Fractional decomposition of matrices and easy parallel computing in FreeFem++

F. Hecht , S-M. Kaber

LJLL, Sorbonne Université, Paris   projet Alpines, Inria de Paris

with F. Nataf, P. Jolivet, P-H. Tournier, A. Fourmont for FreeFem++

16-20 september, 2019

Parallel Solution Methods for Systems Arising from PDEs

CIRM , Marseille

http://www.freefem.org

mailto:frederic.hecht@upmc.fr

Outline

1 Introduction, Heat equation

2 Parallel in time version

3 About FreeFEM

4 Future/Conclusion

Outline

1 Introduction, Heat equation

2 Parallel in time version

3 About FreeFEM

4 Future/Conclusion

Outline

1 Introduction, Heat equation
- Weak form
- Heat equation, Time depend schema
- Optimize version with matrix

Laplace equation, weak form

Let a domain Ω with a partition of $\partial\Omega$ in Γ_2, Γ_e.
Find u a solution in such that:

$$-\Delta u = 1 \text{ in } \Omega, \quad u = 2 \text{ on } \Gamma_2, \quad \frac{\partial u}{\partial \vec{n}} = 0 \text{ on } \Gamma_e \tag{1}$$

Denote $V_g = \{v \in H^1(\Omega) / v_{|\Gamma_2} = g\}$.
The Basic variational formulation with is: find $u \in V_2(\Omega)$, such that

$$\int_\Omega \nabla u.\nabla v = \int_\Omega 1v + \int_\Gamma \frac{\partial u}{\partial n} v, \quad \forall v \in V_0(\Omega) \tag{2}$$

The finite element method is just: replace V_g with a finite element space, and the `FreeFem++` code:

Poisson equation in a fish with FreeFem++

The finite element method is just: replace V_g with a finite element space, and the FreeFem++ code:

```
mesh3 Th("fish-3d.msh");// read a mesh 3d
fespace Vh(Th,P1);   // define the P1 EF space

 Vh u,v;//set test and unknown function in Vh.
macro Grad(u) [dx(u),dy(u),dz(u)] //EOM Grad def
solve laplace(u,v,solver=CG) =
   int3d(Th)(  Grad(u)'*Grad(v)   )
   - int3d(Th) ( 1*v)
   + on(2,u=2);   // BC on Γ2
plot(u,fill=1,wait=1,value=0,wait=1);
```

Run:fish.edp Run:fish3d.edp

Outline

1 Introduction, Heat equation
- Weak form
- Heat equation, Time depend schema
- Optimize version with matrix

Method for Time depend Problem / formulation

The example solve the Thermal Conduction problem of section 5.5 of the documentation. We must solve the temperature equation in Ω on time (0,T).

$$\partial_t u - \nabla \cdot (\kappa \nabla u) = 0 \text{ in } \Omega \times (0,T),$$
$$u(x,y,0) = u_0 + x u_1$$
$$u = 30 \text{ on } \Gamma_{24} \times (0,T), \quad \kappa \frac{\partial u}{\partial n} + \alpha(u - u_e) = 0 \text{ on } \Gamma_{13} \times (0,T).$$

The variational formulation is in $L^2(0,T;H^1(\Omega))$; we shall seek u^n satisfying

$$\forall w \in V_0; \qquad \int_\Omega \frac{u^n - u^{n-1}}{\delta t} w + \kappa \nabla u^n \nabla w) + \int_{\Gamma_{13}} \alpha(u^n - u_{ue}) w = 0$$

where $V_0 = \{w \in H^1(\Omega) / w_{|\Gamma_{24}} = 0\}$.

