```
/*******************************************************************************
/*                              INTEL CONFIDENTIAL
/*   Copyright(C) 2006-2008 Intel Corporation. All Rights Reserved.
/*   The source code contained  or  described herein and all documents related to
/*   the source code ("Material") are owned by Intel Corporation or its suppliers
/*   or licensors.  Title to the  Material remains with  Intel Corporation or its
/*   suppliers and licensors. The Material contains trade secrets and proprietary
/*   and  confidential  information of  Intel or its suppliers and licensors. The
/*   Material  is  protected  by  worldwide  copyright  and trade secret laws and
/*   treaty  provisions. No part of the Material may be used, copied, reproduced,
/*   modified, published, uploaded, posted, transmitted, distributed or disclosed
/*   in any way without Intel's prior express written permission.
/*   No license  under any  patent, copyright, trade secret or other intellectual
/*   property right is granted to or conferred upon you by disclosure or delivery
/*   of the Materials,  either expressly, by implication, inducement, estoppel or
/*   otherwise.  Any  license  under  such  intellectual property  rights must be
/*   express and approved by Intel in writing.
/*
/*******************************************************************************
/*  Content:
/*  Intel MKL example of RCI Flexible Generalized Minimal RESidual method with
/*  ILU0 Preconditioner
/*******************************************************************************

/*---------------------------------------------------------------------------
/*  Example program for solving non-degenerate system of equations.
/*  Full functionality of RCI FGMRES solver is exploited. Example shows how
/*  ILU0 preconditioner accelerates the solver by reducing the number of
/*  iterations.
/*---------------------------------------------------------------------------*/

#include <stdio.h>
#include "math.h"
#include "mkl_blas.h"
#include "mkl_spblas.h"
#include "mkl_rci.h"
#define N 4
#define size 128

int main(void)
{
/*---------------------------------------------------------------------------
/* Define arrays for the upper triangle of the coefficient matrix
/* Compressed sparse row storage is used for sparse representation
/*---------------------------------------------------------------------------*/
        MKL_INT ia[5]={1,4,7,10,13};
        MKL_INT ja[12]={1,2,3,1,2,4,1,3,4,2,3,4};
        double A[12]={4.,-1.,-1.,-1.,4.,-1.,-1.,4.,-1.,-1.,-1.,4.};
/*---------------------------------------------------------------------------
/* Allocate storage for the ?par parameters and the solution/rhs/residual vectors
/*---------------------------------------------------------------------------*/
        MKL_INT ipar[size];
        double dpar[size], tmp[N*(2*N+1)+(N*(N+9))/2+1];
```

```
        double trvec[N],bilu0[12];
        double expected_solution[N]={1.0,1.0,1.0,1.0};
        double rhs[N], b[N];
        double computed_solution[N];
        double residual[N];

        MKL_INT matsize=12, incx=1, ref_nit=2;
        double ref_norm2=7.772387E+0, nrm2;
/*---------------------------------------------------------------------------
/* Some additional variables to use with the RCI (P)FGMRES solver
/*---------------------------------------------------------------------------*/
        MKL_INT itercount,ierr=0;
        MKL_INT RCI_request, i, ivar;
        double dvar;
        char cvar,cvar1,cvar2;

        printf("-------------------------------------------------------\n");
        printf("The FULLY ADVANCED example RCI FGMRES with ILU0 preconditioner\n");
        printf("to solve the non-degenerate algebraic system of linear equations\n");
        printf("-------------------------------------------------------\n\n");
/*---------------------------------------------------------------------------
/* Initialize variables and the right hand side through matrix-vector product
/*---------------------------------------------------------------------------*/
        ivar=N;
        cvar='N';
        mkl_dcsrgemv(&cvar, &ivar, A, ia, ja, expected_solution, rhs);
/*---------------------------------------------------------------------------
/* Save the right-hand side in vector b for future use
/*---------------------------------------------------------------------------*/
        i=1;
        dcopy(&ivar, rhs, &i, b, &i);
/*---------------------------------------------------------------------------
/* Initialize the initial guess
/*---------------------------------------------------------------------------*/
        for(i=0;i<N;i++)
        {
                computed_solution[i]=0.0;
        }
        computed_solution[0]=100.0;

        /*---------------------------------------------------------------------------
        /* Initialize the solver
        /*---------------------------------------------------------------------------*/
        dfgmres_init(&ivar, computed_solution, rhs, &RCI_request, ipar, dpar, tmp);
        if (RCI_request!=0) goto FAILED;

/*---------------------------------------------------------------------------
/* Calculate ILU0 preconditioner.
