

[Help](#)

```
extern "C" {
#include "
href../../mod/jarrowyildirim1d/jarrowyildirim1d_stdh
}
extern "C" {
    extern char premia_data_dir[MAX_PATH_LEN];
    extern char *path_sep;
}

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#else
static double *tm, *PN, *PR, *ZCSR, *ZCSRT, *tm_zcsr, *PNT;
static int Nvalue, Nvalue1; /*Number of value read for PN*/
static char init[] = "nominal_zcb_prices.dat";
static char init1[] = "zcii_swap_rates.dat";
static FILE *Entrees; /*File variable of the code*/
static FILE *Entrees1;
/*Read Nominal Zero Coupon Bond*/
static int lecture_PN()
{

    int i;
    char ligne[20];
    char *pligne;
    double p, tt;
    char data[MAX_PATH_LEN];

    sprintf(data, "%s%s%s", premia_data_dir, path_sep, init);
    Entrees = fopen(data, "r");

    if (Entrees == NULL)
    {
        printf("Le FICHER %s N'A PU ETRE OUVERT. VERIFIER LE CHEMIN\ n", data);
    }

    i = 0;
    pligne = ligne;

    PN = new double[100];
```

```

PNT = new double[100];
tm = new double[100];
PR = new double[100];

while (1)
{
    pligne = fgets(ligne, sizeof(ligne), Entrees);
    if (pligne == NULL) break;
    else
    {
        sscanf(ligne, "%lf t=%lf", &p, &tt);

        PN[i] = p;
        tm[i] = tt;
        i++;
    }
}
Nvalue = i;
fclose(Entrees);

return i;
}

/*Read Zero Coupon Swap Rates*/
static int lecture_ZCSR()
{
    int i;
    char ligne[20];
    char *pligne;
    double p, tt;
    char data[MAX_PATH_LEN];

    sprintf(data, "%s%s%s", premia_data_dir, path_sep, init1);
    Entrees1 = fopen(data, "r");

    ZCSR = new double[100];
    ZCSRT = new double[100];
    tm_zcsr = new double[100];
    if (Entrees1 == NULL)
    {

```

```

    printf("Le FICHIER %s N'A PU ETRE OUVERT. VERIFIER LE CHEMIN\ n", data);
}

i = 0;
pligne = ligne;

while (1)
{
    pligne = fgets(ligne, sizeof(ligne), Entrees1);
    if (pligne == NULL) break;
    else
    {
        sscanf(ligne, "%lf t=%lf", &p, &tt);
        ZCSR[i] = p;
        tm_zcsr[i] = tt;

        i++;
    }
}
Nvalue1 = i;
fclose(Entrees1);

return i;
}

static double bond_zcn(double T)
{
    double POT;
    int i = 0;

    if (T > 0)
    {
        while (tm[i] < T && i < Nvalue)
        {
            i = i + 1;
        }

        if (i == 0)
        {
            POT = 1 * (1 - T / tm[0]) + PN[0] * (T / tm[0]);

```

```

    }
else
{
    if (i < Nvalue)
    {

        POT = PN[i - 1] * (tm[i] - T) / (tm[i] - tm[i - 1]) + PN[i] * (T -
/*printf("values %d %f %f %f %f\ n",i,PN[i-1],PN[i],tm[i-1],tm[i])
    }
    else
    {
        POT = PN[i - 1] + (T - tm[i - 1]) * (PN[i - 1] - PN[i - 2]) / (tm[
    }
}
}
else
{
    POT = 1;
}

return POT;
}
static double bond_zcsr(double T)
{
    double POT;
    int i = 0;

    if (T > 0)
    {
        while (tm_zcsr[i] < T && i < Nvalue1)
        {
            i = i + 1;
        }

        if (i == 0)
        {
            POT = 1 * (1 - T / tm_zcsr[0]) + ZCSR[0] * (T / tm_zcsr[0]);
        }
    }
    else
    {
        if (i < Nvalue)

```

```

        {
            POT = ZCSR[i - 1] * (tm_zcsr[i] - T) / (tm_zcsr[i] - tm_zcsr[i - 1]);
        }
        else
        {
            POT = ZCSR[i - 1] + (T - tm_zcsr[i - 1]) * (ZCSR[i - 1] - ZCSR[i - 2]);
        }
    }
}
else
{
    POT = 0;
}
return POT;
}

```

/*Compute ZeroCoupon Bond Price in Creal Economy*/

```

static void CalculatePR(int tenor_order, double tenor, double swap_mat)
{
    int i, j;

    i = lecture_PN();
    j = lecture_ZCSR();
    i = MIN(i, j);
    if (swap_mat > tm[i - 1])
    {
        printf("\nError : time bigger than the last time value entered in market\n");
        exit(EXIT_FAILURE);
    }
    else
    {
        PNT[0] = 1.;
        for (int j = 1; j < tenor_order + 1; j++)
        {
            PNT[j] = bond_zcn((double)j * tenor);
            ZCSRT[j] = bond_zcsr((double)j * tenor);
            /*printf("%f %f\n", PNT[j], ZCSRT[j]);*/
        }

        PR[0] = 1.0;
    }
}

