Time parallelization
a new (?) and simple (!) method

Sidi-Mahmoud Kaber
http://www.ljll.math.upmc.fr/kaber
kaber@ljll.math.upmc.fr
Laboratoire J.-L. Lions

SORBONNE UNIVERSITÉS

FreeFem++'s days
Paris, december 2017

Parallel computing

Space direction

- Domain decomposition methods (DDM):
  efficient approach to parallelization of the space direction.
  - divide the whole domain in subdomains,
  - solve iteratively and independantly on each subdomain a smaller problem.
  - Corrections steps to propagate information from one subdomain to the others.
- Parallelization is not limited to space direction
  Time direction is also a candidate for parallelization

Parallel computing

Time Parallelization

- Multiple shooting Methods: Chartier, Philippe (1993), . . .
- Domain Decomposition type: Saha, Tradel, Tremaine (1996), Gander (1996), Gander, Halpern, Nataf (1999), . . .
- Parareal method: Lions, Maday, Turinici (2001), . . .
- Direct solvers: Axelson, Verwer (1985), Worley (1991), Sleen, Sloan, Thomée (1999), Maday, Ronquist (2008), Güttel (2012), . . .

M. Gander
50 Years of Time Parallel Time Integration,
Multiple Shooting and Time Domain Decomposition
Springer Verlag, 2015.

Outline

joint work with F. Hecht

initiated with A. Loumi and P. Parnaudeau

Evolution Equations

Implicit discretisation

Consider the linear ODE

$$u'(t) = Au(t)$$

with initial condition $u(0) = u_0$.
FD Discretisation: implicit scheme (backward Euler)

$$(I - \delta t_n A)\mathbf{u}^{n+1} = \mathbf{u}^n.$$

For *m* time steps

$$A_{n+m} \cdots A_{n+1} A_n \; \mathbf{u}^{n+m} = \mathbf{u}^n.$$

with

$$A_k = I - \delta t_k A$$

Evolution Equations

Numerical discretisation

Let us recall what a sequential procedure would be.

- Computing $X = A_1 \cdots A_m$ before solving the linear system $\Longrightarrow$ very expensive
- A sensible sequential computation of the solution:
    - Compute successively $\mathbf{u}^{n+j}$ (for $j = 1, \cdots, m$) solution of
    $$A_j \mathbf{u}^{n+j} = \mathbf{u}^{n+j-1}$$

Cost

Total cost $= m \times$ the cost of solving one linear system.

Evolution Equations

Numerical discretisation

$u'(t) = Au(t)$,
with initial condition $u(0) = u_0$.
Implicit scheme

$$X\mathbf{u}^{n+m} = \mathbf{u}^n.$$

with

$$X = (I - \delta t_{n+m}A)\cdots(I - \delta t_{n+1}A)(I - \delta t_n A)$$

Key idea :

$$\mathbf{u}^{n+m} = \mathbf{X^{-1}}\mathbf{u}^n \quad !!!$$

Evolution Equations

Numerical discretisation

Take $m = 2$.

$$X = (I - h_2 A)(I - h_1 A)$$

For $h_2 \neq h_1$, we have

$$X^{-1} = \alpha_1 (I - h_1 A)^{-1} + \alpha_2 (I - h_2 A)^{-1}$$

with

$$\alpha_1 = \frac{h_1}{h_1 - h_2}, \quad \alpha_2 = \frac{h_2}{h_2 - h_1}.$$

Evolution Equations

Numerical discretisation

Take $m = 2$. The solution of

$$X\mathbf{u}^{n+2} = \mathbf{u}^{n}$$

is broken down into

$$\boxed{x = \alpha_1 x_1 + \alpha_2 x_2}$$

Each vector x_i is the unique solution of the linear system

$$\boxed{A_i x_i = y}$$

There are two independant linear systems to solve.

Cost

Computing x this way is as expensive as solving a single linear system (neglecting the $\alpha_i x_i$ summation).

Parallel computation of Linear Systems

$(A_i)_{i=1}^{m}$ are $n \times n$ nonsingular real matrices

$$X = A_1 \cdots A_m.$$

Problem : how to compute quickly the solution $x \in \mathbb{R}^n$ of the linear system

$$Xx = y$$

where $y \in \mathbb{R}^n$ is any given vector?

Parallel computation of Linear Systems

Consider m distinct real numbers h_i and

$$X = \prod_{i=1}^{m} (I + h_i A)$$

Proposition

The inverse matrix of X (with nonsingular matrices A_i) is

$$X^{-1} = \sum_{i=1}^{m} \alpha_i A_i^{-1}$$

with

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

(partial fraction decomposition of rational functions!!)

Parallel computation of Linear Systems

Accordingly, the solution $Xx = y$ is broken down into

$$\boxed{x = \sum_{i=1}^{m} \alpha_i x_i}$$

Each vector x_i is the unique solution of the linear system

$$\boxed{A_i x_i = y}$$

There are m **independant linear systems** to solve.

Cost

Computing x this way is as expensive as solving a single linear system (neglecting the $\alpha_i x_i$ summation).

