Optimization methods and their applications in DSP

Ivan Tashev
Principal Architect
Microsoft Research

Tutorial outline

- Introduction
- Single parameter optimization
- Multidimensional optimization
- Practical aspects and distributed optimization
- Examples
- Conclusions and Q&A

Introduction

Mathematical optimization

- Subfields
  - Convex programming
    - Linear programming
    - Second order programming
  - Integer programming
  - Quadratic programming
  - Fractional programming
  - Non-linear programming
  - Multi-objective programming
  - Multi-modal optimization
- Optimization algorithms – typically iterative

Optimization problem

- Find the values of the optimization parameters for which the optimization criterion has minimum (maximum):
  - $\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \left(Q(\mathbf{x}) \right), \quad \mathbf{x} = \left[x_1, x_2, \ldots, x_n \right]$
  - Here $Q(\mathbf{x})$ is the optimization criterion
  - $\mathbf{x}=[x_1,x_2,\ldots,x_n]$ are the optimization parameters
- Optimization process:
  - Look around the current point
  - Find a better point
  - Repeat to the moment we can find better points

Optimization criteria

- Also called objective or cost function
- It is a single number only in very simple cases
  - In reality we have multiple requirements (sub-criteria)
- Sum: $Q = \sum_{i=0}^{M-1} w_i q_i$
- Multiplication: $Q = \prod_{i=0}^{M-1} q_i$
- Minimax: $Q = \max\left(w_i q_i\right)$
- Limiting: $q_i = \max\left(\mu_i, \min\left(m_i, q_i\right)\right)$
- Local and global minimum

Constrained optimization

- Notation:
  - $\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \left(Q(\mathbf{x}) \right)$ subject to $\begin{vmatrix} a_i \le x_i \le b_i \\ c_i \le F(\mathbf{x}) \le d_i \end{vmatrix}$
- Constraints convert the optimization problem from mathematical to real
- Parameter values constraints
  - $a_i \le x_i \le b_i$
- Intermediate values constraints
  - $c_i \le F(\mathbf{x}) \le d_i$
- Relation constraints
  - $G(\mathbf{x}) = 0$

Introducing the constraints

- Convert the constrained optimization problem into unconstrained
- Punishing functions:
  - Constant: if $x_i > b_i$ $p_j = 1e10$; else $p_j = 0$;
  - Linear: if $x_i > b_i$ $p_j = (x_i - b_i)$; else $p_j = 0$;
  - Quadratic: if $x_i > b_i$ $p_j = (x_i - b_i)^2$; else $p_j = 0$;
- Adding the punishing function to the optimization criterion:
  - $Q = Q + \sum_J \lambda_j p_j$

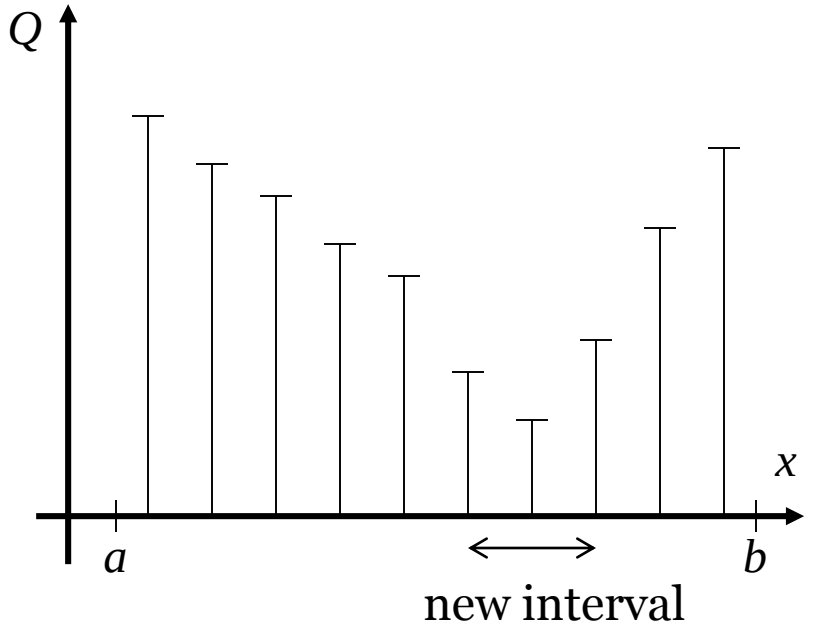
Single parameter optimization

General assumptions

- One minimum in given interval [a, b] – range of uncertainty
- The goal is to reduce the uncertainty interval – process called bracketing
- Can be iterative
- Simple algorithms, predictable performance

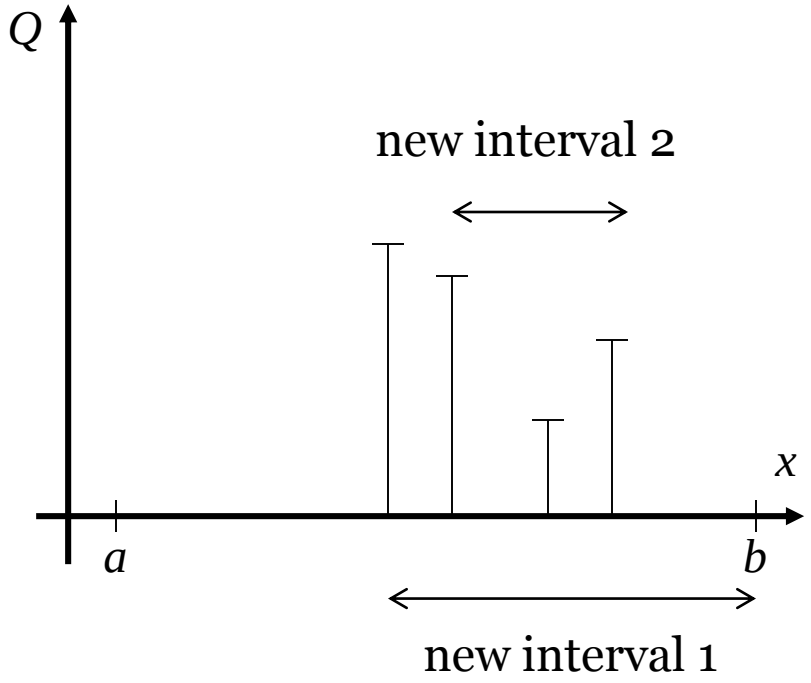
Scanning

- Compute the criterion in the middle of N equally wide intervals
- Find the minimal point
- The new interval is $f = \frac{2}{N+1}$ times smaller
- This can be repeated for another iteration. Then:
  - $N_{opt} = 3, \quad f = \left(\frac{2}{N+1}\right)^K$



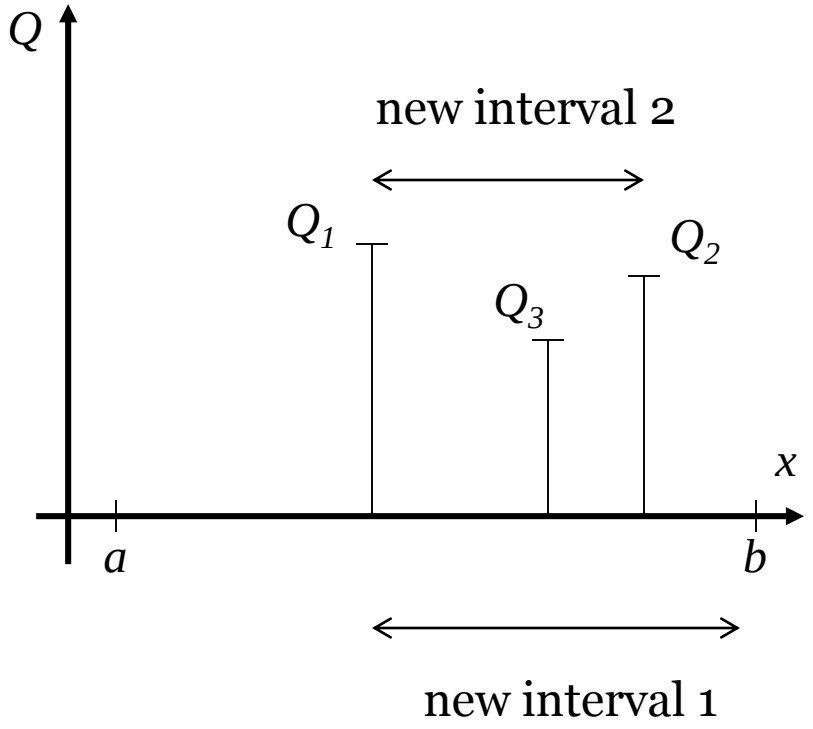
Dichotomy

- "cut in two", bi-section
- Compute the criterion in the middle of the interval in two very close points
- Select the new interval, repeat
  - $f = \left(\frac{1}{2} + \frac{\varepsilon}{2}\right)^K$



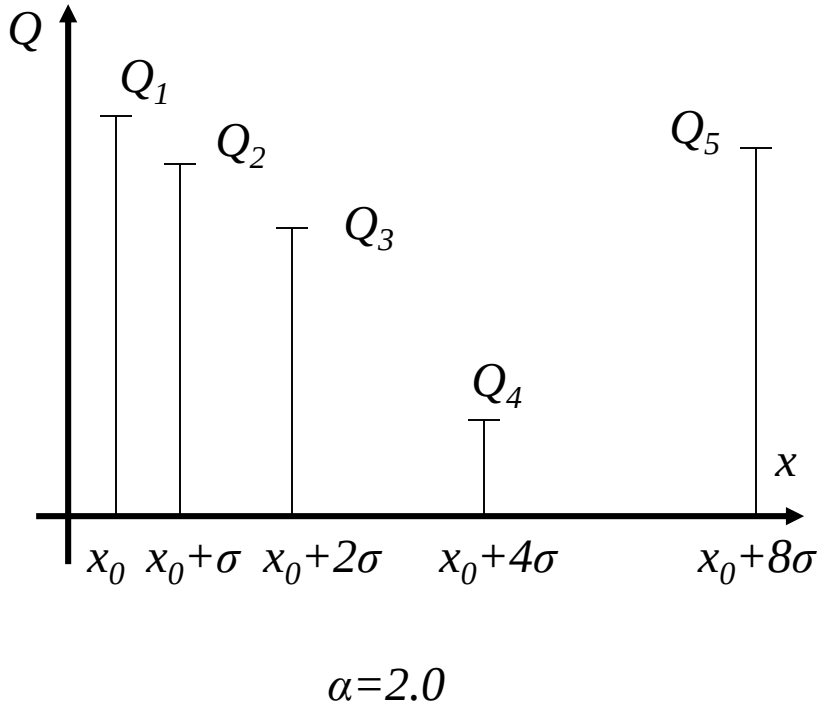
Golden section

- Golden section
  - $\frac{z_1}{z} = \frac{z_1}{z_2} \approx 0.618$
- In each iteration:
  - Divide interval into golden proportion, compute Q
  - Use the computation from the previous iteration to reduce the interval
- Performance:
  - $f = 0.618^{K-1}$



