Linear Algebra Issues in Wireless Communications

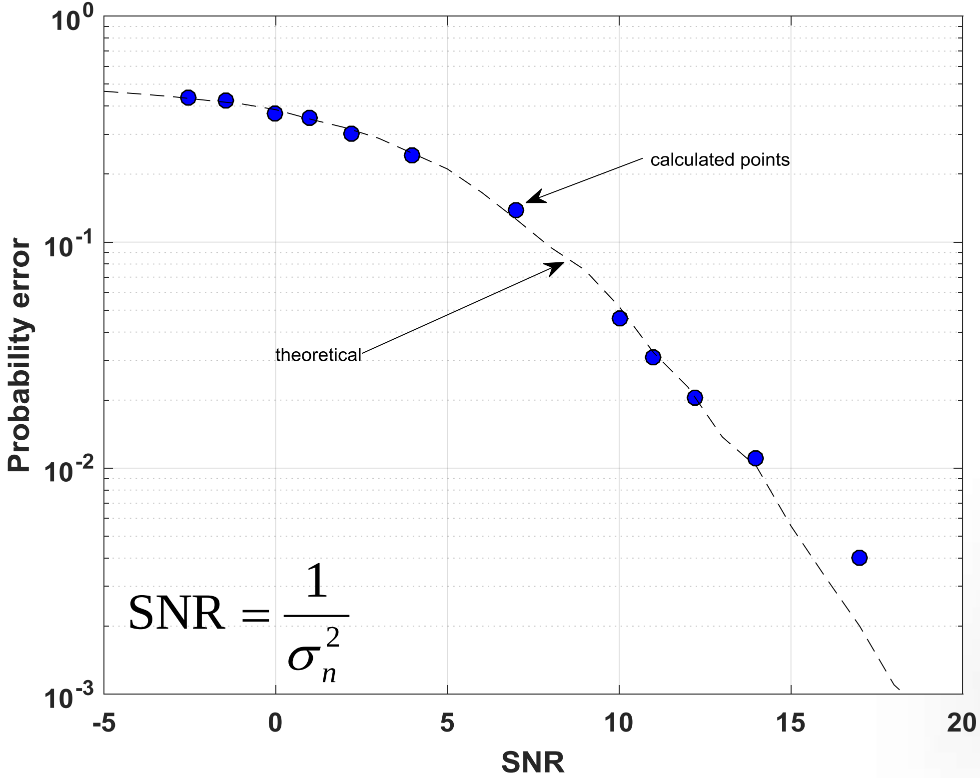
Day 2

Vladimir Lyashev

The Institute of Signal Processing and Control Systems
[lyashev@ieee.org]

SOUTHERN FEDERAL UNIVERSITY

Homework: task #1-3



$$\mathrm{SNR} = \frac{1}{\sigma_n^2}$$

$$\mathbf{A}_k \mathbf{x}_k = \mathbf{b}_k + \boldsymbol{\varepsilon}_k, \text{ where } \mathbf{x}_k = \mathbf{p}_k s_k, \; s_k \in [-1,+1];$$

$$\boldsymbol{\varepsilon}_k \in \mathrm{N}(0,\sigma^2), \; \mathbf{p}_k \text{ is subject to } \left\|\mathbf{A}_k \hat{\mathbf{x}}_k - \mathbf{b}_k\right\|_2 < \sigma^2.$$

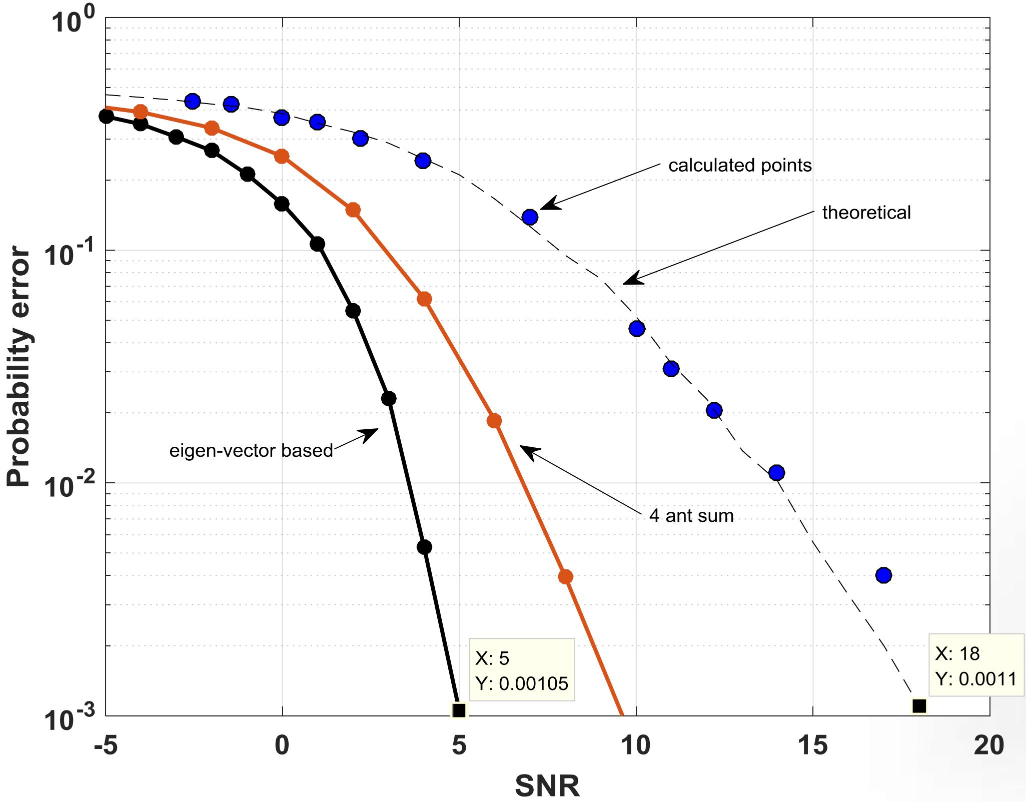
$$\mathbf{A}_k \in \mathbb{R}^{4\times 4}$$

$$\mathbf{p}_k = \tfrac{1}{\sqrt{4}}(1,1,1,1)^T \qquad \left\|\mathbf{p}_k\right\|_2 = 1 \;\; (!)$$

$$\mathbf{A}_k \mathbf{p}_k s_k = \mathbf{b}_k + \boldsymbol{\varepsilon}_k$$

$$P(\sigma_n^2) \to \min$$

Homework: task #4



$$\mathbf{A}_k \mathbf{p}_k s_k = \mathbf{b}_k + \boldsymbol{\varepsilon}_k$$

$$P(\sigma_n^2) \to \min$$

$$\mathbf{p}_k = \frac{1}{\lambda_1^2} \mathbf{v}_1$$

Usage of the 1st eigen-vector allows to reduce variation of equivalent noise in **20 times**.

