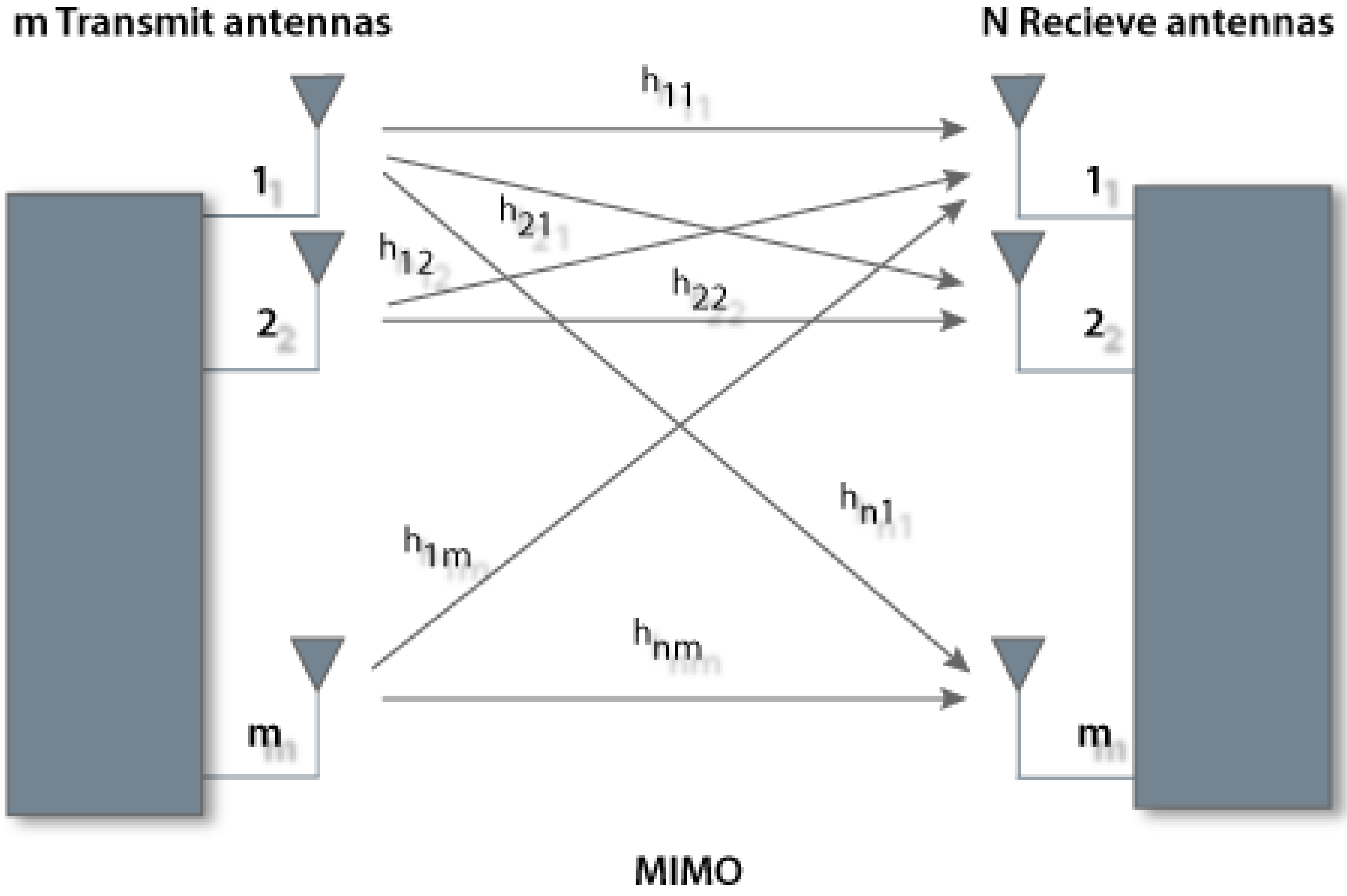
Gaussian Sampling Based Lattice Decoding

- Bragadeesh (EE14B114)
- Nived Rajaraman (EE14B040)
- Dheeraj.M.Pai (EE14B082)

MIMO system



m Transmit antennas
N Recieve antennas
h11
h21
h12
h22
h1m
hn1
hnm
1
2
m
1
2
m
MIMO