Variable step search

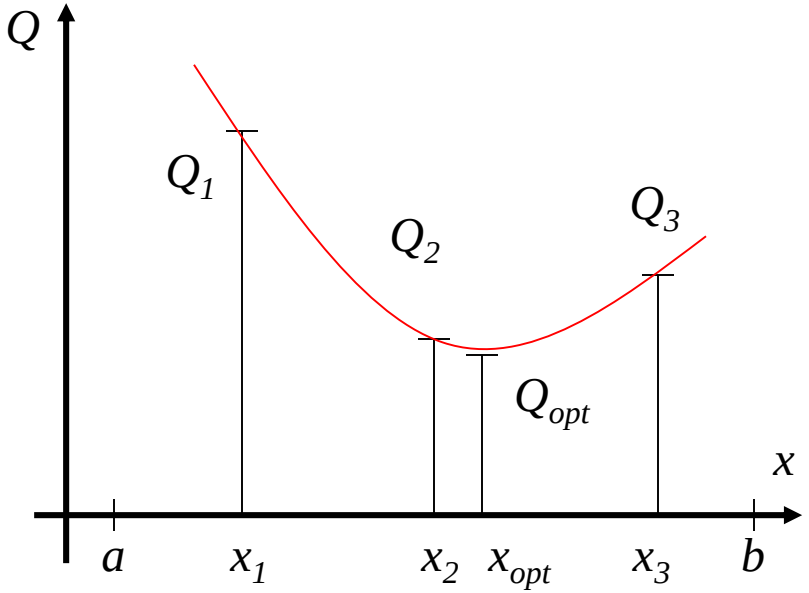
- Given start point x_0 determine the direction
- Make a step σ towards desired direction
- Increase the step if successful:
  - $\sigma_k = \alpha\sigma_{k-1}, \text{ if } Q_k > Q_{k-1}, \quad \alpha > 1$
- Stop when reached worse point
- Repeat from the new point if necessary



$\alpha = 2.0$

Quadratic interpolation

- Also known as Brent's method:
  - Compute Q in three points
  - Find a second degree polynomial going trough these three points: $Q=c_3x^2+c_2x+c_1$
  - Find the minimum: $x_{opt}= - c_2/(2c_3)$
  - Select the best solution from $[Q_1,Q_2,Q_3,Q_{opt}]$
- Good for finalizing any of the previous methods



Multidimensional optimization

General problem

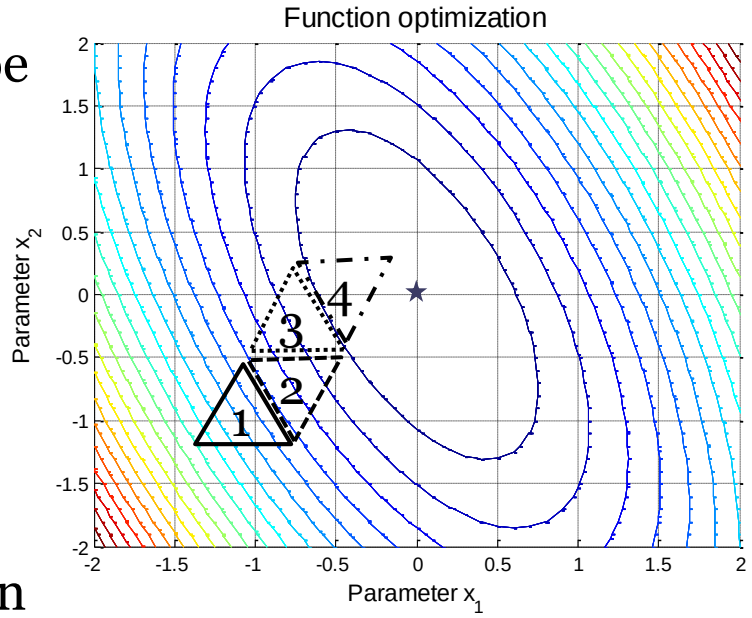
- Lower chances to have unimodal hypersurface
- Efforts increase exponentially with the number of optimization parameters
- Typically non-deterministic
- Iterative, move from the current work point to a better work point.
  - Stop when you cant find better work point
- Critical role of the starting point

Net search

- Multidimensional equivalent of the scanning method
- Compute the criterion in a grid of possible parameter values
- Select the new volume, repeat if necessary
- The only multidimensional method with deterministic performance
- Extremely inefficient, last hope to find a solution

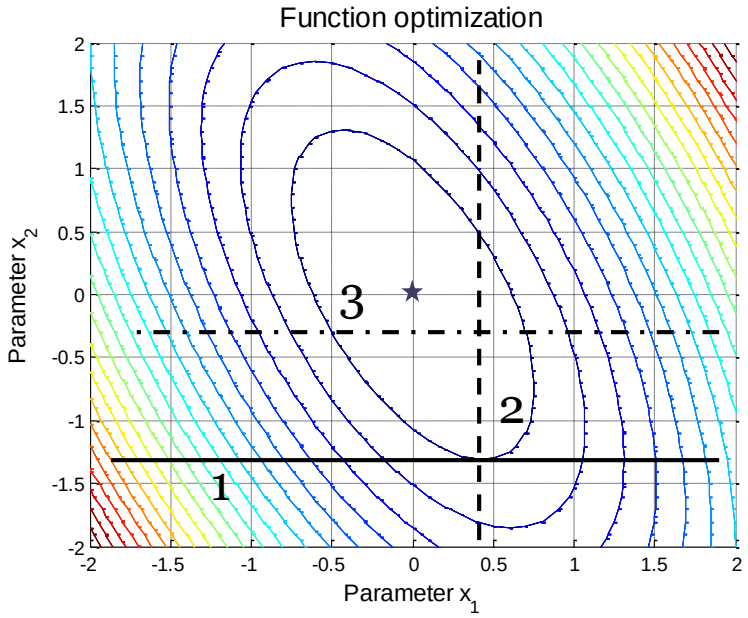
Simplex method

- Simplex: closed *n*-dimensional polytope which is a convex hull of *n*+1 vertices
  - Triangle (2D), tetrahedron (3D), pentachoron (4D), …
- For each iteration:
  - Compute the criterion in vertices
  - Find vertex symmetric to the worst
  - Throw away the worst one
- Stop when the symmetric vortex is even worse
- Variant: variable size simplex
  - Increase it when you have better point, decrease otherwise: reflection and expansion or reflection and contraction



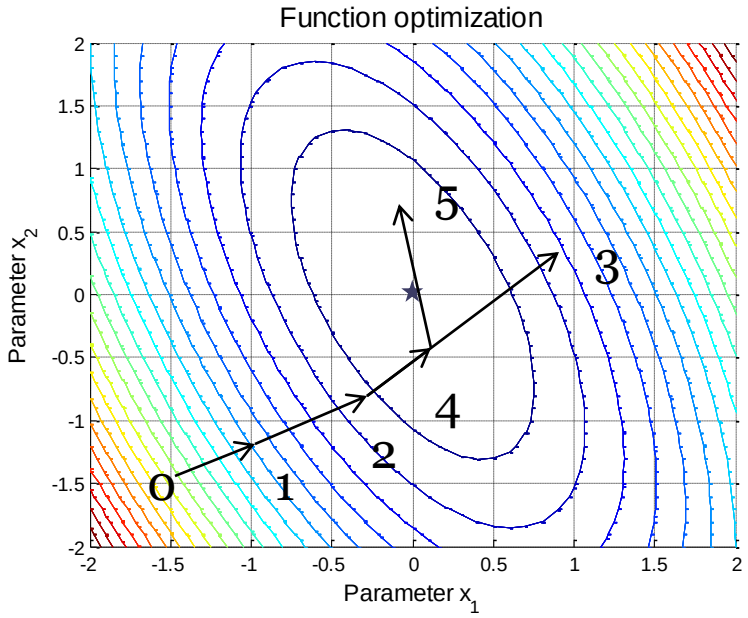
Gaussian optimization

- Fix all parameters, except one, perform single dimensional optimization
- Repeat for the rest of the parameters
- If they are statistically independent – one iteration is sufficient
- Repeat the iteration otherwise



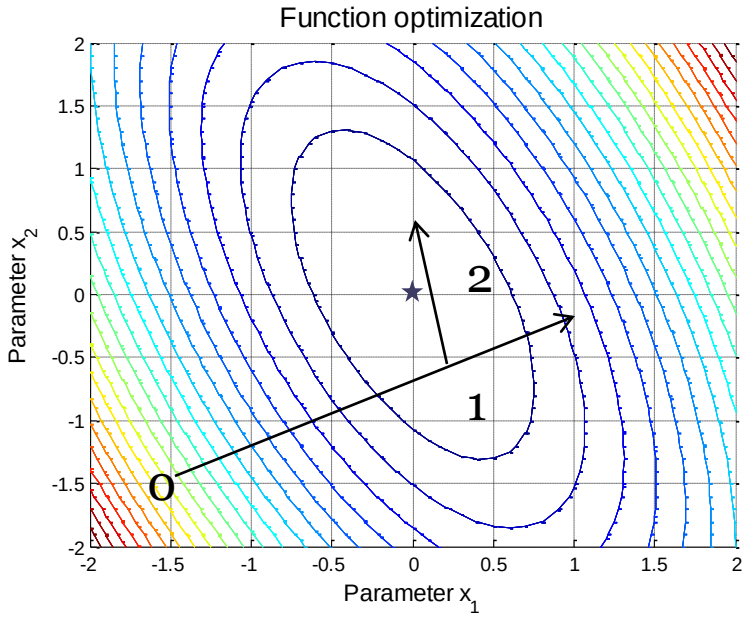
Gradient methods

- The gradient is a vector pointing towards steepest increase
  - $\vec{\nabla} Q = \frac{\delta Q}{\delta x_1}\vec{e}_1 + \frac{\delta Q}{\delta x_2}\vec{e}_2 + \cdots + \frac{\delta Q}{\delta x_n}\vec{e}_n$
- For each iteration
  - Compute the gradient
  - Make a step in the opposite direction, change the work point if successful
  - Increase the step size if successful, decrease otherwise
  - Stop when the step is smaller than given value



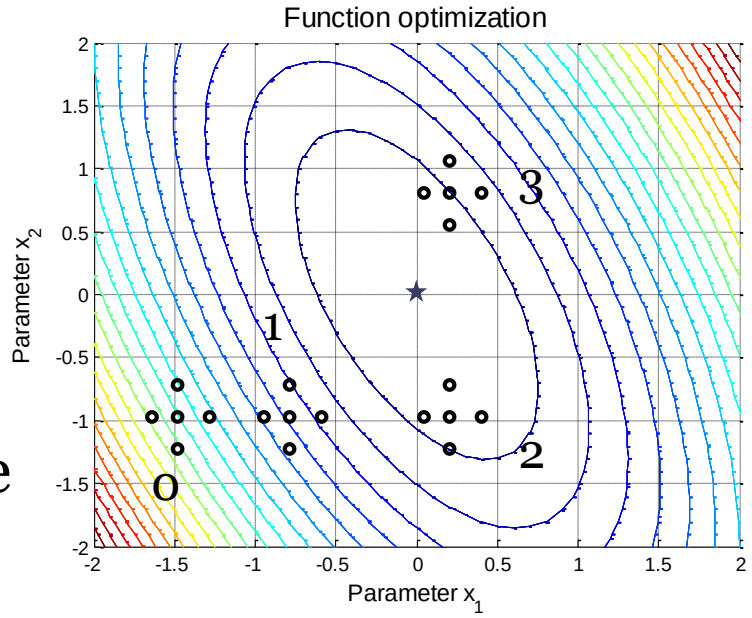
Gradient methods (2)