Run:Heat-v0.edp // a naïve code

Outline

1. Introduction, Heat equation
   - Weak form
   - Heat equation, Time depend schema
   - Optimize version with matrix

a method with sparse matrix

So the to code the method with the matrices $A = (A_{ij})$, $M = (M_{ij})$, and the vectors $u^n, b^n, b', b", b_{cl}$ (notation if w is a vector then w_i is a component of the vector).

$$u^n = A^{-1}b^n, \qquad b' = b_0 + Mu^{n-1}, \quad b" = \frac{1}{\varepsilon}\, b_{cl}, \quad b_i^n = \begin{cases} b"_i & \text{if } \ i \in \Gamma \\ b'_i & \text{else} \end{cases}$$

Where with $\frac{1}{\varepsilon} = \texttt{tgv} = 10^{30}$:

$$\begin{aligned}
A_{ij} &= \begin{cases} \frac{1}{\varepsilon} & \text{if } i \in \Gamma_{24}, \text{and} \\ \displaystyle\int_\Omega w_j w_i / dt + k(\nabla w_j . \nabla w_i) + \int_{\Gamma_{13}} \alpha w_j w_i & \text{else} \end{cases} \\
M_{ij} &= \begin{cases} \frac{1}{\varepsilon} & \text{if } i \in \Gamma_{24}, \text{and} \quad j = i \\ \displaystyle\int_\Omega w_j w_i / dt & \text{else} \end{cases} \\
b_{0,i} &= \int_{\Gamma_{13}} \alpha u_{ue} w_i \\
b_{cl} &= u^0 \quad \text{the initial data}
\end{aligned}$$

The variational form

```
 ...
Vh u0=fu0,u=u0;
```

Create three variational formulations, and build the matrices A,M.

```
varf vthermic (u,v)= int2d(Th)(u*v/dt
           + k*( dx(u)*dx(v) + dy(u)*dy(v)))
  + int1d(Th,1,3)(alpha*u*v)  + on(2,4,u=1);
varf vthermic0(u,v) =  int1d(Th,1,3)(alpha*ue*v);
varf vMass (u,v)= int2d(Th)( u*v/dt)  +
on(2,4,u=1);

real tgv = 1e30;
matrix A= vthermic(Vh,Vh,tgv=tgv,solver=CG);
matrix M= vMass(Vh,Vh);
```

The Time depend Problem with matrix / edp

Now, to build the right hand size we need 4 vectors.

```
real[int]  b0  = vthermic0(0,Vh);//   constant part
of RHS
real[int]  bcn = vthermic(0,Vh);          //   tgv on
Dirichlet part
 //    we have for the node i :  $i \in \Gamma_{24}$  ⇔  $bcn[i] \neq 0$
real[int]  bcl=tgv*u0[];    //    the Dirichlet B.C.
part
```

a fast algorithm: sequential

```
for(real t=0;t<T;t+=dt){
  real[int] b = b0 ;                   //    for the RHS
  b += M*u[];  //   add the the time dependent part
  b = bcn?bcl:b;  //    ∀i:  b[i]=bcn[i]?bcl[i]:b[i]
  u[] = A^-1*b;              //    Solve linear problem
  plot(u);
}
```

Outline

1. Introduction, Heat equation
2. Parallel in time version
3. About FreeFEM
4. Future/Conclusion

Outline

2 Parallel in time version
- Backward Euler method
- A bit of maths:Partial fraction decomposition
- Numerical test

Backward Euler method

Starting the time discretization of this equation $\boldsymbol{u}_t = \boldsymbol{\mathcal{L}u}$ by implicit Euler scheme with time step h_1 reads

$$\frac{U^1 - U^0}{h_1} = \mathcal{L}U^1.$$

With a Finite Difference scheme

$$(I - h_1 X)u^1 = u^0.$$

with X a Finite Difference approximation of the spatial operator $\mathcal{L}$.

Iterating p steps of the implicit scheme, we obtain an equation for u^p

$$(I - h_p X) \cdots (I - h_2 X)(I - h_1 X)u^p = u^0.$$

Because the identity I commute with matrix X, we can see this expression as a polynom expression of matrix.

