ERCIM Working Group Meeting

University of Geneva

May 4, 2001

Solving Economic and Financial Problems with Parallel Krylov Methods

Giorgio Pauletto

Department of Econometrics

Giorgio.Pauletto@metri.unige.ch

http://www.unige.ch/ses/metri/pauletto/

Outline

- Macroeconometric models with forward-looking variables (joint work with M. Gilli)
- Multivariate option pricing (joint work with M. Gilli and E. Kellezi)
- Other Areas of Interest

Forward-Looking Model: MULTIMOD

- Multi-region econometric model of the IMF
- Forward-looking dynamic annual model
- 8 industrial zones: G7 and small industrial countries
- 2 developing zones: high income oil exporters and developing countries
- 3 goods: oil, primary commodities and manufactures
- Short-run behavior described by error correction mechanisms
- Same specification for the models of the industrialized countries (51 equations each)
- Forward-looking wrt real wealth, interest rates, price level

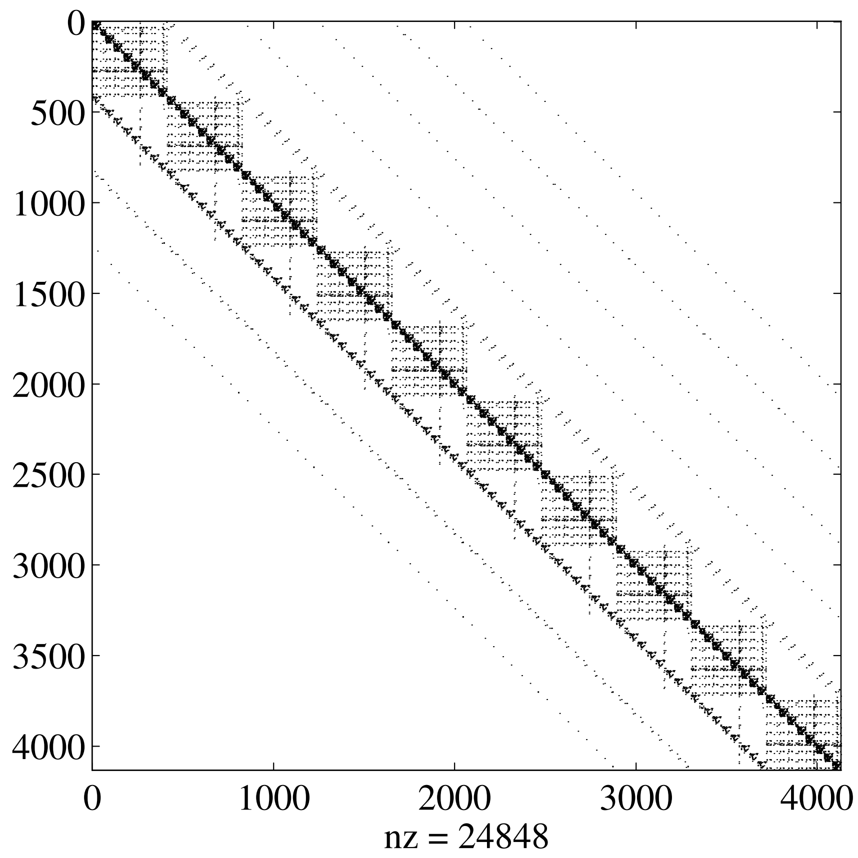
- General form of the model (Simultaneous through future periods)

$$h(y_t, y_{t-1}, \ldots, y_{t-p}, y_{t+1}, \ldots, y_{t+q}, z) = 0$$

- Stacked system of equations:

$$h(y, z) = 0$$

Size $T \times 413$, e.g. for $T = 10$:



- Large and sparse system of nonlinear equations
- Solution algorithm

Algorithm 1 Newton Method

1: Given an initial guess $y^{(0)}$
2: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
3: $\quad b^{(k)} = -h(y^{(k)}, z)$
4: $\quad J^{(k)} = \partial h(y^{(k)}, z)/\partial y'$
5: $\quad \boxed{\textbf{solve } J^{(k)} s^{(k)} = b^{(k)}}$
6: $\quad y^{(k+1)} = y^{(k)} + s^{(k)}$
7: **end for**

- Large and sparse linear system

$$J\, s = b$$

- Large and sparse linear system

$$A\,x = b$$

- Strategies:
  - Sparse direct methods
    - based on LU decomposition
    - reordering necessary to preserve sparsity of factors
  - Krylov subspace methods (inexact Newton)
    - unsymmetric system: GMRES, BiCG, QMR, . . .

Preconditioning

- Convergence of Krylov methods depends on distribution of eigenvalues
- Improve convergence by transforming $Ax = b$ into $M^{-1}Ax = M^{-1}b$ with (hopefully) more clustered eigendistribution
- M should be:
  - a good approximation of A
  - not too expensive to build
  - $Mz = c$ should be easy to solve

So what do we suggest in our application?