Homework: task #4

$$\mathbf{A}_k \mathbf{p}_k s_k = \mathbf{b}_k + \boldsymbol{\varepsilon}_k$$

It is possible to "suppress" noise more if we apply 2nd eigen-vector $\mathbf{v}_2$:

$$\mathbf{p}_k = \begin{bmatrix} \frac{1}{\lambda_1^2}\mathbf{v}_1 & \frac{1}{\lambda_2^2}\mathbf{v}_2 \end{bmatrix}$$

Issue for optimization

IF $\lambda_2^2 << \lambda_1^2$ THEN IT IS BETTER TO USE 1st EIGEN-VECTOR ONLY!!!

IT IS IMPORTANT TO HAVE FAST WAY TO COMPUTE SEVERAL EIGENVECTORS OUT OF ALL.

Vladimir Lyashev

How stochastic information about additive noise can be utilized for minimization of error probability?..

Well-Known Regularization can help us!

$$\mathbf{A}_k \mathbf{p}_k s_k = \mathbf{b}_k + \boldsymbol{\varepsilon}_k$$

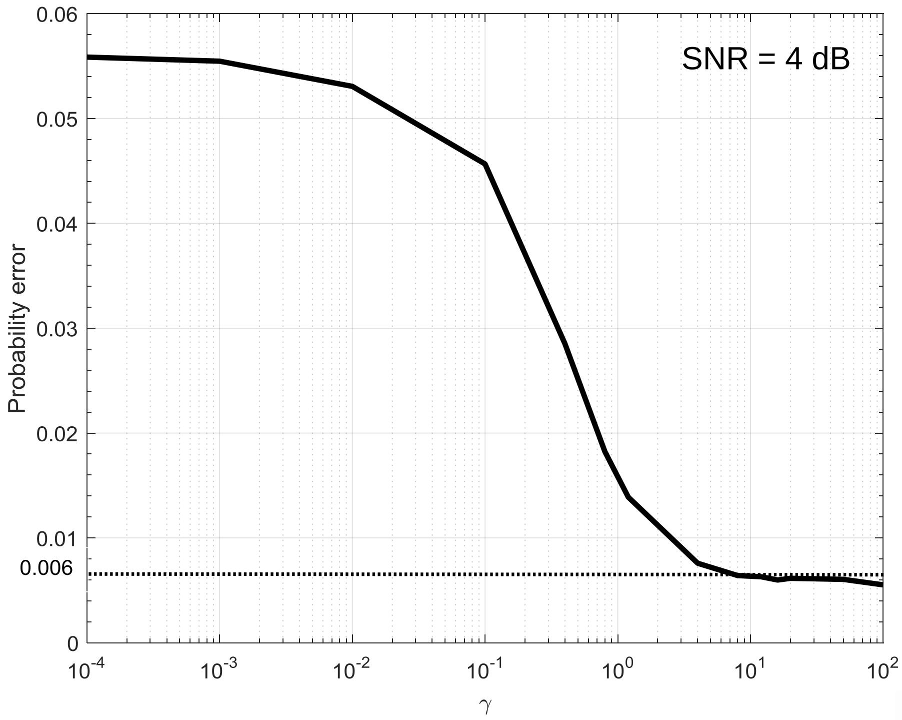
$$s_k = \left(\mathbf{A}_k \mathbf{p}_k\right)^{-1}\left(\mathbf{b}_k + \boldsymbol{\varepsilon}_k\right) = \left(\mathbf{H}_k\right)^{-1}\left(\mathbf{b}_k + \boldsymbol{\varepsilon}_k\right)$$

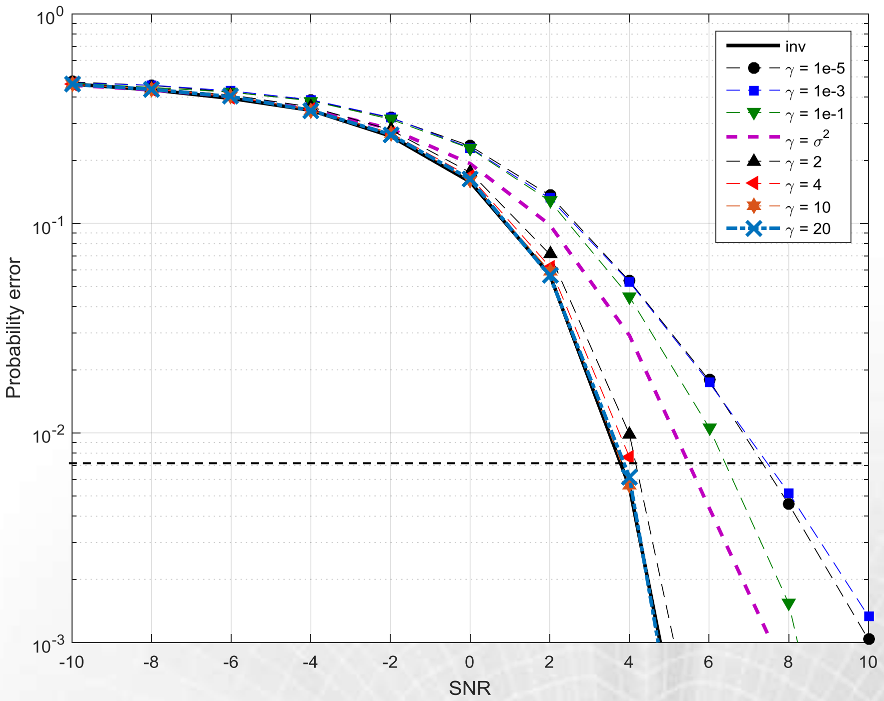
$\mathbf{H}$ is not always square matrix, thus pseudo-inverse is used

$$\left(\mathbf{H}_k\right)^{-1} = \frac{\mathbf{H}_k^H}{\mathbf{H}_k^H \mathbf{H}_k} \xrightarrow{\textit{regularization}} \frac{\mathbf{H}_k^H}{\mathbf{H}_k^H \mathbf{H}_k + \gamma \mathbf{I}}$$

$$\gamma - ?$$

Well-Known Regularization can help us!





