Tensor-Based Computing in Contract Theory, IO, and HJB PDE Macro Models

Victor V. Zhorin

Computation Institute

April 16, 2015

BECKER FRIEDMAN INSTITUTE
FOR RESEARCH IN ECONOMICS
THE UNIVERSITY OF CHICAGO

Based on

- Edward C. Prescott and Robert Townsend, "Pareto Optima and Competitive Equilibria with Adverse Selection and Moral Hazard", Econometrica, 1984
- Alexander Karaivanov and Robert Townsend, "Dynamic Financial Constraints: Distinguishing Mechanism Design from Exogenously Incomplete Regimes", Econometrica, 2014
- Robert Townsend and Victor Zhorin, "Spatial Competition among Financial Service Providers and Optimal Contract Design", 2014
- Benjamin Moll (`http://www.princeton.edu/~moll/HACTproject.htm`): "Heterogeneous Agent Models in Continuous Time" (with Yves Achdou, Jean-Michel Lasry and Pierre-Louis Lions)
- Wolfgang Hackbusch, "Tensor Spaces and Numerical Tensor Calculus", 2012

Key points

- Computers are well-designed and heavily optimized to handle vector- and matrix-based calculations for medium- and large-size problems
- Numerical linear algebra libraries are well-developed and fast
- Tensor-related (multi-dimensional, multivariate) problems are common, highly complex and difficult (often considered impossible/infeasible/intractable) to compute
- Very large-size data intensive processing is typically required, scalability is an issue
- Focus on numerical tensor analysis techniques that are readily available and generally applicable
- The goal is to enable efficient high-performance computations for economic models explicitly formulated in tensor format

Tensor-related problems in economics

- Multivariate statistics and multi-dimensional structural estimations
- Prescott-Townsend linear programming approach for information-constrained non-linear contract models
- Industrial Organization with heterogeneous types and multi-dimensional characteristics
- Stochastic Partial Differential Equation (SPDE) models in macroeconomics and finance
- Also: Hidden Markov Models, big data analysis, belief propagation, **global non-convex optimization problems**, approximation and interpolation of multivariate functions

Speedup: Algorithms vs. Machine improvement

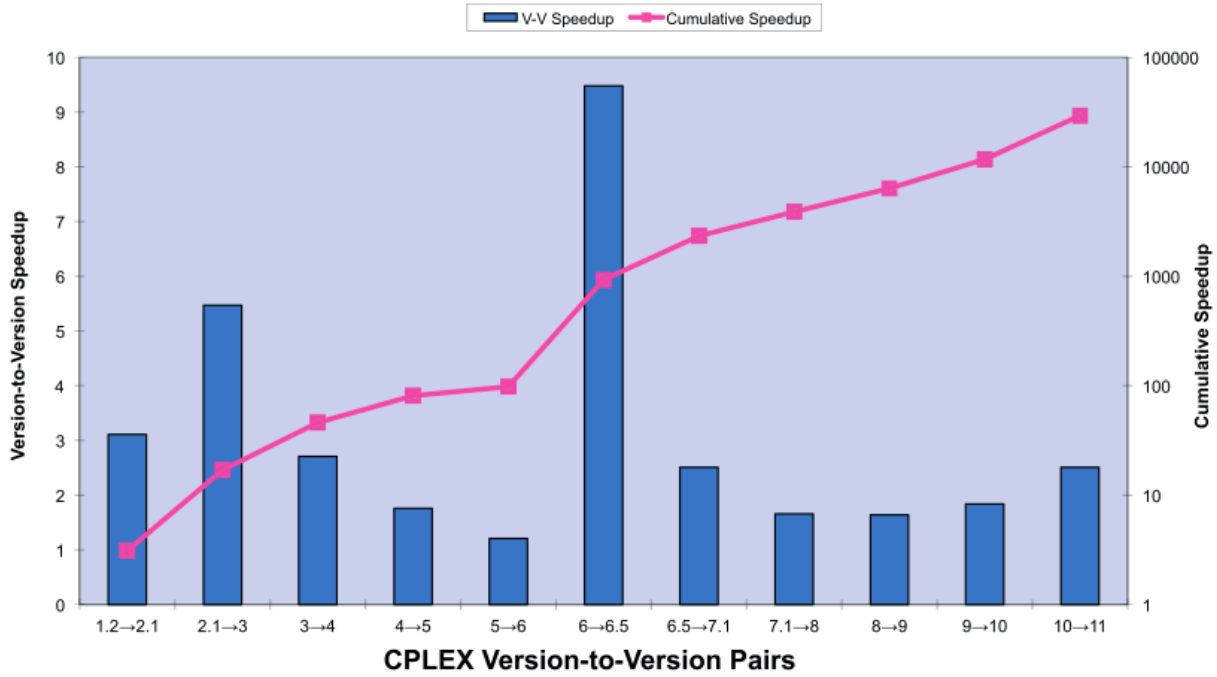
R. E. Bixby, "A Brief History of Linear and Mixed-Integer Programming Computation", Documenta Math., 2012

CPLEX LP code from 1988 through 2002

- **Algorithmic improvement** (machine independent)
  Best of barrier, primal simplex, and dual simplex: 3300x
- **Machine improvement**: 1600x

  Total: 5,280,000x

5 months of computing $\rightarrow \approx$ 1.1 hour (due to better algorithms) $\rightarrow \approx$ 2.5 seconds (machine+algorithms)



Parallel Programming:

same task done faster or more complex task done in feasible time

SAXPY, single-precision real Alpha X Plus Y (BLAS, level 1):

$$Y \leftarrow \alpha * X + Y$$

where $X_i, Y_i, i \in [1, n]$ - vectors

- Instruction (control) parallelism, **strong scaling**
  - Scalar uniprocessor - $2n$ steps
  - Two functional units (an adder and a multiplier) - $n + 1$ steps, speedup $\frac{2n}{n+1} \approx 2$
  - Amdahl's law:
    If s is a fraction of code that is executed serially then speedup from parallelizing $p = 1 - s$ fraction using N processors:

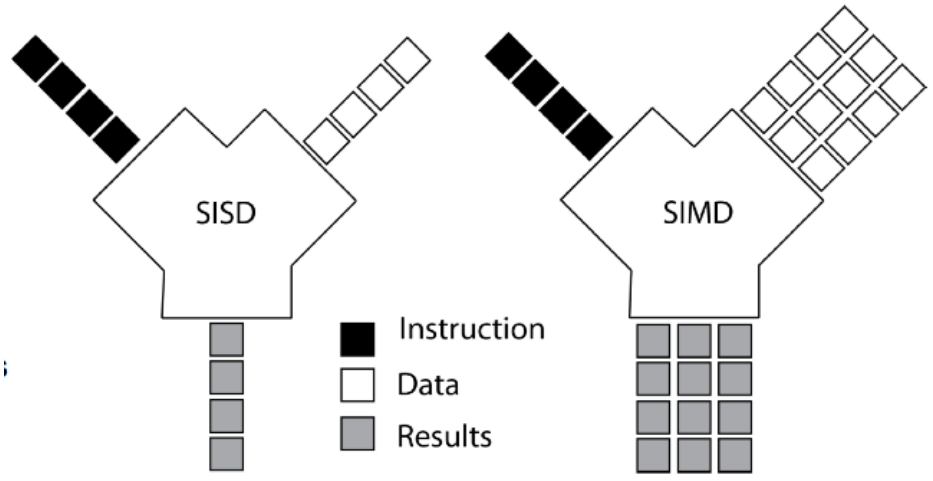
$$Speedup \leq \frac{1}{s + p/N}$$

- Gustafson-Barsis' law, **weak scaling**:

$$Speedup = s + N * (1 - s)$$

- **Data parallelism**: two steps with N processors handling $\alpha * X$ and Y data slices simultaneously, speedup is proportional to $N < n$

