Parallel Multistage Programming Model Used in Portfolio Management

L.Halada
Institute of Informatics,
Slovak Academy of Sciences,
Bratislava, Slovakia
upsyhala@savba.sk

M.Lucka
Institute for Software Science,
University of Vienna,
Vienna, Austria
lucka@par.univie.ac.at

I. Melichercik
Department of Economic and Financial Modeling,
Faculty of Mathematics, Physics and Informatics,
Comenius University, Bratislava,
igor.melichercik@fmph.uniba.sk

AURORA - A Special Research Program funded by ASF

VEGA 1/1004/04 - Slovak Academy Grant Agency

Overview

- Portfolio management problem
- Multiperiod stochastic programming
- Tuning the model on historical data
- Mathematical background -
  Algorithm, Parallelization, Implementation
- Financial and parallel performance
- Concluding remarks

Portfolio Management Problem

- Allocation of a given amount of money in different investments - portfolio selection
- Assets allocation in several currencies
- Uncertainty is of investments
  - foreign exchange risk
  - interest rate risk
  - possible higher returns - the aim
- Portfolio manager has to decide: What bonds should he/she buy or sell?

  In what currencies?

Basic Concept of Multi-period Stochastic Programming

- possibility to correct the decision in the future with respect to developments of financial markets
- make initial decision - in the future correct the decision with respect to developments of financial markets
- the data for the first period are known with certainty-some data concerning the future are stochastic, random
- dynamic nature of decision making – multi-stage stochastic programming model