Ideal knowledge about matrix **A !!! If not ?**

Hometask!

Vladimir Lyashev

Any questions regarding homework?

QR factorization and least squares problems

```
function x=LS(A,b) % Solves the least squares problem || Ax-b || --> min % for the full column rank matrix
A by using QR factorization. [m,n]=size(A);
[Q,R]=my_QR(A);
b1=Q'*b;
c=b1(1:n);
x=R(1:n,1:n)\c;
end
```

The property of the orthogonal martix $\mathbf{Q}$ that enables us to easily solve the least squares problem.

$$\|\mathbf{Qx}\|_2 = \|\mathbf{x}\|_2$$

- MMSE Spatial Multiplexing Detectors (for MIMO-OFDM)
- Cloud eNodeB (for parallel scheduling and processing)
- Distributed MIMO for indoor (for parallel scheduling and pairing)
- Relay network (for backward and forward channels in conjunction with phase control at each relay node)
- Blockwise STBC Matrix Inversion

QR ML Estimator (example)

QR decomposition often used in MIMO systems

Assuming **H** has a rank of *r*, we have: **H = QR**,
Where **Q** is an *N*×*r* orthonormal matrix, **R** is an *r*×*r* upper triangular matrix.

Solution:

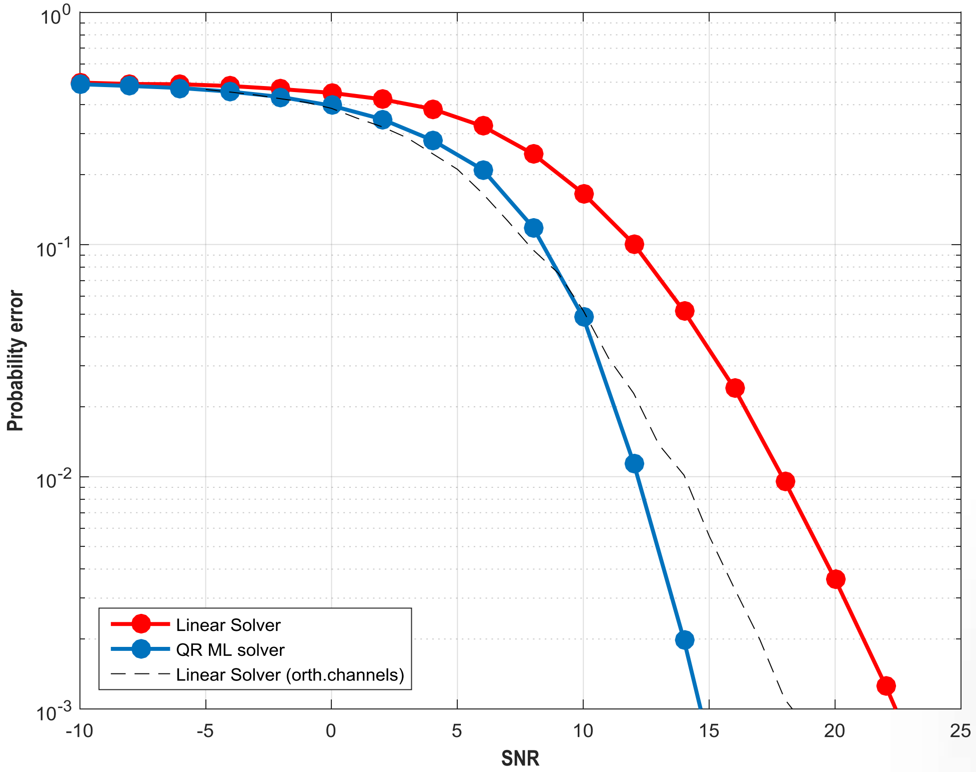
Since **Q** is orthonormal, we have:

$$\left\|\mathbf{y}-\mathbf{H}\hat{\mathbf{x}}_{ML}\right\|^2=\left\|\mathbf{y}-\mathbf{QR}\hat{\mathbf{x}}_{ML}\right\|^2=\left\|\mathbf{Q}\left(\mathbf{Q}^H\mathbf{y}-\mathbf{R}\hat{\mathbf{x}}_{ML}\right)\right\|^2=\left\|\mathbf{Q}^H\mathbf{y}-\mathbf{R}\hat{\mathbf{x}}_{ML}\right\|^2\cong$$

$$\cong\left\|\tilde{\mathbf{y}}-\mathbf{R}\hat{\mathbf{x}}_{ML}\right\|^2=\left\|\begin{pmatrix}\tilde{y}_0\\ \tilde{y}_1\\ \vdots\\ \tilde{y}_{r-1}\end{pmatrix}-\begin{pmatrix}R_{00} & R_{01} & \cdots & R_{0(r-1)}\\ 0 & R_{11} & & R_{1(r-1)}\\ \vdots & & \ddots & \vdots\\ 0 & 0 & 0 & R_{(r-1)(r-1)}\end{pmatrix}\begin{pmatrix}\hat{s}_0\\ \hat{s}_1\\ \vdots\\ \hat{s}_{r-1}\end{pmatrix}\right\|^2$$

Can be viewed as an *r* layer system.