Parallel Programming: types of parallel computing models

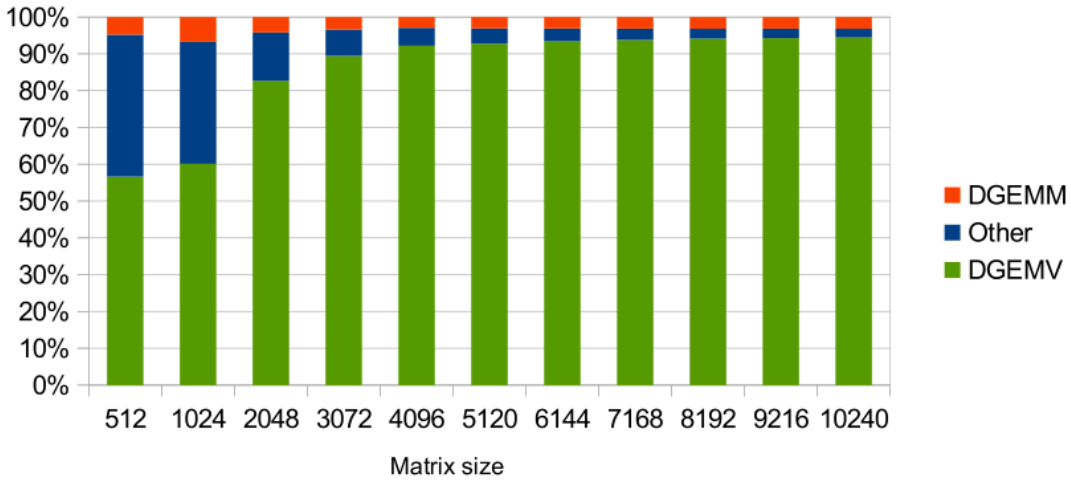


- Data parallel - the same instructions are carried out simultaneously on multiple data items (SIMD)
- Task parallel - different instructions on different data (MIMD)
- MIMD: Message passing (MPI) - overlapping computation and communication (!) , MATLAB Distributed Computing Server with Parallel Computing Toolbox
- SIMD: Array Programming (implicit parallelization), NumPy, High Performance Fortran, **Vectorization** (and **Tensorization**) in Matlab
- Task/data parallel paradigms : OpenMP, Fortran 2008 DO CONCURRENT
- Hybrid Programming: CPU-GPU, Intel Phi MIC architecture, SIMD$\rightarrow$OpenMP$\rightarrow$MPI

Basic Linear Algebra Subroutines

Row-major order storage: C/C++, Mathematica, Python, SAS
Column-major order storage: Fortran, MATLAB, R, Julia
Stride: distance in memory between two adjacent elements of a vector or a matrix

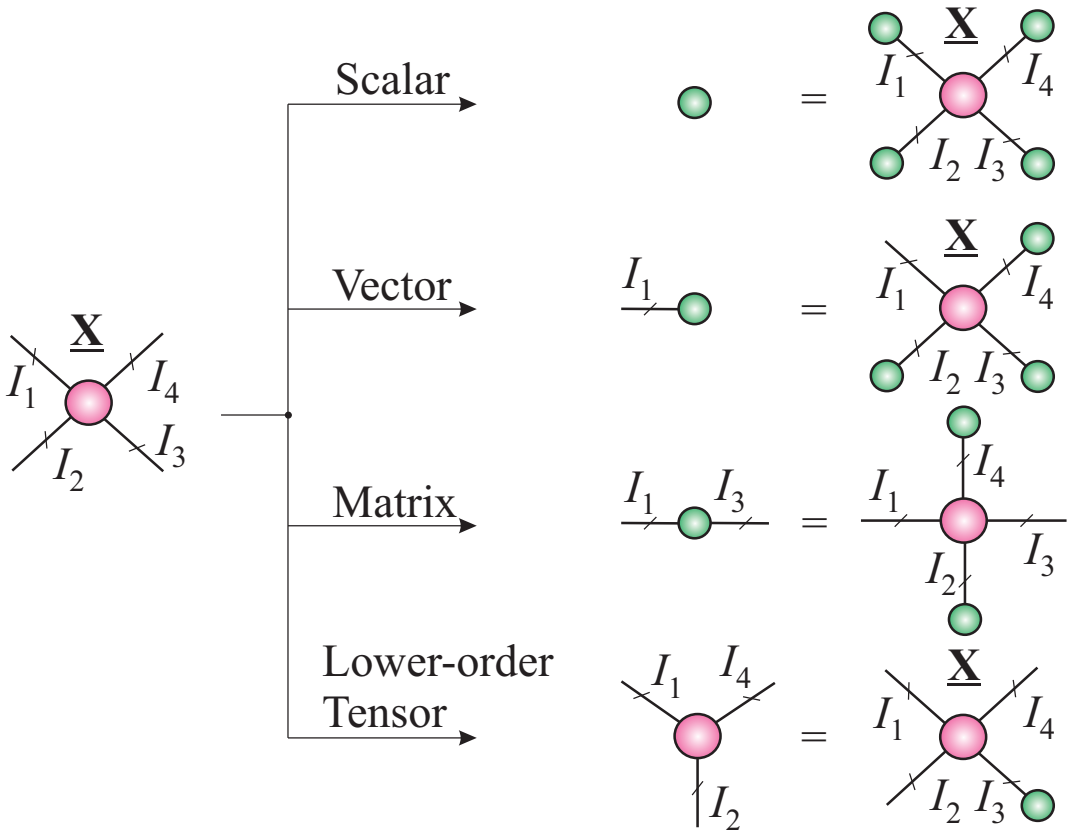
- BLAS level 1: vector-vector multiplication, $\alpha \leftarrow X' * Y$, DOT, (LINPACK '70s)
- BLAS level 2: vector-matrix multiplication, $Y \leftarrow \alpha A * X + \beta Y$, GEMV
- BLAS level 3: matrix-matrix multiplication, $C \leftarrow \alpha A * B + \beta C$, GEMM, (LAPACK 80's)
- parallel BLAS: distributed, heterogeneous, hybrid algorithms, block matrix-level
- multidimensional array BLAS (?)