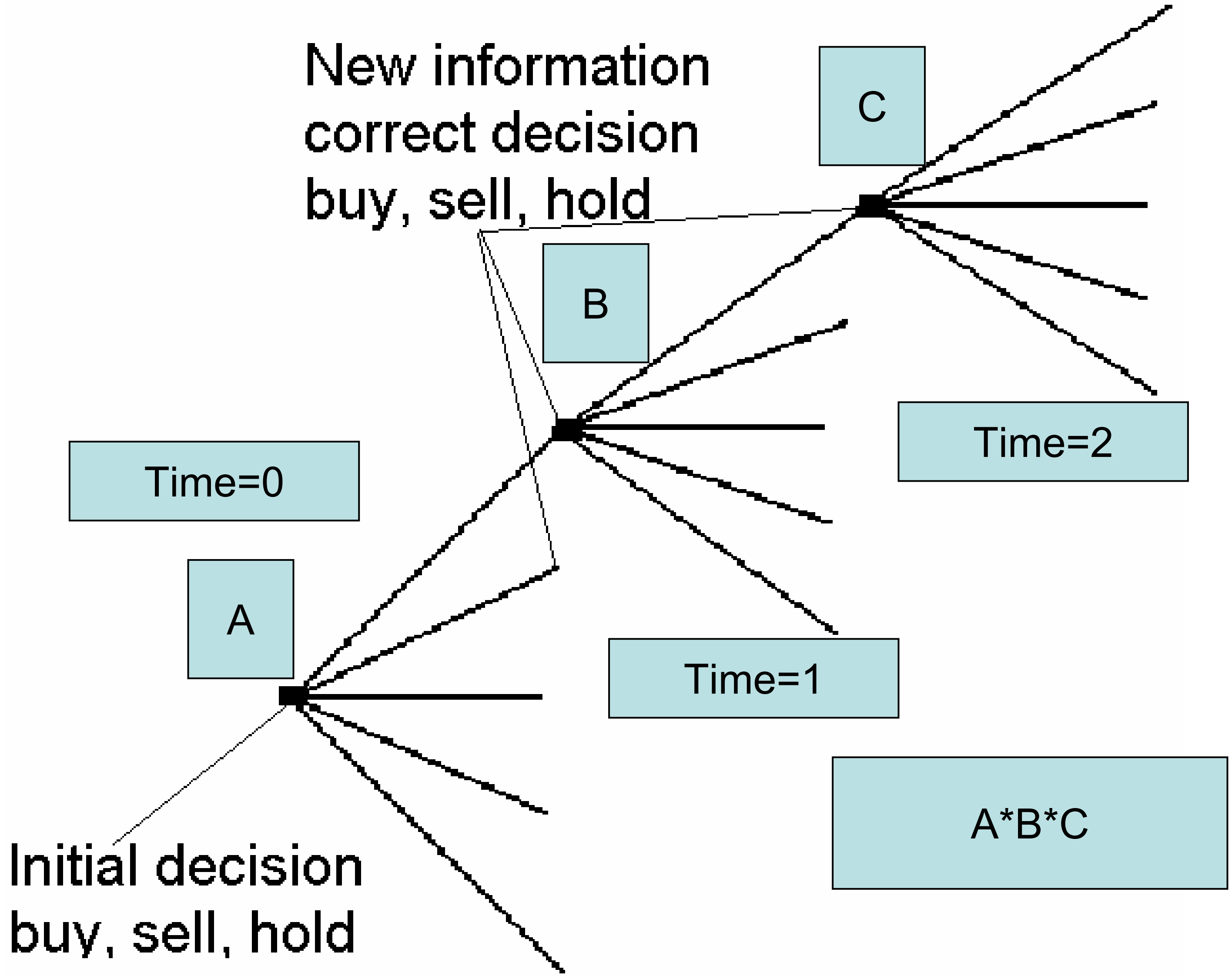
The Problem is …. Uncertainty

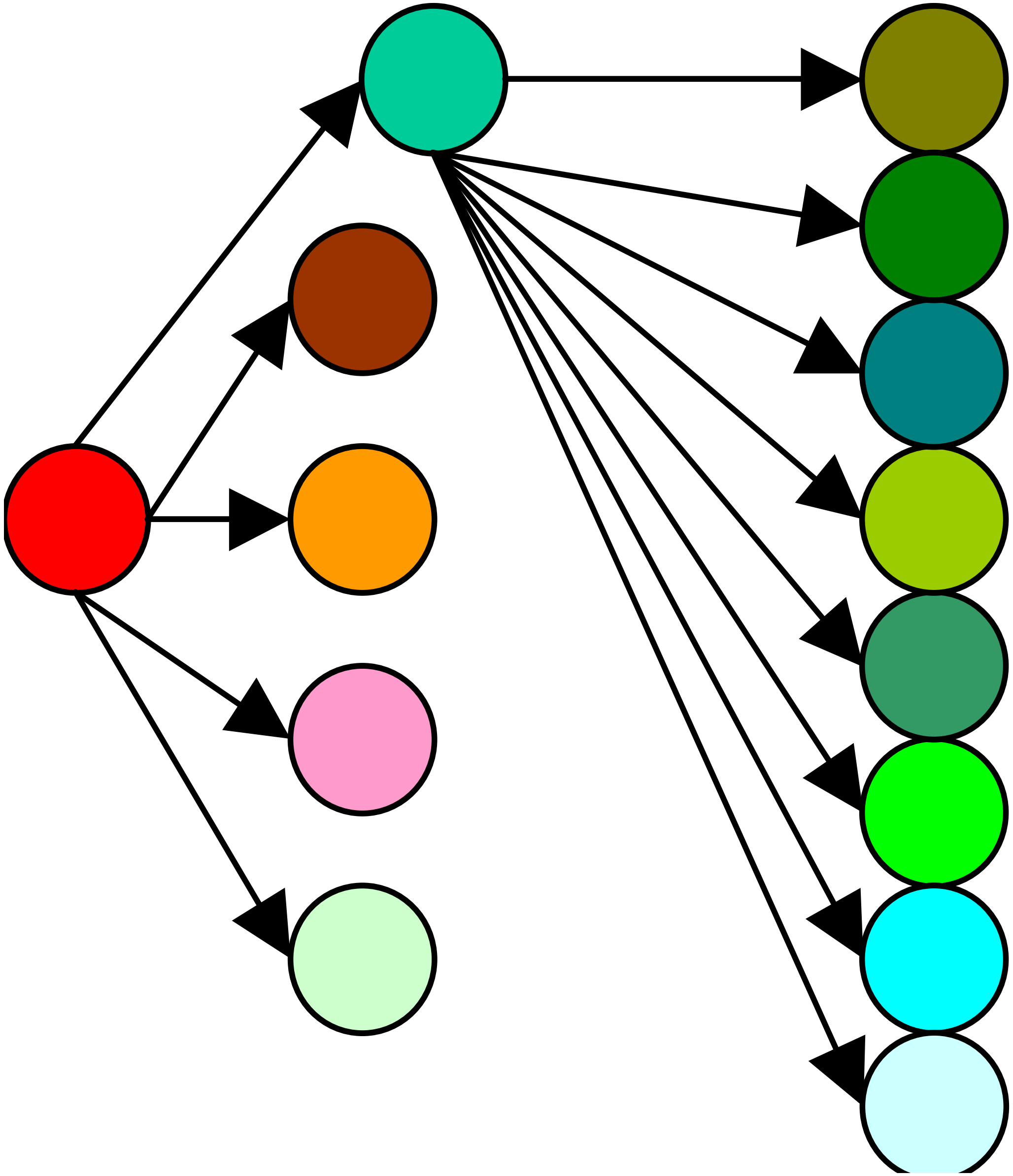
- we do not know the future development of interest rates
- results are very sensitive to the choice of the interest rate scenarios

$\Rightarrow$

  - scenarios have to be chosen carefully
  - enough “standard scenarios”
  - reasonably many “extreme scenarios”
  - no “very extreme scenarios”

The uncertainty is represented by a scenario tree


New information
correct decision
buy, sell, hold
C
B
Time=0
Time=2
A
Time=1
A*B*C
Initial decision
buy, sell, hold



Theoretical Problem Formulation

Inventory balance and cash-flow accounting

Period 1

$$h_j^{(0)} + b_j^{(1)} - s_j^{(1)} = h_j^{(1)}$$

$$c^{(0)} + \sum_j bid_j^{(1)} s_j^{(1)} = \sum_j ask_j^{(1)} b_j^{(1)}$$

Period 1<t<T

j=1,2,...number of currencies

$$h_j^{(\tau-1)}(pred) + b_j^{(\tau)}(node) - s_j^{(\tau)}(node) = h_j^{(\tau)}(node)$$

$$\sum_j bid_j^{(\tau)}(node) s_j^{(\tau)}(node) = \sum_j ask_j^{(\tau)}(node) b_j^{(\tau)}(node)$$

No short positions $\quad b_j^{\tau} \geq 0; s_j^{\tau} \geq 0; h_j^{\tau} \geq 0$

Terminal wealth calculation at time horizon T

$$WT(\omega^T) = \sum_j \xi_j^T(\omega^T) h_j^{T-1}(\alpha(\omega^T)), \quad \text{for all } \omega^{\mathrm{T}} \in F_T$$

$\xi_j^T(\omega^T)$ **Bid price, index j** $1 \le \tau < T$

$h_j^{T-1}(\alpha(\omega^T))$ **Amount of hold units, index j**

ω^τ **is a node in the scenario tree in the time level** τ

Objective function maximizes the terminal welth

$$E(WT) = \sum_{\omega^{\mathrm{T}} \in F\tau} \pi(\omega^\tau) WT(\omega^\tau)$$