Channel model

$$y_c = \boldsymbol{H}_c x_c + \boldsymbol{n_c}$$

$\boldsymbol{H}_c$: $n_r \times n_t$ $matrix$, $\boldsymbol{H_c} \in C^{n_r \times n_t}$, *fading gain channel*
$\boldsymbol{x_c}$: $n_t \times 1$ $matrix$, $\boldsymbol{x_c} \in A^{n_t}$, where A is set of information symbols , sent vector
$\boldsymbol{n}_c$: $n_r \times 1$ $matrix$, iid $\boldsymbol{N}(0, \sigma^2)$, i.i.d Gaussian noise
$\boldsymbol{y}_c$: $n_r \times 1$ $matrix$, $\boldsymbol{y_c} \in C^{n_r}$, received vector

Simplified equation

$$\begin{bmatrix} R(\boldsymbol{H_c}) & -I(\boldsymbol{H_c}) \\ I(\boldsymbol{H_c}) & I(\boldsymbol{H_c}) \end{bmatrix} \begin{bmatrix} R(\boldsymbol{x_c}) \\ I(\boldsymbol{x_c}) \end{bmatrix} + \begin{bmatrix} R(\boldsymbol{n_c}) \\ I(\boldsymbol{n_c}) \end{bmatrix} = \begin{bmatrix} R(\boldsymbol{y_c}) \\ I(\boldsymbol{y_c}) \end{bmatrix}$$

$$y = \boldsymbol{H}x + \boldsymbol{n}$$

$\boldsymbol{H}$: $n_r \times n_t\ matrix,\ \boldsymbol{H_c} \in R^{n_r \times n_t}$

$\boldsymbol{x}$: $n_t \times 1\ matrix,\ \ \boldsymbol{x_c} \in A^{n_t}$, where A is set of information symbols

$\boldsymbol{n}$: $n_r \times 1\ matrix, iid\ \boldsymbol{N}(0, \sigma^2)$

$\boldsymbol{y}$: $n_r \times 1\ matrix, \boldsymbol{y_c} \in R^{n_r}$

The problem

$$y = \boldsymbol{H}x + \boldsymbol{n}$$

$$\hat{x} = \text{argmin}_{x \in A^n} \left|| y - Hx |\right|^2$$

This is identical to the lattice problem

$$\lambda(L) = \min_{v \in L \backslash \{0\}} ||v||$$

which is NP-HARD!!!!! ☹

Lattice decoding algorithm

- Assume all entries of $\boldsymbol{x}$ except at index $\boldsymbol{i}$ is known
- Sample x_i randomly from it’s probability distribution
- Expected to converge to the optimal solution at infinity
- May not give optimal results with negligible probability

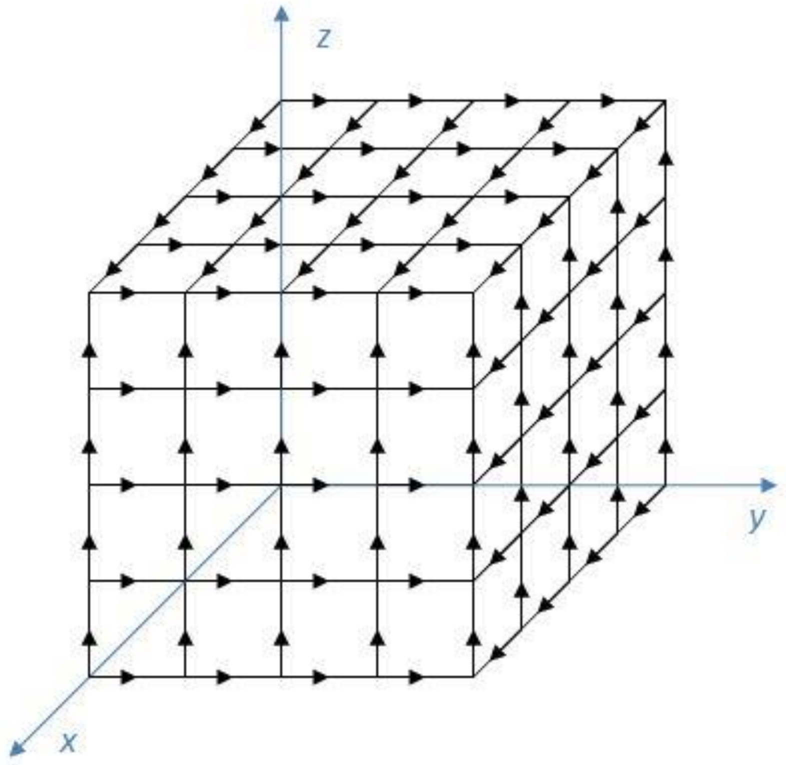
Lattice decoding algorithm

$$p(x_i|y, H, x_{j\neq i}) \propto \exp\left(-\frac{||y - Hx||^2}{\sigma^2}\right) \propto \exp(-\frac{||x_i - \mu_i||^2}{\sigma^2/\left||h_i|\right|^2})$$

where $\mu_i = \frac{\left(y^i\right)^T h_\mathrm{i}}{\left||h_i|\right|^2}$, h_i is i^{th} column of H matrix