Preconditioner Based on Block Structure

- Strucure of Jacobian matrix without leads and lags

$$\begin{bmatrix} D^{t+1} & & & \\ & D^{t+2} & & \\ & & \ddots & \\ & & & D^{t+T} \end{bmatrix}$$

- Use of same matrix D along diagonal
- Factor matrix D once
  - sparse LU with partial pivoting
  - incomplete LU: drop off entries that fill in and are less than a tolerance

- Formally we have
  $M_1 = I_T \otimes P_1 \quad \text{where} \quad P_1 = D^{t+1}$
- Generalization:
  $M_s = I_{T/s} \otimes P_s$
  with $\quad T \bmod s = 0$
- For instance $s = 2$:

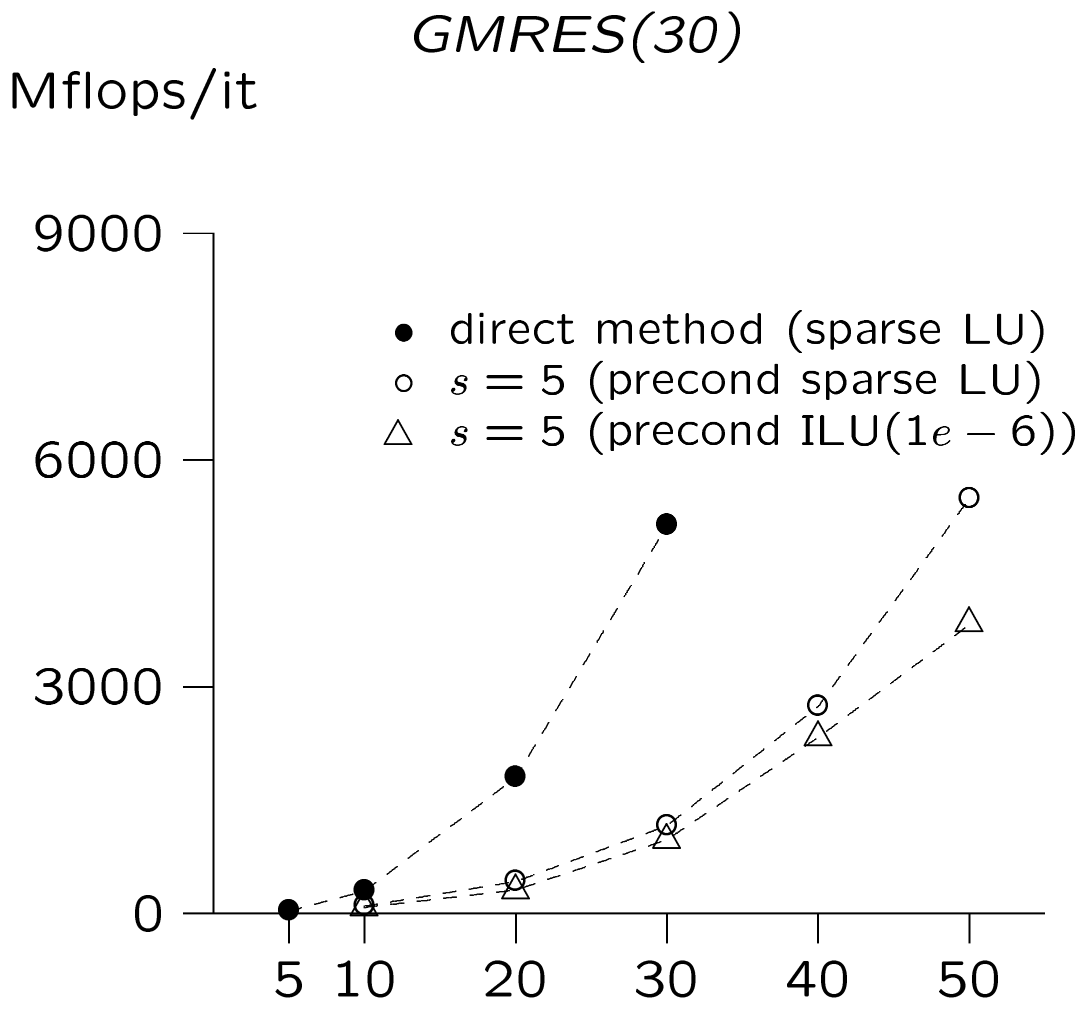
$$P_2 = \begin{bmatrix} D^{t+1} & A_1^{t+2} \\ E_1^{t+1} & D^{t+2} \end{bmatrix}$$