QR ML vs. Linear Detection



Two-state system {-1; +1}

<u>Linear receiver required:</u>

matrix inversion and matrix product (depends of antenna number)

CPLX: (N^3+N^2) MUL

<u>ML receiver required:</u>

matrix product times 2^N

CPLX: $2^N N^2$ MUL

CPLX*: $2^N \Sigma n$ MUL

<u>For system 4x4</u>

- Linear: 80 MUL
- ML: 256 MUL
- ML-QR: 160 MUL (-37.5%)

Parallel

Vladimir Lyashev

What we want?..

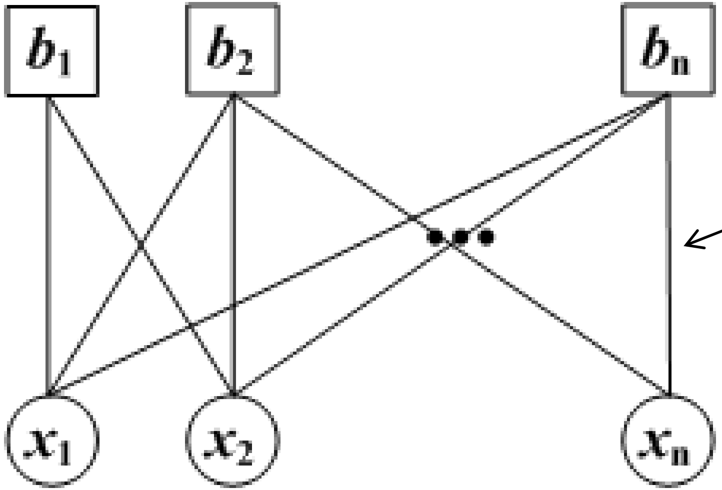
- Performance close to ML solver
- Complexity similar with Linear solver
- Scalability in Time-Frequency-Space dimensions
- Robustness to the noises

From Linear Algebra to Probabilistic Inference

$$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{x} = \mathbf{A}^H \left(\mathbf{AA}^H\right)^{-1} \mathbf{b}$$

N×N: $N^3 + N^2 + \text{inv}()$

$$\Gamma(\chi, \varepsilon): \chi = \{x_1, x_2, \ldots x_n\}; \varepsilon = f(\mathbf{A})$$



functional nodes

Edges of the factor graph, which are corresponded to non-zero matrix elements $a_{b,x}$

variable nodes

More sparsity – less connections – simplier graph

Probabilistic inference

The computation of the solution vector $\mathbf{x}$ is identical to the inference of the vector of marginal means $\boldsymbol{\mu} = \{\mu_1, \mu_2, \ldots, \mu_n\}$ over the graph Γ with the associated joint Gaussian probability density function $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \mathbf{A}^{-1})$.

Solution in quadratic form

$$(\mathbf{Ax}-\mathbf{b})^T(\mathbf{Ax}-\mathbf{b})=\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-2\mathbf{x}^T\mathbf{A}^T\mathbf{b}+\mathbf{b}^T\mathbf{b};$$

$$\text{Quadratic form}: Q(\mathbf{x})=\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-\mathbf{x}^T 2\mathbf{A}^T\mathbf{b}+\mathbf{b}^T\mathbf{b};$$

$$\frac{\partial Q(\mathbf{x})}{\partial \mathbf{x}}=2\mathbf{A}^T\mathbf{Ax}-2\mathbf{A}^T\mathbf{b}=2\mathbf{A}^T(\mathbf{Ax}-\mathbf{b});$$

$$\frac{\partial Q(\mathbf{x})}{\partial \mathbf{x}}=0,\ \text{solution for}\ \mathbf{Ax}=\mathbf{b}.$$

Vladimir Lyashev

Solution in quadratic form

$$p(\mathbf{x}) = \Xi^{-1} e^{-Q(\mathbf{x})} = \Xi^{-1} e^{-\mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} + \mathbf{x}^T 2\mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{b}};$$

$$\boldsymbol{\mu} \leftarrow \mathbf{A}^{-1}\mathbf{b}: \; p(\mathbf{x}) = \Xi^{-1} e^{-\mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} + \mathbf{x}^T 2\mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{b} + \boldsymbol{\mu}^T \mathbf{A}^T \mathbf{A}\boldsymbol{\mu} - \boldsymbol{\mu}^T \mathbf{A}^T \mathbf{A}\boldsymbol{\mu}} =$$

$$= \Xi^{-1} e^{\boldsymbol{\mu}^T \mathbf{A}^T \mathbf{A}\boldsymbol{\mu}} e^{-\mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} + \mathbf{x}^T 2\mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{b} - \boldsymbol{\mu}^T \mathbf{A}^T \mathbf{A}\boldsymbol{\mu}} = \left\{ \begin{array}{l} \mathbf{R} = \mathbf{A}^T \mathbf{A}; \\ \hat{\Xi}^{-1} = \Xi^{-1} e^{\boldsymbol{\mu}^T \mathbf{A}^T \mathbf{A}\boldsymbol{\mu}} e^{-\mathbf{b}^T \mathbf{b}} \end{array} \right\} =$$

$$= \hat{\Xi}^{-1} e^{-\left(\mathbf{x}^T \mathbf{R}\mathbf{x} - \mathbf{x}^T 2\mathbf{R}\mathbf{A}^{-1}\mathbf{b} + \boldsymbol{\mu}^T \mathbf{R}\boldsymbol{\mu}\right)} = \hat{\Xi}^{-1} e^{-(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{R}(\mathbf{x}-\boldsymbol{\mu})} = \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{R}^{-1/2}\right) = \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{A}^{-1}\right).$$

Hence, in order to solve the system of linear equations we need to infer the marginal densities, which must also be Gaussian.

Vladimir Lyashev

Belief Propagation

BP has been found to have outstanding empirical success in many applications, e.g. , in decoding

- ❖ Turbo codes;
- ❖ low-density parity-check (LDPC) codes.

