;
    Iv.resize(Nb);
    if (alpha != 0 && beta != 0)
    {
        cst = expl(alpha * delta / beta + alpha * alpha / (2 * beta * beta)) / bet
        cst3 = -expl(alpha * c) / alpha;
    }

    if (beta > 0 && alpha != 0)
    {
        Hhv.resize(Nb);
        Hhv = Hhn(x, Nb);

        Iv[0] = cst3 * Hhv[0] + h * cst * Hh0(y);
        if (Iv[0] < 0)
            for (int i = 0; i < Nb; i++)
            {
                Iv[i] = 0.0;
            }
        for (n = 1; n < Nb; n++)
        {
            Iv[n] = h * Iv[n - 1] + cst3 * Hhv[n];
            if (Iv[n] < 0)

```

```

        for (int i = n; i < Nb; i++)
        {
            Iv[i] = 0.0;
        }
    }

    return Iv;
}

if (beta < 0 && alpha < 0)
{

    Hhv.resize(Nb);
    Hhv = Hhn(x, Nb);
    Iv[0] = cst3 * Hhv[0] - h * cst * Hh0(-y);
    if (Iv[0] < 0)
        for (int i = 0; i < Nb; i++)
        {
            Iv[i] = 0.0;
        }
    for (n = 1; n < Nb; n++)
    {
        Iv[n] = h * Iv[n - 1] + cst3 * Hhv[n];
        if (Iv[n] < 0)
            for (int i = n; i < Nb; i++)
            {
                Iv[i] = 0.0;
            }
    }
    return Iv;
}

if (beta > 0 && alpha == 0)
{
    Hhv.resize(Nb + 1);
    Hhv = Hhn(x, Nb + 1);
    for (n = 0; n < Nb; n++)
    {
        Iv[n] = Hhv[n + 1] / beta;
    }

    return Iv;
}

```

```

    printf("Fonction long double In : parametres non valides \n");
    exit(0);
}
//fonction qui retourne le vecteur des n premiers fonctions derivés de In % a sa
std::vector<long double> dIn(const long double &c, const long double &alpha, const
                           long double &beta, const long double &delta, const
{
    long double s, x, y, h, cst = 0.0L, cst3 = 0.0L;
    std::vector<long double> Hhv, Iv;
    int i, k;
    h = beta / alpha;
    x = beta * c - delta;
    y = x - 1 / h;
    Iv.resize(Nb);
    if (alpha != 0 && beta != 0)
    {
        cst = (sqrtl(2 * M_PI) / beta) * expl(alpha * delta / beta + alpha * alpha
        cst3 = -expl(alpha * c);
    }
    if (beta > 0 && alpha != 0)
    {
        Hhv.resize(Nb);
        Hhv = Hhn(x, Nb);
        for (k = 0; k < Nb; k++)
        {
            Iv[k] = 0.0L;
            for (i = 0; i <= k; i++)
            {
                Iv[k] = Iv[k] + powl(h, k - i) * Hhv[i];
            }
            s = 0.0L;
            for (i = 1; i <= k; i++)
            {
                s = s + powl(h, k - i) * Hhv[i - 1];
            }
            s = s + Hh(x) * powl(h, k);
            Iv[k] = cst3 * Iv[k] - cst3 * h * s - beta * powl(h, k + 1) * cst * dn
        }
        return Iv;
    }
    if (beta < 0 && alpha < 0)

```

```

{
    Hhv.resize(Nb);
    Hhv = Hhn(x, Nb);
    for (k = 0; k < Nb; k++)
    {
        Iv[k] = 0.0L;
        for (i = 0; i <= k; i++)
        {
            Iv[k] = Iv[k] + powl(h, k - i) * Hhv[i];
        }
        s = 0.0L;
        for (i = 1; i <= k; i++)
        {
            s = s + powl(h, k - i) * Hhv[i - 1];
        }
        s = s + Hh(x) * powl(h, k);
        Iv[k] = cst3 * Iv[k] - cst3 * h * s - beta * powl(h, k + 1) * cst * dn
    }
    return Iv;
}
if (beta > 0 && alpha == 0)
{
    Hhv.resize(Nb);
    Hhv = Hhn(x, Nb);
    for (k = 0; k < Nb; k++)
        Iv[k] = -Hhv[k];
    return Iv;
}
printf("Fonction long double In : parametres non valides \ n");
exit(0);
}
//fonction factorielle
unsigned long long int fact_dia(const int &n)
{
    if (n == 0)
        return 1;
    if (n == 1)
        return 1;
    if (n > 1)
        return n * fact_dia(n - 1);
    printf("Fonction int fact(const int & n) : parametre non valide \ n");
}

```

```

    exit(0);
}
//coefficients binomiaux
unsigned long long int bin_dia(const int &n, const int &k)
{
    if (n >= 0 && n < k)
        return 0;
    if (n >= 0 && n == k)
        return 1;
    if (n >= 0 && n >= k)
    {
        return fact_dia(n) / (fact_dia(k) * fact_dia(n - k));
    }
    else return 0;
    printf("Fonction long double bin(const int& n,const int& k): parametres non va
}

#endif //PremiaCurrentVersion

```