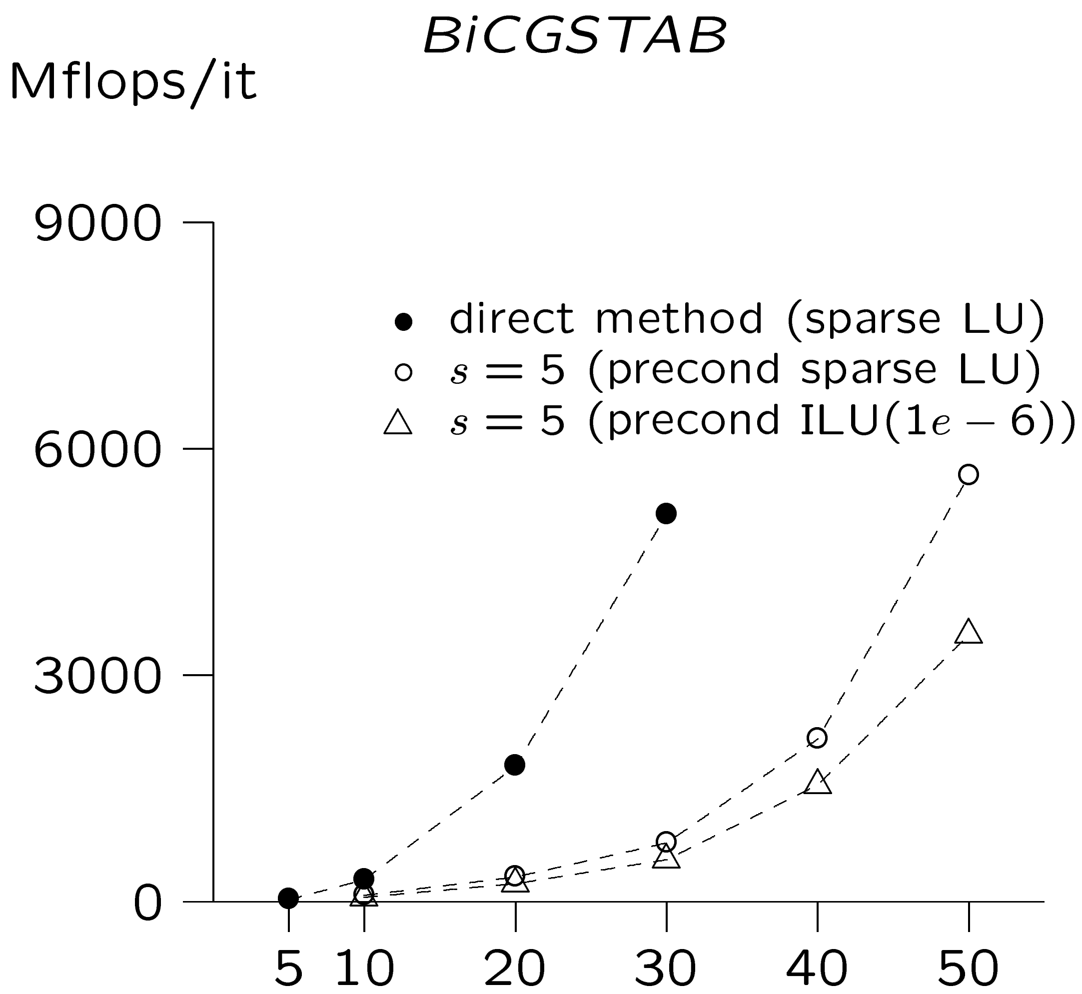
Practical Numerical Implementation

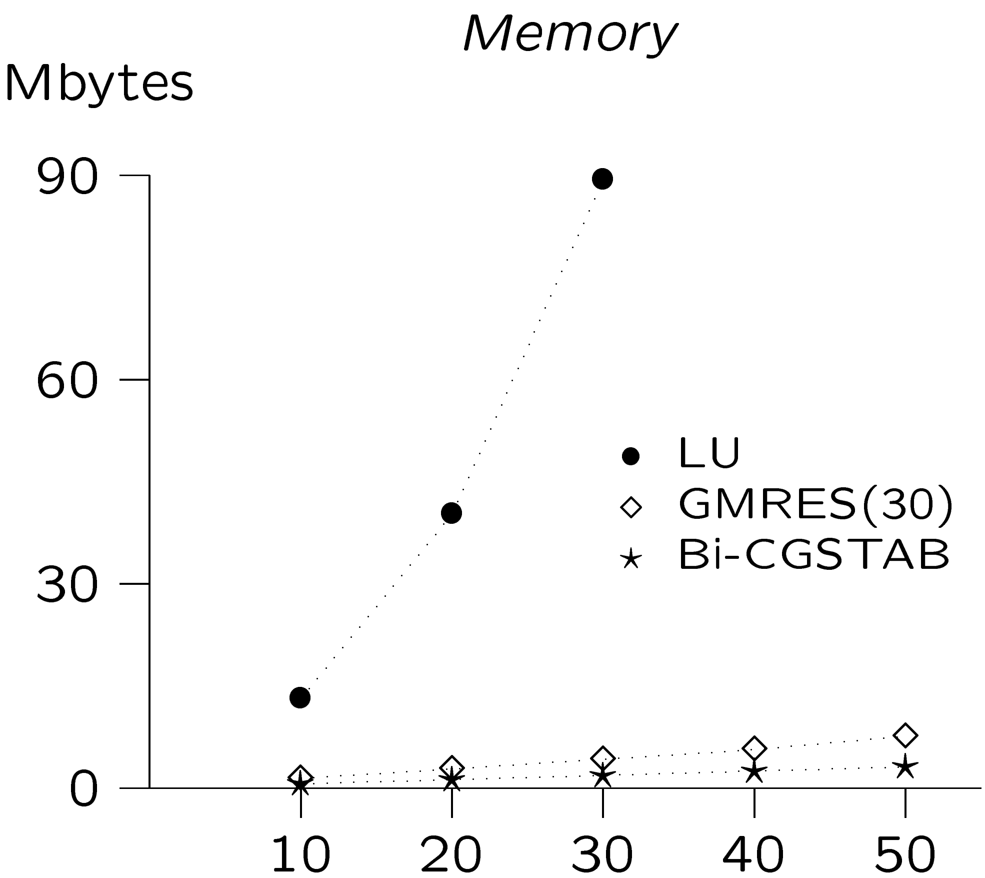
- Implementation in MATLAB:
  - Code modified from Barrett et al. (1994) *"Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods"*
  - Krylov techniques are now available in MATLAB 5.x
- Implementation in Fortran and C:
  - Using codes available in PETSc *"Portable and Extensible Toolkit for Scientific Computation"* Argonne National Laboratory S. Balay, W. Gropp, L. Curfman McInnes and B. Smith
- Allows parallel execution machines using MPI

