Preconditioned Krylov subspace method in FEEL++

Vincent Chabannes

28/04/2014

KSP

Goal : solve a linear system $Ax = F$ with an iterative process.

- Krylov subspace method : find an approximated solution in a space which the dimension increase at each step :

$$u^{n+1} = \underbrace{u^0}_{\text{initial guess}} + \underbrace{Span\left\{r^0, Ar^0, ..., A^{n-1}r^0\right\}}_{\text{Krylov subspace}} \quad \text{with } r^k = F - Ax^k$$

- GMRES, CG, BICG, FGMRES, ...
- Convergence rate depend of condion number of A
- Precondioned Krylov subspace method :
  Transform original system into a equivalent system which have a better conditioning :
  - Left preconditioning : $\underbrace{M_L^{-1}A}_{\tilde{A}} \underbrace{x}_{\tilde{x}} = \underbrace{M_L^{-1}F}_{\tilde{F}}$
  - Right preconditioning : $\underbrace{AM_R^{-1}}_{\tilde{A}} y = F$ et $x = M_R^{-1} y$
  - Left and right preconditioning : $\underbrace{M_L^{-1}AM_R^{-1}}_{\tilde{A}} y = \underbrace{M_L^{-1}F}_{\tilde{F}}$ et $x = M_R^{-1} y$

LU

- $A = LU$

Algorithm 1 Basic algo : step k of LU factorization (a_{kk} pivot)

for i > k **do**
  $l_{ik} = \frac{a_{ik}}{a_{kk}}$
end for
for i > k, j > k **do**
  $u_{ij} = a_{ij} - l_{ik} a_{kj}$
end for

- algo complexity (A is sparse matrice $n \times n$ with bandwidth b) :
  $\mathcal{O}(nb^2)$ in time, $\mathcal{O}(nb)$ in space.
- ordering routine (to reduce fill) to be used in the LU factorization $LU = PAQ$:
  ⚠ not a FEEL++ option but a PETSC option :
  `-pc_factor_mat_ordering_type=nd` [natural, nd, 1wd, rcm, qmd, rowlength]
  $\Longrightarrow$ must be investigated!
- choose a factorisation package [mumps(//),petsc,pastix(//),umfpack, superlu,...]:
  `--pc-factor-mat-solver-package-type=mumps`
- Use a direct solver : **`--ksp-type=preonly`**

ILU

- ILU(k) : approximate factorization

$$\forall_{i,j>k} : if (i,j) \in S \quad a_{ij} \leftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj} \tag{1}$$

- ILUT : only fill in zero locations if $a_{ik} a_{kk}^{-1} a_{kj}$ large enough
- **`--pc-factor-levels=3`**
- **`--pc-factor-fill=6`** :
  Indicate the amount of fill you expect in the factored matrix (fill = number nonzeros in factor/number nonzeros in original matrix).
- factorisation package :
  `--pc-factor-mat-solver-package-type=petsc`
  - petsc : works only in sequential
  - euclid (from HYPRE) : works in sequential (ILU(k) and ILUT) and parallel (ILU(k))
    ⚠the setup phase in // is very long and very not scalable! why???
  - pilut (from HYPRE) : ILUT approximation (works in seq and //) but convergence is very bad and package not maintained

Relaxation methods

Split into lower, diagonal, upper parts: $A = L + D + U$
$x^{k+1} = x^k + P^{-1}(F - Ax^k)$

- Jacobi : **`--pc-type=jacobi`**
  $P^{-1} = D^{-1}$
- Successive over-relaxation (SOR) : **`--pc-type=sor`**

$$P = \frac{1}{\omega} D + U \quad \text{forward} \quad , \quad P = \quad \text{backward}$$

$$x^{k+1} = -(D + \omega L)^{-1} (\omega U + (1 - \omega) D) x^k + \omega (D + \omega L)^{-1} F$$

`--pc-sor-type=local_symmetric` [symmetric, forward, backward, local_symmetric, local_forward, local_backward]
`--pc-sor-omega=1` : $\omega \in]0, 2[$. if $\omega = 1 \Longrightarrow$ Gauss-Seidel
⚠ Not a true parallel SOR, in parallel this implementation corresponds to block Jacobi with SOR on each block.
`--pc-sor-lits=1` : the number of smoothing sweeps on a process before doing a ghost point update from the other processes
`--pc-sor-its=1`
$\Longrightarrow$ The total number of SOR sweeps is given lits*its.