Outline

2. Parallel in time version
    - Backward Euler method
    - A bit of maths:Partial fraction decomposition
    - Numerical test

A bit of maths: Partial fraction decomposition

The fractional decompositions of matrix give

$$((I - h_p X) \cdots (I - h_2 X)(I - h_1 X))^{-1} = \sum_{k=1}^{p} \alpha_i (I - h_i X))^{-1}.$$

where

$$\alpha_i = \prod_{k \neq i} (1 - h_k / h_i)^{-1}. \tag{3}$$

Consequently, the vector u^p could be split into $u^p = \sum_{i=1}^{p} \alpha_i x_i$. where the x_i will be computed with

$$(I - h_i X))x_i = u^0$$

in parallel.

limitation in number of processor

The limitation is the value of α_i coefficient if $p > 6$

	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 6$
$\sum_{i=1}^{p} \lvert\alpha_i\rvert$	10.00	201.00	530.00	7068.33	10895.83

Table: Evolution of the amplification term in (3) as a function of the number of processors p. The h_i are choosen symmetrically : $c = 1/10$ and in the case $p = 2$, $h_i = (1 \pm c)h$, in the case $p = 3$, $h_i \in \{h, (1 \pm c)h\}$, in the case $p = 4$, $h_i \in \{(1 \pm c)h, (1 \pm 2c)h\}$, ...

With Finite element

The matrix discretization, we get

$$(M - h_1 Y)u^1 = Mu^0.$$

with Y a Finite element approximation of the spatial operator $\mathcal{L}$, and M the mass matrix associated the the L^2 scalar product, and we do same with matrix $X = M^{-1}Y$.
Of coarse we never build the matrix, but we just solve:

$$(M - h_i Y))x_i = Mu^0$$

With RHS

the time discretization of this equation $\boldsymbol{u}_t = \mathcal{L}\boldsymbol{u} + f(t)$ by implicit Euler scheme with time step h_1 reads

$$\frac{U^1 - U^0}{h_1} = \mathcal{L}U^1 + f_0$$

where f_1 is a know approximation of f at a time t_0, t_1. Iterating p steps of the implicit scheme, we obtain an equation for u^p
let call $B_q = ((I - h_q X) \cdots (I - h_2 X)(I - h_1 X)), B_0 = I$ so
$B_q = (I - h_q X)B_{q-1}$ and

$$((I - h_q X))u^{q+1} = u^q + f_q, \qquad B_q u^{q+1} = B_{q-1}u^q + h_q B_{q-1} f_{q-1}$$

after a sum we have:

$$B_p^{-1} u^p = u^1 + \sum_{q=1}^{p} h_q B_{q-1} f_{q-1}$$

The problem is the coast to compute the right hand side $F_0 = \sum_{q=1}^{p} h_q B_{q-1} f_{q-1}$, Compute in parallel but it is this a expensive sequential formula with no dependence in time. So we can parallelize.

With RHS

The first problem is to compute in finite element case in parallel :

$$F_{ip} = \sum_{q=1}^{p} h_q B_{q-1} f(ip+q-1), \qquad \text{with} \quad B_q = M^{-1}(M-h_q Y)B_{q-1}, B_0 = I$$

where ip a current temps step after i iteration of p parallel sub iteration.

- set u_0, choose n_k , ...
- sequential loop $k = 0$ to $n_k - 1$ *// extern loop of step p^2*

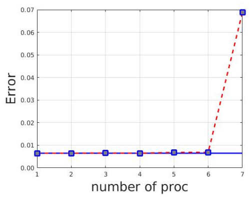
1. parallel loop $j \in \{0, \dots, p-1\}$: $i = j(1+kp)$,Compute F_{ip}
2. sequential Loop $j = 0$ to $p-1$ *// loop of step p*
   - to compute $u_{(j+1)p+kp^2}$ using Partial fraction decomposition :do
   - parallel loop: $i \in \{0, \dots, p-1\}$ *// loop of step 1*
     - compute v_i solution of $(M + h_i X)v_i = M(u_{jp+kp^2} + F_{j(1+kp)p})$
     - $u_{(j+1)p+kp^2} = \sum_k \alpha_k v_k$ *// a mpi Reduce ..*

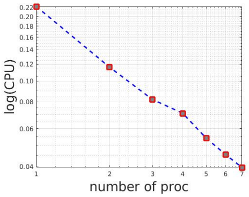
.