Some Computational Results on MULTIMOD

- If enough RAM, use sparse LU
- GMRES with long restart seems to be the most robust
- BiCGSTAB seems to be the cheapest computationally
- Preconditioning with block structure is efficient
- Incomplete factorization with small drop-off is efficient


GMRES(30)
Mflops/it
9000
6000
3000
0
direct method (sparse LU)
$s = 5$ (precond sparse LU)
$s = 5$ (precond ILU($1e-6$))
5
10
20
30
40
50


BiCGSTAB
Mflops/it
9000
6000
3000
0
5
10
20
30
40
50
direct method (sparse LU)
$s = 5$ (precond sparse LU)
$s = 5$ (precond ILU($1e-6$))



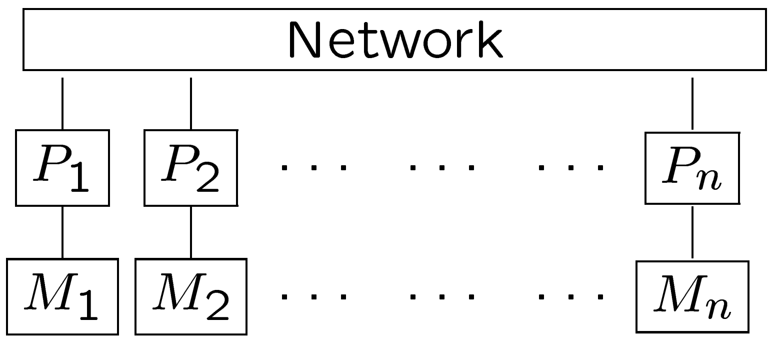
Memory
Mbytes
90
60
30
0
LU
GMRES(30)
Bi-CGSTAB
10
20
30
40
50

Parallel Computing

- Engineering problems solved using parallel computing
- Trends in scientific parallel computing:
  - Solve faster
  - Solve bigger
  - Solve cheaper
  - Solve smarter
- Can this economic problem be efficiently solved on parallel systems? (Sparse but nonsymmetric, nondefinite)

Parallel computing issues

- Locality



- Regularity

Operation count/Memory references

Operation	Flops	Memory references	Ratio
`saxpy` $y = \alpha x + y$	$2n$	$3n + 1$	$2/3$
matrix $\times$ vector	$2n^2$	$n^2 + 3n$	2
matrix $\times$ matrix	$2n^3$	$4n^2$	$n/2$

Parallel Methods for Sparse Linear Systems

- Direct methods: based on LU, submatrix-based or column-based
- Iterative methods: Krylov subspace methods (GMRES, BiCGSTAB, ...)
  - Challenge: $O(n)$ operations on $O(n)$ data
  - Memory traffic cannot be hidden by useful computations
  - Smart ways to avoid the traps
  - Better communication hardware diminishes some problems