$\pi(\omega^T)$ **probability of the scenario** ω^T

Size of the problem

Decision variables: $\mathbf{b_j(node)}$, $\mathbf{s_j(node)}$, $\mathbf{h_j(node)}$

j=1,2,...number of currencies

➔number of variables is proportional to the number

➔of nodes grows exponentially with number of stages

Example 1

4 currencies ➔ 4x3=12 decision variables per node

3 possibilities for each currency development per stage

3 stages

➔ $3^4=81$ possibilities of market development per stage

➔ $12 \times (1+81+81^2) = 79\,716$ decision variables

Size of the problem AxB

Example 2

4 currencies ➔ 4x3=12 decision variables per node

300 possibilities for each currency development at stage 1

300 possibilities for each currency development at stage 2

➔12*(1+A+A*B) possibilities of market development

➔ 1 083 612 decision variables

Scenario tree generation

The price is supposed to follow the discretized lognormal process

$$price_j^{\tau+1} = price_j^{\tau}.\exp(\mu_j.\Delta t + \sigma_j.\sqrt{\Delta t}.Z_j) \quad Z_j \sim N(0,1)$$

Z_j is random variable with normal distribution, Monte Carlo simulations, correlations cor(Z_i,Z_j) and volatilities σ_j are calibrated using historical data

Suppose, that the price process is mean reverting to the prescribed price $P_j^{(\tau+1)}$

$$E(price_j^{(\tau+1)} \mid price_j^{\tau}) = P_j^{(\tau+1)}$$

$$\Rightarrow \mu_j = \log\left(\frac{P_j^{(\tau+1)}}{price_j^{\tau}}\right).\frac{1}{\Delta t} - \frac{1}{2}\sigma_j^2$$

Mathematical background - modeling on historical data

for D= start_day, end_day do

1. Read indices of government bonds for the date D, D-kΔt, k=0,1,2,3, and j=1,2,3,4
2. generate the scenario tree
3. calculate bid and ask prices, and create the input matrix and vectors for the optimization problem
4. solve the optimization problem
5. calculate the new value of the portfolio

end do

Solving the optimization problem

$$\max c^T x, \quad c \text{ and } x \in R^n,$$

x is the vector of decision variables

subject to

$$Ax = b, \quad x \geq 0, \quad b \in R^m$$

A is a constrained matrix

$$\min b^T y, \quad \text{subject to}$$

$$A^T y + z = c, \quad z \geq 0$$

y are dual variables, z are slack variables

Primal-Dual IPM

- N. Karmarkar: *A* new polynomial-time algorithm for linear programming, Combinatorica 4, 1984, pp.373-395

  Polynomial complexity
- Mehrotra's predictor-corrector algorithm
  - Mehrotra, S. 1992. On the Implementation of a Primal-dual Interior Point Method. SIAM J. Optim.2, 575-601.
- Stephen J.Wright: Primal-Dual Interior-Point Methods, SIAM 1997
  - a variant of Mehrotra's IPM – used in the codes such as LIPSOL, LOQO, PCx

Properties of the IPM Method

- A has full row rank, it is fix
- D diagonal matrix with strictly positive entries
- b is a right-hand side vector
- D, b change in each iteration
- $(ADA^{t})\Delta y=b$ 90-95% computing time
- sparse Cholesky decomposition, solving the system of linear equations
- BQ decomposition
  - Birge, J.R.,Qi, L. Management Sci.,34,1988
  - Birge, J.R., Holmes, D.F., Comput.Optim.Appl.,1,1992

MPC ALGORITHM, pp.198

Given a point (x, y, z) with $(x, z) > 0$ the affine - scaling direction is found by solving the system:

$$\boxed{\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{pmatrix}\begin{pmatrix} \Delta x^{aff} \\ \Delta y^{aff} \\ \Delta z^{aff} \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XZe \end{pmatrix}}$$

r_c, r_b and r_μ are residual vectors

$$residuals: r_b = Ax - b, r_c = A^T y + z - c, r_\mu = -XZe$$

$$\Delta x^{aff} := Z^{-1}(XA^T \Delta y^{aff} + r_\mu - Xr_c)$$

$$\Delta y^{aff} := \boxed{(ADA^T)^{-1}} + \left(r_b + AZ^{-1}\left(Xr_c - r_\mu\right)\right),$$

$$\Delta z^{aff} := X^{-1}(r_\mu - Z\Delta x^{aff}),$$

$$D = Z^{-1}X$$

IPM approach

path-following method

iterative Newton method

$$\sigma = \left(\mu_{aff} / \mu\right)^3, \quad \alpha_{\text{aff}}^{\text{pri}} = \arg\max\left\{\alpha \in [0,1], x + \alpha \Delta x^{aff} \geq 0\right\}$$

$$\mu = x^T z / n, \quad \alpha_{\text{aff}}^{\text{dual}} = \arg\max\left\{\alpha \in [0,1], z + \alpha \Delta z^{aff} \geq 0\right\},$$