Domain decomposition

Additive Schwarz with overlap :

$$P = \sum_{i=1}^{p} \hat{R}_i^T \left(R_i A_i R_i^T \right)^{-1} \tilde{R}_i \quad \text{with } R_i, \tilde{R}_i, \hat{R}_i \text{ restriction operators}$$

Example 1 :

```
--pc-type=gasm
--pc-gasm-overlap=2        [size of extended local subdomains ]
--pc-gasm-type=restrict    [basic, restrict, interpolate, none ]
--sub-pc-type=lu           [preconditioner used for A_i^{-1} ]
--sub-pc-factor-mat-solver-package-type=mumps
```

Example 2 :

```
--pc-type=gasm
--pc-gasm-overlap=2        [size of extended local subdomains ]
--pc-gasm-type=restrict    [basic, restrict, interpolate, none ]
--sub-pc-type=ilu          [preconditioner used for A_i^{-1} ]
--sub-pc-factor-levels=3
--sub-pc-factor-fill=6
```

Algebraic multigrid

Algorithm 2 Basic multigrid with two levels

solve on fine grid : $A_1 u_1 = F_1$
compute residual : $r_1 = F_1 - A_1 u_1$
restrict r_1 to the coarse grid : $r_0 = R\ r_1$
solve on coarse grid : $A_0 e_0 = r_0$
interpolate : $e_1 = I\ e_0$
solve on fine grid by starting with $u_1 = u_1 + e_1$

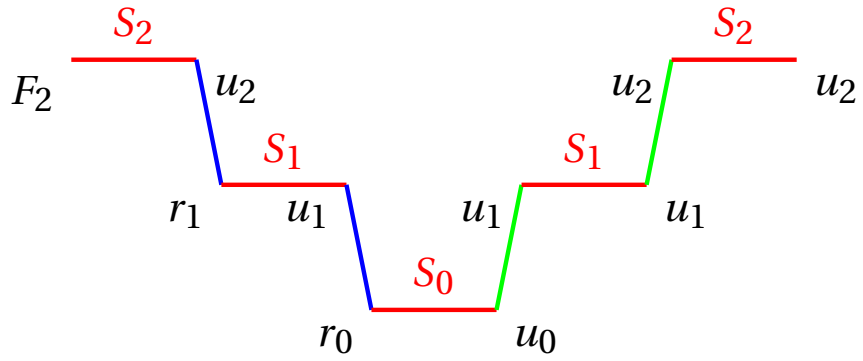

Figure: Multiplicative multigrid

- MultiGrid options :
  `--pc-type=ml` [ml,gamg,boomeramg]
  `--pc-mg-levels=10`
  `--pc-mg-type=multiplicative`
  [multiplicative, additive, full, kaskade]
- Coarse options :
  `--mg-coarse.pc-type=redundant`
- all fine levels options (not including coarse) :
  `--mg-levels.pc-type=sor`
- last fine levels options) :
  `--mg-fine-level.pc-type=sor`
- specific levels options (up to 5) :
  `--mg-levels3.pc-type=sor`

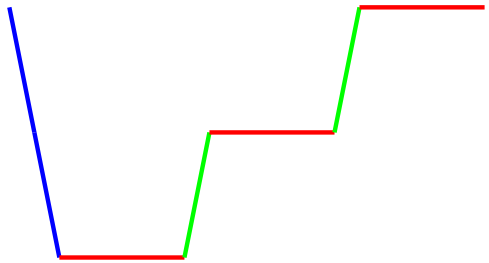

Figure: Kascade multigrid

⚠boomeramg : only **`--pc-mg-levels`** is taken into account. A lot of options can be given with PETSc options (see doc or code). Seams to be very efficient but strange behavior with fine meshes (convergence saturation)

Fieldsplit

FieldSplit : solving bloc matrix $(N \times N)$ ex : $\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix}$

- IndexSplit : computed automatically with composite space,block matrix and block graph
  $\Longrightarrow$ user can redifine split definition (union of split) :
  `--fieldsplit-fields=0->(1),1->(2,0)`
- **`--fieldsplit-type=additive`** [additive, multiplicative, symmetric-multiplicative, schur]

block Jacobi (additive)	:	$\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & A_{11}^{-1} \end{pmatrix}$
block Gauss-Seidel (multiplicative)	:	$\begin{pmatrix} I & 0 \\ 0 & A_{11}^{-1} \end{pmatrix}\begin{pmatrix} I & 0 \\ -A_{10} & I \end{pmatrix}\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & I \end{pmatrix}$
sym block Gauss-Seidel (symmetric-multiplicative)	:	$\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} I & -A_{01} \\ 0 & I \end{pmatrix}\begin{pmatrix} A_{00} & 0 \\ 0 & A_{11}^{-1} \end{pmatrix}\begin{pmatrix} I & 0 \\ -A_{10} & I \end{pmatrix}\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & I \end{pmatrix}$

```
--fieldsplit-0.pc-type=gasm
--fieldsplit-0.sub-pc-type=lu
--fieldsplit-1.pc-type=boomeramg
--fieldsplit-1.ksp-type=gmres
```

FieldSplit in FieldSplit

```
--fieldsplit-fields=0->(0,2),1->(1)
--fieldsplit-0.pc-type=fieldsplit
--fieldsplit-0.fieldsplit-fields=0->(0),1->(2)
```

Schur complement (⚠work only with 2 fields!)