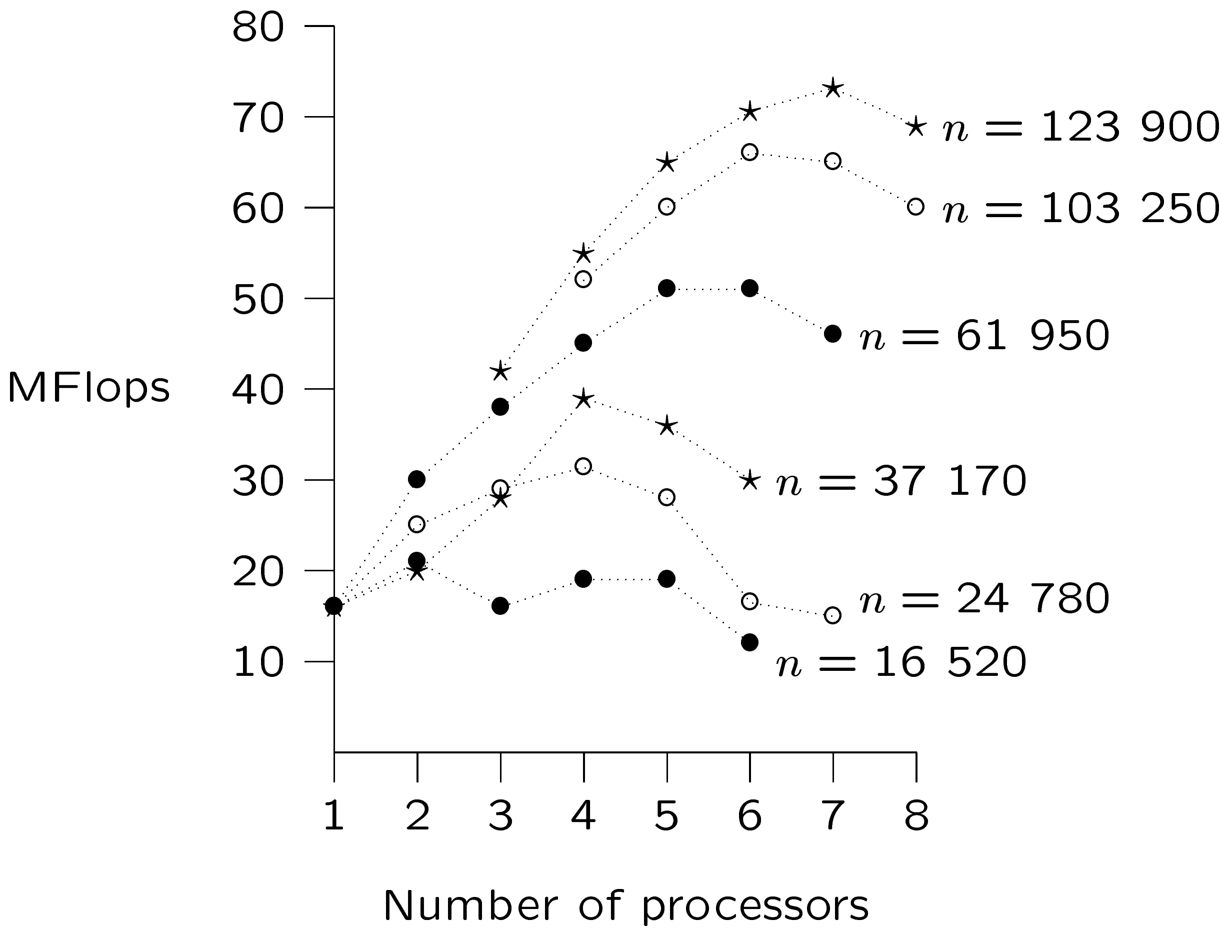
Practical Results

Hardware

- IBM SP2 in Geneva 14+1 nodes
- Each node RS/6000 @ 66 MHz
- Performance
  Peak 266 MFlops — LINPACK 53 MFlops
- High performance switch:
  complete graph
  Latency $\alpha = 0.5\,\mu s$ — $30\mu s$
  Bandwidth $1/\beta = 40$ MBytes/s — 30MBytes/s
- Time needed to move n data items in memory

$$\alpha + \beta\, n \qquad \alpha, \beta \geq 0$$

- Algorithm GMRES(30) with Block Jacobi ILU(0) preconditioner (implemented in PETSc 2.0)
- Performance



Scalability: Speed-up

- T_1 serial time
- T_p parallel time
- $S_p = T_1/T_p$ speed-up
- $E_p = S_p/p$ efficiency
- Isoefficiency

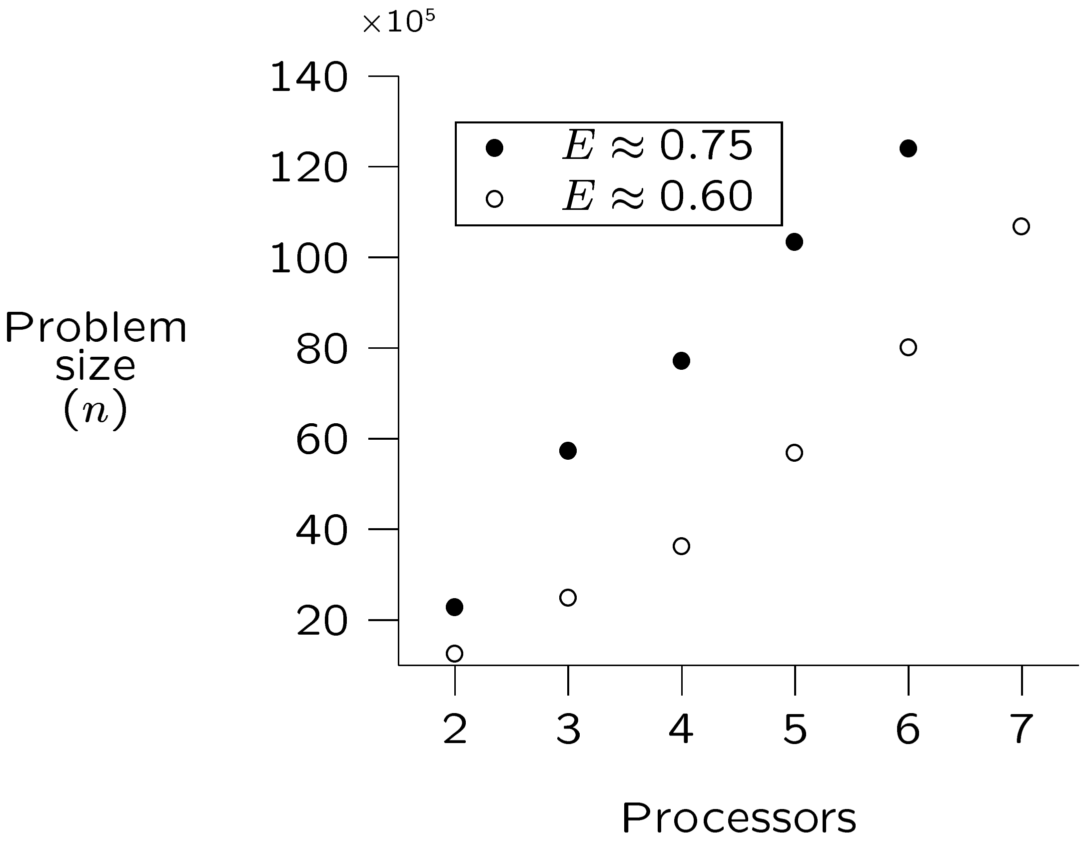
Speed-up S_p

n	$p =$	1	2	3	4	5	6	7	8
16 520		1	1.31	1.00	1.19	1.19	0.75		
24 780		1	1.56	1.81	1.96	1.96	1.76	1.03	0.94
37 170		1	1.25	1.75	2.44	2.25	1.88		
61 950		1	1.88	2.37	2.81	3.19	3.19	2.88	
103 250		1			3.25	3.75	4.13	4.13	3.75
123 900		1		2.63	3.43	4.06	4.41	4.56	4.31

