Help

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

#include "common.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
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 = 1.135652340629414394949193107797076489125 3E4L;
long double SQRTHL = 7.071067811865475244008443621048490392848 4E-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 = 1.027975363806701493173223518428793464602 2E-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 = 1.211888549020167617491408087823246956595 3E-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,
```

```
  1.36364917807100697835511302642785600897 8E3L,
  1.92170797564991589424186498894225532083 3E2L,
  2.58865110065102902306901388590008553322 6E2L,
  2.62875292032145560655894230939685562945 9E1L,
  2.45564903588511430897833374108099138061 0E1L,
  1.37882665359512846438312783641210093912 6E0L
  /* 1.0E0 */
};
/* erfc(0.5) = C15a + C15b to extra precision.  */
static long double C15a = 0.4794921875L;
static long double C15b = 7.93468695346231725334610803547126354842 42E-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] =
{
  -2.34788794320068056378469009400272290682 0E3L,
  8.00859066069210500478072272642102013648 2E2L,
  -5.25736331038411972876018125213231144796 3E2L,
  -4.47173771785780123045029023260024379563 7E1L,
  -4.84954038645257330670879532475930032030 4E1L,
  1.14088526467713467927598678297865595284 3E1L,
  -6.73159108546026944792674687698378615230 0E0L,
  1.37083165303304744034505002587608512123 1E-1L,
  2.02295827998213875502082571707396657667 0E-2L,
};
#define NRDr16 7
static long double RDr16[NRDr16 + 1] =
{
  3.07516617002483721539932326486830808728 1E3L,
  8.73046894216079803160805312727043003662 7E2L,
  1.45847279916634047974258194908845324476 7E3L,
  3.23042368756801970945313078587354038621 7E2L,
  2.80400987271989361208110961798316947465 5E2L,
  4.46533422132322294341808583002697929309 1E1L,
  2.61272325968320592810378784221480913474 6E1L,
  2.34152675118524410972220401854327612499 7E0L,
  /* 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] =
{
  -1.76706873422027772823336437572438036682 6E3L,
  6.69374664566524283242689188880536389870 7E2L,
  -4.74622424183727595812606030740661681775 3E2L,
  -2.27416063772878267514566606484188380319 6E1L,
  -3.54123226614093905009437055253898798263 7E1L,
  6.98895051474705267639449156358517950386 5E0L,
  -5.80768721683654083088135238352928121510 0E0L,
  3.63191598856734643883028350372956944364 2E-1L,
  -1.48894548714963482053734817677028239120 2E-2L
};
#define NRDr17 7
static long double RDr17[NRDr17 + 1] =
{
  2.74845752349815074196446494224691339464 7E3L,
  1.02021339071347768677603733175787125265 2E3L,
  1.38885763593543262197260169529656195273 8E3L,
  3.90336368114381775089599957963731549108 7E2L,
  2.78456834437813949921792896952921988657 8E2L,
  5.55580083021676470277923802006534540114 4E1L,
  2.64621547095905027943044729580129116894 1E1L,
  2.98490528210351749708176675855011201126 5E0L,
  /* 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 = 1.307370576534168546428210115063722402826 7E-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,
```

```
  1.87069525644987274306678320232643667722E0L,
  -4.17748660127310575287986818723700003236 4E0L,
  7.53398037278964614011242481129178252626 3E-1L,
  -8.62994543691775200305806473130876766444 6E-2L
};
#define NRDr19 7
static long double RDr19[NRDr19 + 1] =
{
  2.74430344798113270143271627836341864377 8E3L,
  1.26639635952618706522252805059130217147 1E3L,
  1.46673946142207335149797225551191981427 3E3L,
  4.86871057075969395559749652029805814716 2E2L,
  2.99369430155975604647818963413172257964 3E2L,
  6.86897681951025413974155910269382823744 0E1L,
  2.80150581624767719348019048391375361363 0E1L,
  3.60443990919435026355275034774266395448 1E0L,
  /* 1.0E0 */
};
/* erfc(1.0) = C19a + C19b to extra precision.  */
static long double C19a = 0.15728759765625L;
static long double C19b = 1.16093940351306587793649173907407039330 02E-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] =
{
  -9.65270691645797395636672137961250804764 0E2L,
  5.57706639605093277668346995177364388063 4E2L,
  -4.40633550884849671357222309869357548597 8E2L,
  5.20289346649024273357023268073696665543 4E1L,
  -1.93131184766575791332249594870556393715 9E1L,
  -9.36431826874828766426734145716491809061 1E-2L,
  -3.30639035128635276489135537588258620106 9E0L,
  7.57380604528904464772761300309691651647 5E-1L,
  -9.61174401148909289402747889954563599121 3E-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 = 8.985095167235930421553072836523216156463 6E-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,
```