- Steepest gradient descent
  - Do single dimensional search towards the gradient
- Countless other variations
- Requires analytical derivatives, sensitive to the gradient estimation precision
- Still: $\frac{\delta Q}{\delta x_i} \approx \frac{Q(\mathbf{x}) - Q(\mathbf{x} + \delta_i)}{\delta}$



Configuration method

- A configuration: the work point plus the neighbors
- Compute a configuration, move the work point towards the best point
- Vary the step size to increase the speed
- Stop when the best point is in the middle



Variable metric methods in multiple dimensions

- Quasi-Newton solvers in multiple dimensions
  - Find the root of the gradient function
  - Typically estimate Hessian matrix
- Come in two main flavors:
  - Davidon-Fletcher-Powel (DFP)
  - Broyden-Fletcher-Goldfarb-Shano (BFGS)

Methods comparison: example

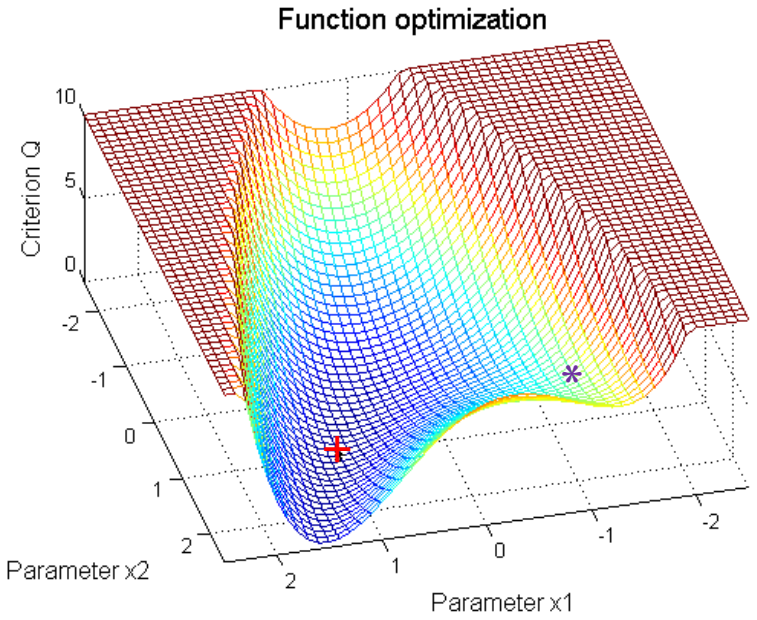
- Rosenbrock's function, k=1:

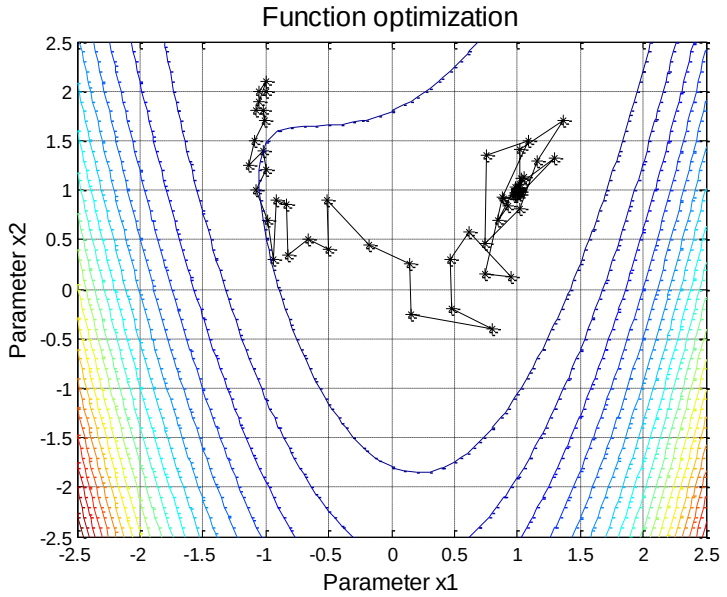
$$Q(x_1, x_2) = k\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

  - $$\frac{\delta Q}{\delta x_1} = -4k\left(x_2 - x_1^2\right)x_1 - 2\left(1 - x_1\right)$$

$$\frac{\delta Q}{\delta x_1} = 2k\left(x_2 - x_1^2\right)$$

- Matlab functions:
  - Simplex method `fminsearch()`
  - Gradient method `fminunc()`
- Gradient method with variants:
  - `findMin()`
- Gaussian method with variants:
  - `findMinGauss()`
- Gaussian search with variants:
  - `findMinGaussSearch()`





Methods comparison: example (2)

Class	Algorithm	Function	Q	Q comp.
Simplex	Matlab	`fminsearch()`	1.02E-13	119
Gradient	Matlab	`fminunc()`	8.72E-19	32
FinDiff			5.15E-15	47
Gradient	variable step	`findMin()`	7.44E-06	194
	steepest descent		5.71E-16	127
	descent, no interpolation		7.81E-06	626
	descent, with interpolation		1.02E-09	154
Gaussian	bracket and select	`findMinGauss()`	4.01E-06	348
	bracket and interpolate		3.42E-06	837
	bracket and golden section		4.28E-14	129
Gaussian	seacrh 60 pts, no interpolation	`findMinGaussSearch()`	6.29E-02	369
	search 20 pts, w/ interpolation		0.00E+00	487

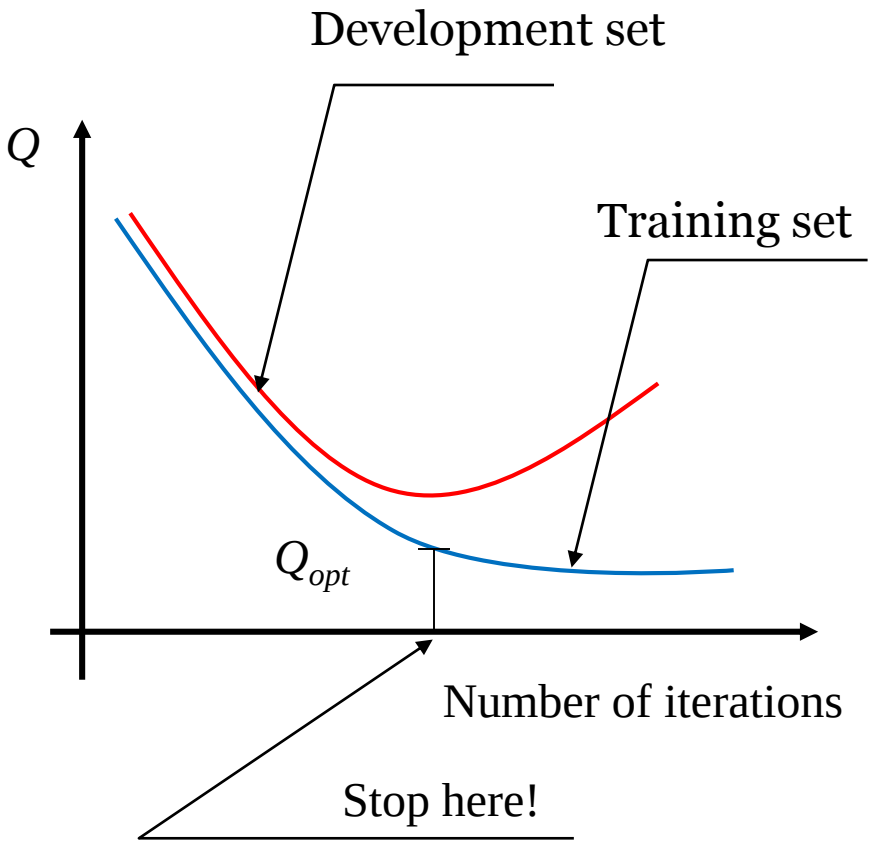
Practical Considerations and Distributed optimization

Practical considerations

- Limiting the search scope
  - Every engineering problem deal with limited parameter values
- Limiting the values of the internal variables
- Limiting the precision
  - Don’t go for too small improvements
- Dealing with discrete values of the parameters
  - Use Gaussian search
- Dealing with the discrete values of the criterion
  - Modified `min()` function in Matlab
- Dynamic range of the variables
  - Scaling to magnitudes of 1’s or 10’s

Training, development, and test sets

- Split the data corpus on three
- The training set is used to compute the optimization criterion
- The development set is computed after each iteration
- Test set is for the final evaluation

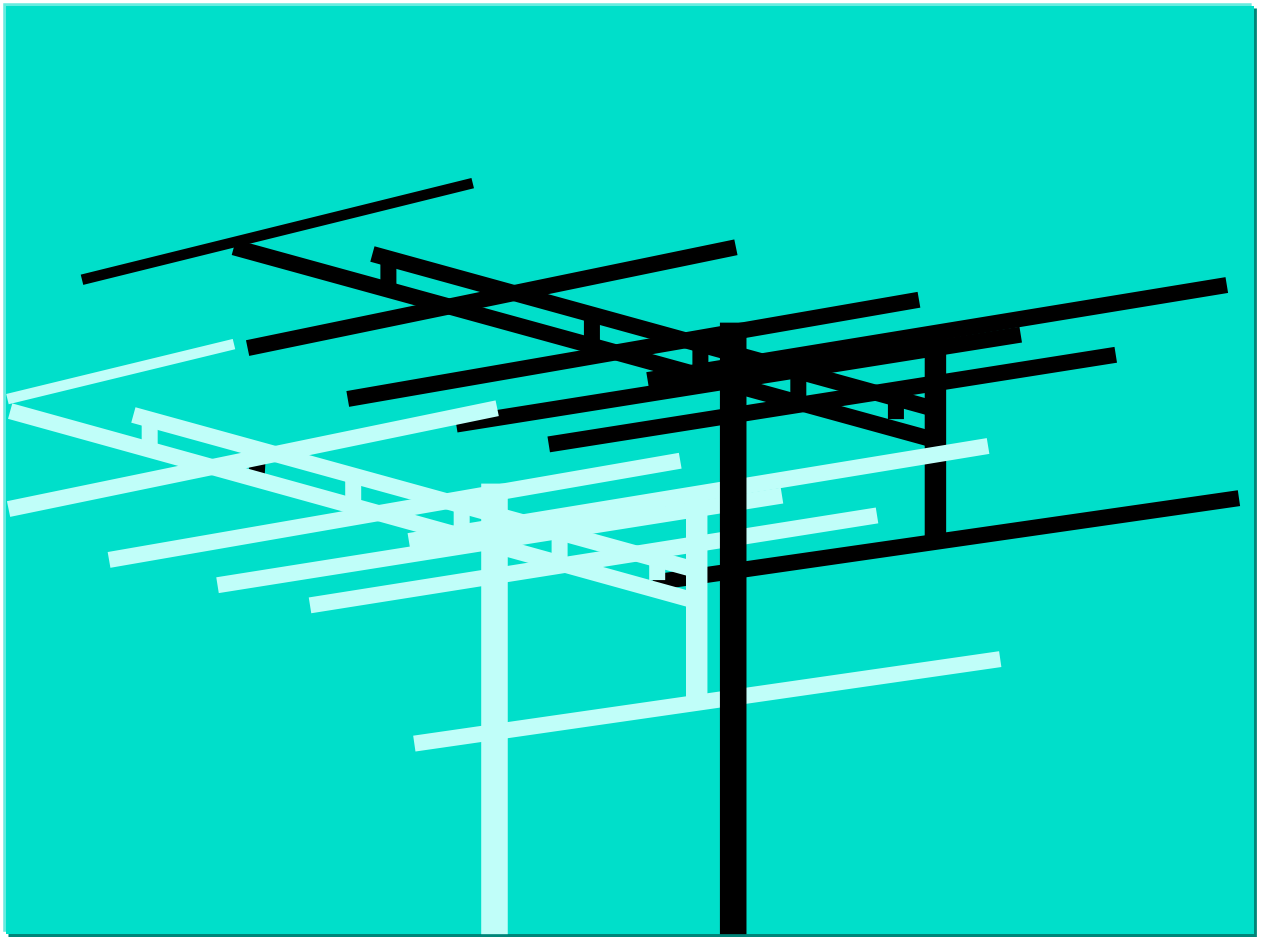


Parallelization of the optimization process

- Parallelization:
  - Running multiple threads on a multi-CPU and/or multicore-CPU
  - Using a computing cluster
- Parallelization of the optimization criterion
  - Run job for each of the components of the data corpus
- Parallelization of the optimization process
  - Computing the gradient in parallel
  - Doing search for many points at once
    - Good for steepest gradient descent and Gaussian methods
- Parallelization has limits
  - Robust code with retries when some job fails
  - Network search still takes a lot of time ☹

