

```
#include "bs1d_doublim.h"
#include "pnl/pnl_specfun.h"

/*Computation of Laplace transform*/
double fnRf_3(dcomplex z, double aa, double bb, double hh, double nn)
{
    double Rfs;
    dcomplex Cun, Cnn, Cnn2, z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11, z12, z13, z14, z15, z16, z17, z18, z19, z20, z21, z22, z23, z24, z25, z26, z27, z28, z29, z30, z31, z32, z33, z34, z35, z36, z37, z38, z39, z40, z41, z42, z43, z44, z45, z46, z47, z48, z49, z50, z51, z52, z53, z54, z55, z56, z57, z58, z59, z60, z61, z62, z63, z64, z65, z66, z67, z68, z69, z70, z71, z72, z73, z74, z75, z76, z77, z78, z79, z80, z81, z82, z83, z84, z85, z86, z87, z88, z89, z90, z91, z92, z93, z94, z95, z96, z97, z98, z99, z100, z101, z102, z103, z104, z105, z106, z107, z108, z109, z110, z111, z112, z113, z114, z115, z116, z117, z118, z119, z120, z121, z122, z123, z124, z125, z126, z127, z128, z129, z130, z131, z132, z133, z134, z135, z136, z137, z138, z139, z140, z141, z142, z143, z144, z145, z146, z147, z148, z149, z150, z151, z152, z153, z154, z155, z156, z157, z158, z159, z160, z161, z162, z163, z164, z165, z166, z167, z168, z169, z170, z171, z172, z173, z174, z175, z176, z177, z178, z179, z180, z181, z182, z183, z184, z185, z186, z187, z188, z189, z190, z191, z192, z193, z194, z195, z196, z197, z198, z199, z200, z201, z202, z203, z204, z205, z206, z207, z208, z209, z210, z211, z212, z213, z214, z215, z216, z217, z218, z219, z220, z221, z222, z223, z224, z225, z226, z227, z228, z229, z230, z231, z232, z233, z234, z235, z236, z237, z238, z239, z240, z241, z242, z243, z244, z245, z246, z247, z248, z249, z250, z251, z252, z253, z254, z255, z256, z257, z258, z259, z260, z261, z262, z263, z264, z265, z266, z267, z268, z269, z270, z271, z272, z273, z274, z275, z276, z277, z278, z279, z280, z281, z282, z283, z284, z285, z286, z287, z288, z289, z290, z291, z292, z293, z294, z295, z296, z297, z298, z299, z300, z301, z302, z303, z304, z305, z306, z307, z308, z309, z310, z311, z312, z313, z314, z315, z316, z317, z318, z319, z320, z321, z322, z323, z324, z325, z326, z327, z328, z329, z330, z331, z332, z333, z334, z335, z336, z337, z338, z339, z340, z341, z342, z343, z344, z345, z346, z347, z348, z349, z350, z351, z352, z353, z354, z355, z356, z357, z358, z359, z360, z361, z362, z363, z364, z365, z366, z367, z368, z369, z370, z371, z372, z373, z374, z375, z376, z377, z378, z379, z380, z381, z382, z383, z384, z385, z386, z387, z388, z389, z390, z391, z392, z393, z394, z395, z396, z397, z398, z399, z400, z401, z402, z403, z404, z405, z406, z407, z408, z409, z410, z411, z412, z413, z414, z415, z416, z417, z418, z419, z420, z421, z422, z423, z424, z425, z426, z427, z428, z429, z430, z431, z432, z433, z434, z435, z436, z437, z438, z439, z440, z441, z442, z443, z444, z445, z446, z447, z448, z449, z450, z451, z452, z453, z454, z455, z456, z457, z458, z459, z460, z461, z462, z463, z464, z465, z466, z467, z468, z469, z470, z471, z472, z473, z474, z475, z476, z477, z478, z479, z480, z481, z482, z483, z484, z485, z486, z487, z488, z489, z490, z491, z492, z493, z494, z495, z496, z497, z498, z499, z500, z501, z502, z503, z504, z505, z506, z507, z508, z509, z510, z511, z512, z513, z514, z515, z516, z517, z518, z519, z520, z521, z522, z523, z524, z525, z526, z527, z528, z529, z530, z531, z532, z533, z534, z535, z536, z537, z538, z539, z540, z541, z542, z543, z544, z545, z546, z547, z548, z549, z550, z551, z552, z553, z554, z555, z556, z557, z558, z559, z560, z561, z562, z563, z564, z565, z566, z567, z568, z569, z570, z571, z572, z573, z574, z575, z576, z577, z578, z579, z580, z581, z582, z583, z584, z585, z586, z587, z588, z589, z590, z591, z592, z593, z594, z595, z596, z597, z598, z599, z600, z601, z602, z603, z604, z605, z606, z607, z608, z609, z610, z611, z612, z613, z614, z615, z616, z617, z618, z619, z620, z621, z622, z623, z624, z625, z626, z627, z628, z629, z630, z631, z632, z633, z634, z635, z636, z637, z638, z639, z640, z641, z642, z643, z644, z645, z646, z647, z648, z649, z650, z651, z652, z653, z654, z655, z656, z657, z658, z659, z660, z661, z662, z663, z664, z665, z666, z667, z668, z669, z670, z671, z672, z673, z674, z675, z676, z677, z678, z679, z680, z681, z682, z683, z684, z685, z686, z687, z688, z689, z690, z691, z692, z693, z694, z695, z696, z697, z698, z699, z700, z701, z702, z703, z704, z705, z706, z707, z708, z709, z710, z711, z712, z713, z714, z715, z716, z717, z718, z719, z720, z721, z722, z723, z724, z725, z726, z727, z728, z729, z730, z731, z732, z733, z734, z735, z736, z737, z738, z739, z740, z741, z742, z743, z744, z745, z746, z747, z748, z749, z750, z751, z752, z753, z754, z755, z756, z757, z758, z759, z760, z761, z762, z763, z764, z765, z766, z767, z768, z769, z770, z771, z772, z773, z774, z775, z776, z777, z778, z779, z780, z781, z782, z783, z784, z785, z786, z787, z788, z789, z790, z791, z792, z793, z794, z795, z796, z797, z798, z799, z800, z801, z802, z803, z804, z805, z806, z807, z808, z809, z810, z811, z812, z813, z814, z815, z816, z817, z818, z819, z820, z821, z822, z823, z824, z825, z82
```

```

z10 = Cmul(mu, Complex(mu.r + nn, mu.i));
z11 = Cmul(z10, Complex(mu.r + nn + 1.0, mu.i));
z12 = Cdiv(Cmul(z8, z9), z11);

z13 = Cadd(z7, z12);

Q_2 = Cdiv(Cmul(z2, z13), z3);

Q = Cadd(Q_1, Q_2);

Rfs = Q.r;

return Rfs;
}