Execution time of QR algorithm in percentage of Intel MKL routines on a 12 core machine

Vectors, matrices and tensors

- A scalar is n order-0 tensor: $\mathcal{X} = (x_0) \in \mathbb{R}$
- A vector is an order-1 tensor: $\mathcal{X} = (x_i) \in \mathbb{R}^n$
- A matrix is an order-2 tensor: $\mathcal{X} = (x_{i_1 i_2}) \in \mathbb{R}^{n_1 \times n_2}$
- kth-order tensor: $\mathcal{X} = (x_{i_1 \ldots i_k}) \in \mathbb{R}^{n_1 \times \ldots \times n_k}$

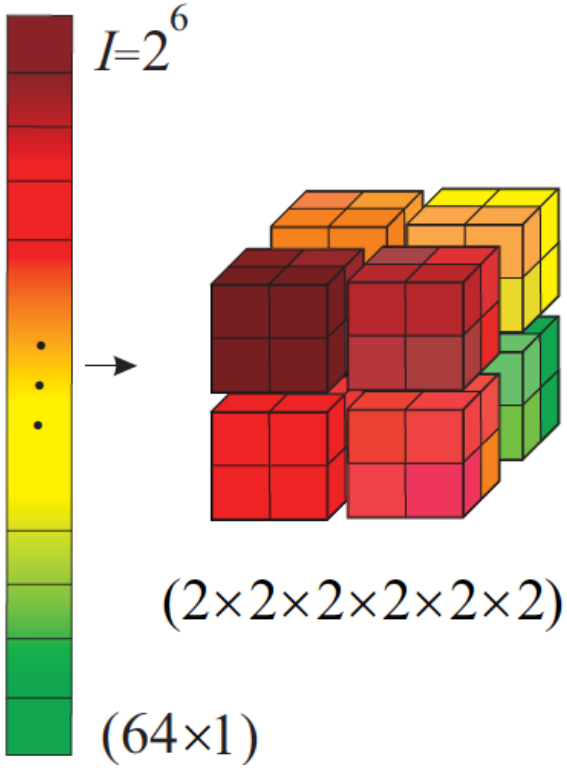


Vectorization of a tensor/tensorization of a vector

Vectorization (stacking):

$$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \Rightarrow vec(X) \in R^{n_1 n_2 n_3}; vec(X) = \begin{bmatrix} x_{111} \\ x_{211} \\ x_{121} \\ x_{221} \\ x_{112} \\ x_{212} \\ x_{122} \\ x_{222} \end{bmatrix}$$

Tensorisation:



Basic Tensor Operations: contraction

Tensor contraction

Summation process applied to tensors - a generalization of vector-matrix and matrix-matrix multiplication

$$\mathcal{C}(i,j,k,m) = \sum_{l} \mathcal{A}(i,l,k) * \mathcal{B}(l,j,m)$$

$\text{Order}(\mathcal{C}) = \text{Order}(\mathcal{A}) + \text{Order}(\mathcal{B}) - 2$

```
rand('seed', 1);
a = rand(2,3,4);
b = rand(3,5,6);
c = zeros(2,5,4,6);
for m =1:6
  for k=1:4
        c(:,:,k,m) = a(:,:,k)*b(:,:,m);
  end
end
a(2,1,3)*b(1,4,5)+a(2,2,3)*b(2,4,5)+a(2,3,3)*b(3,4,5)
c(2,4,3,5)
size(c)
2     5     4     6
```

Basic Tensor Operations: Kronecker product

Kronecker product

Creates tensors whose elements are tensors - a generalization of the outer vector product, can be used to make block vectors, block matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes B = \left[\begin{array}{c|c} a_{11}B & a_{12}B \\ \hline a_{21}B & a_{22}B \end{array}\right]$$

Continuous-time Sylvester linear matrix equation (Lyapunov, Ricatti etc):

$$AX + XB = C, A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{m \times m}, C \in \mathbb{R}^{n \times m},$$

Using $vec(AXB) = (B^\top \otimes A)vec(X)$ the Sylvester equation can be rewritten as a standard linear program:

$$\mathcal{A}x = c, x = vec(X), c = vec(C), \mathcal{A} = \mathcal{I}_m \otimes A + B^\top \otimes \mathcal{I}_n$$

Unobserved action with observed stochastic output

Action-Output

$$\mathcal{A} \otimes \mathcal{Q} : \{a_1, ...a_{n_a}\} \otimes \{q_1, ...q_{n_q}\}$$

Stochastic Production Function $p(q|a)$

Compensation Schedule

$$\mathcal{C}(\mathcal{Q}) = \{c(q_1), ..., c(q_{n_q})\}$$

Expected utility for the agents

$$\omega(c, a) = \sum_{q \in \mathcal{Q}} p(q|a) u(c(q), a)$$

The non-linear profit-optimal mechanism design problem:

$$\max_{c(q), a} \sum_{q \in \mathcal{Q}} p(q|a) \left[q - c(q)\right]$$

s.t.

Incentive Compatibility Constraints (ICC):

$$\omega(c, a) \geq \omega(c, \overline{a}), \forall\ \overline{a} \in \mathcal{A}$$

Utility Assignment Constraint (UAC):

$$\omega(c, a) = \overline{\omega}$$

Linear MH Programs: Prescott-Townsend lottery

Choice: $\pi(c, q, a) = \pi(c|q, a)P(q|a)\pi(a)$ - **probability distribution**, fraction of agents to receive a particular allocation of $\{c, q, a\}$

The linear program for the profit optimal contract:

$$\max_{\pi(c,q,a)} \left[\sum_{\mathcal{C},\mathcal{Q},\mathcal{A}} \pi(c, q, a)[q - c] \right]$$

s.t.

Mother Nature/Technology Constraints:
$\forall \{\overline{q}, \overline{a}\} \in \mathcal{Q} \times \mathcal{A}$

$\sum_{\mathcal{C}} \pi(c, \overline{q}, \overline{a}) = P(\overline{q}|\overline{a}) \sum_{\mathcal{C},\mathcal{Q}} \pi(c, q, \overline{a})$

Incentive Compatibility Constraints (ICC) for action variables:
$\forall\ a, \hat{a} \in \mathcal{A} \times \mathcal{A}$

$\sum_{\mathcal{C},\mathcal{Q}} \pi(c, q, a)\, u(c, a) \geq \sum_{\mathcal{C},\mathcal{Q}} \pi(c, q, a) \frac{P(q,\hat{a})}{P(q,a)} u(c, \hat{a})$

Utility Assignment Constraint:

$\sum_{\mathcal{Q},\mathcal{C}} \pi(c, q, a)\, u(c, a) = \overline{\omega}$

Probability Measure Constraints:

$\sum_{\mathcal{Q},\mathcal{C},\mathcal{A}} \pi(c, q, a) = 1; 0 \leq \pi(c, q, a) \leq 1, \forall \{c, q, a\} \in \mathcal{C} \times \mathcal{Q} \times \mathcal{A}$

Lotteries: tensor vectorization, contraction, tensor products, Karaivanov, 2001

$\mathcal{C} \in \{c_1, c_2\}, \mathcal{Q} \in \{q_1, q_2\}, \mathcal{A} \in \{a_1, a_2\}$

