Q-ary Repeat-Accumulate Codes for Weak Signals Communications

Nico Palermo, IV3NWV

XVII EME Conference
Venice, Italy - 2016

What I'll speak about

- Part I - Introduction to QRA codes and decoders
- Part II - A QRA code for EME. Simulation results
- Part III - Exploiting the redundancy of a QSO
- Part IV - The new QRA64 mode for WSJT-X

I. Introduction to QRA codes and decoders

Historical Perspective

- ~1960 - Low Density Parity Check (LDPC) codes introduced by Robert Gallager at M.I.T.
- 1963...'80s – Nothing happens. Decoding too complicate for those years technology.
- 1993 – Alain Glavieux/Claude Berrou introduce Turbo codes and iterative decoding.
- 1995 – David MacKay resurrects Gallager's LDPC codes and shows how to decode them with Message Passing.
- 2000 – Aamod Khandekar/Robert McEliece at Caltech introduce Irregular Repeat-Accumulate (IRA) codes.
- ...2016 – LDPC codes used everywhere from deep-space probes to mobile phones... and in WSJT-X as well!

LDPC Codes

- Low Density Parity Check means that the parity check matrix of the code is (very) sparse:
  - Each parity check equation involves <u>few</u> codeword symbols
  - Each codeword symbol is involved in <u>few</u> parity check equations
- Parity check matrix H:
  - Rows indicate parity check equations
  - Columns indicate codeword symbols
  - Codewords <u>x</u> satisfy the set of equations H*<u>x</u>=<u>0</u>

  Example: Hamming (7,4) code. Not a LDPC code: H is not sparse

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

x1 + x3 + x5 + x7 = 0

x2 + x3 + x6 + x7 = 0

x4 + x5 + x6 + x7 = 0

QRA Codes

- Class of LDPC codes with Q-ary symbols set
  - Q=4, 8, 16, 32, 64,... or any number for which a finite field exists
  - Maps naturally to orthogonal modulations (i.e. 64-FSK)
- Repeat-Accumulate (RA) encoding:
  - Information symbols are repeated (like in a repetition code),
  - Parity checks are generated as a weighted accumulation of the repeated information symbols sequence
- Same decoding procedure of LDPC codes
  - Maximum A-Posteriori Probability with the Message Passing (MP) algorithm

MAP Decoding

- Maximum A Posteriori (MAP) Probability
- Bayes' rule:

  Prob($\underline{X}$|$\underline{R}$) proportional to Prob($\underline{R}$|$\underline{X}$) * Prob($\underline{X}$),   where:

  $\underline{X}$ = transmitted codeword,

  $\underline{R}$ = received signal sequence

  Prob($\underline{X}$|$\underline{R}$) = a posteriori probability <-- What we need to compute

  Prob($\underline{R}$|$\underline{X}$) = likelihood <-- Channel dependence

  Prob($\underline{X}$) = a priori probability <-- Code and a priori knowledge dependence

- For each codeword symbol we need to maximize the symbol-wise probability Prob(X_j|$\underline{R}$) averaging Prob($\underline{X}$|$\underline{R}$) over all the possible cases we are interested into:
  - Prob(X_j|$\underline{R}$) = sum of Prob($\underline{X}$|$\underline{R}$) over all codewords with given X_j