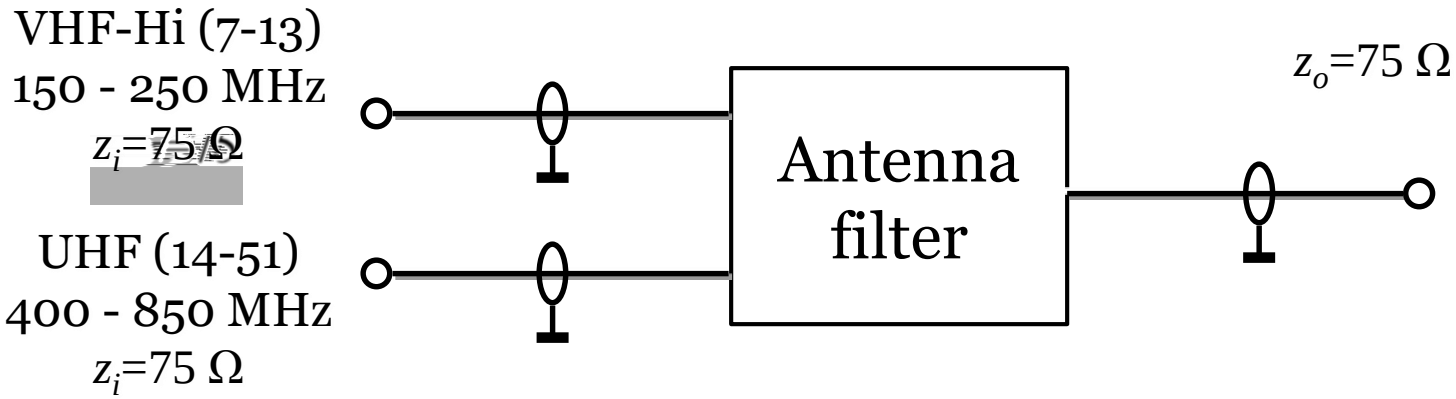
Examples

Simple antenna filter



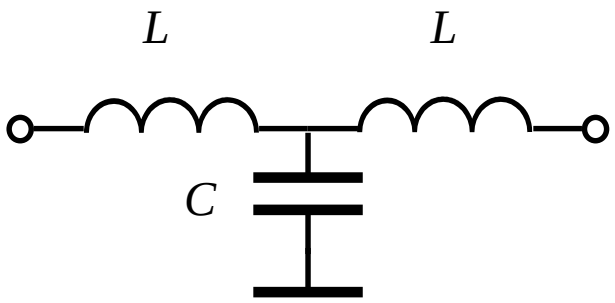
Simple antenna filter: problem formulation

- Need to connect two antennas to one TV



Pass band: higher than -5 dB
Stop band: lower than -12 dB
Input and output impedances: 75 Ω ± 10 Ω

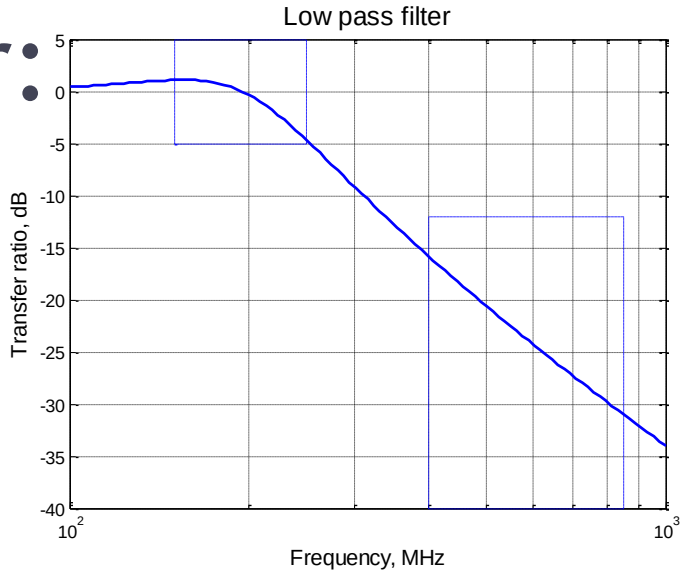
Simple antenna filter: low pass filter

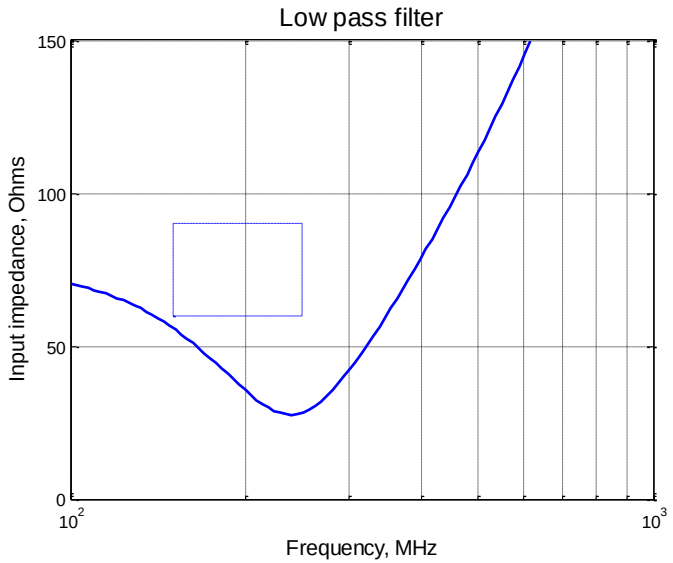


$$L = \frac{z}{2\pi F_S} \qquad C = \frac{2}{2\pi F_S z} \qquad z = \sqrt{\frac{L}{C}}$$

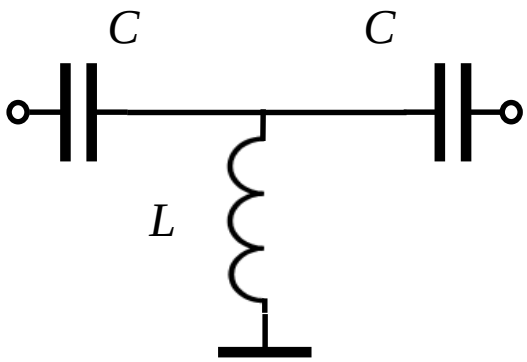
$$z = 75\ \Omega \qquad F_S = 275\ \text{MHz}$$

$$L = 43.4\ \text{nH}$$
$$C = 15.4\ \text{pF}$$





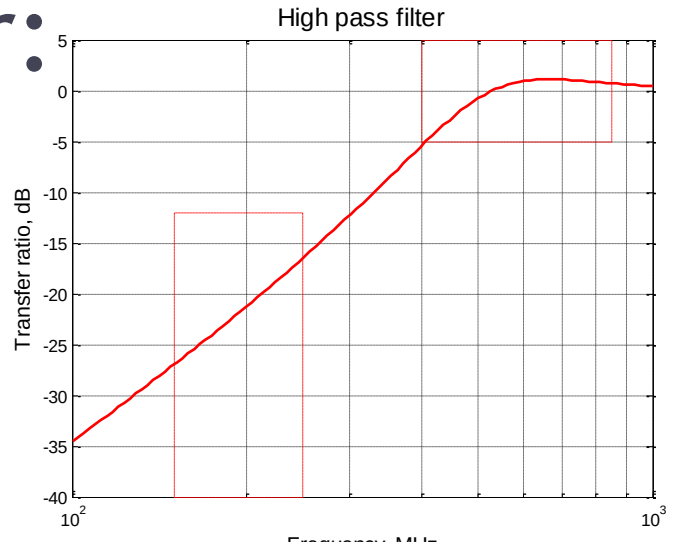
Simple antenna filter: high pass filter

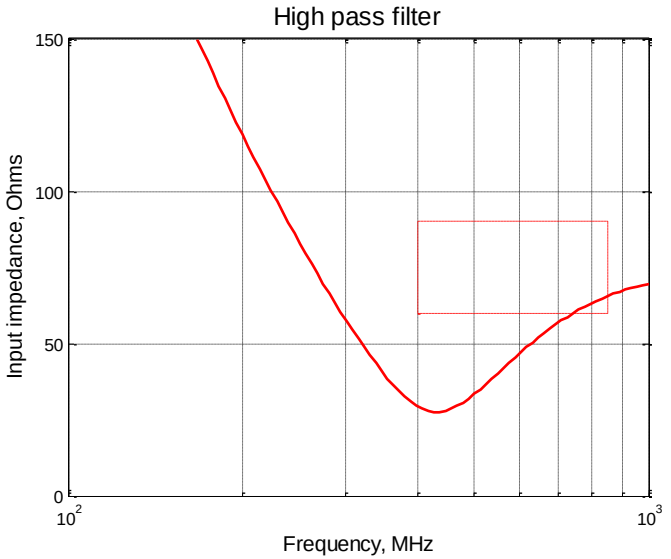


$$L = \frac{z}{4\pi F_S} \quad C = \frac{1}{2\pi F_S z} \quad z = \sqrt{\frac{L}{C}}$$