Visualizing GSLD

- At next iteration, new coordinate is chosen from the particular dimension instead of all points.
- Conditional distribution along each dimension is known.
- Expected to converge probabilistically to ML solution



Implementation Specific Details

Discussing pseudo code, computational complexity and convergence

```
1: input: y, H, n, m, \sigma^2; x: initial vector; I_{max}: max. # iterations;
   \mathcal{A}: Alphabet ; T_1; T_2; \Theta
2: C = 0,  S = 0,  t = 0,
3: Compute \beta = ||y - Hx||^2; z = x;
4: Compute r_i = ||h_i||^2 for i = 1, 2, ..., n; \hat{y} = y - \sum_{j=1}^{n} h_j x_j;
5: while t < I_{max} do
6:   for i = 1 to n do
7:     if C < n - 1 then
8:       Compute \tilde{y}^{(i)} = \hat{y} + h_i x_i;
9:       Compute \mu_i = \frac{(\tilde{y}^{(i)})^T h_i}{r_i};
10:    end if
11:    Generate sample s_i from \mathcal{N}(\mu_i, \frac{\sigma^2}{2r_i});
12:    Generate x_i^{new} from quantization of s_i;
13:    if x_i^{new} \neq x_i then
14:      C = 0; \hat{y} = \tilde{y}^{(i)} - h_i x_i^{new};
15:    else
16:      C = C + 1;
17:    end if
18:    Update ith coordinate of x with x_i^{new};
19:  end for
20:  \gamma = ||y - Hx||^2;
21:  if (\gamma \leq \beta) then
22:    z = x;  \beta = \gamma;  S = 0;
23:  else
24:    S = S + 1;
25:  end if
26:  if \beta < \Theta then
27:    if S \geq T_1 then
28:      goto step 37
29:    end if
30:  else
31:    if S \geq T_2 then
32:      goto step 37
33:    end if
34:  end if
35:  t = t + 1;
36: end while
37: output: z.      z : output solution vector
```

Pseudo code

<u>For every bit i in the vector x_i :</u>

1) Generate = $\tilde{y}^{(i)} = \hat{y} + h_i x_i$

   ŷ is the residual error : $\hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}_{init}$

1) Generate $\mu_i = (\tilde{y}^i)^T h_i \Big/ ||h_i||^2$

2) Sample s_i from $\sim N(\mu_i\,,\, \frac{\sigma^2}{2r_i})$ and quantize s_i to alphabet $(= x_{new}^i)$

3) Update x^i with x_{new}^i and update ŷ (residual error). Repeat from (1)

4) Convergence based on **S**

   S : # consecutive iterations the best solution has not changed

   T_1 : # iterations where best error is below threshold

   and T_2 : # max. number of iterations after which the unchanged estimated best solution is printed

Restarts

Since the algorithm is probabilistic, in terms of both initialization as well as in iteration, the expected output follows a probability distribution.

To compensate for erroneous estimated vectors due to non-convexity of the problem, the algorithm is repeated with different starting vectors and the best solution is chosen based on the residual error of the estimated solutions as:

$$x_{best} = \operatorname{argmin} \left|\left|Y - Hx\right|\right|_2 \qquad x \in X_{solution}$$

P_{ML} vs. number of restarts (ML solution)