/*                      !ATTENTION!
/* DCSRILU0 routine uses some IPAR, DPAR set by DFGMRES_INIT routine.
/* Important for DCSRILU0 default entries set by DFGMRES_INIT are
/* ipar[1] = 6 - output of error messages to the screen,
/* ipar[5] = 1 - allow output of errors,
/* ipar[30]= 0 - abort DCSRILU0 calculations if routine meets zero diagonal element.
```

```
 /*
 /* If ILU0 is going to be used out of MKL FGMRES context, than the values
 /* of ipar[1], ipar[5], ipar[30], dpar[30], and dpar[31] should be user
 /* provided before the DCSRILU0 routine call.
 /*
 /* In this example, specific for DCSRILU0 entries are set in turn:
 /* ipar[30]= 1 - change small diagonal value to that given by dpar[31],
 /* dpar[30]= 1.E-20 instead of the default value set by DFGMRES_INIT.
 /*                  It is a small value to compare a diagonal entry with it.
 /* dpar[31]= 1.E-16 instead of the default value set by DFGMRES_INIT.
 /*                  It is the target value of the diagonal value if it is
 /*                  small as compared to dpar[30] and the routine should change
 /*                  it rather than abort DCSRILU0 calculations.
 /*---------------------------------------------------------------------------*/

         ipar[30]=1;
         dpar[30]=1.E-20;
         dpar[31]=1.E-16;

         dcsrilu0(&ivar, A, ia, ja, bilu0, ipar, dpar, &ierr);
         nrm2=dnrm2(&matsize, bilu0, &incx );

         if (ierr!=0)
         {
           printf("Preconditioner dcsrilu0 has returned the ERROR code %d", ierr);
           goto FAILED1;
         }

         /*---------------------------------------------------------------------
         /* Set the desired parameters:
         /* do the restart after 2 iterations
         /* LOGICAL parameters:
         /* do not do the stopping test for the maximal number of iterations
         /* do the Preconditioned iterations of FGMRES method
         /* Set parameter ipar[10] for preconditioner call. For this example,
     /* it reduces the number of iterations.
         /* DOUBLE PRECISION parameters
         /* set the relative tolerance to 1.0D-3 instead of default value 1.0D-6
     /* NOTE. Preconditioner may increase the number of iterations for an
     /* arbitrary case of the system and initial guess and even ruin the
     /* convergence. It is user's responsibility to use a suitable preconditioner
     /* and to apply it skillfully.
     /*---------------------------------------------------------------------*/
         ipar[14]=2;
         ipar[7]=0;
         ipar[10]=1;
         dpar[0]=1.0E-3;

         /*---------------------------------------------------------------------
         /* Check the correctness and consistency of the newly set parameters
         /*---------------------------------------------------------------------*/
         dfgmres_check(&ivar, computed_solution, rhs, &RCI_request, ipar, dpar, tmp);
         if (RCI_request!=0) goto FAILED;
         /*---------------------------------------------------------------------
```

```
        /* Print the info about the RCI FGMRES method
        /*---------------------------------------------------------------------------*/
        printf("Some info about the current run of RCI FGMRES method:\n\n");
        if (ipar[7])
        {
                printf("As ipar[7]=%d, the automatic test for the maximal number of
 iterations will be\n", ipar[7]);
                printf("performed\n");
        }
        else
        {
                printf("As ipar[7]=%d, the automatic test for the maximal number of
 iterations will be\n", ipar[7]);
                printf("skipped\n");
        }
        printf("+++\n");
        if (ipar[8])
        {
                printf("As ipar[8]=%d, the automatic residual test will be performed\n",
 ipar[8]);
        }
        else
        {
                printf("As ipar[8]=%d, the automatic residual test will be skipped\n",
 ipar[8]);
        }
        printf("+++\n");
        if (ipar[9])
        {
                printf("As ipar[9]=%d the user-defined stopping test will be requested
 via\n", ipar[9]);
                printf("RCI_request=2\n");
        }
        else
        {
                printf("As ipar[9]=%d, the user-defined stopping test will not be requested,
 thus,\n", ipar[9]);
                printf("RCI_request will not take the value 2\n");
        }
        printf("+++\n");
        if (ipar[10])
        {
                printf("As ipar[10]=%d, the Preconditioned FGMRES iterations will be
 performed, thus,\n", ipar[10]);
                printf("the preconditioner action will be requested via RCI_request=3\n");
        }
        else
        {
                printf("As ipar[10]=%d, the Preconditioned FGMRES iterations will not be
 performed,\n", ipar[10]);
                printf("thus, RCI_request will not take the value 3\n");
        }
        printf("+++\n");