Probability distribution $\pi(c, q, a)$

Technology constraints:

$$\begin{cases} \pi(c_1,q_1,a_1)+\pi(c_2,q_1,a_1) = p(q_1,a_1)*(\pi(c_1,q_1,a_1)+\pi(c_2,q_1,a_1)+\pi(c_1,q_2,a_1)+\pi(c_2,q_2,a_1)) \\ \pi(c_1,q_2,a_1)+\pi(c_2,q_2,a_1) = p(q_2,a_1)*(\pi(c_1,q_1,a_1)+\pi(c_2,q_1,a_1)+\pi(c_1,q_2,a_1)+\pi(c_2,q_2,a_1)) \\ \pi(c_1,q_1,a_2)+\pi(c_2,q_1,a_2) = p(q_1,a_2)*(\pi(c_1,q_1,a_2)+\pi(c_2,q_1,a_2)+\pi(c_1,q_2,a_2)+\pi(c_2,q_2,a_2)) \\ \pi(c_1,q_2,a_2)+\pi(c_2,q_2,a_2) = p(q_2,a_2)*(\pi(c_1,q_1,a_2)+\pi(c_2,q_1,a_2)+\pi(c_1,q_2,a_2)+\pi(c_2,q_2,a_2)) \end{cases}$$

$vec(\pi(c,q,a)) \rightarrow \Pi_{n_c*n_q*n_a \times 1}$, $\text{vec}(p(q,a)) \rightarrow P_{n_q*n_a \times 1}$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} \pi_{111} \\ \pi_{211} \\ \pi_{121} \\ \pi_{221} \\ \pi_{112} \\ \pi_{212} \\ \pi_{122} \\ \pi_{222} \end{bmatrix} = \begin{bmatrix} P_1 & P_1 & P_1 & P_1 & 0 & 0 & 0 & 0 \\ P_2 & P_2 & P_2 & P_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_3 & P_3 & P_3 & P_3 \\ 0 & 0 & 0 & 0 & P_4 & P_4 & P_4 & P_4 \end{bmatrix} * \begin{bmatrix} \pi_{111} \\ \pi_{211} \\ \pi_{121} \\ \pi_{221} \\ \pi_{112} \\ \pi_{212} \\ \pi_{122} \\ \pi_{222} \end{bmatrix}$$

$$\left[J_{n_a \times n_q} \otimes I_{1 \times n_c}\right] * \Pi_{n_c*n_q*n_a \times 1} = \left[\left((I_{n_a \times n_a} \otimes J_{n_q \times 1}) \circ (P_{n_q*n_a \times 1} * I_{1 \times n_a})\right) \otimes I_{1 \times n_c*n_q}\right] * \Pi_{n_c*n_q*n_a \times 1}$$

Building block - standard mechanism design problem

from "Spatial Competition among Financial Service Providers and Optimal Contract Design", R. Townsend and V. Zhorin (2014)

The optimal contract to maximize the bank surplus extracted from each agent:

$$S\{\overline{\omega(\theta)}\} : = \max_{\pi(c,q,k,a|\theta)} \left[\sum_{\theta} \sum_{c,q,k,a} \pi\left(c, q, k, a|\theta\right)[q - c - k] \right]$$

where $\pi(c, q, k, a|\theta)$ is a probability distribution over the vector (c, q, k, a) given the agent's type θ.

Mother Nature/Technology Constraints:
$\forall \left\{\overline{q}, \overline{k}, \overline{a}\right\} \in Q \times K \times A$ and $\forall\ \theta \in \Theta$

$$\sum_{c} \pi(c, \overline{q}, \overline{k}, \overline{a}|\theta) = P\left(\overline{q}|\overline{k}, \overline{a}, \theta\right) \sum_{c,q} \pi(c, q, \overline{k}, \overline{a}|\theta)$$

Incentive Compatibility Constraints for action variables:
$\forall\ a, \hat{a} \in A \times A$ and $\forall\ k \in K$ and $\forall\ \theta \in \Theta$:

$$\sum_{c,q} \pi\left(c, q, k, a|\theta\right) u\left(c, a|\theta\right) \geq \sum_{c,q} \pi\left(c, q, k, a|\theta\right) \frac{P(q, |k, \hat{a}, \theta)}{P(q, |k, a, \theta)} u(c, \hat{a}|\theta)$$

Truth-Telling Conditions in Adverse Selection - type θ must not announce type θ' (can add to unobserved a and k):
$\forall\ \theta, \theta' \in \Theta$

$$\sum_{c,q,k,a} \pi(c, q, k, a|\theta) u(c, a|\theta) \geq \sum_{c,q,k,a} \left[\pi\left(c, q, k, a|\theta'\right) \frac{P(q, |k, a, \theta)}{P(q, |k, a, \theta')} u(c, a|\theta') \right]$$

Computing Lottery Programs: tensor vectorization

- Discretization: $\mathcal{C}$, $\mathcal{Q}$, and $\mathcal{A}$ are finite **ordered** sets.
- Tensor product $\mathcal{C} \otimes \mathcal{Q} \otimes \mathcal{A} \to \pi$
- MATLAB Kronecker tensor product: $KRON(X, Y) = X \otimes Y$

```
1  grc = linspace(0.,4,300); %consumption vector
2  grq =[1 4];%output vector
3  gra = linspace(0.,1,300); %effort vector
4  nc= length(grc);nq=length(grq); na=length(gra);
5  %dimension of lottery vector
6  N=na*nq*nc; %=180,000
7
8  %production technology
9  P(2:2:na*nq) = gra;
10 P(1:2:na*nq-1) = 1-gra;
11
12 %tensorization/vectorization
13 C = kron(ones(1,na*nq),grc);
14 Q = kron(kron(ones(1,na),grq),ones(1,nc));
15 A = kron(gra,ones(1,nc*nq));
16
17 %adding up to one for total probability
18 Aeq_1 =  ones(1,N); beq_1 = 1;
19 %set utility offer
20 omega = 2.3; sig = .5; gam = 2;
21 Aeq_ut = C.^(1-sig)/(1-sig) - A.^gam;
22 beq_ut = omega;
23 %objective function
24 Obj = Q - C;
```

Computing Lottery Programs: tensor contraction and decomposition

$$\Big[J_{n_a \times n_q} \otimes I_{1 \times n_c} - \Big(I_{n_a \times n_a} \otimes J_{n_q \times 1} \circ \big(P_{n_q * n_a \times 1} * I_{1 \times n_a}\big)\Big) \otimes I_{1 \times n_c * n_q}\Big] * \Pi_{n_c * n_q * n_a \times 1} = 0$$

```
1 %technology constraints
2 Aeq_mn = kron(eye(na*nq),ones(1,nc))- ...
3 kron(kron(eye(na),ones(nq,1)).*(P'*ones(1,na)),ones(1,nc*nq));
4 beq_mn = zeros(na*nq,1);
```