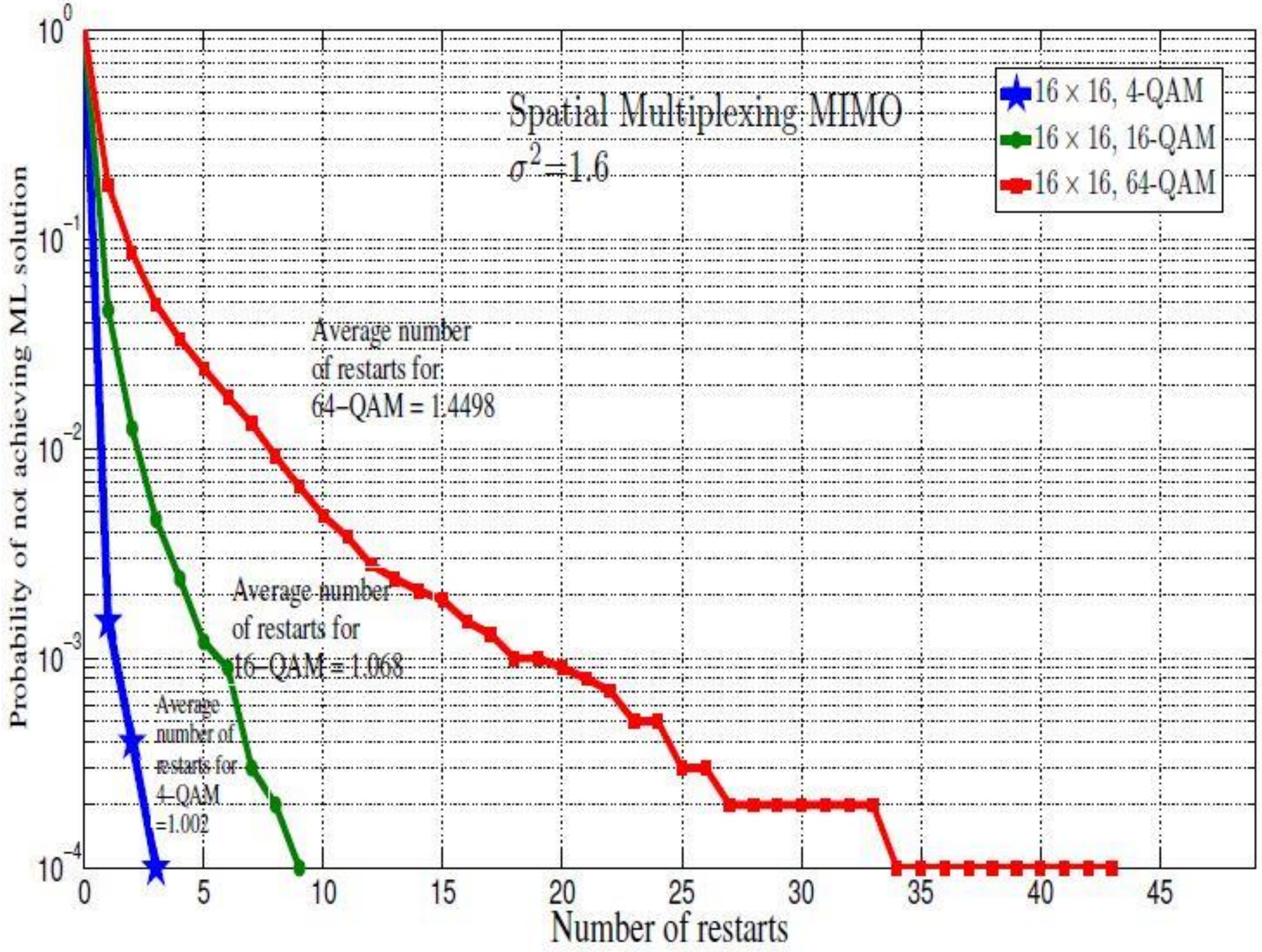


Image courtesy : T. Datta, A. Chockalingam, E. Viterbo, “Gaussian Sampling Based Lattice Decoding”

Complexity

Max # iterations is heuristically chosen to be $O(n)$. In implementation, the constant is chosen as $16.\log_2|A|$