$z = 75\ \Omega$ $F_S = 375$ MHz

$L = 15.9$ nH
$C = 5.7$ pF

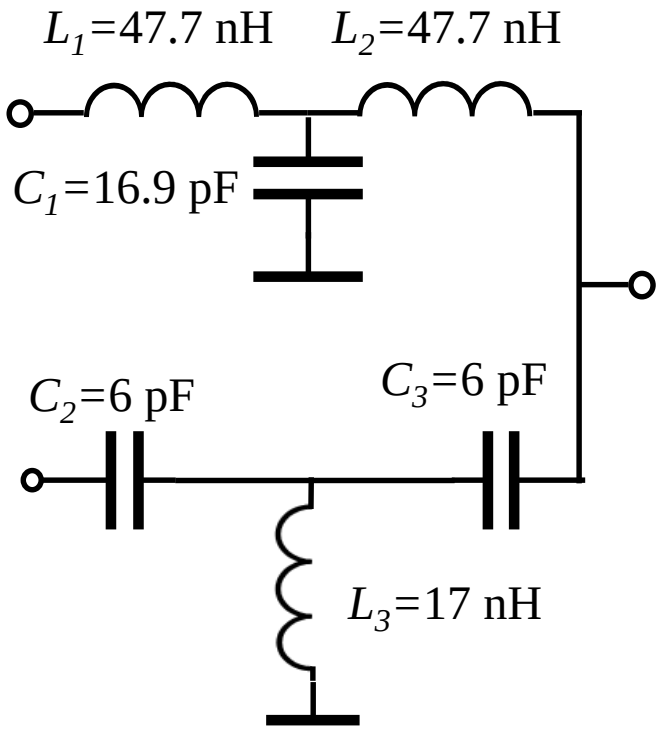




Simple antenna filter: full diagram

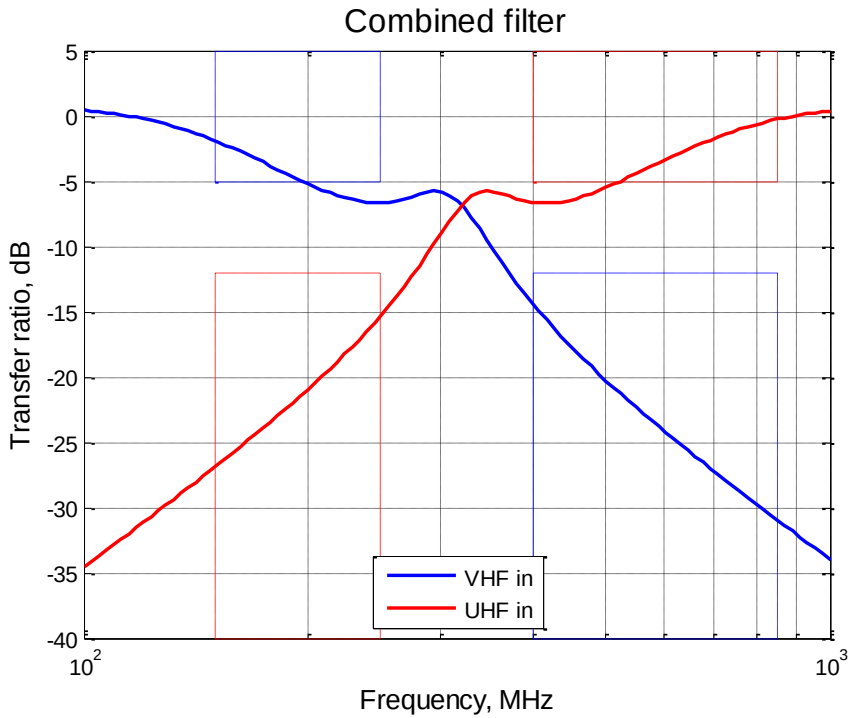
Input 1
150 - 250 MHz
z_i=75 Ω

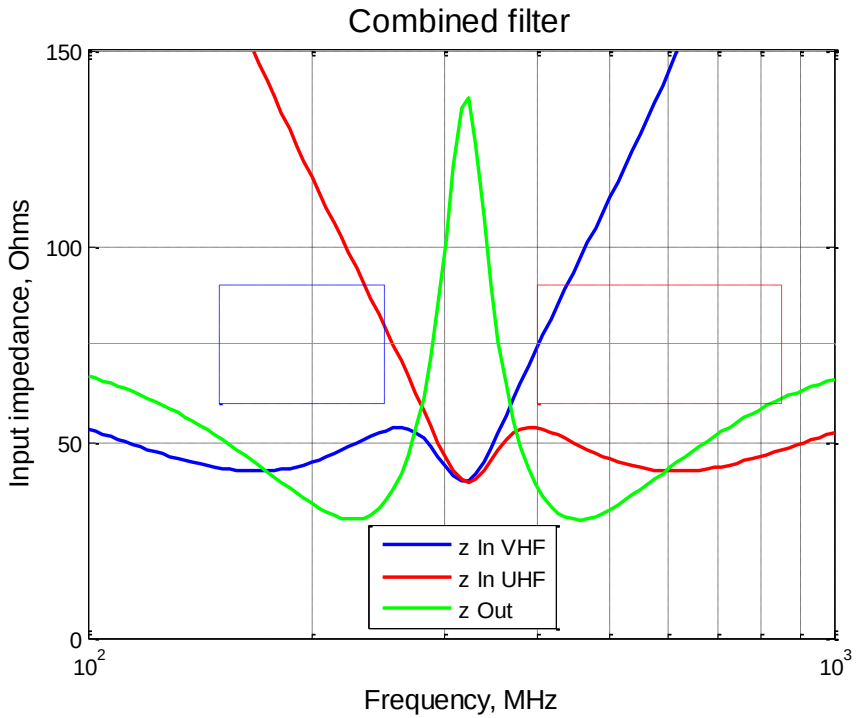
Input 2
400 - 850 MHz
z_i=75 Ω



Output
z_o=75 Ω

Simple antenna filter: staring point





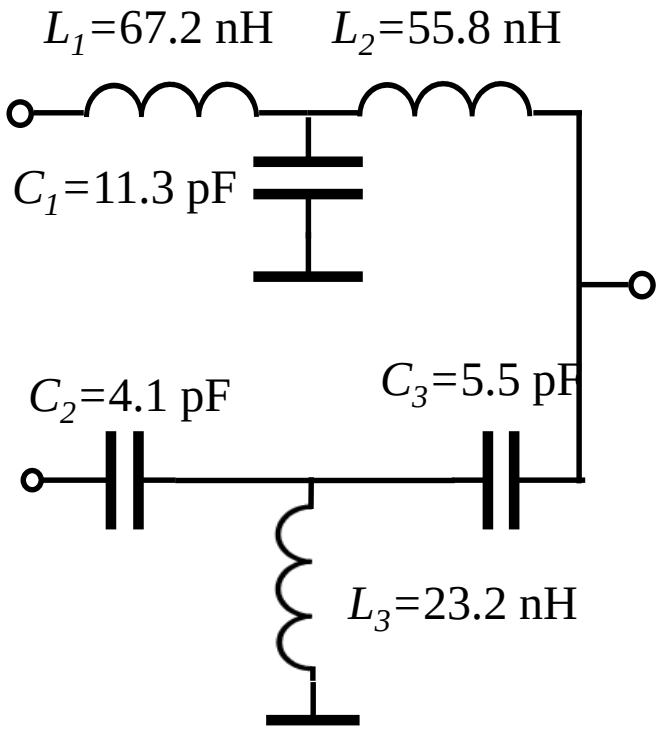
Simple antenna filter: optimization

- Optimization parameters: L_1, L_2, C_1, C_2, C_3, L_3.
- Optimization criterion: weighted sum of 8 sub-criteria
  - Impedance match: VHF and UHF inputs, Output VHF, Output UHF
  - Frequency response: VHF and UHF inputs, stop and pass bands
  - Weights: [0.1 0.1 2.0 2.0 20.0 10.0 20.0 10.0]
- Starting point: estimated with the filter design
- Constraints:
  - $3nH \le L_i \le 300nH$
  - $1pF \le C_i \le 100\,pF$
- $Q_o = 16.801$, $Q_{end} = 2.898$