```
  1.52643856262646570626794373731028297713 8E4L,
  5.56137092214924145713142191414003941178 2E3L,
  9.39403553017970505160907042803683449694 2E2L,
  6.14701959615039457798417518803270734361 5E1L
  /* 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] =
{
  1.29351536404311760170553548578595659249 3E2L,
  2.47453437126918986705325115006345905523 0E3L,
  1.75653756395987573880949132925045748651 0E4L,
  5.97747953537663976377315334467672609160 7E4L,
  1.05431381613967187012317293697205538504 9E5L,
  9.75469977348772695740103809471460303390 4E4L,
  4.57913124257767103833992292521320921488 0E4L,
  1.00071032216451088799711515779771732437 0E4L,
  8.49686325071247144952680527163379470045 2E2L,
  1.79734983138689239693321019923653055733 3E1L
};
#define NRDr7 9
static long double RDr7[NRDr7 + 1] =
{
  2.29269632030703349482039986607553451500 2E2L,
  4.50063260829562696806225840189561005311 6E3L,
  3.32121872348549811153586698851171665933 9E4L,
  1.19608451222184515659678125849084096146 2E5L,
  2.28703388391252984392798340687891093993 0E5L,
  2.37022349579464202726848207502129839442 5E5L,
  1.30517373402243715461093830894499515919 9E5L,
  3.58938625848588763023649000983592855962 1E4L,
  4.33999686404107414972636051633677313610 1E3L,
  1.75313552266546957460538497915286389909 9E2L
  /* 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] =
{
  1.42331356136739492330502517413763912453 3E1L,
  3.24446250327300383751462911384607532720 6E2L,
  2.78493728240329336491167334141284678193 4E3L,
  1.16306068581087486719684989028645547338 2E4L,
  2.55414139493196227610220551735873105375 6E4L,
  2.98273378250072953050333693125869870878 2E4L,
  1.78968356452381060532816971943637474284 0E4L,
  5.05603214222747012126217711282201888275 4E3L,
  5.60534994223478205456126930689570703458 6E2L,
  1.56165259908072950727483224366572606488 1E1L
};
#define NRDr6 9
static long double RDr6[NRDr6 + 1] =
{
  2.52275760661147994606935151941022291332 6E1L,
  5.87679791093189655401422964700660401780 6E2L,
  5.21109212825048071201124821124614475107 4E3L,
  2.28267991085540459927149682740916858079 7E4L,
  5.37124581920559660998632059913310926244 7E4L,
  6.92618610210640035511492567502888892444 5E4L,
  4.79436603336362143257509648772491341447 3E4L,
  1.67319068273406591457381493883567496389 6E4L,
  2.58954484615131312009695701425653623624 2E3L,
  1.34943843258320827688332315620011702743 3E2L
  /* 1.0000000000000000000000000000000000000E0 */
};

/* 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] =
{
  6.74344747827926733913121113724114979676 3E-2L,
  2.03133901593296299816847274335587479635 0E0L,
  2.36923481571319648022180094020136748437 9E1L,
```