Complexity per iteration: $O(n)$ - Due to vector-vector dot products

# of updates/iteration is $O(n)$ - each (of n) bit needs to be updated

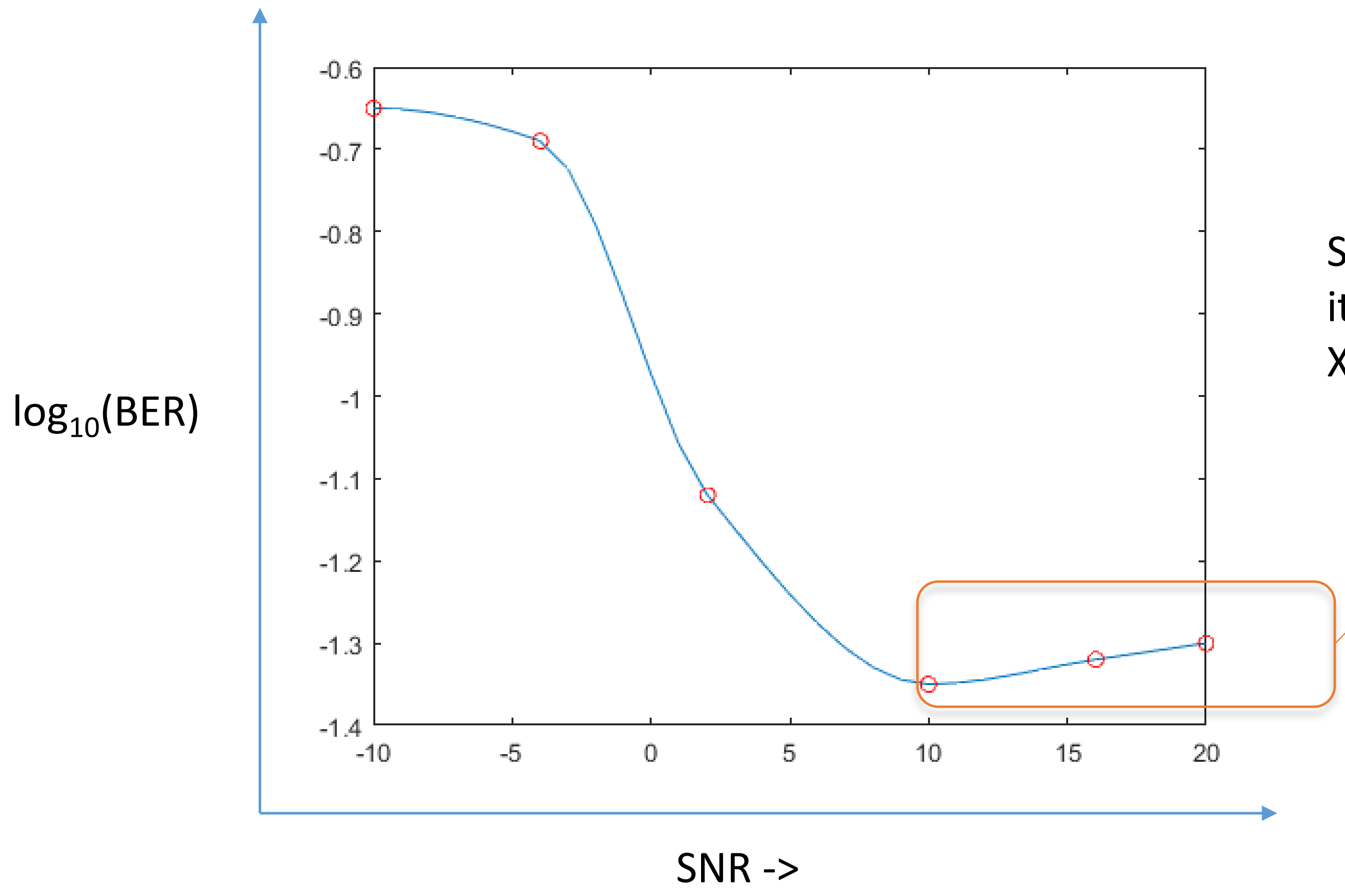
Overall computational complexity of algorithm : $O(n^3)$

Convergence

As σ->0, the solution **does not converge to the optimal solution**. The algorithm becomes deterministic (except for the random initialization), but the expected solution does not converge to the optimal $x = H^{-1}y$