Simple antenna filter: optimization results

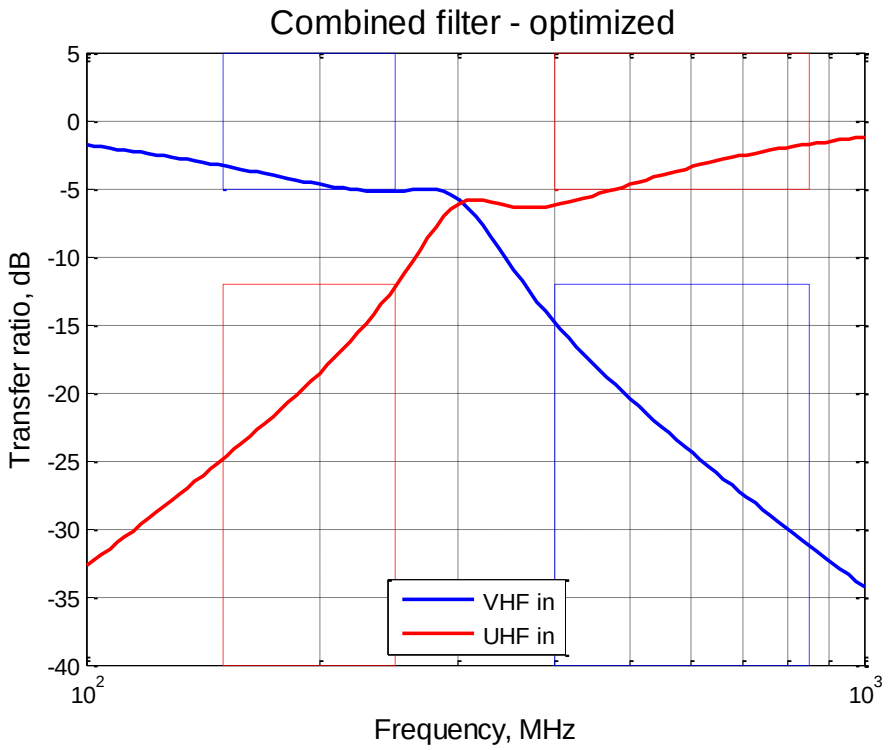
Input 1
150 - 250 MHz
z_i=75 Ω

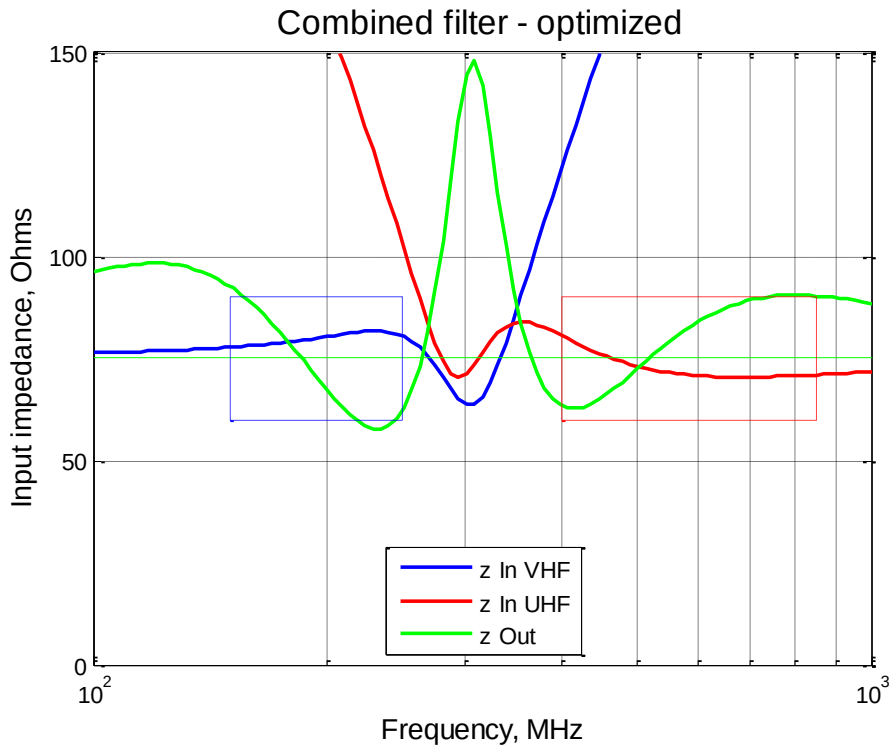
Input 2
400 - 850 MHz
z_i=75 Ω



Output
z_o=75 Ω

Simple antenna filter: after optimization



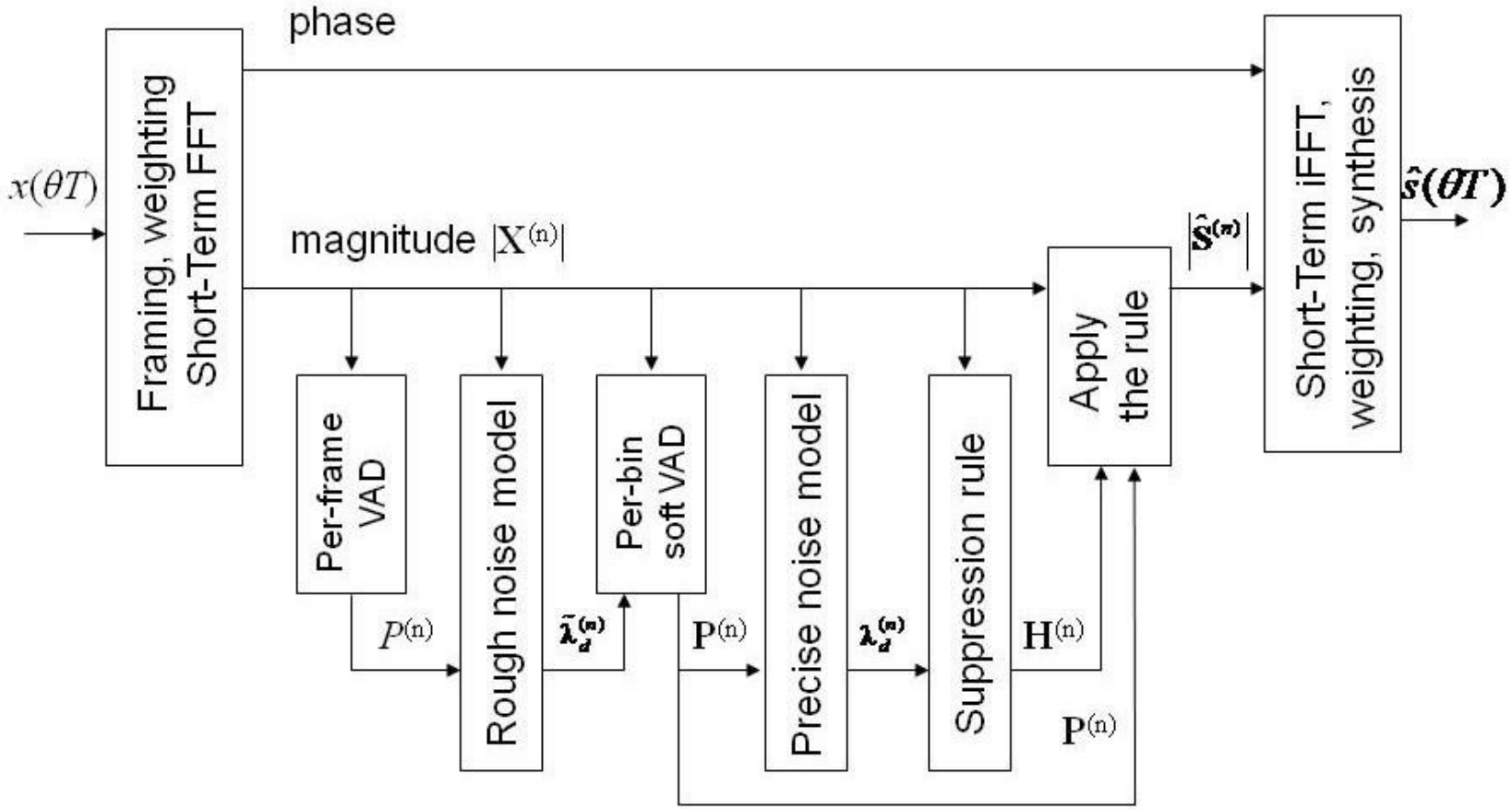


Simple antenna filter: methods comparison

Class	Algorithm	Function	Q	Q comp.
Simplex	Matlab	`fminsearch()`	2.965	826
Gradient	Matlab	`fminunc()`	2.964	1962
FinDiff			2.964	310
Gradient	variable step	`findMin()`	2.974	9507
	steepest descent		2.976	5108
	descent, no interpolation		2.976	9558
	descent, with interpolation		2.975	6764
Gaussian	bracket and select	`findMinGauss()`	2.921	279
	bracket and interpolate		2.898	380
	bracket and golden section		2.903	666
Gaussian	seacrh 60 pts, no interpolation	`findMinGaussSearch()`	2.986	1467
	search 20 pts, with interpolation		2.971	795

Noise suppressor optimization

Noise suppressor architecture



Per-frame VAD and rough noise model

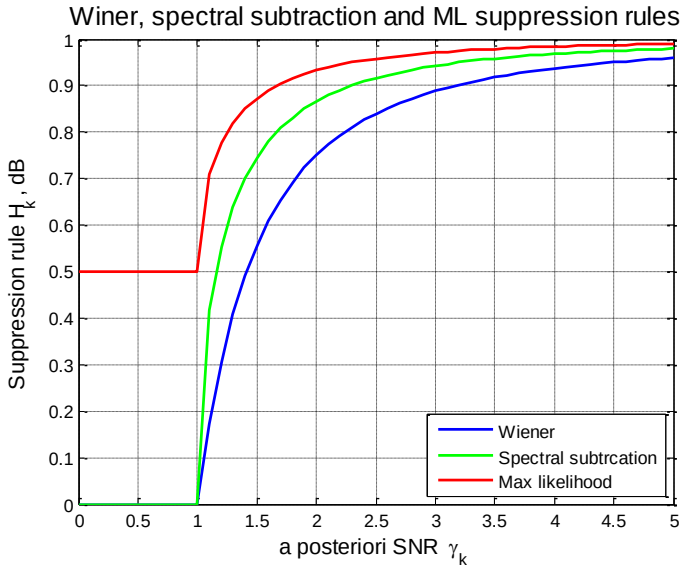
- Filter to maximize SNR
  - $L^{(n)} = \sum_{k=0}^{K-1} \left(W_k . \left| X_k^{(n)} \right| \right)^2 \qquad \mathbf{W} = \arg\max_{\mathbf{W}} \left[\sum_{k=0}^{K-1} \left(W_k . \left| \bar{S}_k^{(n)} \right| \right)^2 - \sum_{k=0}^{K-1} \left(W_k . \left| \bar{D}_k^{(n)} \right| \right)^2 \right]$
- Apply double time constant integrator to track the noise
  - $L_{\min}^{(n)} = \begin{cases} \left(1 - \frac{T}{\tau_{up}}\right) L_{\min}^{(n-1)} + \frac{T}{\tau_{up}} L^{(n)} & L^{(n)} > L_{\min}^{(n-1)} \\ \left(1 - \frac{T}{\tau_{down}}\right) L_{\min}^{(n-1)} + \frac{T}{\tau_{down}} L^{(n)} & L^{(n)} \le L_{\min}^{(n-1)} \end{cases} \qquad V^{(n)} = \begin{cases} \tilde{\mathrm{H}}_0 & if \quad L^{(n)} / L_{\min}^{(n)} < \eta_{down} \\ \tilde{\mathrm{H}}_1 & if \quad L^{(n)} / L_{\min}^{(n)} > \eta_{up} \\ V^{(n-1)} & \text{otherwise} \end{cases}$
- Update the rough noise model when $V^{(n)}=H_0$:
  - $\tilde{\lambda}_d^{(n)}(k) = \tilde{\lambda}_d^{(n-1)}(k) + \frac{T}{\tau_r} \left(\left| \bar{X}_k^{(n)} \right|^2 - \tilde{\lambda}_d^{(n-1)}(k) \right)$

Soft VAD and precise noise model

- From the PDFs of the noise $p(X_k|\mathrm{H}_0)$ and the speech $p(X_k|\mathrm{H}_1)$ compute the likelihood ratio (Sohn et all, 1998):
  - $$\Lambda_k \triangleq \frac{p(X_k \mid \mathrm{H}_1)}{p(X_k \mid \mathrm{H}_0)} = \frac{1}{1+\xi_k}\exp\left(\frac{\gamma_k \xi_k}{1+\xi_k}\right)$$
- Smooth using first order HMM
  - $$\Gamma_k^{(n)} = \frac{a_{01} + a_{11}\Gamma_k^{(n-1)}}{a_{00} + a_{10}\Gamma_k^{(n-1)}}\Lambda_k^{(n)}$$
- Compute the speech presence probability
  - $$P_k^{(n)} = \frac{\widehat{\Lambda}_k^{(n)}}{1+\widehat{\Lambda}_k^{(n)}} \qquad \widehat{\Lambda}_k^{(n)} = \frac{P(\mathrm{H}_0)}{P(\mathrm{H}_1)}.\Gamma_k^{(n)}$$
- Update the precise noise model
  - $$\lambda_{d,k}^{(n)} = \lambda_{d,k}^{(n-1)} + \left(1-P^{(n)}\right)\left(1-P_k^{(n)}\right)\frac{T}{\tau_p}\left(\left|X_k^{(n)}\right|^2 - \lambda_{d,k}^{(n-1)}\right).$$

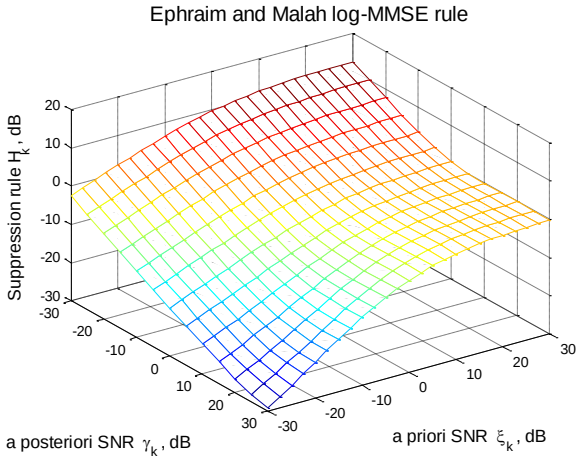
Suppression rules

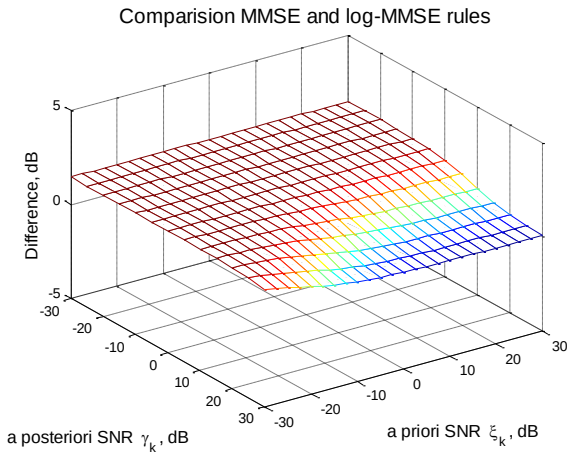
- Prior and posterior SNRs:
  - $\xi_k \triangleq \frac{\lambda_s(k)}{\lambda_d(k)}, \gamma_k \triangleq \frac{|X_k|^2}{\lambda_d(k)}$
  - $\lambda_d(k) \triangleq E\left\{|D_k|^2\right\} \quad \lambda_s(k) \triangleq E\left\{|S_k|^2\right\}$
- MMSE, Wiener (1947)
  - $H_k = \frac{\lambda_s(k)}{\lambda_s(k)+\lambda_d(k)} = \frac{\xi_k}{1+\xi_k}$
- Spectral subtraction, Boll (1975): $H_k = \sqrt{\frac{\xi_k}{1+\xi_k}}$
- Maximum Likelihood, McAulay & Malpass (1981):
  - $H_k = \frac{1}{2} + \frac{1}{2}\sqrt{\frac{\xi_k}{1+\xi_k}}$