Computing Lottery Programs: calling LP solver

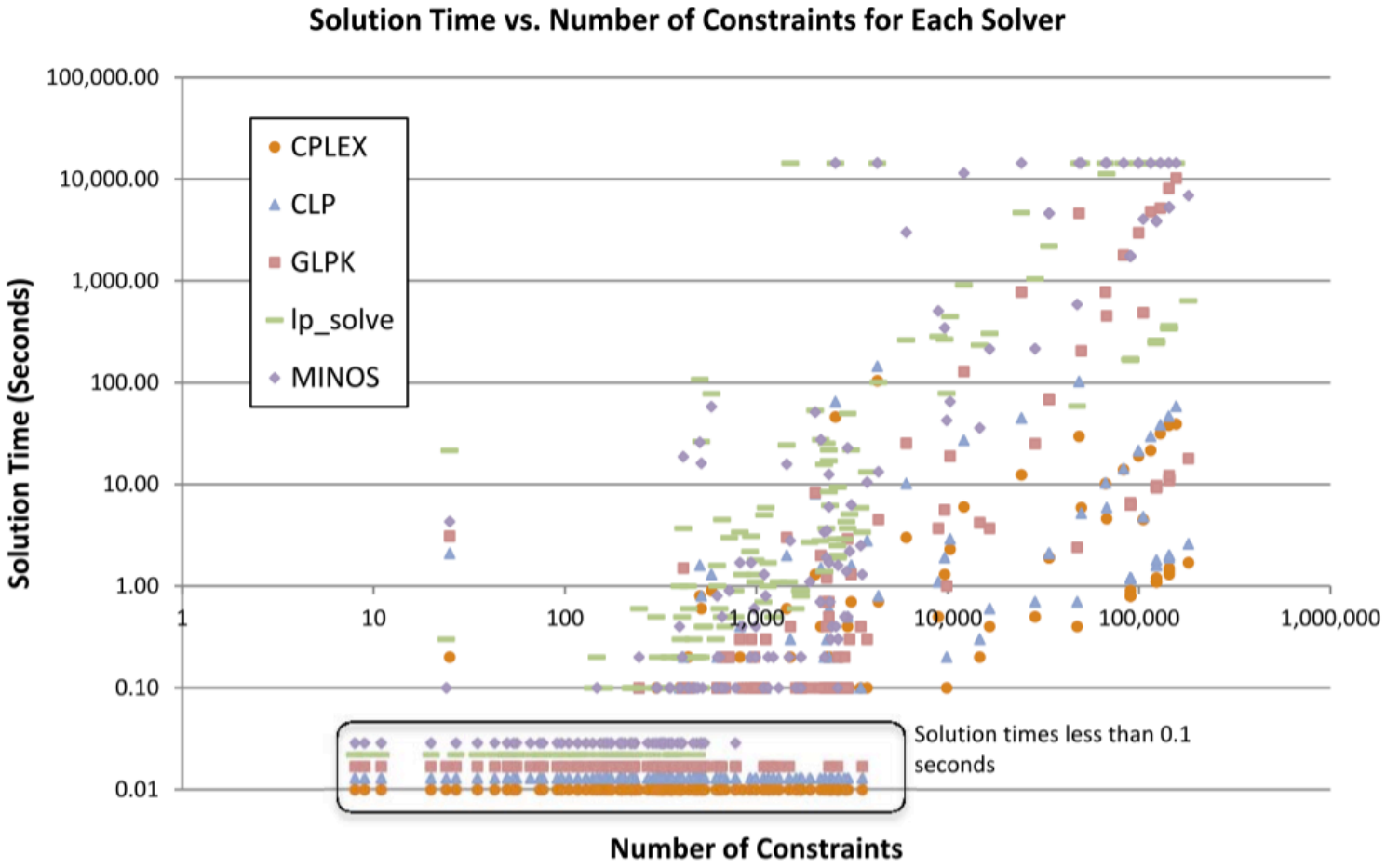
```
A_ineq = []; b_ineq = [];
Aeq = [Aeq_1;Aeq_ut;Aeq_mn;];
beq = [beq_1;beq_ut;beq_mn;];
%if use linprog
[x,fval,eflag,nn,la] = linprog(-Obj,A_ineq,b_ineq,Aeq,beq,zeros(N,1),ones(N,1));
Cp = C*x;  Qp = Q*x; Ap = A*x;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Number of variables: 180000
 Number of linear inequality constraints:    0
 Number of linear equality constraints:      602
 Number of lower bound constraints:          180000
 Number of upper bound constraints:          180000
Algorithm selected   large-scale: interior point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Qp  Cp Ap Qp-Cp]
3.8280    2.5418    0.9427    1.2862
%MATLAB linprog  on X5460 @ 3.16GHz, CPU Mark 4437
Elapsed time is 26 seconds.
%MATLAB linprog on E5-2670 @ 2.60GHz, CPU Mark 12849
Elapsed time is 10 seconds.
%GUROBI lp solver on X5460 @ 3.16GHz, CPU Mark 4437
Elapsed time is 2 seconds.
```

Utility: $\frac{c^{1-\sigma}}{1-\sigma} - a^{\theta} \Rightarrow 2.3000$

FOC check: $\theta a^{\theta-1} = c^{-\sigma}(\overline{q} - \underline{q}) \Rightarrow 1.8853 \approx 1.8817$

Linear Programming: solvers

- MATLAB linprog
- Open source: GLPK, lp_solve, CLP, SoPlex
- IBM CPLEX , XPRESS, Gurobi: interfaces to R, MATLAB, Python



"Comparison of Open-Source Linear Programming Solvers", Sandia Report, 2013

Endogenous utility distribution: a contract-based model of IO (Townsend-Zhorin, 2014)

Tensor-based contract competition

The agents are distributed uniformly in $\mathbb{R}^1 : [0; 1]$ with total market mass set to one. The household cost to access financial services is $\overline{L} * |x - x_i|$, x is location of the agent, x_i **is location of bank** i, $\overline{L}$ is a spatial cost or disutility from accepting a contract. The agents of type θ at location x choose to go to bank i if contract utility from bank i satisfies participation constraint and the real value offered is better than the one from bank i'

$$V_{diff} = u_i(\theta) - \overline{L} * |x - x_i| \geq u_{i'}(\theta) - \overline{L} * |x - x_{i'}| \geq \hat{u_0}(\theta)$$

where $\hat{u_0}(\theta)$ is autarky value.