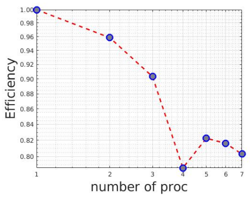
Outline

2 Parallel in time version
- Backward Euler method
- A bit of maths:Partial fraction decomposition
- Numerical test

Numerical test







The $2D$ nonhomogeneous heat equation. Run:Para-rhs.edp

see
`https://hal.archives-ouvertes.fr/hal-01878765`

Outline

1. Introduction, Heat equation
2. Parallel in time version
3. About FreeFEM
4. Future/Conclusion

Outline

3 About FreeFEM

- Introduction, FreeFEM
- In progress or not
- Linear elasticty equation
- Bose Einstein Condensate, result analyse
- Anaylse BEC Solution
- ffddm
- Surface finite element

Introduction FreeFEM

FreeFem++ / FreeFEM is a software to solve numerically partial differential equations (PDE) in $\mathbb{R}^2)$, $\mathbb{R}^3)$ or on surface with finite elements methods. We used a user language to set and control the problem. The FreeFem++ language allows for a quick specification of linear PDE's, with the variational formulation of a linear steady state problem and the user can write they own script to solve no linear problem and time depend problem. You can solve coupled problem or problem with moving domain or eigenvalue problem, do mesh adaptation , compute error indicator, etc ...

By the way, FreeFem++ is build to play with abstract linear, bilinear form on Finite Element Space and interpolation operator.

FreeFem++ is a freeware and this run on Mac, Unix and Window architecture, in parallel with MPI.
To try of cell phone
`https://www.ljll.math.upmc.fr/lehyaric/ffjs/`

The 11th FreeFem++ days: 18,19 of Dec, 2019, Sorbonne Université, Jussieu, Paris, France

Outline

3 About FreeFEM

- Introduction, FreeFEM
- In progress or not
- Linear elasticty equation
- Bose Einstein Condensate, result analyse
- Anaylse BEC Solution
- ffddm
- Surface finite element

In progress (future proche ou pas)

- to get Version 4.4 , just do do:

```
git clone -b develop  https://github.com/FreeFem/FreeFem-sources f
```

- Mesh intersection of get conservative formulation in 2d (too hard => no)
- Element on curve and Surface (Ok)
- BEM method (near futur)
- DG in 3d
- cmake (????)
- rewrite of sparse matrix kernel (Done) ($\Longrightarrow$ new GMRES, new CG, ...)
- rewrite of the Finite element kernel of isoparametric FE and to mixte surface FE and 3d FE (in future) in a problem.(no template).
- debugger (client-server graphics services) (Good idea but technical)
- more general problem (possible with new matrix)

Outline

3 About FreeFEM

- Introduction, FreeFEM
- In progress or not
- Linear elasticty equation
- Bose Einstein Condensate, result analyse
- Anaylse BEC Solution
- ffddm
- Surface finite element

Linear Lame equation, weak form

Let a domain $\Omega \subset \mathbb{R}^d$ with a partition of $\partial\Omega$ in Γ_d, Γ_n.
Find the displacement $\boldsymbol{u}$ field such that:

$$-\nabla.\sigma(\boldsymbol{u}) = \boldsymbol{f} \text{ in } \Omega, \quad \mathbf{u} = \mathbf{0} \text{ on } \mathbf{\Gamma_d}, \quad \sigma(\mathbf{u}).\mathbf{n} = \mathbf{0} \text{ on } \mathbf{\Gamma_n} \tag{4}$$

Where $\varepsilon(\boldsymbol{u}) = \frac{1}{2}(\nabla\boldsymbol{u} + {}^t\nabla\boldsymbol{u})$ and $\sigma(\boldsymbol{u}) = \boldsymbol{A}\varepsilon(\boldsymbol{u})$ with $\boldsymbol{A}$ the linear positif operator on symmetric $d \times d$ matrix corresponding to the material propriety. Denote $V_{\boldsymbol{g}} = \{\boldsymbol{v} \in H^1(\Omega)^d / \boldsymbol{v}_{|\Gamma_d} = \boldsymbol{g}\}$.
The Basic displacement variational formulation is: find $\boldsymbol{u} \in V_0(\Omega)$, such that:

$$\int_\Omega \varepsilon(\boldsymbol{v}) : \boldsymbol{A}\varepsilon(\boldsymbol{u}) = \int_\Omega \boldsymbol{v}.\boldsymbol{f} + \int_\Gamma ((\boldsymbol{A}\varepsilon(\boldsymbol{u})).n).v, \quad \forall \boldsymbol{v} \in V_0(\Omega) \tag{5}$$