Scalability: Isoefficiency

Efficiency E_p

n	$p=$ 1	2	3	4	5	6	7	8
16 520	1	0.66	0.33	0.30	0.24	0.13		
24 780	1	0.78	0.60	0.49	0.35	0.17	0.13	
37 170	1	0.63	0.58	0.61	0.45	0.31		
61 950	1	0.94	0.79	0.70	0.64	0.53	0.41	
103 250	1			0.81	0.75	0.69	0.59	0.47
123 900	1		0.88	0.86	0.81	0.74	0.65	0.54



Option pricing

- An **option** is a contract that gives the holder the **right** to buy (*call*) an asset (*underlying*) at a certain date (*expiration*) for a given price (*strike*)
- If price S_T at expiration $>$ strike K then exercise the option: gain $S_T - K$
- Otherwise don't do anything: gain 0
- Payoff is $\left[S_T - K\right]_+$
- What is the fair price C of such a contract?
- Depends on strike, time to expiration, underlying price, but also on the interest rate and volatility of underlying

Option pricing

- Evolution of the underlying price

$$\frac{dS}{S} = \mu \, dt + \sigma \, dZ$$

  where μ is the expected rate of return σ the volatility and dZ the standard Wiener process

- For constants μ and σ, the stochastic differential equation has explicit solution
  $\longrightarrow$ Black and Scholes / Merton (1973)

- What about more general models?

- Numerical computation:
  Trees, PDE, Monte Carlo

Description of the Instrument

- Multi-asset European option
- $C = \Big[\text{max}(S_1, S_2, \ldots, S_n) - K, 0\Big]_+$
- $n = 3$ the volatilities are 0.2 and the correlations 0.5
- Same strike prices
- We can easily allow for different volatilities and correlations

Fundamental PDE

$$\frac{\partial C}{\partial t} = -\sum_{i=1}^{3} r S_i \frac{\partial C}{\partial S_i} - \frac{1}{2}\sum_{i,j=1}^{3} \frac{\partial^2 C}{\partial S_i \partial S_j}\rho_{ij}\sigma_i\sigma_j + rC$$

with terminal condition at time T

$$C = \Big[\max\{S_1, S_2, S_3\} - K\Big]_+$$

$$\frac{\partial u}{\partial \tau} = V \cdot \nabla u + (D\nabla)\cdot \nabla u - ru$$

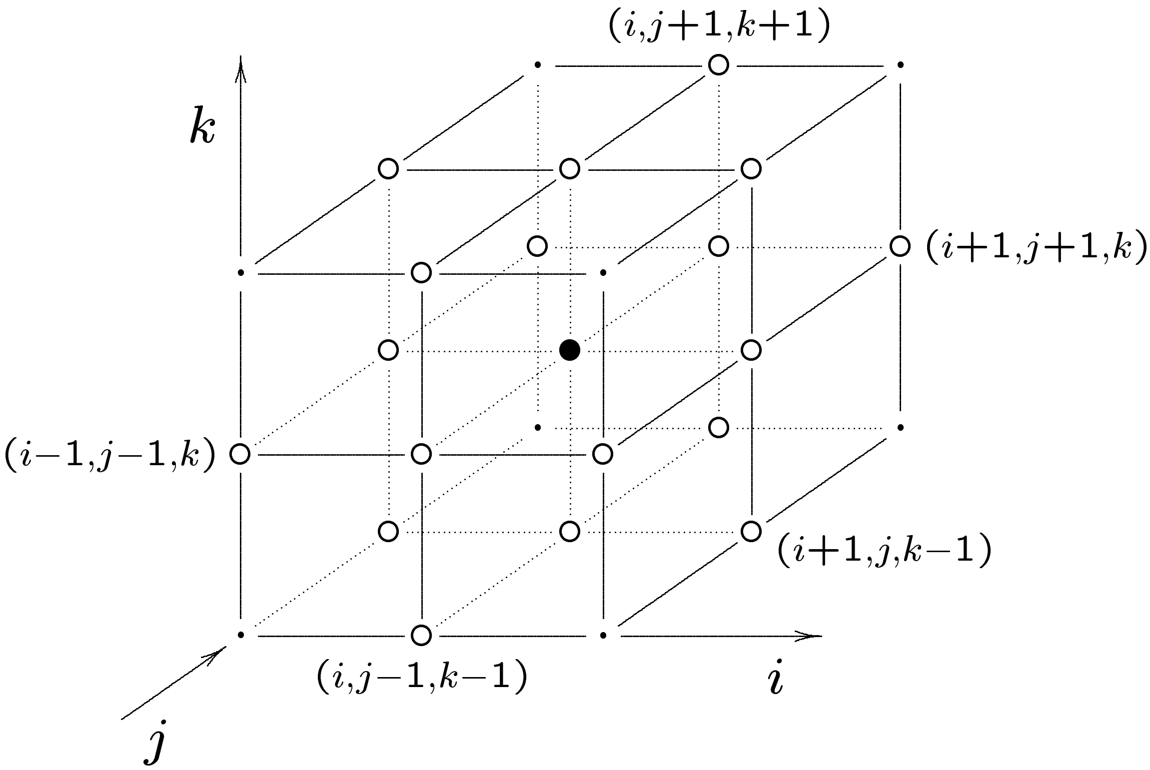
$$V = \begin{bmatrix} r - \frac{1}{2}\sigma_x^2 \\ r - \frac{1}{2}\sigma_y^2 \\ r - \frac{1}{2}\sigma_z^2 \end{bmatrix}$$