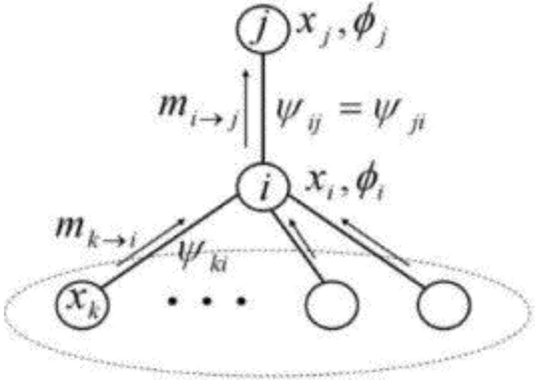
Gaussian Belief Propagation

$$p(\mathbf{x}) \propto \prod_{i=1}^{n} \phi_i(x_i) \prod_{\{i,j\}} \psi_{ij}(x_i, x_j) \qquad \psi_{ij}(x_i, x_j) = e^{-x_i a_{ij} x_j} \qquad \phi_i(x_i) = e^{b_i x_i - a_{ii} x_i^2 / 2}$$

$N(i)$ – set of neighbour nodes to i

$N(i) \setminus j$ – the same excluding j node

The marginals are computed (as usual) according to the product rule



$$p(x_i) = \alpha \phi_i(x_i) \prod_{k \in N(i)} m_{ki}(x_i)$$

$$m_{ki}(x_i) \propto N\left(\mu_{ki} = -p_{ki}^{-1} a_{ki} \mu_{k \setminus i}, \, p_{ki}^{-1} = -a_{ki}^{-2} p_{k \setminus j}\right)$$

BP Algorithm

1. *Initialize:*
   - ✓ Set the neighborhood $\mathrm{N}(i)$ to include $\forall k \neq i \exists A_{ki} \neq 0$.
   - ✓ Set the scalar fixes $P_{ii} = A_{ii}$ and $\mu_{ii} = b_i / A_{ii}$, $\forall i$.
   - ✓ Set the initial $\mathrm{N}(i) \ni k \rightarrow i$ scalar messages $P_{ki} = 0$ and $\mu_{ki} = 0$.
   - ✓ Set a convergence threshold ϵ.
2. *Iterate:*
   - ✓ Propagate the $\mathrm{N}(i) \ni k \rightarrow i$ messages P_{ki} and μ_{ki}, $\forall i$ (under certain scheduling).
   - ✓ Compute the $\mathrm{N}(j) \ni i \rightarrow j$ scalar messages
     $P_{ij} = -A_{ij}^2 / (P_{ii} + \sum_{k \in \mathrm{N}(i) \setminus j} P_{ki})$,
     $\mu_{ij} = (P_{ii}\mu_{ii} + \sum_{k \in \mathrm{N}(i) \setminus j} P_{ki}\mu_{ki}) / A_{ij}$.

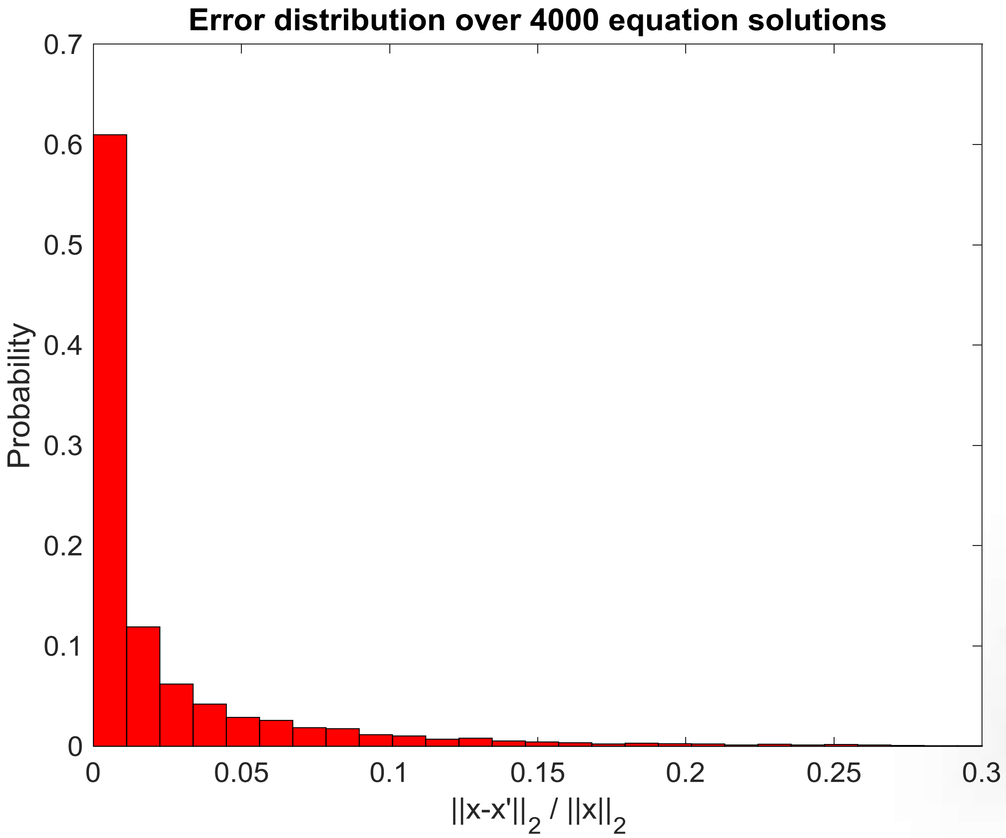
Vladimir Lyashev

BP Algorithm

3. *Check:*
   - ✓ If the messages P_{ij} and μ_{ij} did not converge (w.r.t. ϵ), return to Step 2.
   - ✓ Else, continue to Step 4.
4. *Infer:*
   - ✓ Compute the marginal means $\mu_i = \left(P_{ii}\mu_{ii} + \sum_{k \in \mathrm{N}(i)} P_{ki}\mu_{ki}\right) / \left(P_{ii} + \sum_{k \in \mathrm{N}(i)} P_{ki}\right),\ \forall i.$
   - (✓ Optionally compute the marginal precisions $P_i = P_{ii} + \sum_{k \in \mathrm{N}(i)} P_{ki}$)
5. *Solve:*
   - ✓ Find the solution $x_i^* = \mu_i,\ \forall i.$