- can restrict choice by finite number of potential locations (even easier) - but we want to know if unrestricted competition in space delivers interesting patterns
- spatially different agent's characteristics, all we need to do is to integrate over densities and we have that already built-in
- can do $\mathbb{R}^2$, put in roads just like on real maps

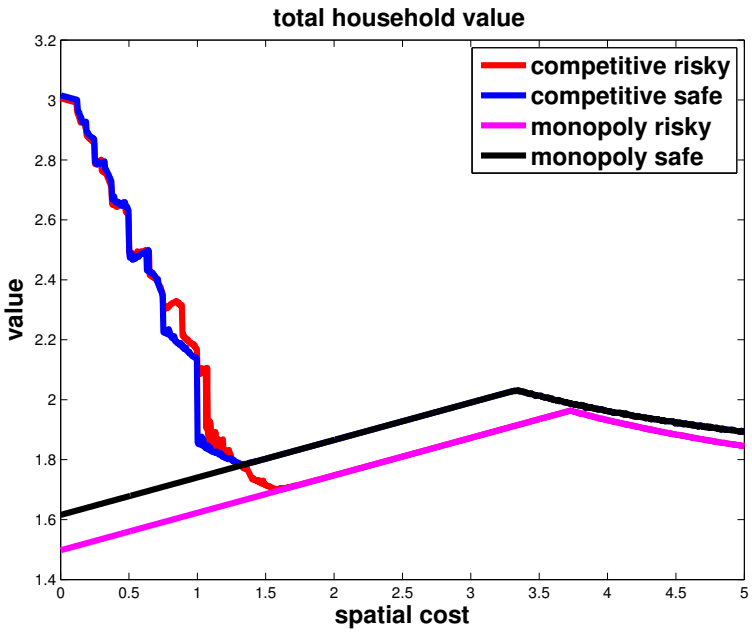
MARKET STRUCTURE

- collusion (two-branch monopoly)
- simultaneous Nash in contracts at fixed location (**no commitment**)
  - welfare implications of liberalization
- full commitment to location and simultaneous Nash on contracts (**partial commitment**)
- sequential Nash equilibrium (SNE) with **full commitment** to location and contract (business model)
  - local informed player vs outside entrant facing adverse selection

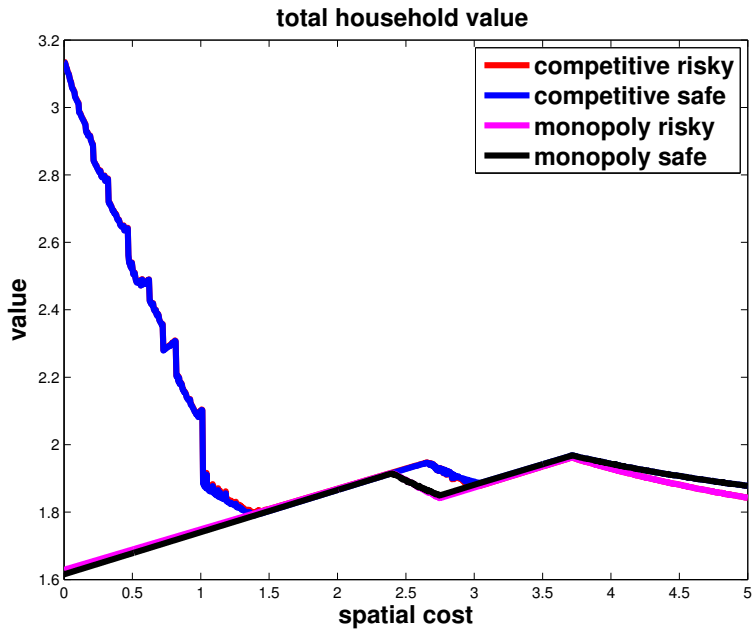
Welfare implications of financial liberalization (Townsend-Zhorin, 2014)

- real value computed for risky and safe households at all locations
- local two-branch monopoly at fixed locations [1/4; 3/4]
- at relatively low spatial costs the switch from monopoly to competition increases the household utility, but with some twists
- with full information the biggest gain is for the risky type
- with adverse selection it is much harder to distinguish across types, so the overall gain from liberalization/competition is similar for both types
- safe type gains more from liberalization in the adverse selection regime than in the full information regime
- at yet higher spatial costs there is no gain for either type

(a) Full Information



(b) Adverse Selection



HJB PDE for Robust Economic-Climate Models

- finite difference for HJB PDE: simple and standard
- technique is general and common in optimal robust control literature
- numerical global solution of robust stochastic climate-economic models in continuous time with optimal control
- allows to do robustness analysis over a range of different models with unknown drift distortions altering dynamics of stochastic processes for temperature, damages, productivity, climate sensitivity
- carbon-climate impact with multiplicative component of uncertainty for the evolution of temperature

Non-linear stochastic HJB PDE with robustness

$$W(X_1, ..., X_I)$$
$$= \max_{C_1,...,C_m} \min_{\Lambda_1,...,\Lambda_n} \mathbb{E}_0 \int_0^\infty e^{-\rho t} \left[u(C_1, ..., C_m) + \sum_{k=1}^{n} \frac{1}{2\theta_k} \Lambda_k^2 \right] dt$$

s.t. laws of motion for states $\iff$

$$0 = \max_{C_1,...,C_m} \min_{\Lambda_1,...,\Lambda_n} \left[u(C_1, ..., C_m) + \sum_{k=1}^{n} \frac{1}{2\theta_k} \Lambda_k^2 \right.$$
$$+ \sum_{i=1}^{I} \frac{\partial W(X_1, ..., X_I)}{\partial X_i} f_i(X_1, ..., X_I; C_1, ..., C_m; \Lambda_1, ..., \Lambda_n)$$
$$+ \sum_{i=1}^{I} \frac{\partial^2 W(X_1, ...X_I)}{\partial X_i^2} g_i(X_1, ..., X_I; C_1, ..., C_m; \Lambda_1, ..., \Lambda_n)$$
$$\left. + W(X_1, ..., X_I) h(X_1, ...X_I; C_1, ..., C_m; \Lambda_1, ..., \Lambda_n) \right]$$

Convergent approximation: numerical methods

- theory of viscosity solutions by Crandall and Lions (84,92,96)
- Barles-Souganidis (91) proof: monotone finite-difference (FD) method can converge to a well-behaved unique solution of the fully nonlinear Hamilton-Jacobi-Bellman (HJB) equation with no explicit boundary conditions
- Oberman (06): convergent numerical schemes for PDE are constructed and implemented
- we apply a FD numerical approach to solve Ramsey-type economic growth models with carbon-climate response (CCR) and Hansen-Sargent robustness channels

Robust Economic-Climate model: basic formulation

Four state variables: $K, \log A, T, \log \lambda$ and four controls. One maximizer: C, and 3 minimizers: G_T, G_a, G_λ. Value function $W(K, T, \log A, \log \lambda; t)$. The laws of motion for state variables in continuous time:

$$dK_t = [A_t K_t^\alpha - C_t - \delta K_t]\, dt$$

$$d \log A_t = [\mu_{a0} - \mu_{aT}(T_t - T_0) - \sigma_a G_{at}]\, dt + \sigma_a d\tilde{B}_t^1$$

$$dT_t = d\hat{T}_t + \lambda_t A_t K_t^\alpha dt$$

$$d\hat{T}_t = -\sigma_T G_{Tt} dt + \sigma_T d\tilde{B}_t^2$$

$$d \log \lambda_t = -\kappa_\lambda (\log \lambda_t - \log \overline{\lambda}) dt - \sigma_\lambda G_{\lambda t} dt + \sigma_\lambda d\tilde{B}_t^3$$