$$D = \begin{bmatrix} \frac{1}{2}\sigma_x^2 & \rho_{xy}\sigma_x\sigma_y & \rho_{xz}\sigma_x\sigma_z \\ \rho_{yx}\sigma_y\sigma_x & \frac{1}{2}\sigma_y^2 & \rho_{yz}\sigma_y\sigma_z \\ \rho_{zx}\sigma_z\sigma_x & \rho_{zy}\sigma_z\sigma_y & \frac{1}{2}\sigma_z^2 \end{bmatrix}$$

Solution by Finite Difference Methods

Discretization

Grid fragment



Finite Difference Method

Solve for u_{Ω}^{m+1}

$$(I - \theta A)u_{\Omega}^{m+1} = \Big(I + (1-\theta)A\Big)u_{\Omega}^{m} + \theta B u_{\partial\Omega}^{m+1} + (1-\theta) B u_{\partial\Omega}^{m}$$

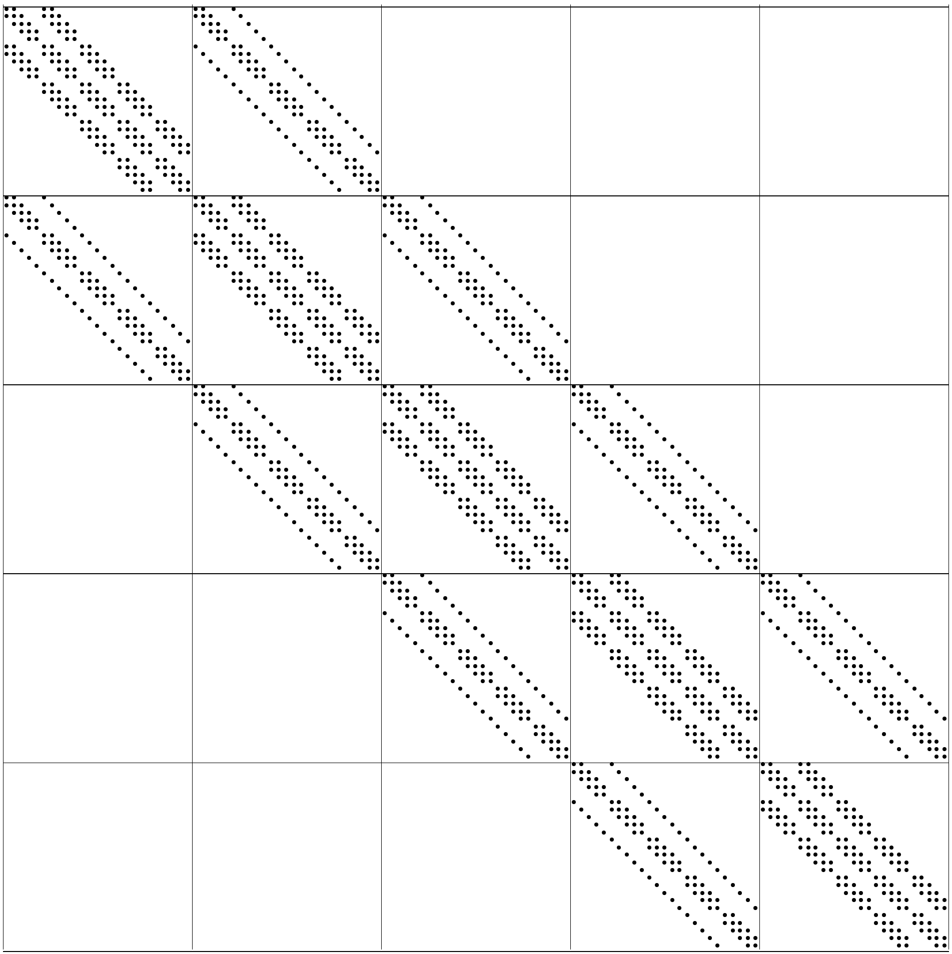
- $\theta = 0$ explicit method
- $0 < \theta < 1$ implicit method $\longrightarrow$ solution of linear system

Implicit methods are numerically more stable and more precise for a given discretization