        if (ipar[11])
```

```
        {
                printf("As ipar[11]=%d, the automatic test for the norm of the next
 generated vector is\n", ipar[11]);
                printf("not equal to zero up to rounding and computational errors will be
 performed,\n");
                printf("thus, RCI_request will not take the value 4\n");
        }
        else
        {
                printf("As ipar[11]=%d, the automatic test for the norm of the next
 generated vector is\n", ipar[11]);
                printf("not equal to zero up to rounding and computational errors will be
 skipped,\n");
                printf("thus, the user-defined test will be requested via RCI_request=4\n");
        }
        printf("+++\n\n");
        /*---------------------------------------------------------------------------
        /* Compute the solution by RCI (P)FGMRES solver with preconditioning
        /* Reverse Communication starts here
        /*---------------------------------------------------------------------------*/
 ONE:  dfgmres(&ivar, computed_solution, rhs, &RCI_request, ipar, dpar, tmp);
        /*---------------------------------------------------------------------------
        /* If RCI_request=0, then the solution was found with the required precision
        /*---------------------------------------------------------------------------*/
        if (RCI_request==0) goto COMPLETE;
        /*---------------------------------------------------------------------------
        /* If RCI_request=1, then compute the vector A*tmp[ipar[21]-1]
        /* and put the result in vector tmp[ipar[22]-1]
        /*---------------------------------------------------------------------------
        /* NOTE that ipar[21] and ipar[22] contain FORTRAN style addresses,
        /* therefore, in C code it is required to subtract 1 from them to get C style
        /* addresses
        /*---------------------------------------------------------------------------*/
        if (RCI_request==1)
        {
                mkl_dcsrgemv(&cvar, &ivar, A, ia, ja, &tmp[ipar[21]-1], &tmp[ipar[22]-1]);
                goto ONE;
        }
        /*---------------------------------------------------------------------------
        /* If RCI_request=2, then do the user-defined stopping test
        /* The residual stopping test for the computed solution is performed here
        /*---------------------------------------------------------------------------
        /* NOTE: from this point vector b[N] is no longer containing the right-hand
        /* side of the problem! It contains the current FGMRES approximation to the
        /* solution. If you need to keep the right-hand side, save it in some other
        /* vector before the call to dfgmres routine. Here we saved it in vector
        /* rhs[N]. The vector b is used instead of rhs to preserve the
        /* original right-hand side of the problem and guarantee the proper
        /* restart of FGMRES method. Vector b will be altered when computing the
        /* residual stopping criterion!
        /*---------------------------------------------------------------------------*/
        if (RCI_request==2)
        {
                /* Request to the dfgmres_get routine to put the solution into b[N] via
```

```
 ipar[12]
                /*---------------------------------------------------------------------------
 -
                /* WARNING: beware that the call to dfgmres_get routine with ipar[12]=0 at
 this stage may
                /* destroy the convergence of the FGMRES method, therefore, only advanced
 users should
                /* exploit this option with care */
                ipar[12]=1;
                /* Get the current FGMRES solution in the vector b[N] */
                dfgmres_get(&ivar, computed_solution, b, &RCI_request, ipar, dpar, tmp,
 &itercount);
                /* Compute the current true residual via MKL (Sparse) BLAS routines */
                mkl_dcsrgemv(&cvar, &ivar, A, ia, ja, b, residual);
                dvar=-1.0E0;
                i=1;
                daxpy(&ivar, &dvar, rhs, &i, residual, &i);
                dvar=dnrm2(&ivar,residual,&i);
                if (dvar<1.0E-3) goto COMPLETE;

                else goto ONE;
        }
        /*---------------------------------------------------------------------------
        /* If RCI_request=3, then apply the preconditioner on the vector
        /* tmp[ipar[21]-1] and put the result in vector tmp[ipar[22]-1]
        /*---------------------------------------------------------------------------
        /* NOTE that ipar[21] and ipar[22] contain FORTRAN style addresses,
        /* therefore, in C code it is required to subtract 1 from them to get C style
        /* addresses
        /* Here is the recommended usage of the result produced by ILU0 routine
    /* via standard MKL Sparse Blas solver routine mkl_dcsrtrsv.