$$A = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} = LDU = \begin{pmatrix} I & 0 \\ A_{10}A_{00}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{00} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A_{00}^{-1}A_{01} \\ 0 & I \end{pmatrix}$$

with Schur complement of A_{00} : $S = A_{11} - A_{10}A_{00}^{-1}A_{01}$

$$A^{-1} = \underbrace{\begin{pmatrix} I & -A_{00}^{-1}A_{01} \\ 0 & I \end{pmatrix}}_{U^{-1}} \underbrace{\begin{pmatrix} A_{00}^{-1} & 0 \\ -S^{-1}A_{10}A_{00}^{-1} & S^{-1} \end{pmatrix}}_{(LD)^{-1}} = \underbrace{\begin{pmatrix} A_{00}^{-1} & -A_{00}^{-1}A_{01}S^{-1} \\ 0 & S^{-1} \end{pmatrix}}_{(DU)^{-1}} \underbrace{\begin{pmatrix} I & 0 \\ -A_{10}A_{00}^{-1} & 0 \end{pmatrix}}_{L^{-1}}$$

$$= \underbrace{\begin{pmatrix} I & -A_{00}^{-1}A_{01} \\ 0 & I \end{pmatrix}}_{U^{-1}} \underbrace{\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix}}_{D^{-1}} \underbrace{\begin{pmatrix} I & 0 \\ -A_{10}A_{00}^{-1} & I \end{pmatrix}}_{L^{-1}}$$

`--fieldsplit-type=schur`
`--fieldsplit-schur-fact-type=full` [diag, lower, upper, full]

- diag : $\tilde{D}^{-1}$: positive definite, suitable for MINRES
- lower : $(LD)^{-1}$ (suitable for left preconditioning)
- upper : $(DU)^{-1}$ (suitable for right preconditioning)
- full : $U^{-1}(LD)^{-1} = (DU)^{-1}L^{-1}$ (an exact solve if applied exactly, needs one extra solve with A)

Schur complement

By default, all A_{00}^{-1} approximation use same KSP

- Inner solver in the Schur complement of $S = A_{11} - A_{10} A_{00}^{-1} A_{01}$
  --fieldsplit-schur-inner-solver.use-outer-solver=false
  --fieldsplit-schur-inner-solver.pc-type=jacobi
  --fieldsplit-schur-inner-solver.ksp-type=preonly
- Upper solver in full Schur factorisation :

$$A^{-1} = \underbrace{\begin{pmatrix} I & -A_{00}^{-1} A_{01} \\ 0 & I \end{pmatrix}}_{U^{-1}} \underbrace{\begin{pmatrix} A_{00}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix}}_{D^{-1}} \underbrace{\begin{pmatrix} I & 0 \\ -A_{10} A_{00}^{-1} & I \end{pmatrix}}_{L^{-1}}$$

--fieldsplit-schur-upper-solver.use-outer-solver=false
--fieldsplit-schur-upper-solver.pc-type=jacobi
--fieldsplit-schur-upper-solver.ksp-type=preonly

Schur complement

- SIMPLE : Semi-Implicit Method for Pressure-Linked Equations (Patankar and Spalding 1972)
  Variants : SIMPLER (Patankar - 1980), SIMPLEC (Van Doormaal and Raithby - 1984)

$$P_{SIMPLE} = \begin{pmatrix} A_{00} & 0 \\ A_{10} & \tilde{S} \end{pmatrix} \begin{pmatrix} I & D_{00}^{-1} A_{01} \\ 0 & I \end{pmatrix} \qquad \text{with } \tilde{S} = A_{10} D_{00}^{-1} A_{01}$$

```
--fieldsplit-schur-precondition=user
--fieldsplit-schur-upper-solver.use-outer-solver=false
--fieldsplit-1.pc-type=lu
--fieldsplit-1.ksp-type=gmres
--fieldsplit-1.ksp-maxit=10
--fieldsplit-1.rtol=1e-3
```

- Approximate S^{-1} with preconditioner PCLSC : $S^{-1} \approx (A_{10} A_{01})^{-1} A_{10} A_{00} A_{01} (A_{10} A_{01})^{-1}$

```
--fieldsplit-schur-precondition=self
--fieldsplit-1.pc-type=lsc
--fieldsplit-1.ksp-type=gmres
--fieldsplit-1.ksp-maxit=10
--fieldsplit-1.ksp-rtol=1e-3
--fieldsplit-1.lsc.ksp-type=gmres
--fieldsplit-1.lsc.pc-type=boomeramg
--fieldsplit-1.lsc.ksp-rtol=1e-4
```

- Others preconditioner (work in progress) : Yosida, PCD (Pressure Convection Diffusion), aSIMPLE, aPCD,...

Conclusion

work done

- refactor the preconditioner interface of PETSc (in order to mix all preconditioner)
- understand how PETSc build and apply these prec
- improve management of split

TODO

- Improve the robustness of default options
- continue refactoring and build precondioner option on the fly (?)
- LU/ILU : study the reordoring
- MultiGrid : add DM structure in ordre to use also a geometric multigrid with gamg
- FieldSplit/Schur : Devlopement of new preconditioner