General Case MAP Decoding

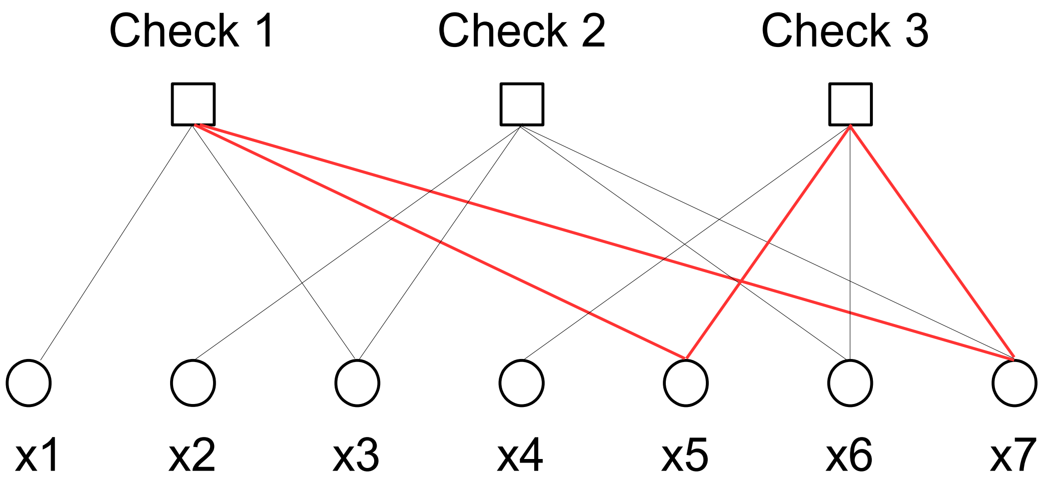
- Given the likelihoods and any a priori knowledge:
  1. Compute <u>ALL</u> the codewords a posteriori probabilities
  2. For each information symbol:
     a) Sum the probabilities of <u>ALL</u> the codewords in which a symbol assumes a given value, and
     b) Select as the best estimate of a symbol the value which maximizes its a posteriori probability distribution
- Complexity scales <u>exponentially</u> with codeword length
- Example: K=72 information bits => ~2^72 operations =>

<u>Hundreds thousands years to decode a single message</u>

(using a good PC)

Tanner Graphs

- Alternative representation of a code parity check matrix
  - Mark codeword symbols with circles
  - Mark parity check equations with boxes
  - Connect circles to boxes with edges to indicate which symbol is involved in a given check equation
- Immediate sight of code properties (i.e. cycles)



Example: Hamming (7,4) code

x1 + x3 + x5 + x7 = 0 (Check 1)

x2 + x3 + x6 + x7 = 0 (Check 2)

x4 + x5 + x6 + x7 = 0 (Check 3)