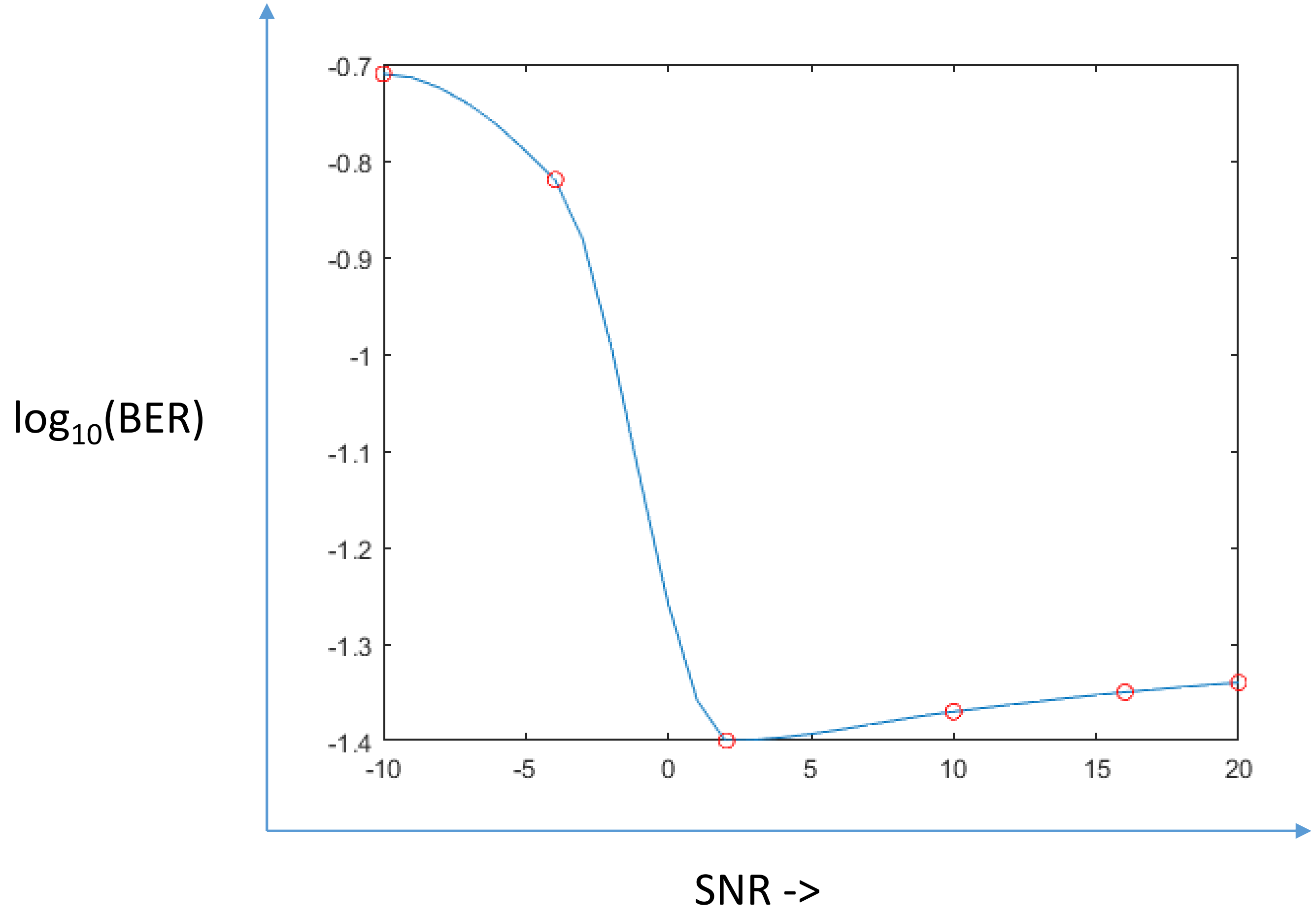
As σ becomes very large, the algorithm in principle would take a significant number of iterations to converge. This would be because the noise variance becomes significant to the distance between symbols and hence, the sampling and quantization step would not generate the same result over iterations with high probability.

BER vs. SNR 16 x 16 16 QAM



Solution is non-optimal: hence it **does not** converge to the $X = H^{-1}y$ solution as $\sigma \to 0$

BER vs SNR 32 x 32 16 QAM



Solution is non-optimal: hence it **does not** converge to the $X = H^{-1}y$ solution as σ -> 0

Comparison with other models

- Exact algorithms : ML decoding (Sphere decoder)[2].
- Sub – optimal algorithms : K-best sphere decoding algorithm.
- Fixed complexity sphere decoder
- Randomized lattice decoding[Shuiyin et.al][4].
- Other randomized algorithms [3].

[2]E. Viterbo and J. Boutros, “A universal lattice code decoder for fading channels,”
[3]T. Datta, N. A. Kumar, A. Chockalingam, and B. S. Rajan, “A novel MCMC algorithm for near-optimal detection in large-scale uplink multiuser MIMO systems,”
[4]S. Liu, C. Ling, and D. Stehle, “Randomized lattice decoding: Bridging the gap between lattice reduction and sphere decoding,”

Comparison with other models

Complexity :

ML Decoder(Sphere decoder) : Exponential Complexity. Not feasible for higher dimensions.

Comparison with other models

Complexity and SNR to achieve BER 0.01

Algorithm	Complexity in average number of real operations in $\times 10^6$ and SNR in dB required to achieve 10^{-2} BER for 16×16 MIMO			
	16-QAM		64-QAM	
	Complexity	SNR	Complexity	SNR
Prop. GSLD	0.93	16.9	4.85	23.8
R-MCMC-R [8]	1.71	17	11.18	24
R3TS [9]	3.96	17	25.42	24.2
FSD [3]	4.83	17.6	305.72	24.3

Comparison with other models

Complexity and SNR to achieve BER 0.01

Algorithm	Complexity in average number of real operations in $\times 10^6$ and SNR in dB required to achieve 10^{-2} BER for 16×16 MIMO			
	16-QAM		64-QAM	
	Complexity	SNR	Complexity	SNR
Prop. GSLD	0.93	16.9	4.85	23.8
R-MCMC-R [8]	1.71	17	11.18	24
R3TS [9]	3.96	17	25.42	24.2
FSD [3]	4.83	17.6	305.72	24.3

