Domain

- uint nx
  nt
- double X
  T
- uint n/m
  d

getters

Decomposition

- Domain domain
- uint nsubx
  nsubt
- double theta
- vector<uint> subsizes [n,m]
- Matrix Xi overlap_forw
  overlap_back
- vector<uint> start_elem

Decomposition (dom, subx, subt, theta,
(dom, subx, subt)
(dom, subx, subt, n, m)

getters
createDec(double n, double m)
tuple<..uint..> get_info_subk(k)

SolverTraits

- uint max_it
- double tol
  tol_pipe_sx
- uint it_wait

getters

DomainDecSolverBase

- Domain domain
- decomposition DataDD
- vec<SpMat> R
  $\tilde{R}$
  locA
- int locA_created

VectorXd solve(A, b, SolveTraits) = 0
pair<SpMat -> createRK(k)
createRmatrices()
createAlocal(SpMat A)
pair< > getRk(k)
SpMat getAk(k)
Costructor (dom, dec, A) → chiama createR() e createA(A)

Ras

stesso costruttore e metodi
...
VectorXd precondAction(SpMat x)

RasPipelined

DomainDecSolverFactory

- Domain domain
- decomposition DataDD

costructor(dom, dec)

VectorXd ()(method, A, b, SolveTraits)
unique ptr createSolver(name, Args)
↳ puntatore ad oggetto RAS che viene inizializzato

chiama create solver che inizializza RAS con A, b, SolveTraits. Poi usa solve di RAS

DA RAGIONARE:

- SolverBase riceve nel costruttore A in modo che quando creo l'obj vengono inizializzate tutte le local matrices (R, A). Perché poi nel costruttore chiamo createAlocal(). In alternativa devo chiamarlo dopo separatamente e in questo caso è importante controllare flag localA_created

- se voglio accedere a localA, localR non lo faccio dal puntatore della factory ma creo oggetto Ras e poi ci accedo da lì. Alternativa è creare classe localmatrix e passarla come membro a SolverBase.

- SolverFactory non ha A nel costruttore, viene passato come parametro quando chiamo ( ) in modo che A venga usata nel costruttore di un SolverBase

- decomposizione non cambia. quindi neanche localA. Se chiamo solver prima con un metodo e poi con un altro tutto il processo di creazione delle localA viene rifatto. 2) come è ora
Unico modo per evitare è quello di passare A a decomposition (1) e far creare Alocal e Rlocal da Decomposition. Però vuol dire che se cambio A sulla stessa decomposizione devo rifare tutti i campi di DataDD (overlap ---)
  - -> se cambio decomposizione tutto deve essere rifatto, nessuna differenza
  - -> cambio A (stessa dec): 1) rifare DataDD 2) niente
  - -> cambio metodo, stessa A, stessa dec: 1) niente 2) rifare Alocal, Rlocal

  => forse meglio 1). Quindi cambiare e passare A a Decomposition che dovrà crearsi subito Alocal Rlocal

- capire in genere feurefon. collegare elementi a dof. Per noi con griglia cartesiana è facile e ci muoviamo sapendo come sono numerati elementi e con le grandezze dei subs.