 Vladimir Lyashev

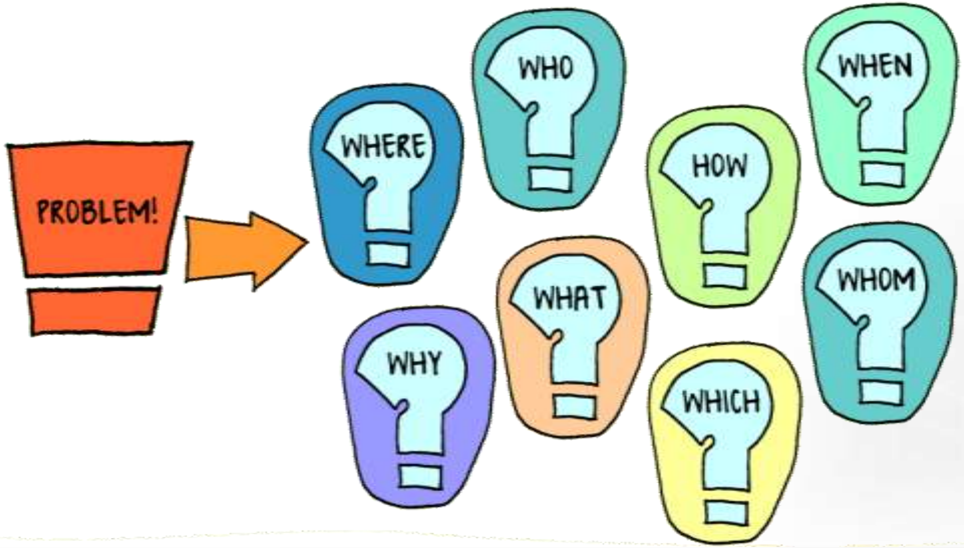
Error distribution



- ✓ Complexity is similar to GE $\mathcal{O}(n^3)$
- ✓ Good approximation of ML solution that is beneficial in noisy matrix equations
- ✓ Can solve matrix equation, which solution is defined on the grid only, smoothly

Vladimir Lyashev

Brief Overview of some other problems in wireless communication

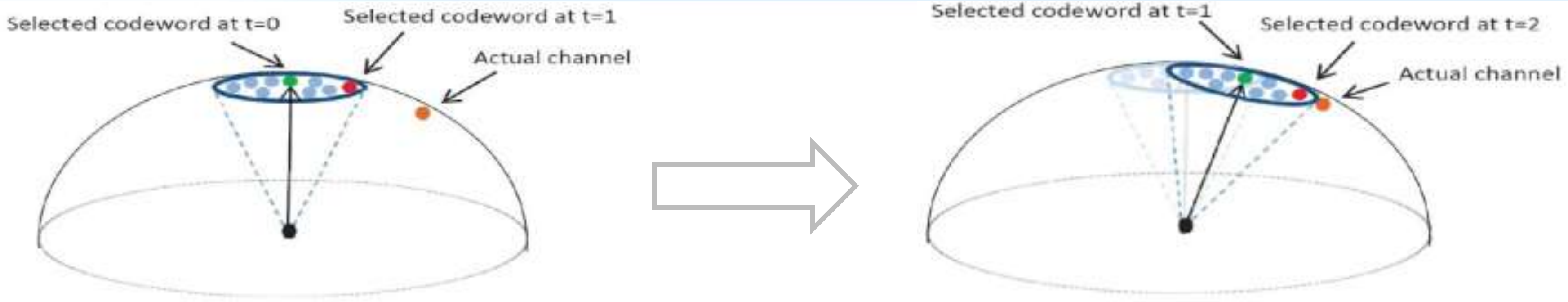


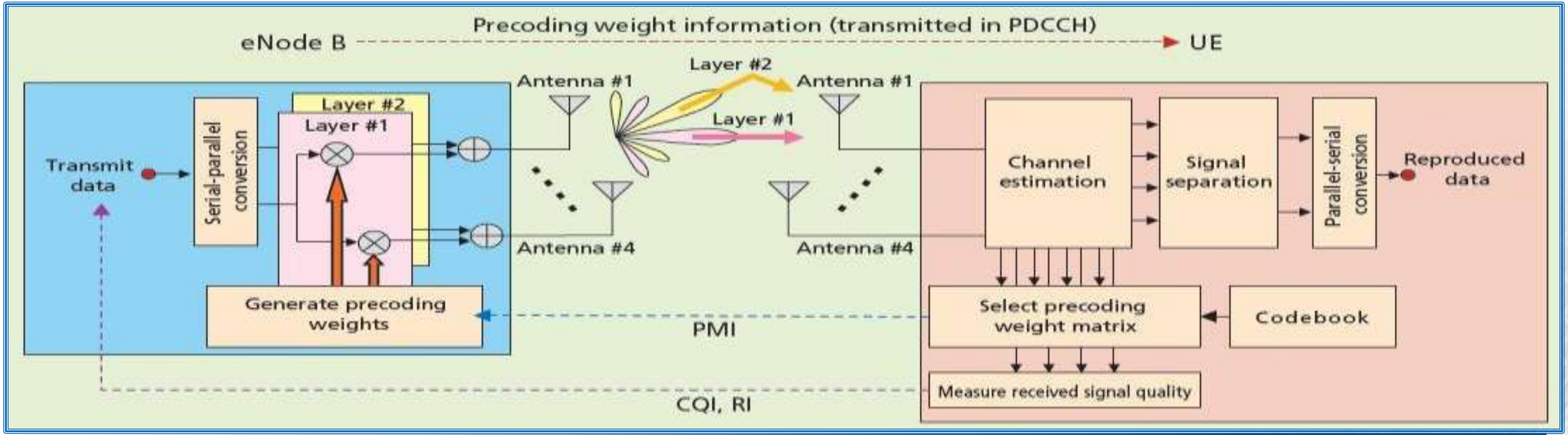
Vector Weighting

In TDD systems, the channel is the same on transmitter and receiver, but it is changing in time and prediction of the channel is challengeable problem in wireless communication.

In FDD systems, the channels are different for uplink and downlink, because of channel reciprocity.