```
  1.38781961401610743360310154559479087592 2E2L,
  4.43560025693651583993772090717196612178 6E2L,
  7.88157794993681750798117089241773973304 7E2L,
  7.61574909929154597617990528185176573468 0E2L,
  3.75248452866344246708960666300677115777 7E2L,
  8.27964428602714521430830329253700956472 6E1L,
  6.20146298341373816270972277096004004264 7E0L,
  6.64963160872006233304350624950337828269 7E-2L
};
#define NRDr5 9
static long double RDr5[NRDr5 + 1] =
{
  1.19524494516188982201817827070690397234 3E-1L,
  3.66021690815325302138486242719766599131 1E0L,
  4.37340588324307801965572177902199515985 4E1L,
  2.65330596305623500891673340278687712186 5E2L,
  8.92132979049115204631842212441589550633 5E2L,
  1.70555223155560075972926064056236330431 2E3L,
  1.83271110960689344676317460347724462532 5E3L,
  1.05682395327583564997399816874426108331 6E3L,
  2.97556198179290972212645699707434489558 4E2L,
  3.39314909515823252189453700847220348743 6E1L
  /* 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] =
{
  3.55868591923642007387245955488561299400 7E-2L,
  1.46022364249695065156181719525327792452 8E0L,
  2.37985674655518954687672030884106657726 8E1L,
  2.00520552124655486033406469881722016011 7E2L,
  9.53337492866498995562912002741949051759 6E2L,
  2.62302457699443833613042171131456042537 3E3L,
  4.12644634603735586340585027628851620886 3E3L,
  3.54067586159668780182965538786765403001 3E3L,
  1.50603708489106457265327376198761739425 9E3L,
  2.63071569918270674586727245222889175235 3E2L,
```

```
  1.20247662965290061963540924274975036487 8E1L
};
#define NRDr4 10
static long double RDr4[NRDr4 + 1] =
{
  6.30760656171459059039968318441033658373 9E-2L,
  2.61971705113427124929305683608272177666 5E0L,
  4.34444140268172501763045152296841084460 8E1L,
  3.75289111640839944095319518430102339917 6E2L,
  1.84930598880465465392197280438800635550 2E3L,
  5.35850526199167589183588565449988344940 3E3L,
  9.09189099540525131463142872109070547582 5E3L,
  8.73141831394929179785635174527828751641 6E3L,
  4.42021128504327033749232540076427186874 0E3L,
  1.03148736302185610688230650910792320083 2E3L,
  8.38703608484604612180514505604042946178 3E1L
  /* 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] =
{
  4.58448154295627535458231931304041831675 5E-5L,
  2.67492315828884844211088394843793034912 8E-3L,
  6.34423253205521224801721124374046684731 1E-2L,
  7.98514596599200274493355045045151351396 3E-1L,
  5.84507806188828166506474634766312394627 0E0L,
  2.56662531881686658748275949760802952259 6E1L,
  6.73622518234344660526883782795085664094 8E1L,
  1.02179646013959808940934776171273051205 3E2L,
  8.34433661551543053092995561540070661976 4E1L,
  3.20774901152824935628389735627737630696 7E1L,
  4.38618512386341208685642397169514202603 6E0L,
  8.97157644858120835182686834802352886385 6E-2L
};
#define NRDr3 10
static long double RDr3[NRDr3 + 1] =
{
```

```
  8.12578196521811230328165706532040966137OE-5L,
  4.78180676261150468524781781842894529552OE-3L,
  1.14778553841379831779035799684576761456lE-1L,
  1.46928555200708810661421899646475230760GE0L,
  1.10171226134988033922103993899991240776501E1L,
  5.00850752709509341371317165526827686107lE1L,
  1.38305869161346897048642514633682944743 3E2L,
  2.26411425027891252050101010873633959975 2E2L,
  2.08137719769859868057633017997999694003 9E2L,
  9.72443812969001360944015178160178113794 4E1L,
  1.90790505077183237208997587758929176012 1E1L
  /* 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] =
{
  6.92861515800525688569804584058951372839 9E-7L,
  5.61624593894207582602638233792241300787 9E-5L,
  1.87162498071526179483235843889421969611 3E-3L,
  3.34992206379579237164202376525374756300 9E-2L,
  3.53186523397434994395634550246313569583 4E-1L,
  2.26471415762507277397646882516090634236 0E0L,
  8.81072029448925377674731973063821488302 6E0L,
  2.01405668557165583301918324893144288843 7E1L,
  2.52458694765719074703955431081412874332 0E1L,
  1.52065694093720888624618894024458167160 9E1L,
  3.33414550079096367503584148233449368049 8E0L,
  1.12210838000710924589653424515114063245 7E-1L
};
#define NRDr2 10
static long double RDr2[NRDr2 + 1] =
{
  1.22806506182487479598493709242778108925 6E-6L,
  1.00159399952015916755912904289380223596 9E-4L,
  3.36652755569936724142145074982103097444 6E-3L,
  6.09862694719586525415226558599186115036 9E-2L,
  6.54154792250861398581318938719880466023 5E-1L,
```