    /*---------------------------------------------------------------------------*/
        if (RCI_request==3)
        {
                cvar1='L';
                cvar='N';
                cvar2='U';
                mkl_dcsrtrsv(&cvar1,&cvar,&cvar2,&ivar,bilu0,ia,ja,&tmp[ipar[21]-1],trvec);
                cvar1='U';
                cvar='N';
                cvar2='N';
                mkl_dcsrtrsv(&cvar1,&cvar,&cvar2,&ivar,bilu0,ia,ja,trvec,&tmp[ipar[22]-1]);
                goto ONE;
        }
        /*---------------------------------------------------------------------------
        /* If RCI_request=4, then check if the norm of the next generated vector is
        /* not zero up to rounding and computational errors. The norm is contained
        /* in dpar[6] parameter
        /*---------------------------------------------------------------------------*/
        if (RCI_request==4)
        {
                if (dpar[6]<1.0E-12) goto COMPLETE;
                else goto ONE;
        }
```

```
        /*---------------------------------------------------------------------------
        /* If RCI_request=anything else, then dfgmres subroutine failed
        /* to compute the solution vector: computed_solution[N]
        /*---------------------------------------------------------------------------*/
        else
        {
                goto FAILED;
        }
        /*---------------------------------------------------------------------------
        /* Reverse Communication ends here
        /* Get the current iteration number and the FGMRES solution (DO NOT FORGET to
        /* call dfgmres_get routine as computed_solution is still containing
        /* the initial guess!). Request to dfgmres_get to put the solution
        /* into vector computed_solution[N] via ipar[12]
        /*---------------------------------------------------------------------------*/
COMPLETE:   ipar[12]=0;
        dfgmres_get(&ivar, computed_solution, rhs, &RCI_request, ipar, dpar, tmp,
&itercount);
        /*---------------------------------------------------------------------------
        /* Print solution vector: computed_solution[N] and the number of iterations:
itercount
        /*---------------------------------------------------------------------------*/
        printf("The system has been solved \n");

        printf("\nThe following solution has been obtained: \n");
        for (i=0;i<N;i++)
        {
                printf("computed_solution[%d]=%e\n",i,computed_solution[i]);
        }
        printf("\nThe expected solution is: \n");
        for (i=0;i<N;i++)
        {
                printf("expected_solution[%d]=%e\n",i,expected_solution[i]);
        }
        printf("\nNumber of iterations: %d\n" ,itercount);
        printf("\n");

        if(itercount==ref_nit && fabs(ref_norm2-nrm2)<1.e-6)
        {
        printf("-------------------------------------------------------\n");
        printf("C example of FGMRES with ILU0 preconditioner \n");
        printf("has successfully PASSED all stages of computations\n");
        printf("-------------------------------------------------------\n");

        return 0;
        }
        else
        {
        printf("Probably, the preconditioner was computed incorrectly:\n");
        printf("Either the preconditioner norm %e differs from the expected norm
%e\n",nrm2,ref_norm2);
        printf("and/or the number of iterations %d differs from the expected number
%d\n",itercount,ref_nit);
        printf("-------------------------------------------------------------------\n");
```

```
        printf("Unfortunately, FGMRES+ILU0 C example has FAILED\n");
        printf("-----------------------------------------------------------------\n");
        return 0;
        }
FAILED:
        printf("The solver has returned the ERROR code %d \n", RCI_request);
FAILED1:
        printf("-----------------------------------------------------------------\n");
        printf("Unfortunately, FGMRES+ILU0 C example has FAILED\n");
        printf("-----------------------------------------------------------------\n");
        return 0;
}
```