Vector Weighting





Vladimir Lyashev

Optimization Problem

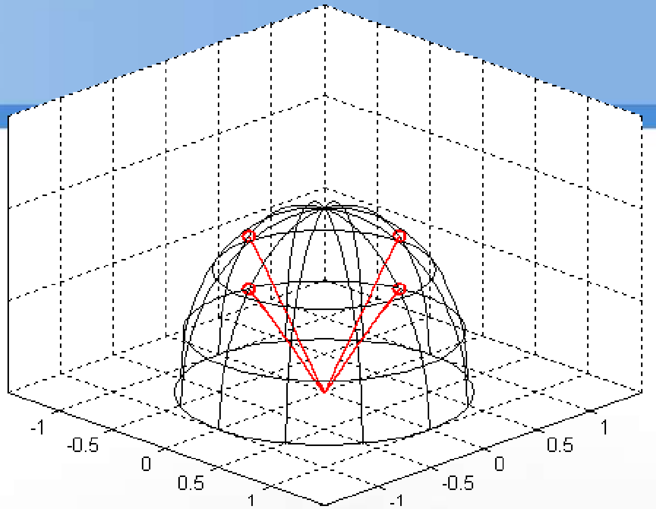
Let us define following variables:

$\mathbf{h} \in \mathbb{C}^N$	downlink channel
Ω	set of precoding vectors (codebook)
ω	precoding matrix index (PMI) reported by user
$\mathbf{w} \in \mathbb{C}^N$	weighting vector from codebook according PMI
$\mathbf{Q}$	rotation matrix $N \times N$ for codebook

Thus we can define precoding vector, which is reported by user (receiver):

$$\omega = \arg\max_i \mathbf{h}^H \mathbf{w}_i$$

after that transmitter (base station) assume the channel as $\mathbf{w}_\omega$ and will use it for antenna weighting $\mathbf{w}_\omega s$. If downlink channel is equal weighting vector (from codebook): $\mathbf{h} = \mathbf{w}_\omega$, then we obtain bound performance, otherwise some performance degradation by quantization error in codebook.



In single user case (at the beginning) we can consider some rotation operator $\mathbf{Q}$, which allows to reduce quantization error influence. Thus,

$$\frac{1}{N}\left\|\sum_{n=1}^{N}\left(\mathbf{h}_n\mathbf{h}_n^H - \mathbf{Q}_n\left[\mathbf{w}_n\mathbf{w}_n^H\right]\mathbf{Q}_n^H\right)\right\| \to \min \qquad (1)$$

where each $\mathbf{w}_n$ is defined by

$$n = \arg\max_i \mathbf{w}_i^H\left(\mathbf{Q}_n\mathbf{h}_n\mathbf{h}_n^H\mathbf{Q}_n^H\right)\mathbf{w}_i$$

and we assume that $\mathbf{h}_n \approx \mathbf{h}_{n+1} = \mathbf{h}$. Thus

$$\left\|\mathbf{h}\mathbf{h}^H - \sum_{n=1}^{N}\left(\mathbf{Q}_n\left[\mathbf{w}_n\mathbf{w}_n^H\right]\mathbf{Q}_n^H\right)\right\| \to \min \quad \text{or} \quad \sum_{n=1}^{N}\left(\mathbf{Q}_n\left[\mathbf{w}_n\mathbf{w}_n^H\right]\mathbf{Q}_n^H\right) \xrightarrow{N\to\infty} \mathbf{h}\mathbf{h}^H \qquad (2)$$

In this problem we need to define the best choise for set of the matrices $\{\mathbf{Q}_n\}_{n=1}^N$ with limited N to minimize expression (2).

1. How can we minimize quantization error in optimal way ? (realtime)
2. How can we define mechanism for adjustment an algorithm to terminal distribution in the cell ? (offline processing is assumed here)

Ways for matrix generation

- Grassmannian Line Packing
- DFT based design (DFT)
- Mutually Unbiased Bases Construction (MUB)
- Vector Quantization Construction (VQ)

Stochastic Precoder and Decorrelator Control Problem

For some positive constants $\boldsymbol{\gamma} = \{\gamma_k > 0 : \forall k\}$ and $\boldsymbol{\beta} = \{\beta_k > 0 : \forall k\}$, the stochastic precoder and decorrelator control problem is formulated as

$$\min_{\Omega} L^{\Omega}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{\chi}(0)) = \sum_{k=1}^{K} \left(\overline{P}_k^{\Omega} \boldsymbol{\chi}(0) + \gamma_k \overline{I}_k^{\Omega} \boldsymbol{\chi}(0) + \beta_k \overline{B}_k^{\Omega} \boldsymbol{\chi}(0) \right) =$$

$$= \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=0}^{T-1} E^{\Omega}\left(c(\mathbf{Q}(t), \boldsymbol{\Omega}(\boldsymbol{\chi}(t))) \right)$$

$\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ can also be interpreted as the corresponding Lagrange Multipliers associated with the playback interruption probabilities and buffer overflow probabilities of the K users

$c(\mathbf{Q}, \mathbf{F}) = \sum_{k=1}^{K} c_k(Q_k, \mathbf{F}_k)$ - per-stage cost function with

$$c_k(Q_k, \mathbf{F}_k) = \mathrm{Tr}\left(\mathbf{F}_k \mathbf{F}_k^H\right) + \gamma_k e^{-\eta[Q_k - Q^l]} + \beta_k e^{-\eta[Q^h - Q_k]}$$

