# MKL a lineární solvery PARDISO, CG a FGMRES

## Solver PARDISO

– Parallel Sparse Direct Linear Solver
– přímý solver pro soustavy lineárních algebraických rovnic v R nebo C ve Fortranu nebo C
  tj. řeší A*X=B, kde A je n x n, X a B jsou n x nrhs (provádí se LU, LDL nebo $LL^T$ faktorizace)
– matice soustavy A: řídká (a velká)
  symetrická ($a_{kl}=a_{lk}$), strukturálně symetrická (když $a_{kl}$, tak i $a_{lk}$) nebo nesymetrická
  reálná, hermitovská nebo komplexní
  pozitivně definitní nebo indefinitní
– využíváno procedur knihovny BLAS
– paralelizováno pro SMP (symmetric multiprocessing = sdílená paměť), zrychlení až 7krát na 8 procesorech
– http://www.pardiso-project.org, v MKL jedna z implementací
– MKL (verze 10): dokumentace – Reference Manual str. 2161–2252
  rozhraní – originál PARDISO
  – DSS (direct sparse solver)
  verze OOC (out-of-core) pro soustavy nad rozsah fyzické paměti
– algoritmus:
  1. analýza a symbolická faktorizace
  2. numerická faktorizace
  3. substituce a iterační zpřesnění
– formát uložení matice A:
  PARDISO-varianta formátu CSR (compressed sparse row), viz MKL Reference Manual str. 2860
  a(*) pro prvky, ia(*) pro řádkový index, ja(*) pro sloupcový index
  velikost polí a, ja rovna počtu nenulových prvků matice A, velikost pole ia rovna počet_řádků+1
  nenulové prvky řádku ukládány zleva doprava
  pro symetrické matice musí být zahrnuty diagonální prvky a prvky horního trojúhelníka
  A => a: a(i)=...
  a => A: A(i,j)=...

## Solvery CG a FGMRES

– Conjugate Gradient Solver & Flexible Generalized Minimal Residual Solver
– iterační solvery pro soustavy lineárních algebraických rovnic v R (ne v C) ve Fortranu nebo C
  tj. řeší A*X=B, kde A je n x n, X a B jsou n x nrhs
– matice soustavy A: symetrická pozitivně definitní (pro CG) nebo nesymetrická (pro FGMRES)
– rozhraní RCI ISS (iterative sparse solvers based on reverse communication interface)
– akcelerátory: preconditioners ILU0 a ILUT

Rozhraní originál PARDISO: řešení „vše v jednom"

– volání podprogramu:

```
SUBROUTINE pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja, perm, nrhs, iparm, msglvl, b, x, error)
INTEGER(4)  pt(64)                              ! pro 32bitové operační systémy
INTEGER(8)  pt(64)                              ! pro 64bitové operační systémy
INTEGER(4)  maxfct, mnum, mtype, phase, n, nrhs, error, ia(*), ja(*), perm(*), iparm(*)
REAL(8)     a(*), b(n,nrhs), x(n,nrhs)          ! pro real soustavu
COMPLEX(8)  a(*), b(n,nrhs), x(n,nrhs)          ! pro complex soustavu (neboli COMPLEX*16)
```

– kde jsou:

| | |
|---|---|
| pt(64) | interní ukazatele |
| maxfct | počet různých matic A téže struktury (obvykle 1) |
| mnum | index aktuální matice (obvykle 1) |
| mtype | 11 reálná nesymetrická, –2 reálná symetrická indefinitní, 2 reálná symetrická poz. def. aj. |
| phase | ij, kde i a j jsou pořadová čísla počáteční, resp. koncové fáze solveru<br>(viz pořadí fází algoritmu: 1 analýza, 2 faktorizace, 3 substituce)<br>př. 13 všechno, 22 pouze numerická faktorizace, –1 uvolnění paměti (0 jen pro mnum) |
| n | počet rovnic (a neznámých) |
| a | real nebo complex vektor nenulových prvků matice |
| ia | i-tý prvek ukazuje pořadí prvního nenulového prvku A(i,:) v ja(:),<br>ia(n+1) je počet nenulových prvků v A plus 1 |
| ja | sloupcové indexy nenulových prvků z A |
| perm | permutační vektor |
| nrhs | počet pravých stran |
| iparm(64) | řídicí parametry, výstupní informace<br>iparm(0)=0 pro default hodnoty všech ostatních iparm s výjimkou iparm(3)<br>iparm(3)=počet procesorů, stejné číslo jako v proměnné MKL_NUM_THREADS |
| msglvl | 0 pro mlčení, 1 pro výpis statistiky |
| b | real nebo complex pravá strana (pravé strany), při iparm(6)=1 přepsáno řešením |
| x | real nebo complex řešení při iparm(6)=0 |
| error | 0 ok, –1 inconsistent input, –2 not enough memory, –4 zero pivot, –8 integer overflow aj. |