$\alpha_{\text{aff}}^{\text{dual}}, \alpha_{\text{aff}}^{\text{pri}}$ are centering parameters

$$\mu_{\text{aff}} = \left(x + \alpha_{aff}^{pri} \Delta x^{aff}\right)^T \left(z + \alpha_{aff}^{dual} \Delta z^{aff}\right) / n$$

$$r_b = 0, \quad r_c = 0, \quad r_\mu = \sigma \mu e - \Delta X^{aff} \Delta Z^{aff} e, \qquad \left(\Delta x^{cc}, \Delta y^{cc}, \Delta z^{cc}\right)$$

$$\left(\Delta x^k, \Delta y^k, \Delta z^k\right) = \left(\Delta x^{aff}, \Delta y^{aff}, \Delta z^{aff}\right) + \left(\Delta x^{cc}, \Delta y^{cc}, \Delta z^{cc}\right)$$

$$\alpha_{\max}^{pri} = \arg\max\left\{\alpha \geq 0; x^k + \alpha \Delta x^k \geq 0\right\} \quad \alpha_{\max}^{dual} = \arg\max\left\{\alpha \geq 0; z^k + \alpha \Delta z^k \geq 0\right\}$$

$$\alpha_k^{pri} = \min\left(0.99 * \alpha_{\max}^{pri}, 1\right) \qquad \alpha_k^{dual} = \min\left(0.99 * \alpha_{\max}^{dua}, 1\right)$$

$$x^{k+1} = x^k + \alpha_k^{pri} \Delta x^k, \qquad k = 0,1,2,\ldots$$

$$\left(y^{k+1}, z^{k+1}\right) = \left(y^k, z^k\right) + \alpha_k^{dual} \left(\Delta y^k, \Delta z^k\right)$$

Matrix A for Three-Stage Stochastic Model

$$A^{(3)} = \begin{pmatrix} A_0^{(3)} & & & & & & & & \\ T_{10} & A_{10} & & & & & & & \\ & T_{11} & A_{11} & & & & & & \\ & T_{12} & & A_{12} & & & & & \\ & T_{13} & & & A_{13} & & & & \\ T_{20} & & & & & A_{20} & & & \\ & & & & & T_{21} & A_{21} & & \\ & & & & & T_{22} & & A_{22} & \\ & & & & & T_{23} & & & A_{23} \end{pmatrix} = \begin{pmatrix} A_0^{(3)} & & \\ T_1^{(3)} & A_1^{(2)} & \\ T_2^{(3)} & & A_2^{(2)} \end{pmatrix}$$

Matrix A for the Two-Stage Stochastic Model

$$A^{(2)} = \begin{pmatrix} A_0^{(2)} & & & & & & & & \\ T_{11} & A_{11}^{(1)} & & & & & & & \\ T_{12} & & A_{12}^{(1)} & & & & & & \\ T_{12} & & & A_{12}^{(1)} & & & & & \\ T_{12} & & & & .. & & & & \\ & & & & & .. & & & \\ & & & & & & .. & & \\ & & & & & & & A_{1k-1}^{(1)} & \\ T_{1k} & & & & & & & & A_{1k}^{(1)} \end{pmatrix}$$

Matrix $A^{(1)}_{ij}$

$$\begin{bmatrix} p_b(0,1) & p_b(0,2) & p_b(0,3) & p_b(0,4) & p_s(0,1) & p_s(0,2) & p_s(0,3) & p_s(0,4) & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T^{(1)}ij$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solving the system $(ADA^T)\Delta y=b$

- Birge, J.R.,Qi, L. Management Sci.,34,1988
- Birge, J.R., Holmes, D.F., Comput.Optim.Appl.,1,1992
  - direct solver
  - $(ADA^T)^{-1}$ Sherman-Woodbury-Morrison formula
  - decomposition of the system to smaller systems
  - almost independent calculations suited for parallel execution
  - sparse matrix multiplication
  - about 0,001% nonzero elements
  - 300x300 the system (ADA^T) has 451 056 unknowns

$$R^{(3)} = A^{(3)} D^{(3)} (A^{(3)})^t$$

$$R^{(3)} = \mathrm{R}^{(3)} + U^{(3)} \mathrm{D}^{(3)} (V^{(3)})^t$$

$$\mathrm{R}^{(3)} = Diag(I_{m_0}, A_1^{(2)} D_1^{(2)} \left(A_1^{(2)}\right)^t, A_1^{(2)} D_1^{(2)} \left(A_1^{(2)}\right)^t) =$$

$$Diag(I_{m_0}, R_1^{(2)}, R_2^{(2)})$$

$$U^{(3)} D^{(3)} (V^{(3)})^t = \begin{pmatrix} A_0 I_{m_0} \\ T_1^{(3)} \\ T_2^{(3)} \end{pmatrix} \begin{pmatrix} D_0 & \\ & I_{m_0} \end{pmatrix} \begin{pmatrix} A_0^t (T_1^{(3)})^t (T_2^{(3)})^t \\ -I_{m_0} \end{pmatrix}$$

Sherman – Morrison – Woodbury - formula

$$(R^{(3)})^{-1} = (\mathrm{R}^{(3)})^{-1} - (\mathrm{R}^{(3)})^{-1} U^{(3)} (G^{(3)})^{-1} (V^{(3)})^{t} (\mathrm{R}^{(3)})^{-1}$$

it holds if and only if $\mathrm{R}^{(3)}$ and $G^{(3)}$ are nonsingular

$$\mathrm{G}^{(3)} = \begin{pmatrix} D_0^{-1} + A_0^t A_0 + \sum_{i=1}^{2} \left(T_i^{(3)}\right)^t \left(R_i^{(2)}\right)^{-1} T_i^{(3)} & A_0^t \\ -A_0 & 0 \end{pmatrix} = \begin{pmatrix} G^{(3)} & A_0^t \\ -A_0 & 0 \end{pmatrix}$$