$Q^l < Q^h$ - corresponding *min* and *max* buffer size

η – smooth coefficient

Ω - set of control policies

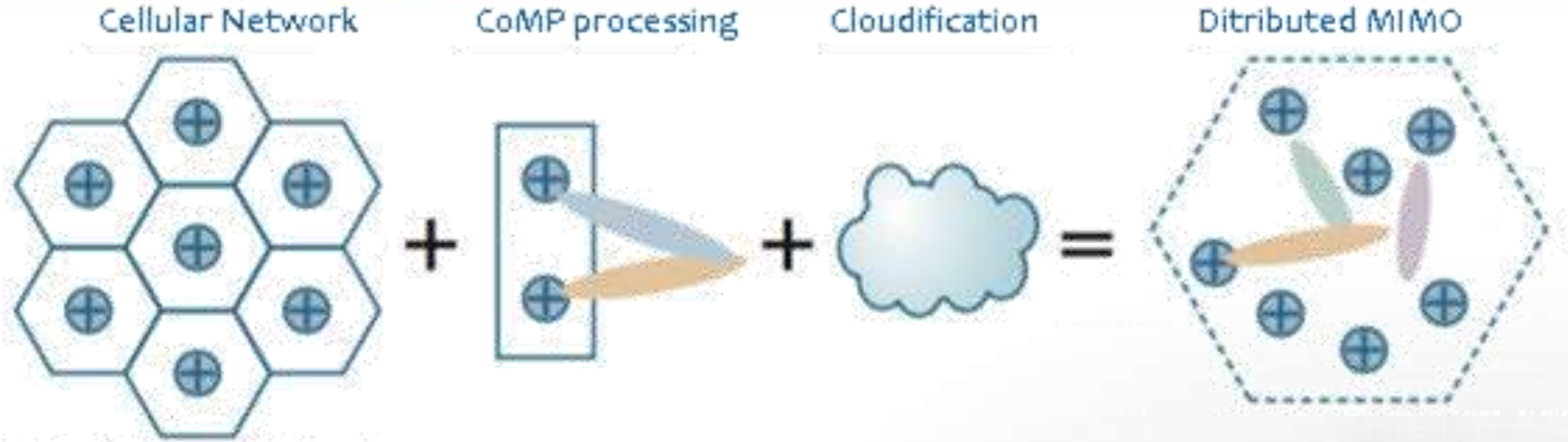
$\chi(t)$ - system state defined as set of Markov chains

$Q_k(t)$ – a controlled Markov chain

$\mathbf{F}_k$ - precoders $\mathbf{U}_k$ - decorrelators

$Q_k(t+1) = f\left(Q_k(t), \mathbf{H}(t), \{\mathbf{F}, \mathbf{U}_k\}\right)$

D-MIMO and User Centric Approach



Cell-Centric Approach

serving cell uses
query for processing users

/* in fact, each user has priority,
which is defined by serving cell */

User-Centric Approach

no serving cell, each base-station wants to attach user and competes with others

/* different number of processing units requires scalable matrix methods and efficient rank approx.*/

Low-Rank Approximation

Objective:

Maximum $\Sigma_{user}\log(R)$ (through traditional algorithms, such as zero-forcing, user-pairing in MU-MIMO, or new algorithms)

Simplify the computational complexity

Simplify the time complexity

- Distributed algorithm supporting parallel computation
- Simplify the time complexity of all J,M,K,N combination

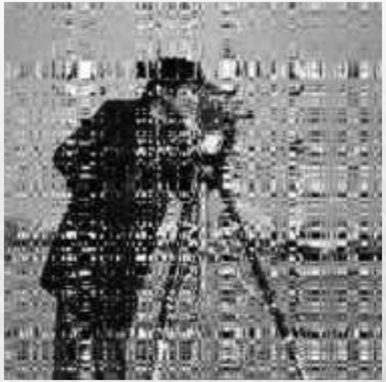
The theory of pseudo-skeleton approximations

$$\mathbf{H} = \mathbf{H}_0 + \mathbf{E}_0,\ \ rank(\mathbf{H}_0) = r,\ \ \|\mathbf{E}_0\| < \varepsilon,\ \ \mathbf{H} \in \mathbb{C}^{n\times m}$$

$$\mathbf{H} = \mathbf{H}_1 + \mathbf{E}_1,\ \ rank(\mathbf{H}_1) = r,\ \ \|\mathbf{E}_1\| < k\varepsilon,\ \ k = const.$$

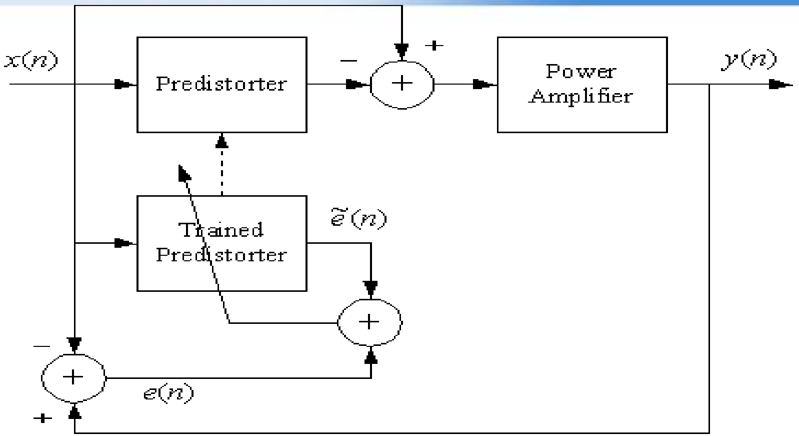
$$\mathbf{H} = \mathbf{CUR} + \mathbf{E}_1,\ \text{where}\ \ \mathbf{C} \in \mathbb{C}^{n\times r}\ \text{and}\ \mathbf{R} \in \mathbb{C}^{r\times m}$$

$\mathbf{U} \in \mathbb{C}^{r\times r}$ - skeleton matrix of $\mathbf{H}$



Vladimir Lyashev

Nonlinear Model Optimization Problem



$$y = F_{\varepsilon}[x]$$

$$\| y(n) - F_{\varepsilon}[x(n)] \| \to \min_{\vec{c}}$$

Main problems:

I. Optimal nonlinear model selection;

II. Model optimization method design.

1. Cholesky decomposition;
2. QR and SVD decomposition;
3. LMS, NLMS (gradient) algorithm;
4. RLS, QR RLS, affine projection algorithm;
5. Back propagation algorithm for neural networks.
6. Tensor approach

$$\sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} \dots \sum_{m_k=0}^{\infty} h_k(m_1, m_2, \dots, m_k) \prod_{r=1}^{k} x(n - m_r) - y^o(n) \to \min_{h_k}$$

Content of Day#3

- Matrix problems in radio resource management
- Criteria for user scheduling and its matrix formalization
- Robustness & accuracy
- The simplest one doesn't mean the fastest one

Thanks for your attention!