```

```

static int Out_Laplace(double s, NumFunc_1 *L, NumFunc_1 *Up, NumFunc_1 *Rebate
{
    int N = 15, M = 11;
    int i;
    double price, delta, price2, delta2;
    double xx, y, hh, sum, sum2, Avg, Avg2, Fun, Fun2, j, S[13], Q[13], U, tt, d,
    double St, St2, Lower, Upper, v, pp;
    double sigma2;
    double nu, h, h2, a, a2, b, b2, CTtK;

    /* Inversion Variables*/
    double A;
    dcomplex z;

    /*Inversion parameters*/
    A = 19.1;

    Upper = (Up->Compute)(Up->Par, 0.0);
    Lower = (L->Compute)(L->Par, 0.);
    K = PayOff->Par[0].Val.V_PDDOUBLE;
    pp = 1.e-8;
    St = s;
    St2 = s * (1. + pp);
    v = r - divid;
    sigma2 = sigma * sigma;

```

```

nu = (1.0 / sigma2) * (v - 0.5 * sigma2);
h = K / St;
h2 = K / St2;
a = log(St / Lower);
a2 = log(St2 / Lower);
b = log(Upper / St);
b2 = log(Upper / St2);

/* INVERSION */
tt = t;
xx = A / (2 * tt);
hh = M_PI / tt;
z = Complex(xx / sigma2, 0.0);

sum = fnRf_3(z, a, b, h, nu) * .5 / sigma2;
sum2 = fnRf_3(z, a2, b2, h2, nu) * .5 / sigma2;

/* Computation of S[0]=s(n) which approximate f(t) */
for (i = 1; i <= N; i++)
{
    y = (double)i * hh;
    z = Complex(xx / sigma2, y / sigma2);
    j = PNL_ALTERNATE(i);
    sum = sum + j * fnRf_3(z, a, b, h, nu) / sigma2;
    sum2 = sum2 + j * fnRf_3(z, a2, b2, h2, nu) / sigma2;
}

S[0] = sum;
Q[0] = sum2;
/* End of Inversion */

/* Computation of s(n+p) p<=M+1 for Euler appromations */
for (i = 1; i <= M + 1; i++)
{
    y = (double)(N + i) * hh;
    z = Complex(xx / sigma2, y / sigma2);
    j = PNL_ALTERNATE(N + i);
    S[i] = S[i - 1] + j * fnRf_3(z, a, b, h, nu) / sigma2;
    Q[i] = Q[i - 1] + j * fnRf_3(z, a2, b2, h2, nu) / sigma2;
}

```

```

/* Computation of Euler appromations */
Avg = 0.;
Avg2 = 0.;
for (i = 1; i <= M + 1; i++)
{
    Avg = Avg + pnl_sf_choose(M, i - 1) * S[i - 1];
    Avg2 = Avg2 + pnl_sf_choose(M, i - 1) * Q[i - 1];
}

d = pow(2.0, (double)M);
U = exp(A / 2.) / tt;

/*f(t) values*/
Fun = U * Avg / d;
Fun2 = U * Avg2 / d;

/*Black-Sholes price for call option*/

pnl_cf_call_bs(1., h, t, r, divid, sigma, &price, &delta);
pnl_cf_call_bs(1., h2, t, r, divid, sigma, &price2, &delta2);

CTtK = St * price - St * exp(-r * t) * Fun;

/*Price*/
*ptprice = CTtK;

/*Delta*/
*ptdelta = (CTtK - (price2 - price) / (h2 - h) * K) / St - exp(-r * t) * (Fun2 - Fun);

return OK;
}

int CALC(AP_Out_Laplace)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, divid;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);

```

```

    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

    return Out_Laplace(ptMod->S0.Val.V_PDOUBLE, ptOpt->LowerLimit.Val.V_NUMFUNC_1
}

static int CHK_OPT(AP_Out_Laplace)(void *Opt, void *Mod)
{
    Option *ptOpt = (Option *)Opt;
    TYPEOPT *opt = (TYPEOPT *) (ptOpt->TypeOpt);

    if ((opt->TwoDoubleStep).Val.V_BOOL == FALSE)
        if ((opt->Parisian).Val.V_BOOL == FALSE)
            if ((opt->RebOrNo).Val.V_BOOL == NOREBATE)
                if ((strcmp(((Option *)Opt)->Name, "DoubleCallOutEuro") == 0))
                    return OK;

    return WRONG;
}

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    return OK;
}

PricingMethod MET(AP_Out_Laplace) =
{
    "AP_Out_Laplace",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(AP_Out_Laplace),
    {"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
    CHK_OPT(AP_Out_Laplace),
    CHK_ok,
    MET(Init)
} ;

```