MAP Decoding of LDPC codes

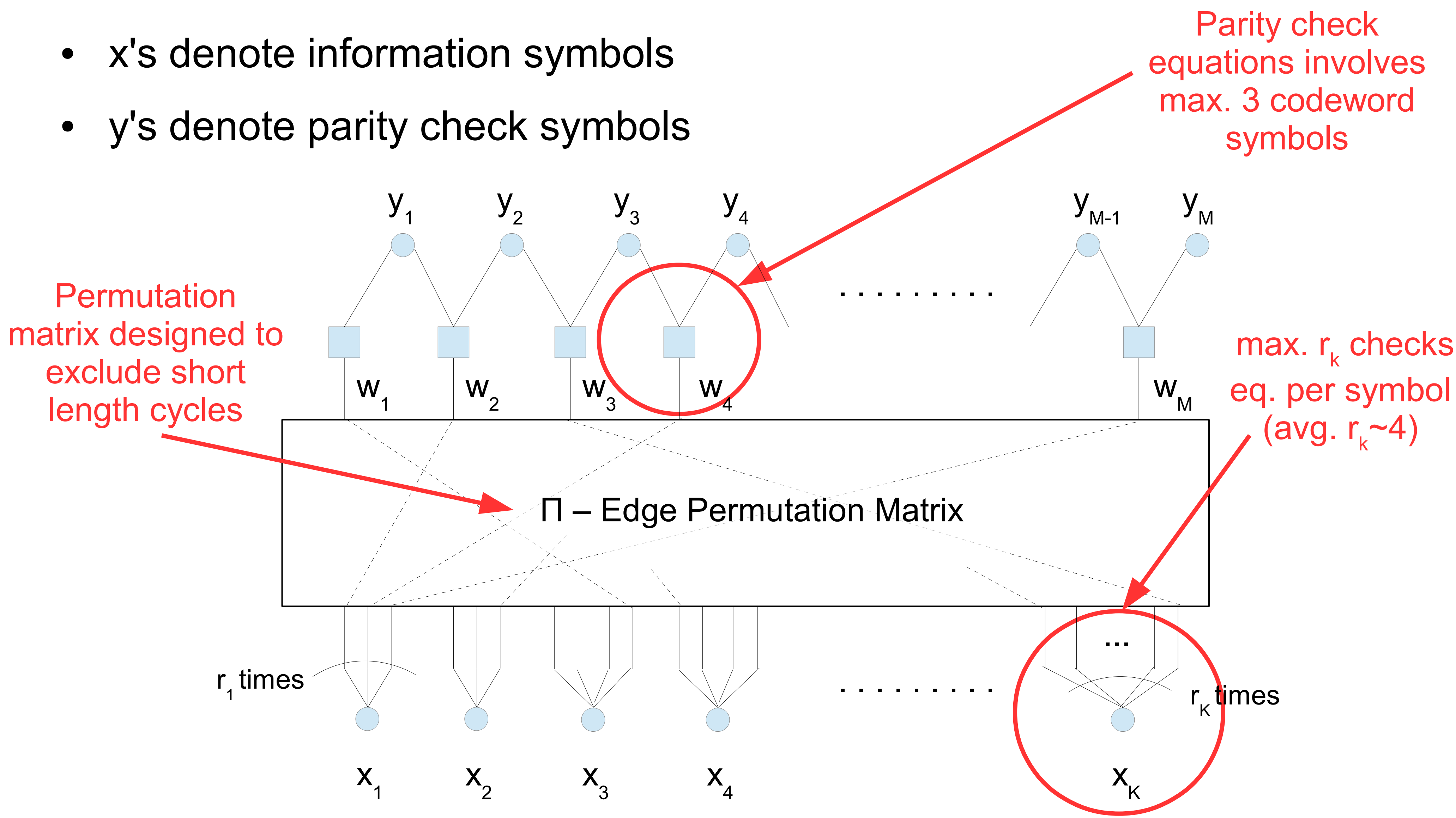
- A posteriori probabilities can be computed exactly if the code Tanner graph is a tree (has no cycles)
- Parity check equations with few variables and variables involved in few checks => very fast evaluation of probabilities factors
- LDPC codes can be designed to have few and sufficiently large length (girth) cycles (no good code graph is a tree),
- LDPC codes involve few variables per parity check equation and few equations per variable =>

A posteriori probabilities can be evaluated with good precision and much more quickly than in the general case