Suppression rules (2)

- ST-MMSE, Ephraim & Malah (1984):
  - $$H_k = \frac{\sqrt{\pi \nu_k}}{2\gamma_k}\left[(1+\nu_k)\mathrm{I}_0\left(\frac{\nu_k}{2}\right)+\nu_k\,\mathrm{I}_1\left(\frac{\nu_k}{2}\right)\right]\exp\left(\frac{\nu_k}{2}\right)$$
  - $$\nu(k) \triangleq \frac{\xi_k}{1+\xi_k}\gamma_k$$
- ST-logMMSE, Ephraim & Malah (1985):
  - $$H_k = \frac{\xi_k}{1+\xi_k}\left\{\frac{1}{2}\int_{\upsilon_k}^{\infty}\frac{\exp(-t)}{t}dt\right\}$$
- Efficient alternatives, Wolfe&Godsill (2001):
  - Joint Maximum A Posteriori Spectral Amplitude and Phase (JMAP SAP) Estimator
  - Maximum A Posteriori Spectral Amplitude (MAP SA) Estimator
  - MMSE Spectral Power (MMSE SP) Estimator





Prior SNR estimation and rule modifier

- Prior SNR using decision directed approach, Ephraim & Malah (1984):
  - $\xi_k^{(n)} = \alpha \frac{\left|\hat{S}_k^{(n-1)}\right|^2}{\lambda_d^{(n)}} + (1-\alpha).\max\left[0,\left(\gamma_k^{(n)} - 1\right)\right]$
- Uncertain presence of the speech signal
  - Pauses are integral part of the speech signal
  - McAulay & Malpass (1981):
    - $\tilde{H}_k^{(n)} = P_k^{(n)}.H_k^{(n)}$
  - Ephraim & Malah (1984):
    - $\tilde{H}_k = \frac{\Lambda(X_k, q_k)}{1+\Lambda(X_k, q_k)}.H_k \qquad \Lambda(Y_k, q_k) = \mu_k \frac{P(X_k \mid \mathrm{H}_1)}{P(X_k \mid \mathrm{H}_0)} \qquad \mu_k \triangleq (1-q_k)/q_k$
  - Middleton & Esposito (1968)

Practical aspects

- Avoid division by zero ☺:
  - Check for, or add a small constant
- Avoid overflow:
  - exp(max(700,x))
  - ln(x+ε)
- Limit the values in realistic boundaries:
  - probabilities [0.001,0.999], etc.
- Smooth in time
  - First order integrator
  - HMM based
- Limiting suppression gains
  - $\tilde{H}_k^{(n)} = \left[G_{Um} + \left(1 - G_{Um}\right) P_k^{(n)} \right] \left[G_m + \left(1 - G_m\right) H_k^{(n)} \right]$

Tuning as optimization problem

- Numerous parameters that can't be computed
  - Adaptation time constants, thresholds, feedback parameters
- Formulate good optimization criterion
  - "Better" noise suppressor? SNR? MMSE? LSD? ML? log-MMSE?
    - We actually don't want MMSE, or ML, or even log-MMSE estimators
    - We want humans to perceive the output signal as "better" quality
  - Perceptual quality
    - Mean Opinion Score (MOS), ITU.T Recommendation P.800
    - Perceptual Evaluation of Sound Quality (PESQ)
      - Computational proxy of MOS measurements
      - ITU-T Recommendation P.864.2
  - Combined optimization criterion:
    - Q = w1*PESQ + w2*SNR - w3*LSD - w4*MMSE

Tuning as optimization problem (2)

- Generate proper data corpus (synthetic vs. real recordings)
  - Measure the impulse responses between various positions of the driver's head (9) and microphone (12)
  - Record noise corpus under various driving conditions (5x3)
  - Clean speech corpus: TMIT (6300 utterances)
  - Combine randomly, correct for the Lombard effect, split on 3
- Perform the optimization
  - Gaussian minimization
  - Parallelized for execution on 240 CPUs
- See Tashev, Lovitt, Acero – PacRim 2009 for details

Tuning as optimization problem (2)

- Generat…eal recordin…
  - Measu…sitions of the driver's…
  - Record…ns (5x3)
  - Clean s…
  - Combi… split on 3
- Perform…
  - Gaussi…
  - Parall…
- See Tashev, Lovitt, Acero – PacRim 2009 for details

Tuning as optimization problem (2)

- Generate proper data corpus (synthetic vs. real recordings)
  - Measure the impulse responses between various positions of the driver's head (9) and microphone (12)
  - Record noise corpus under various driving conditions (5x3)
  - Clean speech corpus: TMIT (6300 utterances)
  - Combine randomly, correct for the Lombard effect, split on 3
- Perform the optimization
  - Gaussian minimization
  - Parallelized for execution on 240 CPUs
- See Tashev, Lovitt, Acero – PacRim 2009 for details

Results

- Suppression rule independent parameters

τ_{down}	τ_{up}	η_{down}	η_{up}	τ_r	τ_p	a_{01}	a_{10}	b_{01}	b_{10}	α_{VAD}	α
0.035	20.0	1.1	2.9	1.39	0.825	0.37	0.11	0.41	0.73	0.940	0.974

- Suppression rules

Rule	PESQ	SNR	MSE	LSD	G_m	G_{Um}
Base line	2.449	7.62	0.718820	7.654		
MMSE	2.839	22.11	0.434427	9.410	0.329	0.092
MMSE DDA	***3.020***	***31.58***	***0.419704***	***10.256***	***0.020***	***0.100***
ML	2.795	19.24	0.457038	8.188	0.001	0.018
SS	2.983	25.02	0.421912	9.394	0.001	0.083
ST-MMSE	3.003	28.15	0.419987	9.316	0.024	0.100
ST log-MMSE	***3.018***	***30.49***	***0.420202***	***9.797***	***0.001***	***0.103***
JMAP SAP	3.011	28.18	0.419490	9.392	0.020	0.100
MAP SAE	**3.021**	**29.42**	**0.419297**	**9.640**	**0.020**	**0.100**
MMSE SP	3.011	28.70	0.419637	9.468	0.001	0.097

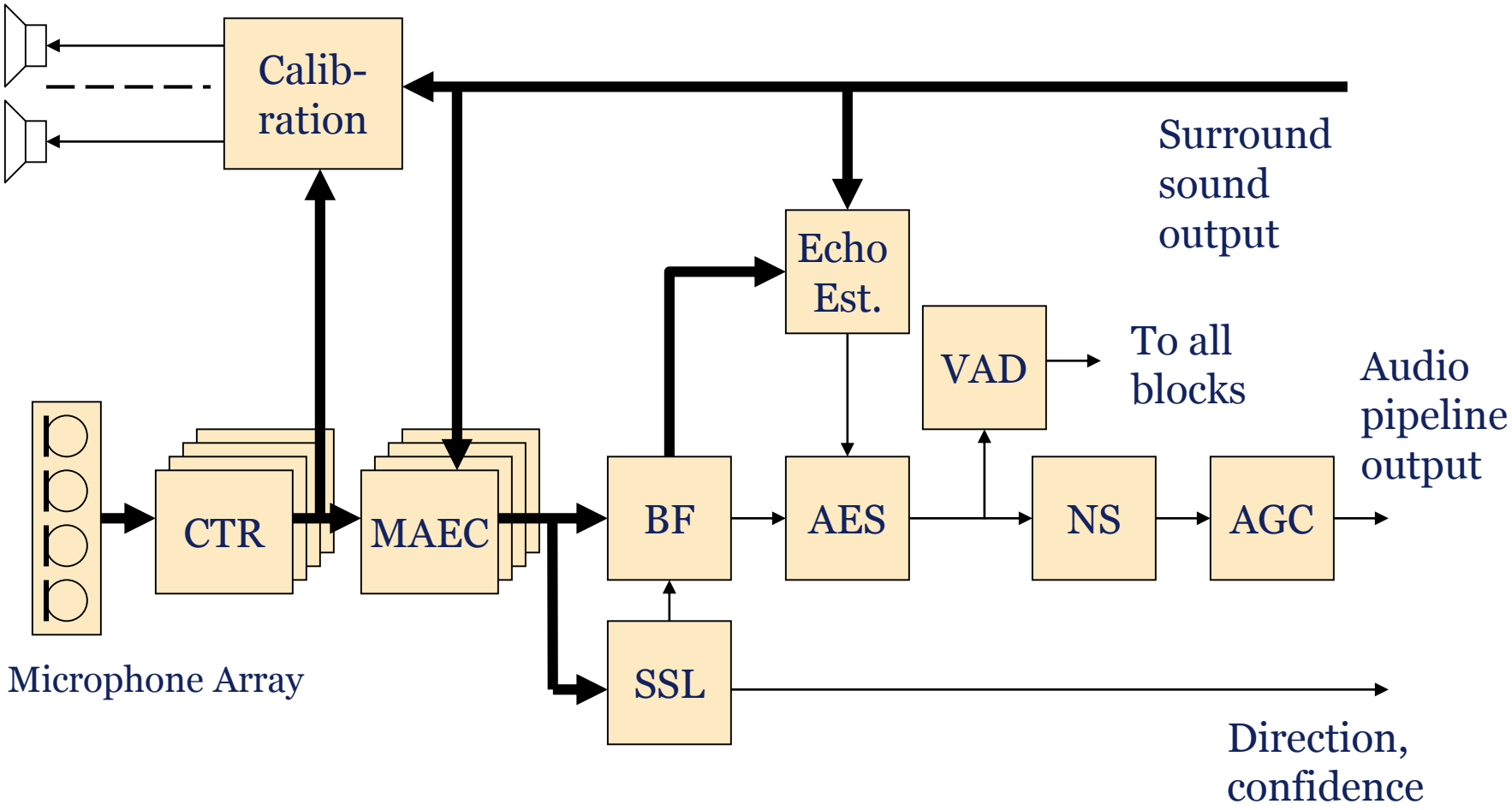
- Improvements:

0.57 – baseline
0.17-0.27 – industrial
(in PESQ points)

20-22 dBC – baseline
10-14 dBC – industrial
(in SNR)