Continuous time HJB PDE framework

$$0 = \max_{C} \min_{G_T, G_a, G_\lambda} \Bigg[\frac{1}{1-\gamma} C^{1-\gamma} + \frac{1}{2\theta} \left(G_T^2 + G_a^2 + G_\lambda^2 \right) - \rho * W + \frac{\partial W}{\partial K} \left[A K^\alpha - C - \delta K \right] + \frac{\partial W}{\partial T} \left[\lambda A K^\alpha - \sigma_T G_T \right] + \frac{\partial W}{\partial \log \lambda} \left[-\kappa_\lambda (\log \lambda_t - \log \overline{\lambda}) - \sigma_\lambda G_\lambda \right] + \frac{\partial W}{\partial \log A} \left[\mu_{a0} - \mu_{aT} (T - T_0) - \sigma_a G_a \right] + \ldots \Bigg]$$

Controls:

$$G_\lambda^*(K, T, \log A, \log \lambda) = \theta \sigma_\lambda \frac{\partial W(K, T, \log A, \log \lambda)}{\partial \log \lambda}$$

HJB PDE versus discrete-time dynamic programming

optimal consumption feedback in HJB PDE approach:

$$C^*(K, T, \log A, \log \lambda) = \left(\frac{\partial W(K, T, \log A, \log \lambda)}{\partial K} \right)^{-1/\gamma}$$

compare to standard VFI in discrete time DP approach:

$$\beta \mathbb{E} \frac{\partial W(K', T', \log A', \log \lambda')}{\partial K} = [K' - AK^{\alpha} - (1-\delta)K]^{-\gamma}$$

Efficient discrete-time DP methods: L. Maliar, S. Maliar "Envelope condition method versus endogenous grid method for solving dynamic programming problems", Economics Letters, 2013

> *...continuous-time approach sidesteps this difficulty completely. In this regard, it shares some similarities with the "endogenous grid method" of Carroll (2006). The difference is that in continuous-time this also works with "exogenous grids". Intuitively, discrete time distinguishes between "today" and "tomorrow" but in continuous time, "tomorrow" is the same thing as "today".*

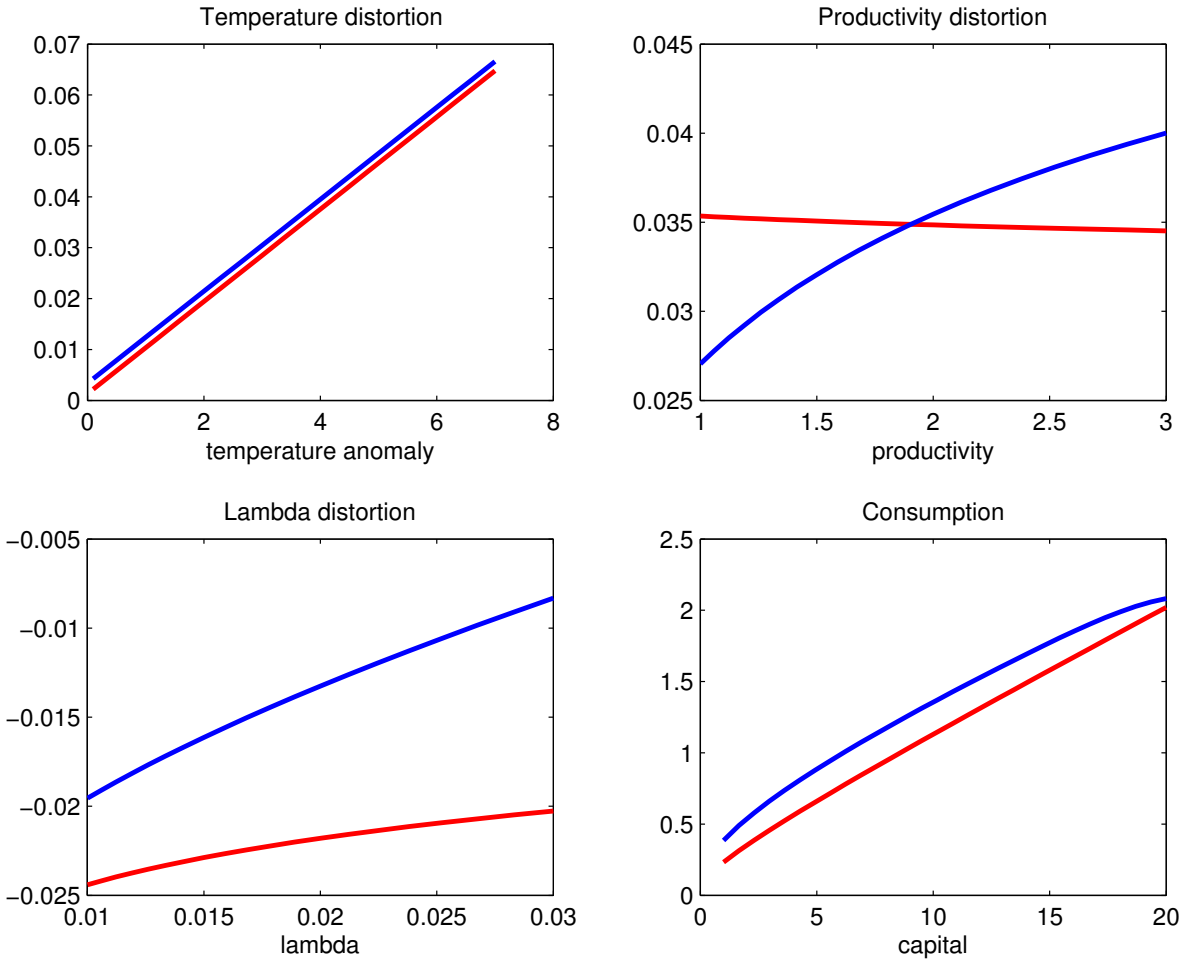
from Benjamin Moll et al, "Heterogeneous Agent Models in Continuous Time", 2014

Finite-differencing schemes and numerical solutions

stable upwind schemes

dimension-adaptive tensor products

six continuous states, $\approx$ 500 million points for the PDE grid



Tensors in discretized partial differential equations

A separable second-order differential operator $\mathcal{L}$:

$$\mathcal{L} = L^{(1)} + L^{(2)} + L^{(3)}, L^{(1)} = \frac{\partial}{\partial x_i} a_i(x_i) \frac{\partial}{\partial x_i} + b_i(x_i) \frac{\partial}{\partial x_i} + c_i(x_i)$$

when discretized on grid

$$G_{n_1 \times n_2 \times n_3} = \left\{ \frac{i}{n_1}, \frac{j}{n_2}, \frac{k}{n_3} : 0 \leq i, j, k \leq 1 \right\}$$

can be written using Kronecker product:

$$\mathcal{L} = L^{(1)} \otimes I_{n_2 \times n_2} \otimes I_{n_3 \times n_3} + I_{n_1 \times n_1} \otimes L^{(2)} \otimes I_{n_3 \times n_3} + I_{n_1 \times n_1} \otimes I_{n_2 \times n_2} \otimes L^{(3)}$$

Finite difference for vectorized tensors

$$vec(\mathcal{L}) = \mathcal{V}_j^+(:) - \mathcal{V}_j^-(:)$$

Tensor rank and tensor decomposition

C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," J. ACM, 60 (2013)