the solution $\quad R^{(3)} \Delta y^{(3)} = b^{(3)}$

$$\Delta y^{(3)} = p^{(3)} - s^{(3)}$$

$$\mathrm{R}^{(3)} p^{(3)} = b^{(3)}$$

$$G^{(3)} q^{(3)} = (V^{(3)})^t p^{(3)}$$

$$\mathrm{R}^{(3)} s^{(3)} = U^{(3)} q^{(3)}$$

Three-stage procedure

1. solve $(A_k^{(2)} D_k^{(2)} \; (A_k^{(2)})^{\dagger}; \; b_k^{(2)})$
2. compute matrix $G^{(3)}$ - dependent on all $T_k^{(3)} p_k^{(2)}$
3. solve $(A_k^{(2)} D_k^{(2)} \; (A_k^{(2)})^{\dagger}; \; T_k^{(3)} q_1^{(3)}))$
4. calculate $\Delta y_k^{(3)} = p_k^{(2)} - s_k^{(2)}$

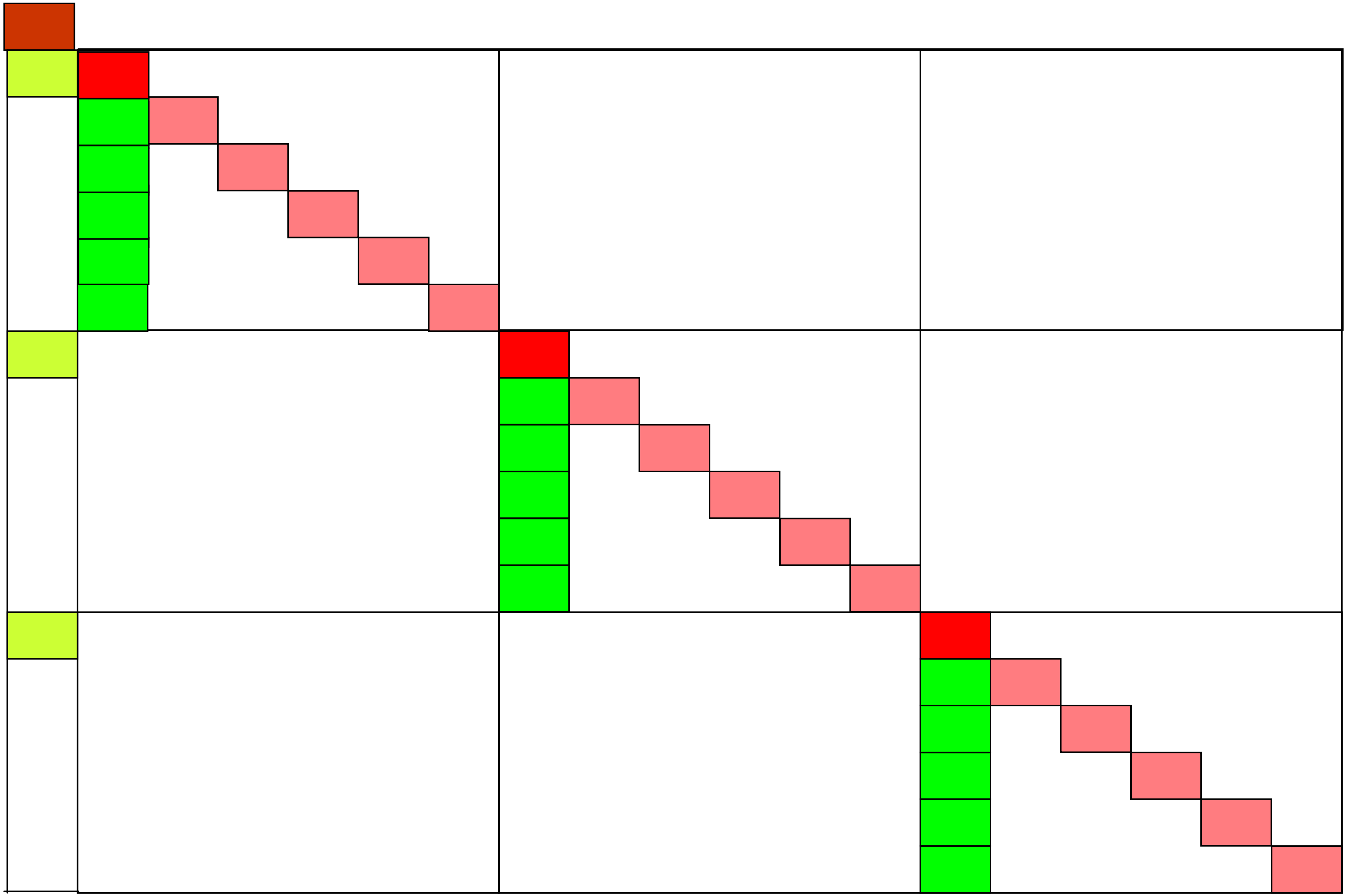
$k=1,2,...N^{(3)}$

Two-stage procedure

1. solve $(A_j^{(1)} D_j^{(1)} (A_j^{(1)})^{\dagger}; \; b_j^{(1)})$
2. compute matrix $G^{(2)}$ - dependence on all $T_j^{(2)} p_j^{(1)}$
3. solve $(A_j^{(1)} D_j^{(1)} (A_j^{(1)})^{\dagger}; \; T_j^{(1)} q_j^{(1)}))$
4. calculate $\Delta y_j^{(2)} = p_j^{(1)}) - s_j^{(1)}$ $j=1,2,...M$