– vstupní parametry iparm:

| | | |
|---|---|---|
| | 1 | 0 pro default ostatních (kromě iparm(3)), 1 nikoliv |
| | 4 | 0 pro přímý solver (default), jinak řízení iteračních solverů (CG, CGS) |
| | 6 | 0 pro zápis řešení do x (default), 1 pro přepsání b |
| | 8 | max. počet iteračních zpřesnění (default 0, ale po perturbační pivotaci vždy 2) |
| | aj. | |
| MKL: | 60 | 0 pro in-core verzi, 1 pro out-of-core verzi podle potřeby (porovnáním s údajem v MB<br>v proměnné MKL_PARDISO_OOC_MAX_CORE_SIZE), 2 pro out-of-core verzi |

– výstupní parametry iparm:

| | |
|---|---|
| 7 | počet iteračních zpřesnění |
| 14 | počet perturbovaných pivotů |
| 15 | max. potřeba paměti v KB ve fázi 1 |
| 16 | permanentní potřeba paměti v KB pro fáze 2 a 3 |
| 17 | max. potřeba paměti pro fáze 2 a 3 |
| celková potřeba paměti: max(iparm(15), iparm(16)+iparm(17)) | |
| 18 | počet nenulových prvků ve faktorech (pokud na vstupu -1) |
| 19 | Mflops (pokud na vstupu –1) |
| 20 | počet CG iterací |
| MKL: | 61, 62, 63 jako 15, 16, 17, ale po in-core verzi |

Rozhraní DSS (Direct Sparse Solver): řešení po krocích

– potřeba deklarovat interní ukazatele (handles typu MKL_DSS_HANDLE) a jejich pomocí alokovat paměť
– možnost volit mezi in-core (všechna podle v paměti) a out-of-core (OOC) verzemi
– hlavičkové soubory s definovanými konstantami a rozhraním funkcí

```
f90:  INCLUDE 'mkl_dss.f90'
      TYPE(MKL_DSS_HANDLE) handle
f77:  INCLUDE 'mkl_dss.f77'
      INTEGER*8 handle      nebo    DOUBLE PRECISION handle
```

– volání INTEGER FUNCTIONs:

```
1. nejdříve...
   dss_create
   dss_define_structure
   dss_reorder
2. i v cyklu...
   dss_factor_real (nebo complex)          ! vícekrát pro více matic téže struktury
   dss_solve_real (nebo complex)           ! vícekrát pro více pravých stran
3. nakonec...
   dss_delete
4. ve vhodný čas...
   dss_statistics
```

– FUNCTION dss_create(handle, opt)

```
počáteční inicializace ukazatele handle
opt:    integer součet konstant definovaných v souboru mkl_dss
        default MKL_DSS_MSG_LVL_WARNING + MKL_DSS_TERM_LVL_ERROR
        možnost OOC:  ... + MKL_DSS_OOC_VARIABLE (default: OOC při potřebě nad 2 GB)
                      ... + MKL_DSS_OOC_STRONG (OOC vždy)
návratová hodnota: integer chybový kód (MKL_DSS_SUCCESS neboli 0 pro ok)
```

– FUNCTION dss_define_structure(handle, opt, rowIndex, nRows, nCols, columns, nNonZeros)

```
informace o struktuře matice A velikosti nRows x nCols (musí být čtvercová)
pole rowIndex a columns odpovídají polím ia a ja z rozhraní PARDISO
opt:    MKL_DSS_SYMMETRIC, MKL_DSS_SYMMETRIC_STRUCTURE, MKL_DSS_NON_SYMMETRIC
```

– atd. viz příklad

Rozhraní RCI ISS (Iterative Sparse Solver based on Reverse Communication Interface)

– algoritmy: RCI CG – Conjugate Gradient Solver pro symetrické pozitivně definitní matice
– verze s jednou nebo více pravými stranami (single nebo multiple rhs, srhs/mrhs)
RCI FGMRES – Flexible Generalized Minimal Residual Solver pro nesymetrické indefinitní matice