```
  4.30113023330537197672711748092567658320 4E0L,
  1.73715589235089171152771112169299476290 9E1L,
  4.20689211211055821468064940123687382880 1E1L,
  5.78748799602501684340352426157477963121 9E1L,
  4.09404714859082271516318150781377486162 1E1L,
  1.23060393005694487583654971651564399709 4E1L
  /* 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] =
{
  1.29311180113819979525003522938303332253 9E-6L,
  9.78522472088098045617143875916116181670 6E-5L,
  2.93247439623321216605633143062117606594 3E-3L,
  4.49642959571984708391743533778069743692 1E-2L,
  3.80598985592772084487747886984671887784 6E-1L,
  1.78974553222242629212678172457015259007 1E0L,
  4.46573737963438931890323730659417176462 8E0L,
  5.26853582225808227840124017148885043376 7E0L,
  2.25835776680743383949427668109271399165 1E0L,
  1.50445933407875000296653803665286080949 7E-1L
};
#define NRDr1 8
static long double RDr1[NRDr1 + 1] =
{
  2.29198099157877007017917730290672837140 6E-6L,
  1.74584582880802855202967469453493462038 4E-4L,
  5.28324884198210231707292386957678527801 9E-3L,
  8.22121229707814147094445480743463484801 8E-2L,
  7.12050067486190295042351093906023094562 1E-1L,
  3.47543536756080962218398343913366459815 5E0L,
  9.24325339198923353387438604361130438711 3E0L,
  1.22789479247528094151175887731890319718 8E1L,
  6.78936141039884131663861762439271907772 4E0L
  /* 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] =
{
  -4.08888969707748530101048693181735700023 5E1L,
  7.15704643068180855384230750282696005103 6E3L,
  -2.19156191257440986555001548545137373178 0E3L,
  2.18017491655531687498898117765405733721 9E3L,
  2.84857865804967066823133368237972094345 5E2L,
  1.63036249095251283676281046217479892527 4E2L,
  6.31771235396186697414373939686529359689 5E0L,
  2.45044103418349243465558649652285757806 6E1L,
  5.12766227770678766495602554589705089620 3E-1L
};
#define NTD2 8
static long double TD2[NTD2 + 1] =
{
  1.73102644592683400827376892401516104888 5E4L,
  1.20968223900799037079611260428604817375 0E4L,
  1.16095029021799364132060228246297616385 7E4L,
  5.39429464512712657782550716906135569815 7E3L,
  2.79123934053363266944215849753252177609 3E3L,
  8.98936557133731903294300538737899382768 4E2L,
  2.97401649376634940972538571089729806967 7E2L,
  6.14819275459037637874026107253352727194 7E1L,
  1.17850289249073844565546892740844084748 0E1L
  /* 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
  /* 1.000000000000000000000000000000000000000E0 */
};

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

/*abs*/
long double polevll(long double x, void *PP, int n)
{
  register long double y;
  long double *P;

  P = (long double *) PP;
  y = *P++;
  do
    {
      y = y * x + *P++;
    }
  while (--n);
```

```
  return (y);
}

long double p1evl1(long double x, void *PP, int n)
{
  register long double y;
  long double *P;

  P = (long double *) PP;
  n -= 1;
  y = x + *P++;
  do
    {
      y = y * x + *P++;
    }
  while (--n);
  return (y);
}

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;
}

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);

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) * erfcel(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 - erfcl(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) */

long double erfcel(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);
}

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 * erfcl(z);
      if (x > 0)
        y = 1.0L - y;

    }
  else
    {
      y = 0.5L * (erfcel(z));
      z = expl(-a * a);
      y = y * sqrtl(z);
      if (x > 0.0L)
        y = 1.0L - y;
    }

  return (y);
}
//fonction Hh0
```

```
long double Hh0(const long double &x)
{
  return sqrtl(2.0L * M_PI) * (cumnorm(-x));
}
//fonctions qui retourne le vecteur des n premiers fonctions Hhn>
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 Ãłtre > 0 \ n");
      exit(0);
    }
}
//fonction I-1
long double I(const long double &c, const long double &alpha, const long double
{
  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 I0
long double I0(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, cons
```

```
                                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 derivÃ‰s de In % a s
std::vector<long double> dIn(const long double &c, const long double &alpha, con
                             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
```