```

```

        for (int j = 1; j < tenor_order + 1; j++)
        {
            PR[j] = PNT[j] * pow((1.0 + ZCSRT[j]), (double)j * tenor);
        }
    }
}

/*Compute Function Beta in Page 91 of Moreni thesis*/
static double Beta(double a, double t1, double t2)
{
    double beta = 0.0;
    if ((t2 - t1) == 0.0)
    {
        beta = 1.0;
    }
    else
    {
        beta = (1.0 - exp(-a * (t2 - t1))) / a;
    }

    return beta;
}

/*compute function gamma in page 92 of Moreni thesis */
static double ParameterGamma(double t, double tenor, int tenor_order, double si

{
    double a1 = sigmar * Beta(ar, (tenor_order - 1) * tenor, tenor_order * tenor);
    double a2 = Beta(ar, t, (tenor_order - 1) * tenor);
    double a3 = rhorcpi * sigma_cpi;

    double a4 = 0.5 * sigmar * Beta(ar, t, (tenor_order - 1) * tenor);
    double a5 = rhonr * sigman / (an + ar) * (1 + ar * Beta(an, t, tenor * (tenor_order - 1)));
    double a6 = rhonr * sigman / (an + ar) * Beta(an, t, tenor * (tenor_order - 1));
    double result = a1 * (a2 * (a3 - a4 + a5) - a6);

    return result;
}

/*Compute Year-On-Year Inflation-Indexed Swaps*/
static int cf_yyis1d(NumFunc_1 *p, double t, double swap_mat, double tenor, dou

```

```

{
    int TenorMax = (int)((swap_mat - t) / tenor);

    /*Compute ZeroCoupon Bond Price in Creal Economy*/
    CalculatePR(TenorMax, tenor, swap_mat);
    double strike = p->Par[0].Val.V_DOUBLE;

    /*compute floating leg and fixed leg*/
    double fixedleg = 0.0;    //fixed leg
    double fl = 0.0;         // floating leg

    for (int i = 1; i < TenorMax + 1; i++)
    {
        fl = fl + psi * PNT[i - 1] * PR[i] / PR[i - 1] * exp(ParameterGamma(t, tenor));
        fixedleg = fixedleg + (phi * strike + psi) * PNT[i];
    }

    /*Price*/
    * price = Nominal * fl - Nominal * fixedleg;

    delete [] tm;
    delete [] PN;
    delete [] PNT;
    delete [] PR;
    delete [] ZCSR;
    delete [] ZCSRT;
    delete [] tm_zcsr;

    return OK;
}
#endif //PremiaCurrentVersion

extern "C" {
    #if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#else"
        static int CHK_OPT(CF_YI_YYIIS)(void *Opt, void *Mod)
        {

```

```

        return NONACTIVE;
    }
    int CALC(CF_YI_YYIIS)(void *Opt, void *Mod, PricingMethod *Met)
    {
        return AVAILABLE_IN_FULL_PREMIA;
    }
#else
    int CALC(CF_YI_YYIIS)(void *Opt, void *Mod, PricingMethod *Met)
    {
        TYPEOPT *ptOpt = (TYPEOPT *)Opt;
        TYPEMOD *ptMod = (TYPEMOD *)Mod;

        return cf_yyiis1d(ptOpt->PayOff.Val.V_NUMFUNC_1, ptMod->T.Val.V_DATE, ptOpt->
    }

    static int CHK_OPT(CF_YI_YYIIS)(void *Opt, void *Mod)
    {
        if ((strcmp(((Option *)Opt)->Name, "YearOnYearInflationIndexedSwap") == 0))
            return OK;
        else
            return WRONG;
    }

#endif //PremiaCurrentVersion
    static int MET(Init)(PricingMethod *Met, Option *Opt)
    {
        if (Met->init == 0)
        {
            Met->init = 1;
        }
        return OK;
    }

    PricingMethod MET(CF_YI_YYIIS) =
    {
        "CF_JarrowYildirim1d_YYIIS",
        {" ", PREMIA_NULLTYPE, {0}, FORBID}},
        CALC(CF_YI_YYIIS),
        {"Price", DOUBLE, {100}, FORBID} , {" ", PREMIA_NULLTYPE, {0}, FORBID}},
        CHK_OPT(CF_YI_YYIIS),

```



```
        CHK_ok,  
        MET(Init)  
    } ;  
}
```