MPI Parallelization with LAPACK

- NP = PxQ virtual processor array
- P processors processed in parallel large block rows
- Q processors processed in parallel two-stage problems
- mostly independent calculations in both levels
- two steps with collective gathering in every two-stage problem and every iteration
- two steps with collective gathering in three-stage problem and every iteration



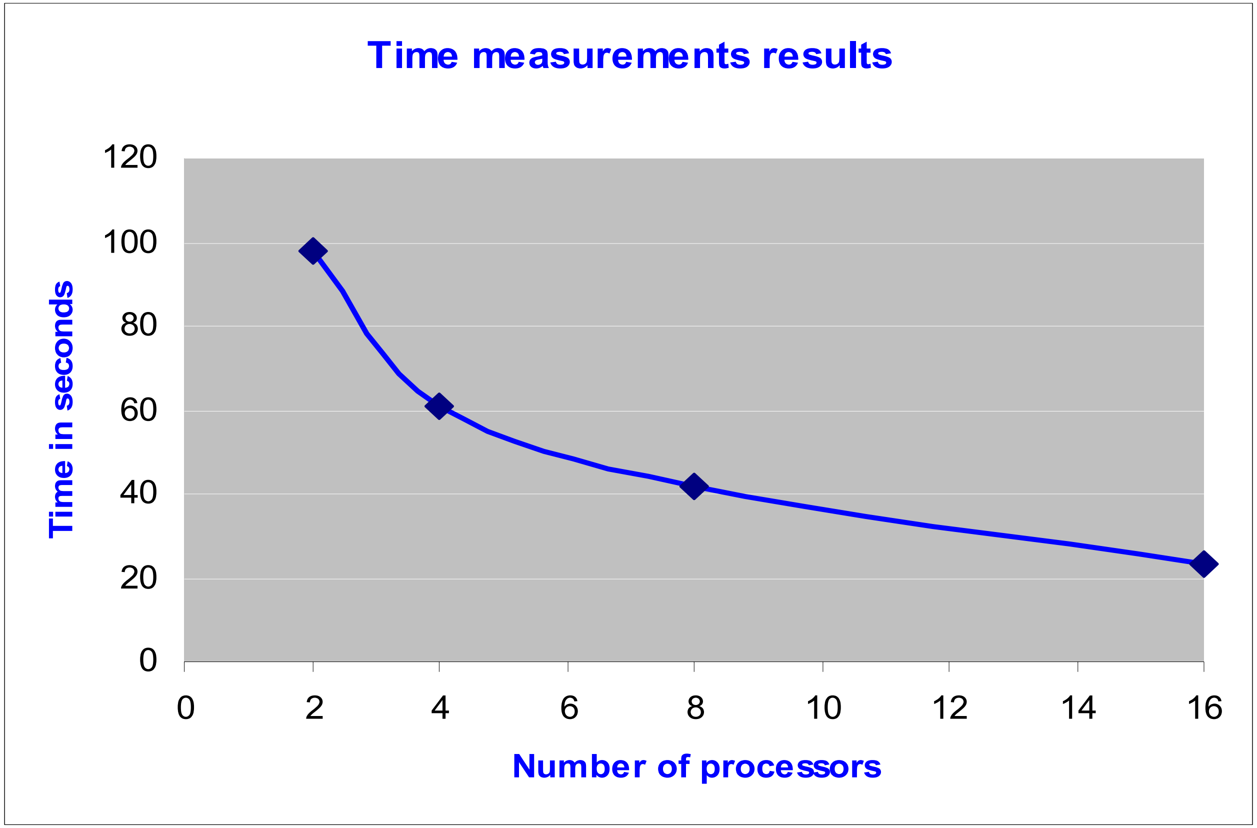
Hardware Facilities

- IBM 1350 cluster (aurora.tuwien.ac.at)
- 72 IBM x335 nodes
- 144 Pentium IV Nocona 3.6 Ghz processors (2 cpus per node)
- Linux operating system (Redhat)
- infiniband low latency node interconnect

Parallel performance

- the size of $A^{(3)}$ is 451 505 x 1 083 612
- scenarios: 300 - 300
- ADA^T 451 505 unknowns
- 2 167 220 nonzero 203 856 765 025 $A^{(3)}$
- one day modeling

NP	PxQ								
	1x2	2x1	1x4	2x2	4x1	1x8	2x4	4x2	8x1
2	171.7	97.67							
4			119.3	73.33	60.92				
8						95.58	71.42	48.33	41.74
	1x16	2x8	4x4	8x2	16x1				
16	71.92	52.4	41.17	36.4	23.4				


Time measurements results
120
100
80
60
40
20
0
Time in seconds
0
2
4
6
8
10
12
14
16
Number of processors