Run:beam-3d.edp Run:beam-EV-3d.edp Run:free-cyl-3d.edp
Run:beam-3d-Adapt.edp

Outline

3. About FreeFEM
   - Introduction, FreeFEM
   - In progress or not
   - Linear elasticty equation
   - Bose Einstein Condensate, result analyse
   - Anaylse BEC Solution
   - ffddm
   - Surface finite element

Bose Einstein Condensate

With. I. Danaila (Univ. Rouen), G. Vergez (Phd Becasim), P-E Emeriau (Stage ENS/2016)

Just a direct use of `Ipopt` interface (2 day of works to start script see, + n month)
The problem is find a complex field u on domain $\mathcal{D}$ such that:

$$u = \underset{||u||=1}{\operatorname{argmin}} \int_{\mathcal{D}} \frac{1}{2}|\nabla u|^2 + V_{trap}|u|^2 + \frac{g}{2}|u|^4 - \Omega i\overline{u}(\left(\begin{smallmatrix}-y\\x\end{smallmatrix}\right).\nabla)u$$

to code that in FreeFem++
use

- Ipopt interface (`https://projects.coin-or.org/Ipopt`)
- Adaptation de maillage

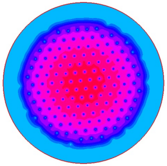
The idea to mixte Ipopt and adapt mesh is play this stop criterion, and finally use freefem++ to analyse the result.

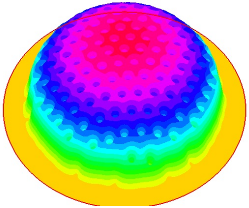
Run:BEC.edp

Outline

3. About FreeFEM
   - Introduction, FreeFEM
   - In progress or not
   - Linear elasticty equation
   - Bose Einstein Condensate, result analyse
   - Anaylse BEC Solution
   - ffddm
   - Surface finite element

Anaylse BEC, Best Fit (With. P-E Emeriau)





The function `findalllocalmin` find all the le local min and use a greedy algorithm to to the local attraction zone, by adding the triangle through the minimal vertices.

Run:findalllocalmin.edp Run:findalllocalminbec.edp

Just use `ipopt` to find the arg min of $J(\alpha) = \int_\Omega (u - \phi_\alpha)^2)$ where $alpha$ is the set of parameters.

On the domain with no vortex, all just do the L2 projection on axisymmetric space with a laplace regularisation

1. Run:fit-axi.edp
2. Run:analyssolbec.edp

$$\frac{\partial u}{\partial t} + (u\cdot\nabla)u - \nu\Delta u + \nabla p = 0, \quad \nabla\cdot u = 0$$

with the same boundary conditions and with initial conditions $u = 0$.
This is implemented by using the interpolation operator for the term $\frac{\partial u}{\partial t} + (u\cdot\nabla)u$, giving a discretization in time

$$\begin{array}{ll} \frac{1}{\tau}(u^{n+1} - u^n \circ X^n) - \nu\Delta u^{n+1} + \nabla p^{n+1} & = 0, \\ \nabla\cdot u^{n+1} & = 0 \end{array} \quad (6)$$

The term $X^n(x) \approx x - \tau u^n(x)$ will be computed by the interpolation operator or convect operator.
Or better we use an order 2 schema, BDF1

$$\frac{\partial u}{\partial t} + (u\cdot\nabla)u \approx \frac{(3u^{n+1} - 4u^n \circ X_1^n + u^{n-1}\circ X_2^n)}{2\tau}$$

with $u^* = 2u^n - u^{n-1}$, and $X_1^n(x) \approx x - \tau u^*(x), X_2^n(x) \approx x - 2\tau u^*(x)$