- GSLD > R-MCMC > FSD

Comparison with other models

Performance:

Algorithm	Complexity in average number of real operations in $\times 10^6$ and SNR in dB required to achieve 10^{-2} BER for 16×16 MIMO			
	16-QAM		64-QAM	
	Complexity	SNR	Complexity	SNR
Prop. GSLD	0.93	16.9	4.85	23.8
R-MCMC-R [8]	1.71	17	11.18	24
R3TS [9]	3.96	17	25.42	24.2
FSD [3]	4.83	17.6	305.72	24.3

- ML Decoder > GSLD > R-MCMC > FSD

BER vs SNR curve for 4QAM and 16-QAM

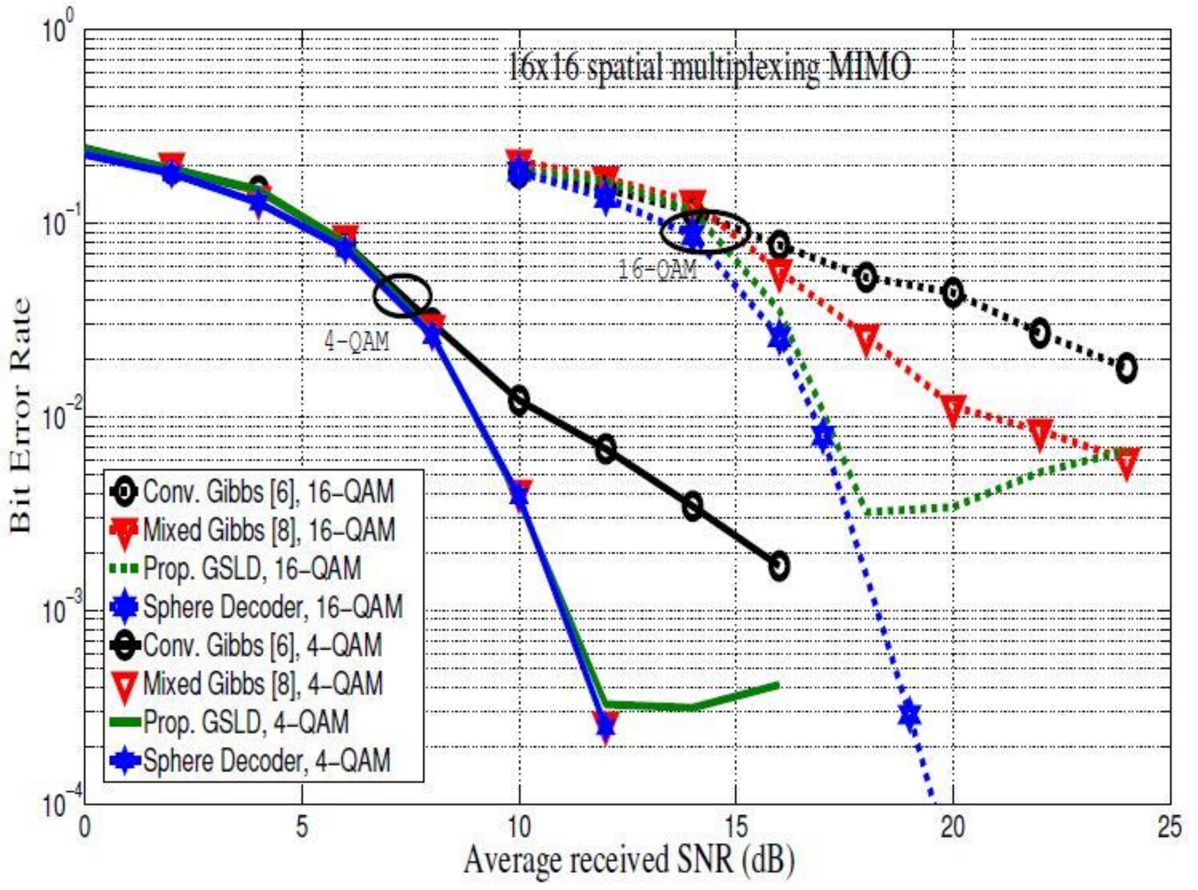


Image courtesy : T. Datta, A. Chockalingam, E. Viterbo, "Gaussian Sampling Based Lattice Decoding"