Table I. Tractability of Tensor Problems

Problem	**Complexity**
Bivariate Matrix Functions over $\mathbb{R}$, $\mathbb{C}$	Undecidable (Proposition 12.2)
Bilinear System over $\mathbb{R}$, $\mathbb{C}$	NP-hard (Theorems 2.6, 3.7, 3.8)
Eigenvalue over $\mathbb{R}$	NP-hard (Theorem 1.3)
Approximating Eigenvector over $\mathbb{R}$	NP-hard (Theorem 1.5)
Symmetric Eigenvalue over $\mathbb{R}$	NP-hard (Theorem 9.3)
Approximating Symmetric Eigenvalue over $\mathbb{R}$	NP-hard (Theorem 9.6)
Singular Value over $\mathbb{R}$, $\mathbb{C}$	NP-hard (Theorem 1.7)
Symmetric Singular Value over $\mathbb{R}$	NP-hard (Theorem 10.2)
Approximating Singular Vector over $\mathbb{R}$, $\mathbb{C}$	NP-hard (Theorem 6.3)
Spectral Norm over $\mathbb{R}$	NP-hard (Theorem 1.10)
Symmetric Spectral Norm over $\mathbb{R}$	NP-hard (Theorem 10.2)
Approximating Spectral Norm over $\mathbb{R}$	NP-hard (Theorem 1.11)
Nonnegative Definiteness	NP-hard (Theorem 11.2)
Best Rank-1 Approximation over $\mathbb{R}$	NP-hard (Theorem 1.13)
Best Symmetric Rank-1 Approximation over $\mathbb{R}$	NP-hard (Theorem 10.2)
Rank over $\mathbb{R}$ or $\mathbb{C}$	NP-hard (Theorem 8.2)

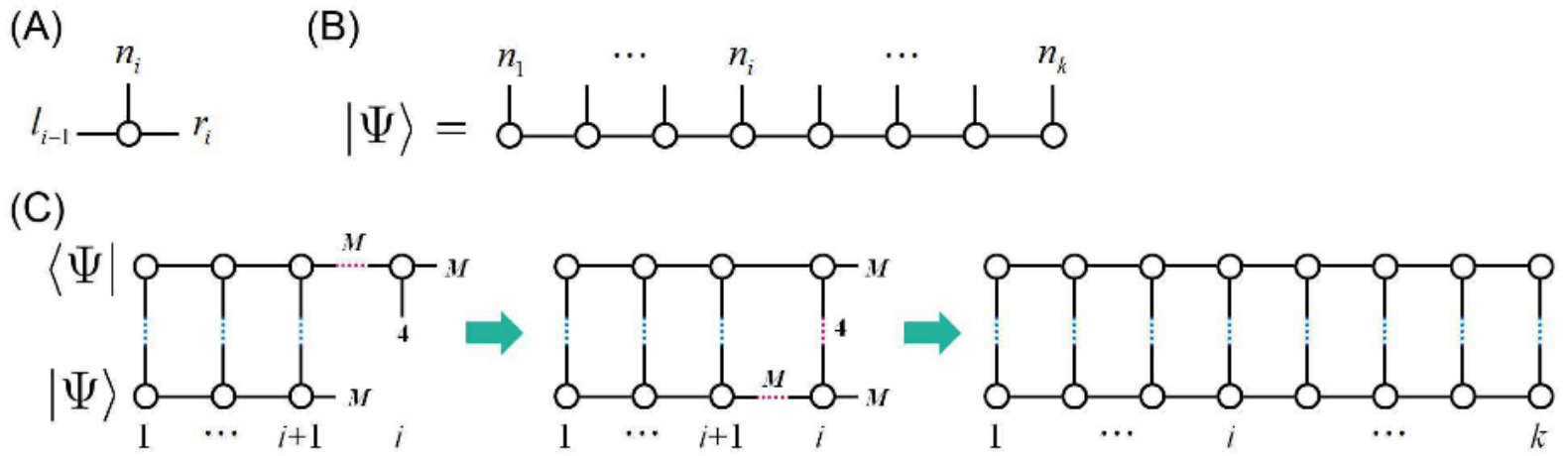
$$rank(\mathcal{A}) = \min \left\{ r : \mathcal{A} = \sum_{i=1}^{r} \lambda_i x_i \otimes y_i \otimes z_i \right\}$$

$\mathcal{A} \in \mathbb{R}^{l \times m \times n}, \mathcal{X} \in \mathbb{R}^{l}, \mathcal{Y} \in \mathbb{R}^{m}, \mathcal{Z} \in \mathbb{R}^{n}$

Tensor networks and quantum entanglement

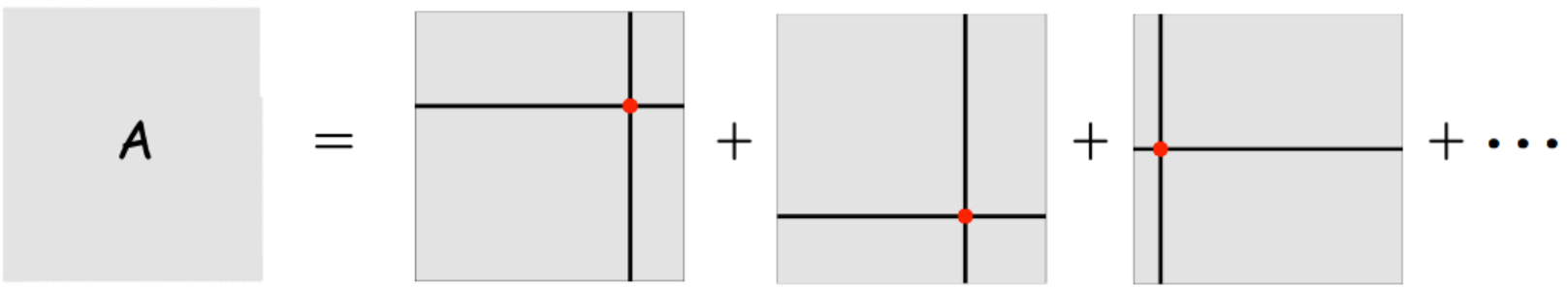
U. Schollwök, "The density-matrix renormalization group in the age of matrix product states," Annals of Physics, 2011

$$\Phi = \otimes_{i=1}^{d} \psi_i$$



Given a d-dimensional tensor, find **an approximation** by tensor decomposition

$$f(x_1, ..., x_d) \approx \sum_{k=1}^{r} f_1^k(x_1) \dots f_d^k(x_d)$$



Given Chebyshev interpolation nodes z_k

$$z_k = -cos\left(\frac{2k-1}{2m}\pi\right)$$

and Chebychev coefficients $a_{ij}, i, j = 0, \ldots, n$ computed on Chebychev nodes we can approximate

$$p(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}\, T_i\left(2\frac{x-a}{b-a}-1\right)\, T_j\left(2\frac{y-c}{d-c}-1\right)$$

Chebfun is freely-available open-source software for MATLAB for computing using Chebyshev technology.
`http://www.chebfun.org`

> *"As we enter an era of parallel computing, with the rules of the game changing fast, it is intriguing that so many fundamental issues remain unsettled in the game we have been playing for decades."*
>
> *L. Trefethen, "Three mysteries of Gaussian elimination", 1985*