Incompressible Navier-Stokes UZWAZA

The generalise Stokes problem is find u, p solution of

$$Au + Bp = f, \quad {}^tBu = 0$$

with $A \equiv (\alpha Id + \nu\Delta)$ and $B \equiv \nabla$. remark,if A est symmetric positive the you can use a conjugate gradient to solve the the following problem

$${}^tBA^{-1}B\,p = {}^tBA^{-1}f$$

Now in a periodic domain, all differential operators commute and the Uzawa algorithm comes to solving the linear operator $-\nabla.((\alpha Id - \nu\Delta)^{-1}\nabla$, where Id is the identity operator. So the preconditioner suggested is $-\alpha\Delta^{-1} + \nu Id$.
Run:NSUzawaCahouetChabart.edp
Run:NSUzawaCahouetChabart-3d-aorte.edp

Outline

3 About FreeFEM

- Introduction, FreeFEM
- In progress or not
- Linear elasticty equation
- Bose Einstein Condensate, result analyse
- Anaylse BEC Solution
- ffddm
- Surface finite element

Domain decomposition: ffddm, hpddm

With Pierre-Tournier , Pierre Jolivet, Frédéric Nataf.

- Documentation FFDDM
- tutorial FFDDM
- Run:DDM-Schwarz-Lap-2dd.edp
- Run:diffusion-3d-minimal-ddm.edp

Outline

3 About FreeFEM

- Introduction, FreeFEM
- In progress or not
- Linear elasticty equation
- Bose Einstein Condensate, result analyse
- Anaylse BEC Solution
- ffddm
- Surface finite element

Surface finite element on Tore

With Axel.Fourmont.

```
load "msh3"
real R = 3, r=1, h = 0.2; //
int nx = R*2*pi/h, ny = r*2*pi/h;
mesh Th2 = square(nx,ny,[x*pi*2,y*pi*2],region=1);
func torex= (R+r*cos(y))*cos(x);
func torey= (R+r*cos(y))*sin(x);
func torez= r*sin(y);
meshS Th=movemesh2S(Th2,transfo=[torex,torey,torez
    ],orientation=1);
plot(Th);
fespace VhS(Th,P1);
VhS u,v;
macro grad3(u) [dx(u),dy(u),dz(u)]  // EOM
solve Lap(u,v) = int2d(Th)( u*v+grad3(u)'*grad3(v))
     -int2d(Th)(x*v);
plot(u,wait=1,nbiso=20,fill=1);
```

Run:Lap-on-Tore.edp

Outline

1 Introduction, Heat equation

2 Parallel in time version

3 About FreeFEM

4 Future/Conclusion

Parallelization in time

This is almost a 0 cost method get a real speed up on few processor.

future:

- Other schema in time: BDF-2 (not possible due to problem of commutation of iteration)
- Approximation of the exponential of matrix (in progress).
- Other equation : Wave equation (not trivial, due to problem of commutation) but possible thought approximation of sin and cos.

Domain decomposition: FFDDM/HPDDM

To accelerate again do: Domain decomposition: FFDDM/HPDDM
With Pierre-Tournier , P. Jolivet, Frédéric Nataf.

- Documentation FFDDM
- tutorial FFDDM
- Run:DDM-Schwarz-Lap-2dd.edp
- Run:diffusion-3d-minimal-ddm.edp

Some hpddm examples from the distribution

Diffusion	Run:diffusion-2d.edp , Run:diffusion-3d.edp
Elastically	Run:elasticity-2d.edp , Run:elasticity-3d.edp

Conclusion/Future

FreeFEM(++) v4.4 is

- very good tool to solve non standard PDE in 2D/3D and of surface
- to try new domain decomposition domain algorithm

The future we try to do:

- 3d anisotrope mesh adaptation (with new version mmg3d software)
- link with MFront: a code generation tool dedicated to material knowledge (CEA/EDF France)
- automate the parallel tool (see ffddm, in progress P-H Tournier, F. Nataf, P. Jolivet)
- Add integral method (in progress, A. Fourmont, X. Claeys) , and finite volume method (works of G. Sadaka).
- Build more graphic with VTK, paraview , web , ... (in progress)

Thank for you attention.