BER vs SNR curve for 16QAM and 64-QAM

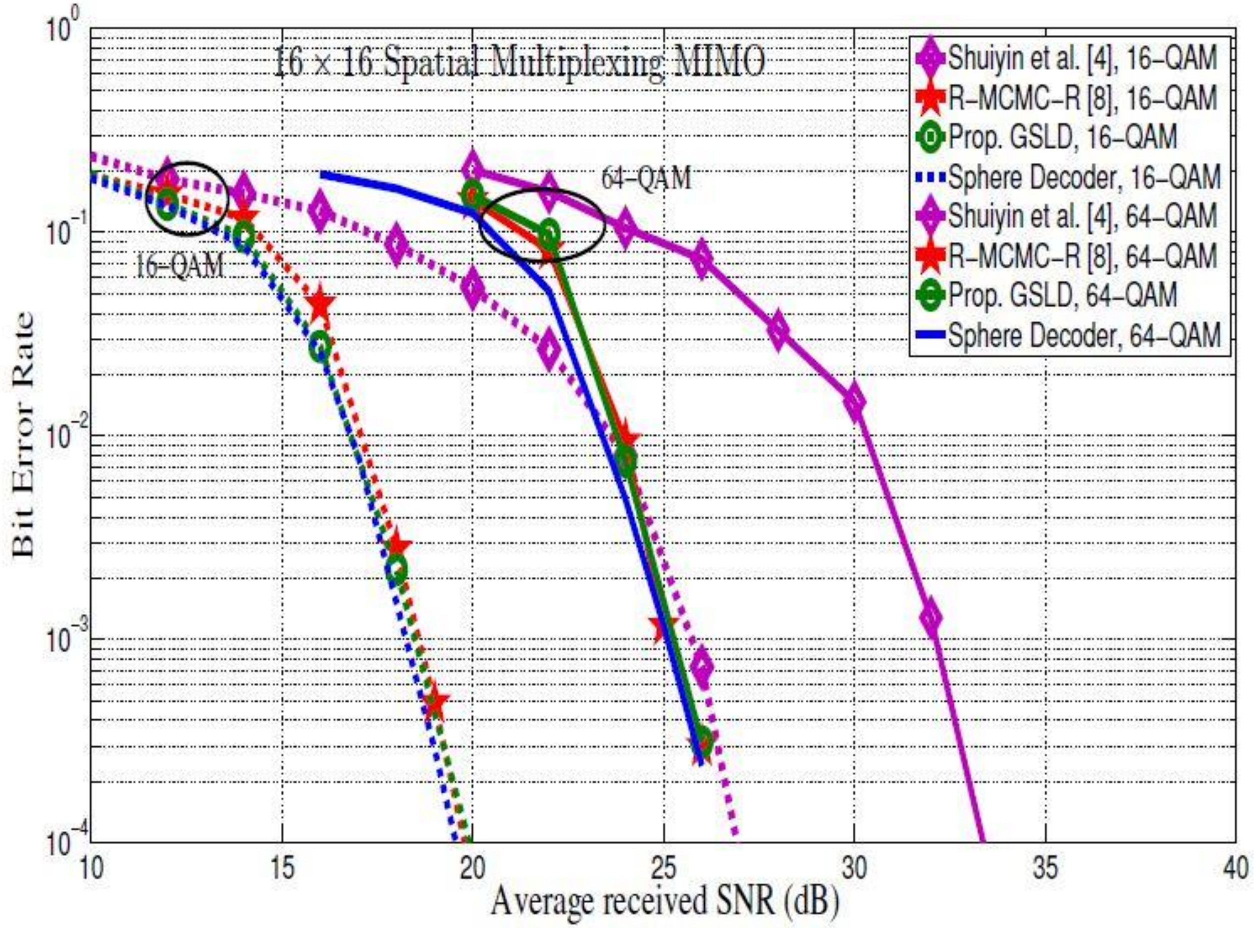


Image courtesy : T. Datta, A. Chockalingam, E. Viterbo, “Gaussian Sampling Based Lattice Decoding”

Conclusion

- The proposed GSLD performs competitively with the other existing models with much lower decoding complexity.
- The algorithm is much easier to implement as compared to other lattice decoding algorithms

References

- [1]T. Datta, A. Chockalingam, E. Viterbo, "Gaussian Sampling Based Lattice Decoding"
- [2]E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels,"
- [3]T. Datta, N. A. Kumar, A. Chockalingam, and B. S. Rajan, "A novel MCMC algorithm for near-optimal detection in large-scale uplink multiuser MIMO systems,"
- [4]S. Liu, C. Ling, and D. Stehle, "Randomized lattice decoding: Bridging the gap between lattice reduction and sphere decoding,"