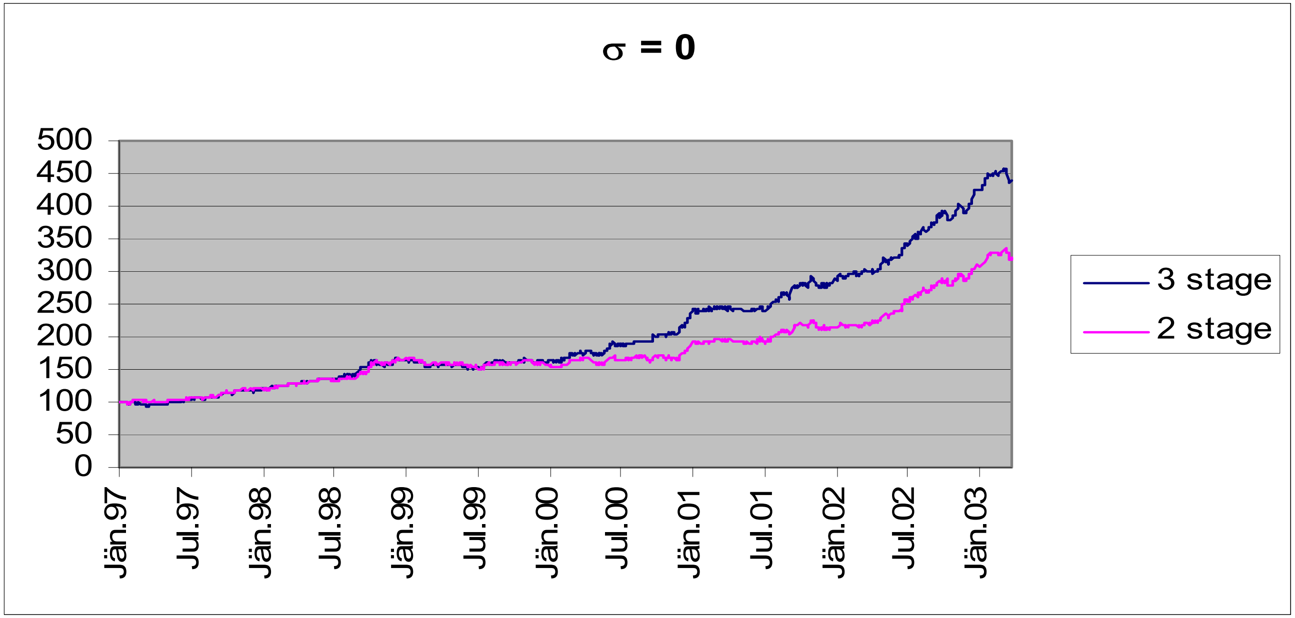
Financial performance

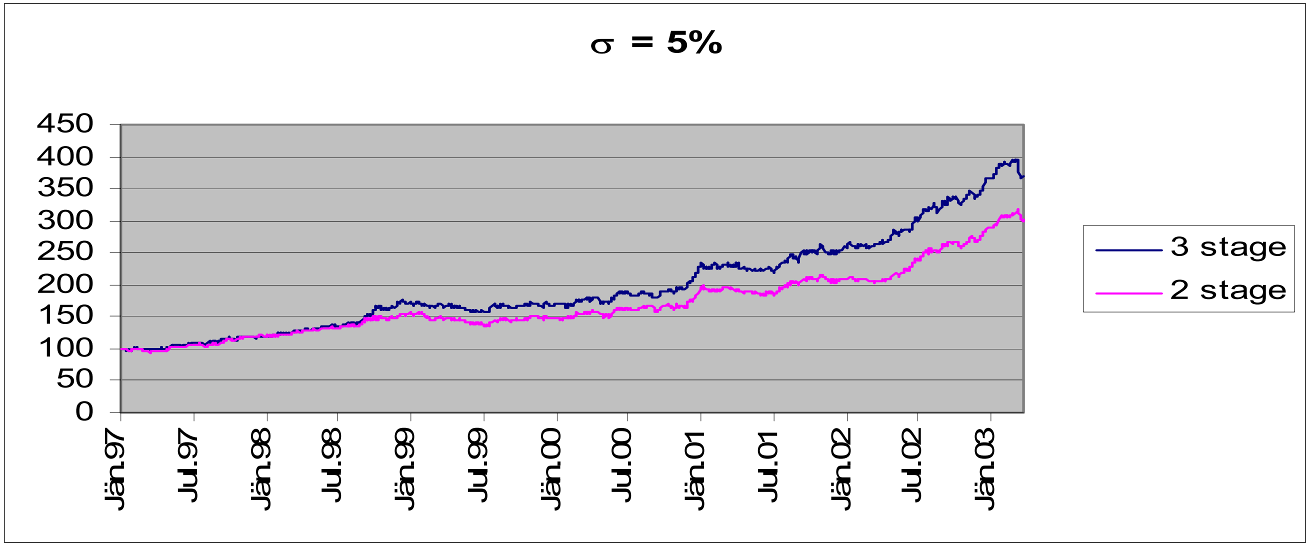
- Tests on historical data January 1997- January 2003
- Bond indices of 10 years government bonds USD, EUR, CHF, GBP
- Domestic currency was USD
- 300 scenarios per stage
- Δt = 6 weeks, it means 2 weeks for the 3-stage and 3 weeks for the 2 stage model
- Expected prices were real future returns with normal perturbation