Audio pipeline in Kinect

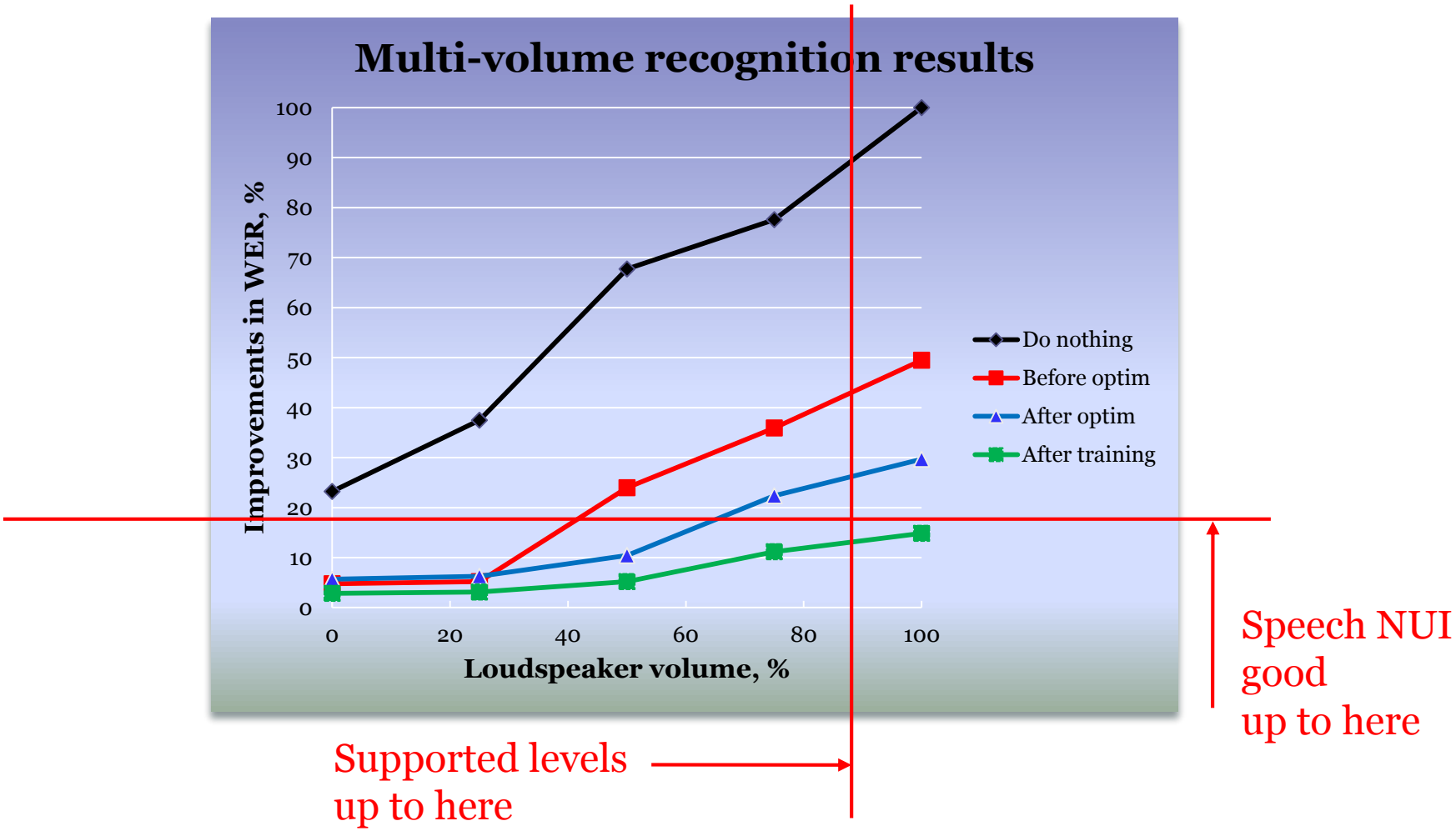
Audio pipeline architecture



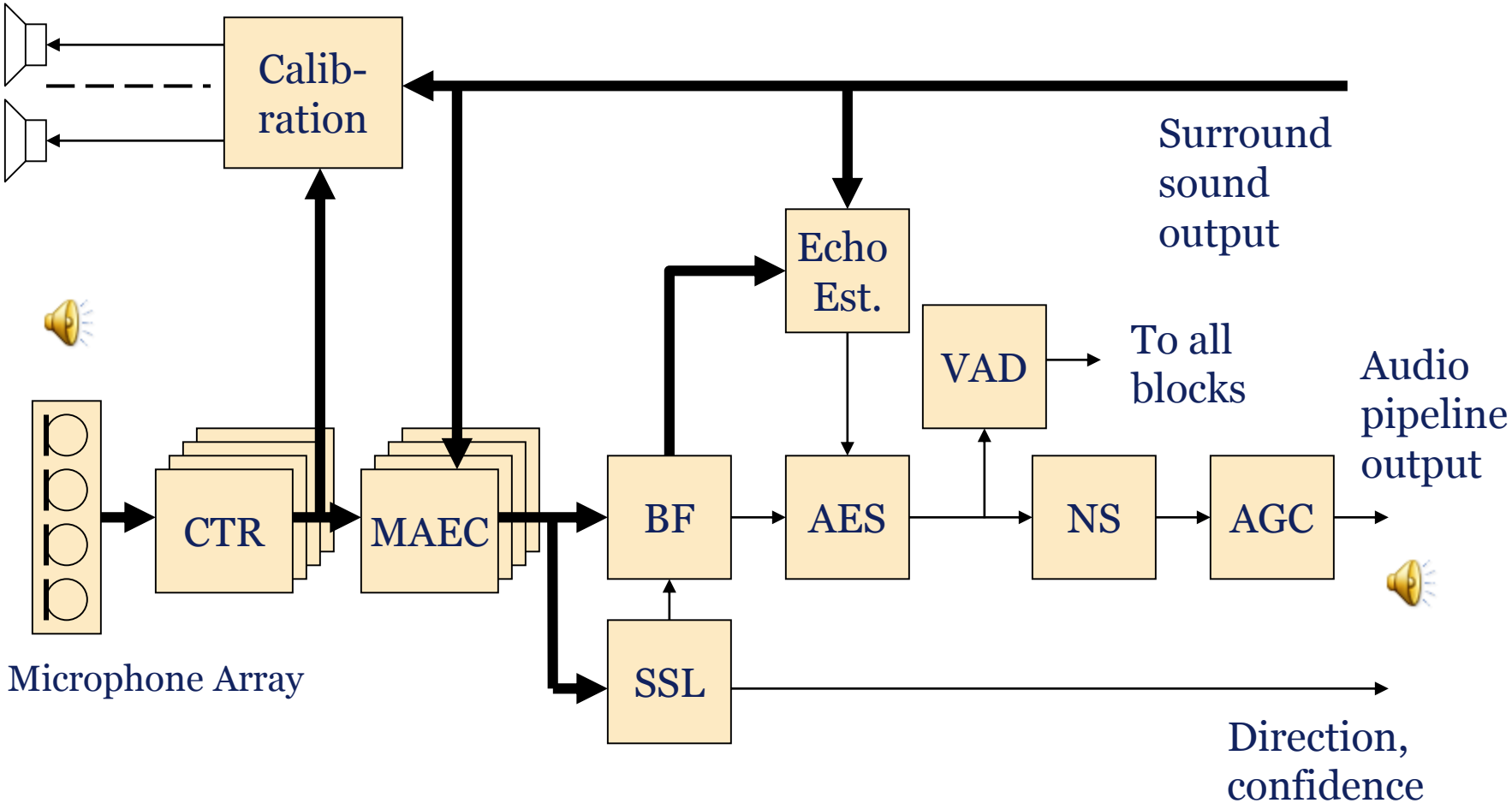
End-to-end optimization

- Mean Opinion Score (MOS) and Perceptual Evaluation of Sound Quality (PESQ)
- 75 parameters for optimization
- Optimization criterion:
  - Q = PESQ+0.01*ERLE+0.001*SNR-0.001*LSD-0.01*MSE
- Optimization algorithm
  - Gaussian minimization
- Data corpus with various distance, levels, reverberation
- Parallelized processing on computing cluster, 216 CPUs
- Optimization time: ~50 hours

End-to-end optimization: results



Results: demo



Conclusions and Q&A

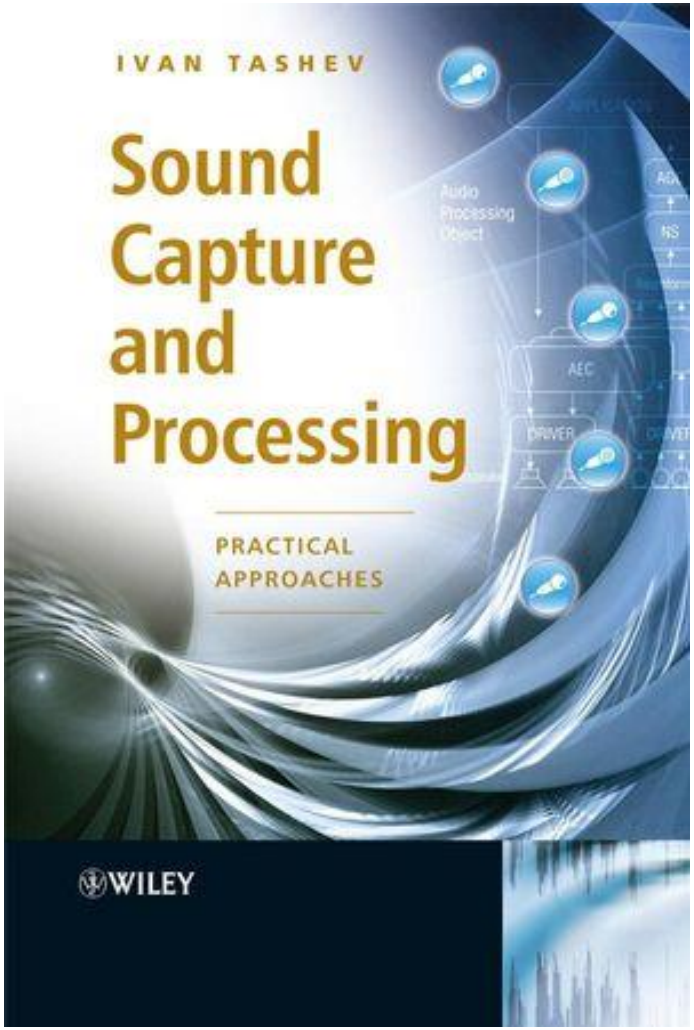
Conclusions

- Optimization methods are powerful tool for improving the parameters of complex DSP systems
- Your data corpus is the model of the world you want your system to work – it is not going to work in conditions not in the corpus
- Use training, development, and test data
- Don’t chase the mathematical optimum, use engineering precision and common sense

Shameless plug-in #1

Ivan Tashev. *Sound Capture and Processing: Practical Approaches*. Willey, 2009

Contains algorithms, a lot of figures, and sample Matlab code



Shameless plug-in #2

Saturday, January 14

10:15 – 10:45, Milano I

"Audio for Kinect: nearly impossible"

Dr. Ivan Tashev, Microsoft Research

References

- W. Press, S. Teukolsky, W. Vetterling, B. Flannery. *Numerical Recipes in C++*, Cambridge University Press, 2002.
- A. Antoniou, W.-S. Lu. *Practical Optimization*, Springer, 2007.
- T. Shoup. *A Practical Guide to Computer Methods for Engineers*, Prentice Hall, 1979.
- R. P. Brent. *Algorithms for Minimization without Derivatives*, Prentice Hall, 1973.
- J. E. Dennis, R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, 1983.
- D. A. H. Jacobs. *The State of the Art in Numerical Analysis*, Academic Press, 1977.
- S. Boyd, L. Vandenberghe. *Convex Optimization*, Cambridge University Press, 2004

Finally

Questions?

Bring your USB memory sticks to copy the Matlab software!

Contact info:
ivantash@microsoft.com
http://research.microsoft.com/en-us/people/ivantash/