In practice, the **interconnections between processors are far from being neglectible**.

Linear Systems arising from FEM

$$M\frac{u^{n+1}-u^n}{h_1} = Bu^{n+1}$$

with

- M: mass matrix
- B: stiffness matrtix

For $m = 2$

$$(I - h_1 A)(I - h_2 A)u^{n+2} = u^n$$

with

$$A = M^{-1}B$$

Exactly the same framework!

FEM : Laplace equation (2D)

Consider the homogeneous equation

$$u'(t) = Au(t)$$

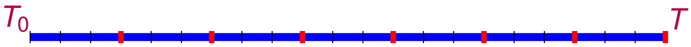
with initial condition $u(0) = u_0$.

Implicit scheme using m distinct time steps h_i

Notations:

- intial and final times: T_0, T
- $t_n = T_0 + n$ time steps : $h_1 + h_2 + \cdots + h_m + h_1 + h_2 + \cdots$

Remark: solution computed only at times t_{km}, $k = 1, \cdots, K$



example: m=3, K=7

FEM : Laplace's equation (2D)

Approximation from t_{km} to $t_{(k+1)m}$

- $$\begin{aligned} A_1 u_{km+1} &= u_{km} \\ A_2 u_{km+2} &= u_{km+1} \\ &\vdots \\ A_m u_{km+m} &= u_{km+m-1} \end{aligned}$$
- $X u_{(k+1)m} = u_{km}$

with $X = A_1 \cdots A_m$

FEM : Laplace's equation (2D)

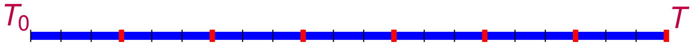
For $k = 1, \cdots K$

- Compute $\mathbf{u}_{(k+1)m}$ solution of

$$Xu_{(k+1)m} = u_{km} + g_{km}$$

$u_{(k+1)m} = \sum_{i=1}^{m} \alpha_i x_i$ `(Reduce step)`

x_i **computed in parallel**



example: m=3, K=7

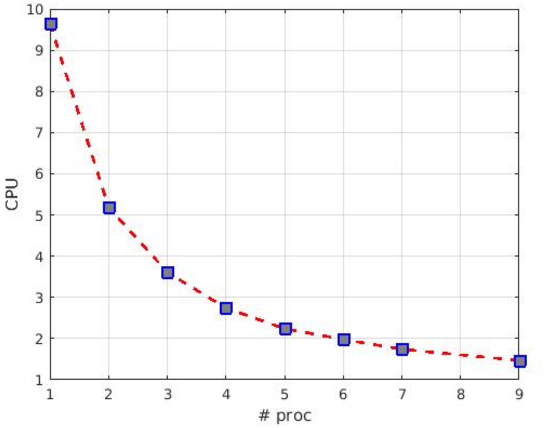
FreeFem

Domain: unit square

```
nb vertices =3721 , nb triangles = 7200 , nb boundary edges 240
real T = .1; int Ndt=3000;
```

Applications

FEM : Laplace's equation (2D)



Efficient parallel solvers

Let us introduce some definitions related to the computation time.

- $f(1,T,N) = f(1, mK\delta t, N)$ is the sequential time to compute the solution at the final time $T = mK\delta t$, for a problem of size N:

$$f(1,T,N) = mK\, f(1,\delta t, N).$$

- $f(m,T,N)$ is the parallel time to solve a problem of size N using m processors

$$f(m,T,N) = Kf(1,\delta t,N) + KC(m,N).$$

with $C(m,N)$ the cost of the communications between m processors sharing a data of size N.

Efficient parallel solvers

Two quantities are of interest in parallel computing:

- Speedup

$$S(m,T,N) = \frac{f(1,T,N)}{f(m,T,N)} = m\frac{f(1,\delta t,N)}{f(1,\delta t,N)+C(m,N)}.$$

- Efficiency

$$E(m,T,N) = \frac{1}{m}S(m,T,N) = \frac{f(1,\delta t,N)}{f(1,\delta t,N)+C(m,N)}.$$

In a perfect world, the communications have no cost $\Longrightarrow$ ideal speedup and ideal efficiency

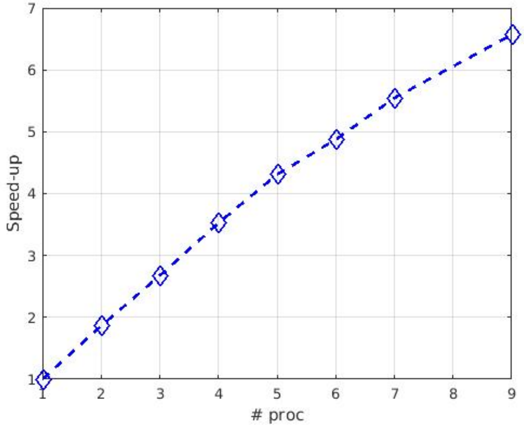
$$S^*(m,T,N) = m, \quad E^*(m,T,N) = 1.$$

Unfortunately our world is not perfect and the communication time $C(m,N)$ is not at all neglectible.