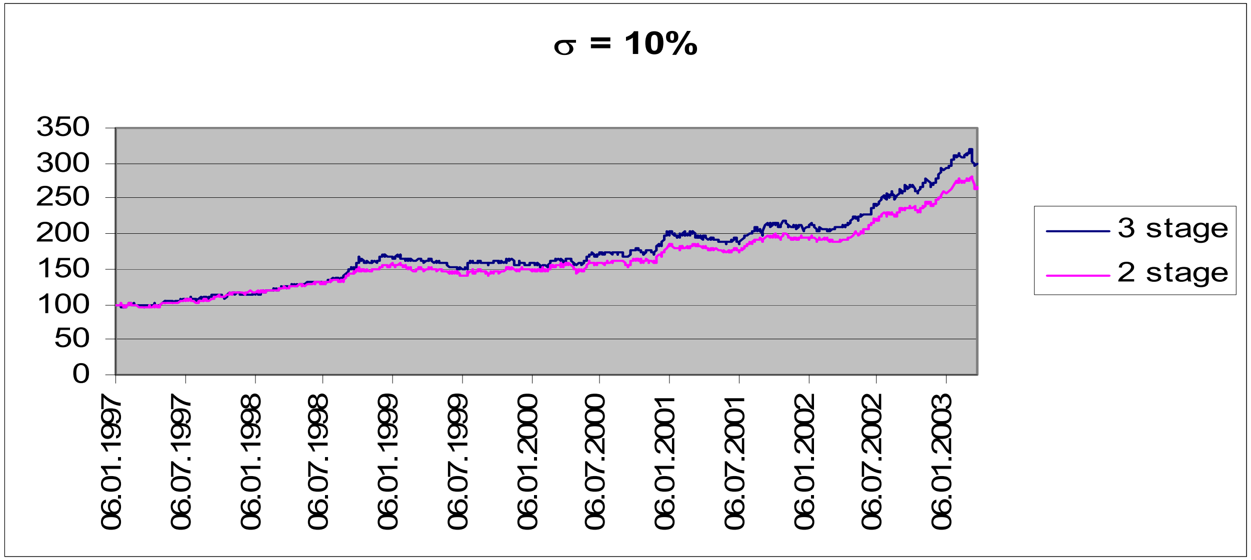
$$\frac{P_j^{(2)}}{I_j^{(1)}} = \frac{I_j^{(2)}}{I_j^{(1)}} + \sigma_e Z_{1,j}, \quad \frac{P_j^{(3)}}{P_j^{(2)}} = \frac{I_j^{(3)}}{I_j^{(2)}} + \sigma_e Z_{2,j}, \quad \frac{P_j^{(4)}}{P_j^{(3)}} = \frac{I_j^{(4)}}{I_j^{(3)}} + \sigma_e Z_{3,j}$$

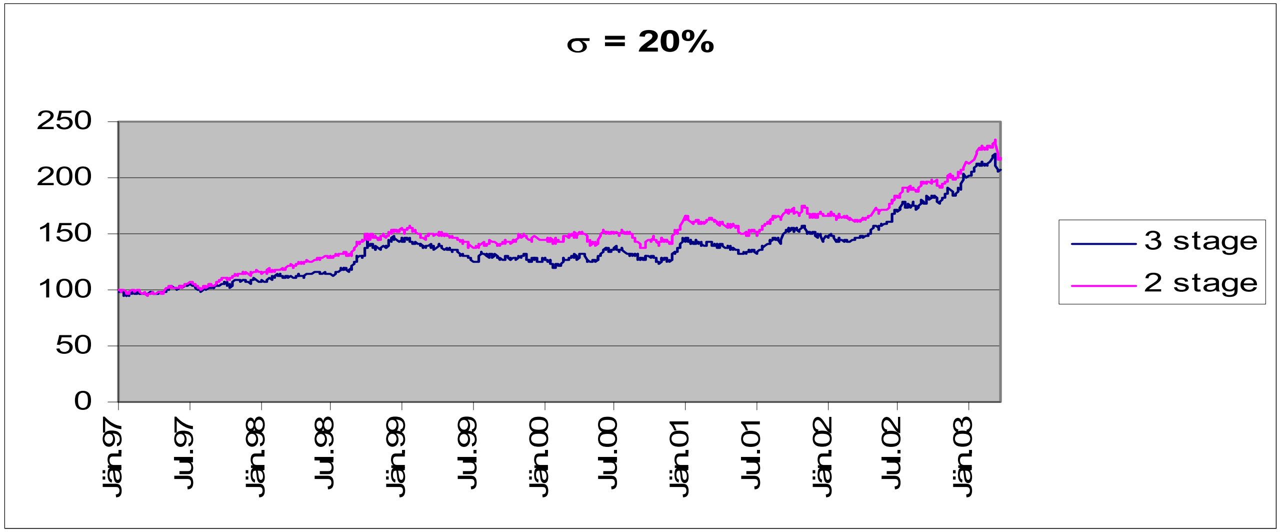
σ_e standard deviation of the error

$$\sigma_e = 0, \quad \sigma_e = 5\%, \quad \sigma_e = 10\%, \quad \sigma_e = 20\%$$


σ = 0
500
450
400
350
300
250
200
150
100
50
0
Jän.97
Jul.97
Jän.98
Jul.98
Jän.99
Jul.99
Jän.00
Jul.00
Jän.01
Jul.01
Jän.02
Jul.02
Jän.03
3 stage
2 stage


σ = 5%
450
400
350
300
250
200
150
100
50
0
Jän.97
Jul.97
Jän.98
Jul.98
Jän.99
Jul.99
Jän.00
Jul.00
Jän.01
Jul.01
Jän.02
Jul.02
Jän.03
3 stage
2 stage


σ = 10%
350
300
250
200
150
100
50
0
06.01.1997
06.07.1997
06.01.1998
06.07.1998
06.01.1999
06.07.1999
06.01.2000
06.07.2000
06.01.2001
06.07.2001
06.01.2002
06.07.2002
06.01.2003
3 stage
2 stage


σ = 20%
250
200
150
100
50
0
Jän.97
Jul.97
Jän.98
Jul.98
Jän.99
Jul.99
Jän.00
Jul.00
Jän.01
Jul.01
Jän.02
Jul.02
Jän.03
3 stage
2 stage

Conclusion remarks

- Does it make sense to increase number of stages?

  YES , IF

  we work with "good" information
- increasing the number of stages need not to improve the quality of the solution – it depends of the quality of information
- stability issues - preconditioning
- comparison with sparse solvers