Finite Difference Method

Dimension of matrix
$|\Omega| = \prod_{i=1}^{d}(N_i - 1) \approx N^d$

Sparsity pattern of matrix A



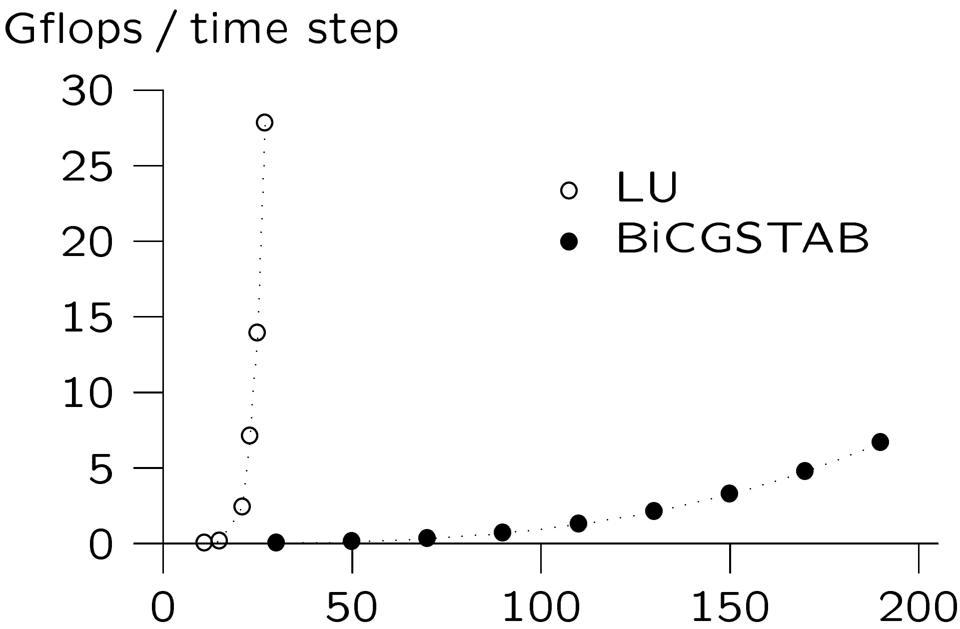
Implicit Method

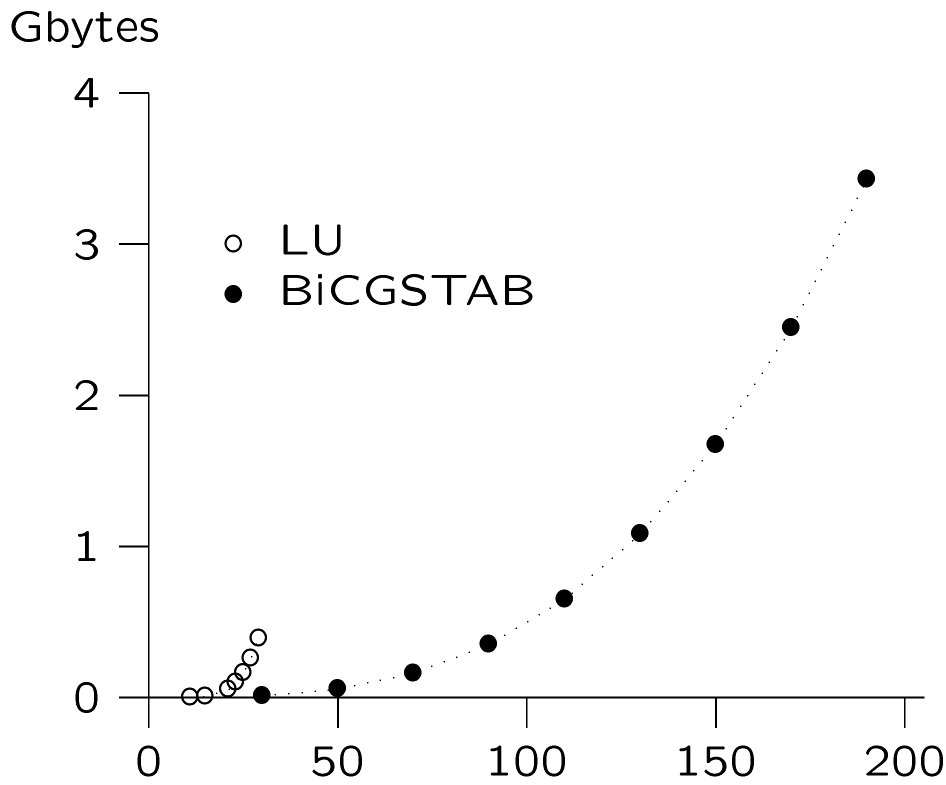
- Need to solve large linear system for each time-step
- For reasonable grid sizes
  $N = 140 \quad \longrightarrow \quad 2.5 \times 10^6$ equations
- Krylov subspace methods, BiCGSTAB
- Serial implementation $\longrightarrow$ memory bottleneck

Implicit Method

- Parallel implementation on a 32 node Beowulf
- Pentium II machines at 220 MHz with 128 MBytes memory
- Software written in C uses PETSc and MPI

Computational Results





Other Areas of Interest

- Application of Krylov methods to other economics problems (Dynamic Programming)
  [Giorgio Pauletto]
- Parallel Monte Carlo (PRNG)
  [Giorgio Pauletto]
- Model selection
  [Manfred Gilli, Erricos Kontoghiorghes, Elvezio Ronchetti]
- Artificial financial markets
  [Manfred Gilli]