Applications

FEM : Laplace's equation (2D)

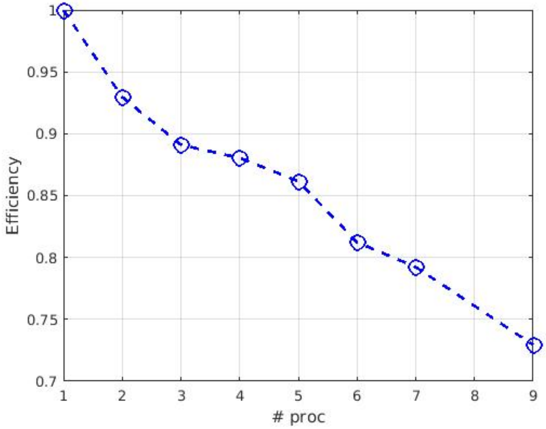
Speedup



Applications

FEM : Laplace's equation (2D)

Efficiency



FEM : Poisson equation (2D)

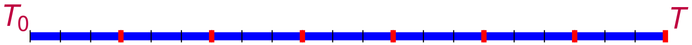
Consider the non-homogeneous equation

$$u'(t) = Au(t) + f$$

with initial condition $u(0) = u_0$.
Implicit scheme using m distinct time steps h_i
Notations:

- intial and final times: T_0, T
- $t_n = T_0 + n$ time steps : $h_1 + h_2 + \cdots + h_m + h_1 + h_2 + \cdots$

Remark: solution computed only at times t_{km}, $k = 1, \cdots, K$



FEM : Poisson's equation (2D)

Approximation from t_{km} to $t_{(k+1)m}$

$$\begin{array}{rcl} A_1 u_{km+1} & = & u_{km} + h_1 f_{km+1} \\ A_2 u_{km+2} & = & u_{km+1} + h_2 f_{km+2} \\ \vdots & & \vdots \\ A_m u_{km+m} & = & u_{km+m-1} + h_m f_{km+m} \end{array}$$

$$X u_{(k+1)m} = u_{km} + g_{km}$$

with $X = A_1 \cdots A_m$ and

$$g_{km} = h_1 f_{km+1} + A_1 \left\{ h_2 f_{km+2} + A_2 \left\{ h_3 f_{km+3} + \cdots + A_{m-1} \left\{ h_m f_{km+m} \right\} \right\} \right\}$$

FEM : Poisson's equation (2D)

- $h = \frac{1}{m}\sum_i h_i$, $T = Km^2 h$
- Outer loop $k = 1, \cdots$
    - input: u_{km^2}
        - For $j = 1, \cdots, m$
          compute u_{km^2+jm} by solving **sequentially**

$$Xu_{km^2+jm} = u_{km^2+(j-1)m} + \underbrace{g_{km^2+(j-1)m}}_{!!}$$

    - output: $u_{(k+1)m^2}$

FEM : Poisson's equation (2D)

- For $k = 1, \cdots$
  - Compute **in parallel** $g_{km^2+(j-1)m}$ for $j = 1, \cdots, m$
  - For $j = 1, \cdots, m$
    - compute **sequentially** u_{km^2+jm} by solving

$$Xu_{km^2+jm} = u_{km^2+(j-1)m} + g_{km^2+(j-1)m}$$

each u_{km^2+jm} being computed **in parallel**

See the computer sessions of the workshop

Limitation(s) of the method

Accuracy. In practical situations, the solution x_i is known only through an approximation $\tilde{x}_i$ with an error ε_i :

$$\|\tilde{x}_i - x_i\| \leq \varepsilon_i.$$

¿From which we deduce an upper bound on the error:

$$\|\tilde{x} - x\| \leq (\max_{1 \leq i \leq m} \varepsilon_i) \sum_{i=1}^{m} |\alpha_i|$$

this bound may be very large as m grows.

Conclusion

- New method to solve linear time-dependent systems using a fractional decomposition of the matrix resulting from the discretisation of the time evolution operator.
- This method has been applied to solve the bidimensional heat equation. We obtain good results.
- However, for precision and stability reasons, the use of our method should be limited to moderate number of processors.
- We are currently investigating several developpements of the method.
    - **combine low-order methods (computed in parallele) to get a high-order one**
    - **combine with space Domain Decomposition**.
    - …
- In summary, the method is efficient, but should be limited to a small number of processors

References

1. M.J. Gander: 50 Years of Time Parallel Time Integration
2. M.J. Gander, L. Halpern, and T.T.B.: A direct solver for time parallelization
3. J.-L. Lions, Y. Maday, and G. Turinici: Résolution d’EDP par un schéma en temps “pararéel”
4. Y. Maday and E.M. Rønquist: Parallelization in time through tensor-product space-time solvers
5. W. L. Miranker and W. Liniger: arallel methods for the numerical integration of ordinary differential equations
6. J. Nievergelt: Parallel methods for integrating ordinary differential equations