Decoding complexity scales linearly with codeword length

Tanner Graph of a QRA Code

- x's denote information symbols
- y's denote parity check symbols



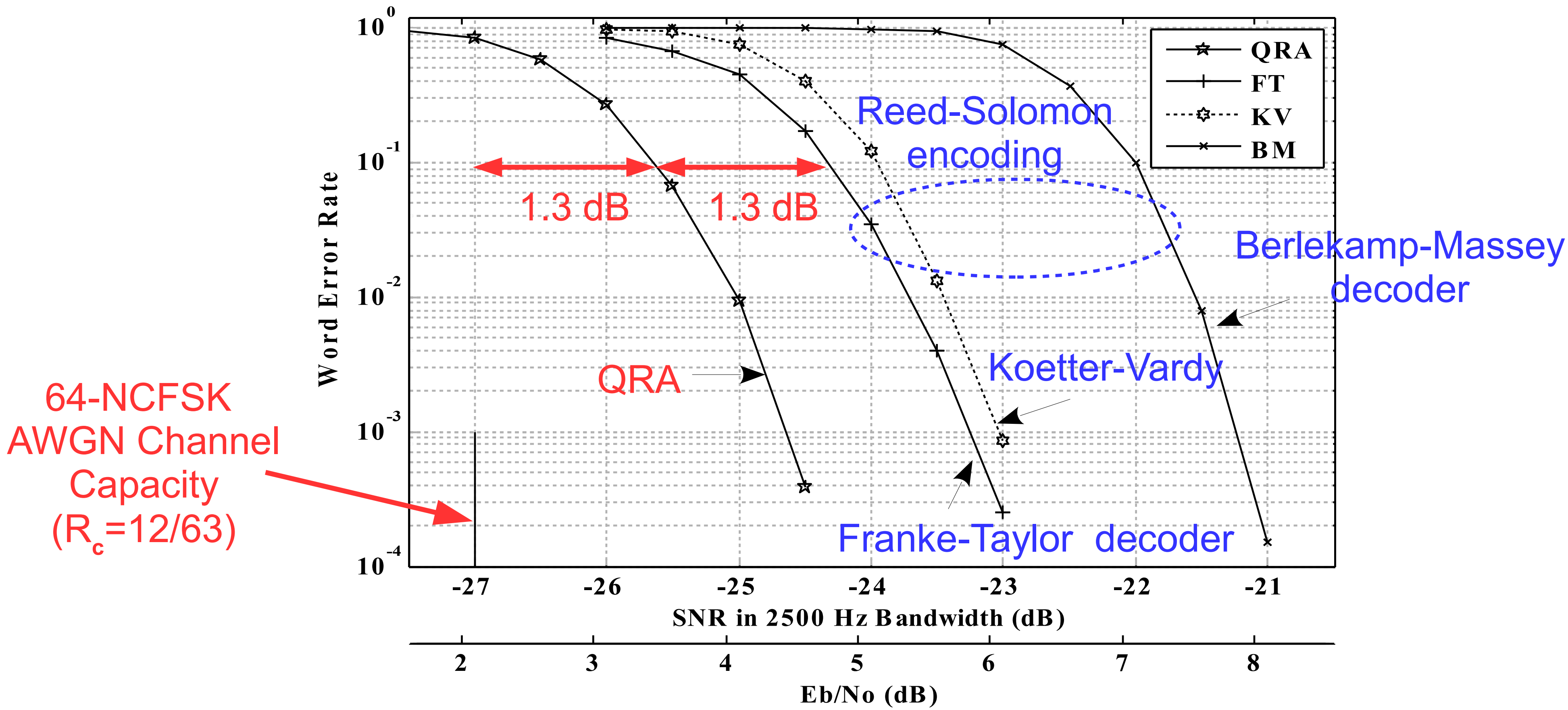
Message Passing Decoder

- MAP probabilities evaluated iteratively exchanging “messages” among circles (codeword variables) and boxes (check equations)
- The messages are actually probability distributions
- Each iteration is a two step process:
  - $c \rightarrow v$ step : send messages from checks to variables
  - $v \rightarrow c$ step : send messages from variables to checks
- After each iteration find the symbol values which maximize the (approximate) a posteriori probability and check if all parity check equations are satisfied (successful decode)
- Stop if no success within a max. number of iterations