– reverzní komunikace: potřeba poskytovat solveru na jeho žádost (např.) součin matice a vektoru

– volání INTEGER SUBROUTINEs:

1. nejdříve...

| | | | |
|---|---|---|---|
| dcg_init | dcgmrhs_init | dfgmres_init | ! nastavení parametrů |

a případná změna parametrů

2. v cyklu

| | | | |
|---|---|---|---|
| dcg_check | dcgmrhs_check | dfgmres_check | ! kontrola parametrů |
| dcg | dcgmrhs | dfgmres | ! solver |
| dcg_get | dcgmrhs_get | dfgmres_get | ! vrací číslo iterace |

– rozhraní podprogramů:

SUBROUTINE dcg_init (n, x, b, RCI_request, ipar, dpar, tmp)
SUBROUTINE dcg_check (n, x, b, RCI_request, ipar, dpar, tmp)
SUBROUTINE dcg (n, x, b, RCI_request, ipar, dpar, tmp)
SUBROUTINE dcg_get (n, x, b, RCI_request, ipar, dpar, tmp, itercount)
SUBROUTINE dcgmrhs_init (n, x, nrhs, b, method, RCI_request, ipar, dpar, tmp)
atd., viz příklady v Reference Manual str. 2199-2201

– parametry pro skupinu podprogramů dcg:

| | |
|---|---|
| n | počet rovnic, zkopíruje se do ipar(1) |
| x | počáteční, pak okamžité řešení, pro srhs real(8) vektor velikosti n, pro mrhs matice n x nrhs |
| nrhs | počet pravých stran pro mrhs verzi |
| b | pravá strana, vektor nebo (pro mrhs) matice (nrhs,n) |
| method | jen 1 |
| RCI_request | integer hodnota |
| < 0 | chybový kód |
| 0 | hotovo |
| 1 | požadavek násobit A krát tmp(1:n,1), uložit do tmp(1:n,2) a volat dcg/dcgmrhs |
| 2 | požadavek provést zastavovací test, při neúspěchu volat dcg/dcgmrhs |
| 3 | požadavek předpodmínění vektoru tmp(1:n,3), uložit do tmp(1:n,4) a volat dcg, pro mrhs podobně (trochu jinak) |
| ipar | integer pole velikosti 128 (srhs), pro mrhs více (aktuálně však stačí 11) |
| prvky: | 1 pro n, 2/6/7 typ výpisu, 4 číslo iterace, 5 max. počet iterací, 8/9/10 typ zastavovacího testu (8: 1 pro test max. počtu iterací, 9: 1 pro test velikosti rezidua, 10: 1 pro zastavování s RCI_request=2, nuly pro vypnutí testů), 11 předpodmiňování (0 pro ne) |
| dpar | real(8) pole velikosti 128 (srhs), pro mrhs více (stačí 8) |
| prvky: | 1 relativní tolerance (default 1e-6_8), 2 absolutní tolerance (default 0._8), 3-6 L2-normy reziduí z různých fází výpočtu, 7 alpha parametr, 8 beta parametr |
| tmp | real(8) matice n x 4, pro mrhs více |

– parametry pro skupinu podprogramů dfgmres:

| | |
|---|---|
| RCI_request | integer hodnota |
| < 0 | chybový kód |
| 0 | hotovo |
| 1 | požadavek násobit A krát tmp(ipar(22)), uložit do tmp(ipar(23)) a volat dfgmres |
| 2 | požadavek provést zastavovací test, při neúspěchu volat dfgmres |
| 3 | požadavek předpodmínění vektoru tmp(ipar(22)), uložit do tmp(ipar(23)) a volat dfgmres |
| 4 | požadavek testu nulovosti normy vektoru v dpar(7) |
| ipar | integer pole velikost 128 (stačí 23) |
| | 1 až 11 viz výše, 12 pro automatický test nulovosti normy vektoru dpar(7), 15 pro verzi algoritmu s restarty |

– akcelerátory (preconditioners) ILU0 a ILUT pro FGMRES

ILU0 – neúplný LU rozklad se zachováním struktury původní matice
call dcsrilu0(n, a, ia, ja, bilu0, ipar, dpar, ierr)
ILUT – neúplný LU rozklad se zachováním některých prvků mimo strukturu (omezení na velikost a počet)
call dcsrilut(n, a, ia, ja, bilut, ibilut, jbilut, tol, maxfil, ipar, dpar, ierr)
př. Reference Manual str. 2237-2239