II. Simulation Results

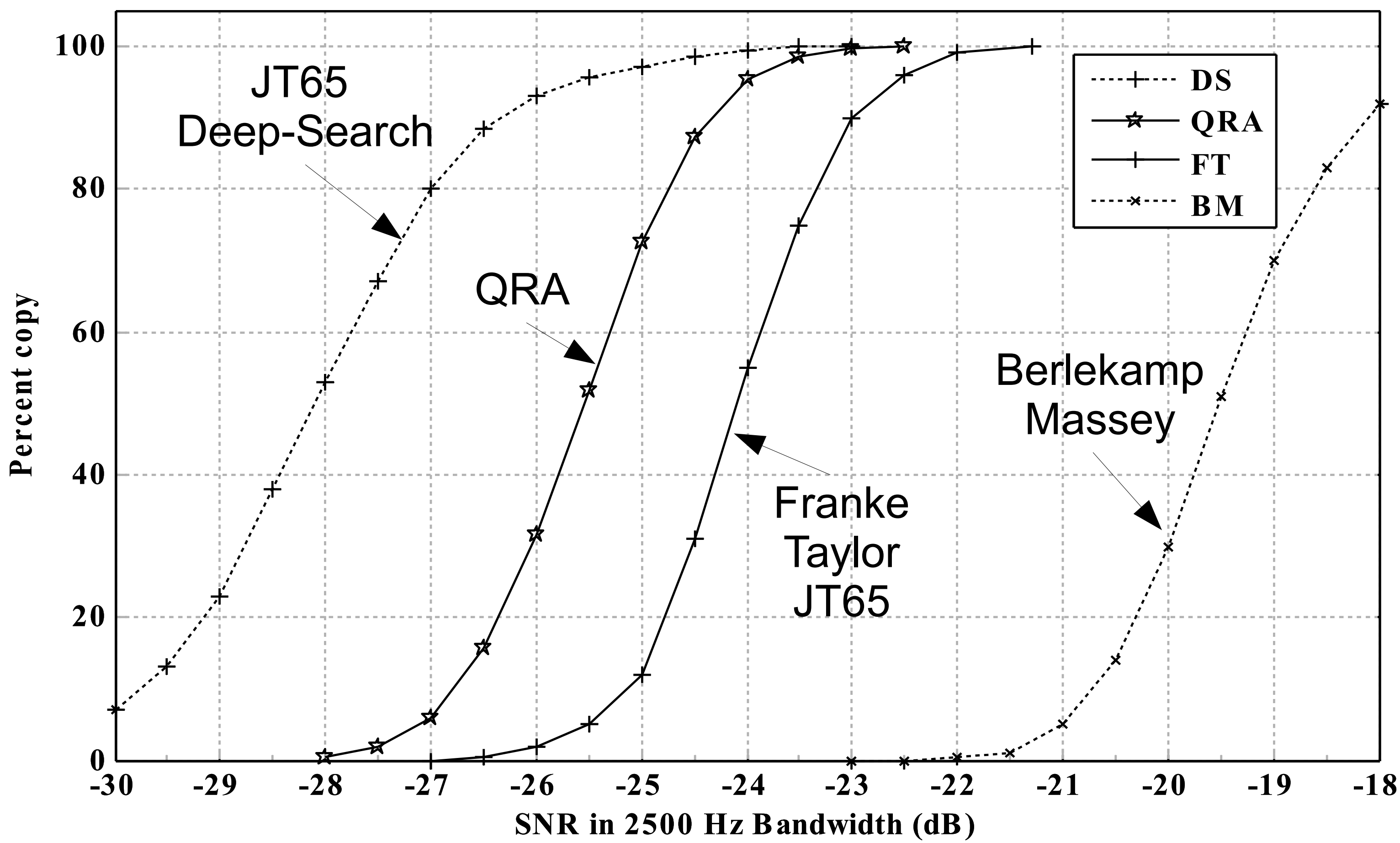
QRA(12,63) ↔ RS(12,63)

- AWGN channel, QRA MP decoder with 100 iterations
- Same code parameters/modulation/sync. pattern of JT65:
  - K=12, N=63, 64-FSK (non coherent demod.), 63 sync. symbols



QRA(12,63) ↔ RS(12,63)

- Rayleigh channel, QRA MP decoder with 100 iterations
- Same code parameters/modulation/sync. pattern of JT65



III. Exploiting the redundancy of a QSO

Decoding with "a priori" knowledge

1) No a priori avail. => Maximum Likelihood (ML) decoder

2) A priori available => Maximum A Posteriori (MAP) prob. decoder

<u>MAP decoders easily handle both cases</u>

ML is just a special case of MAP

MAP is much better than ML

- A two-way QSO is a sequence of messages with decreasing amount of uncertainty/increasing amount of a priori (AP) knowledge:
  - First message in a QSO is a CQ call, i.e. [<u>CQ</u> IV3NWV JN66]
  - First replies (if any) directed to our call, i.e. [<u>IV3NWV</u> SM5BSZ JO89]
  - Further replies come from known source, i.e. [<u>IV3NWV SM5BSZ</u> -25]
  - Last reply is just an acknowledge, i.e. [<u>IV3NWV SM5BSZ 73</u>]

  => <u>INSTRUCT THE DECODER TO HANDLE ALL THESE CASES!</u>

Typical QSO “a priori”

Sample QSO between IV3NWV and SM5BSZ:

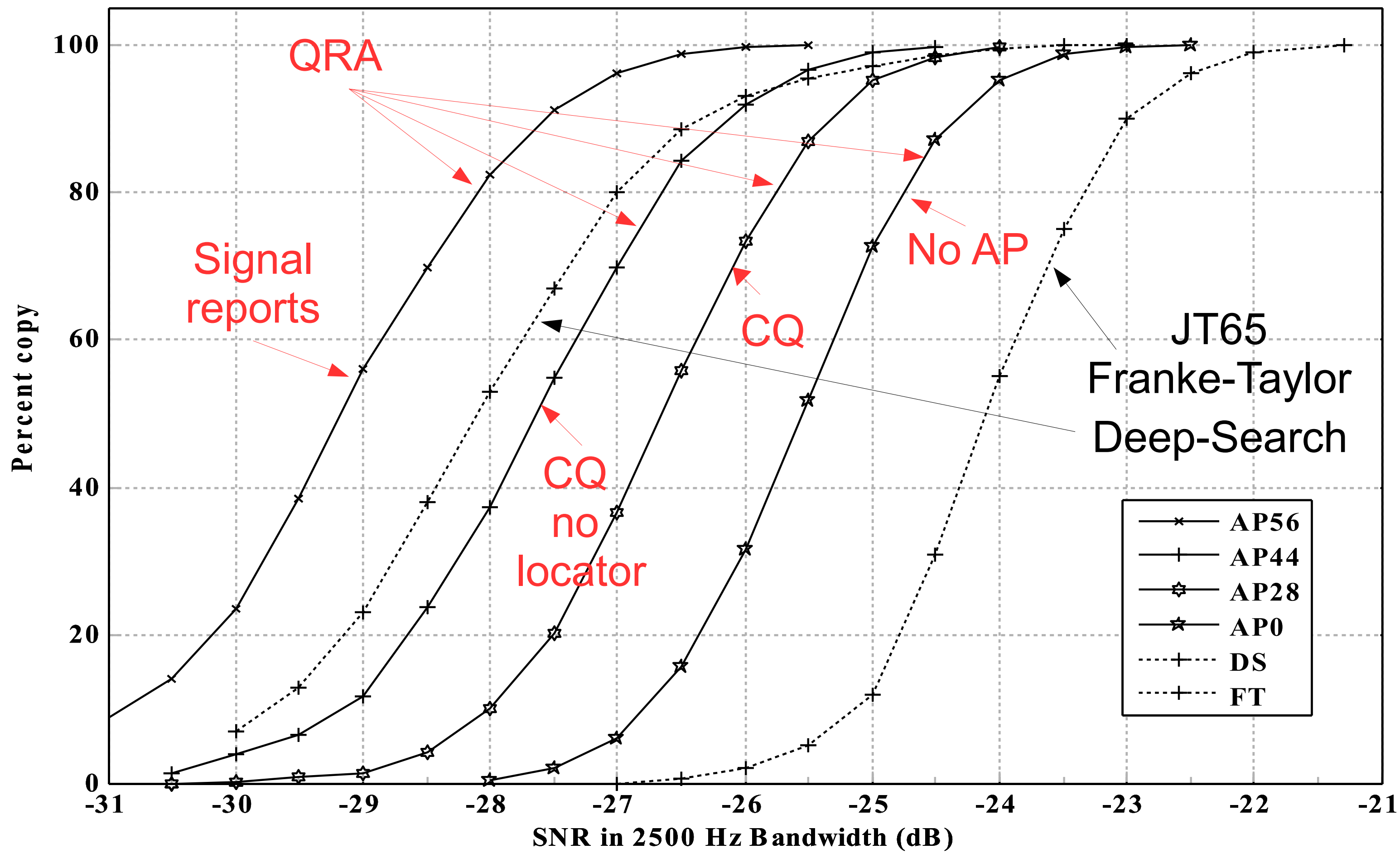
What SM5BSZ's decoder knows from QSO semantics

1. CQ IV3NWV JN66
2. IV3NWV SM5BSZ JO89
3. SM5BSZ IV3NWV -25
4. IV3NWV SM5BSZ R-25
5. SM5BSZ IV3NWV 73
6. IV3NWV SM5BSZ 73

What IV3NWV's decoder knows from QSO semantics

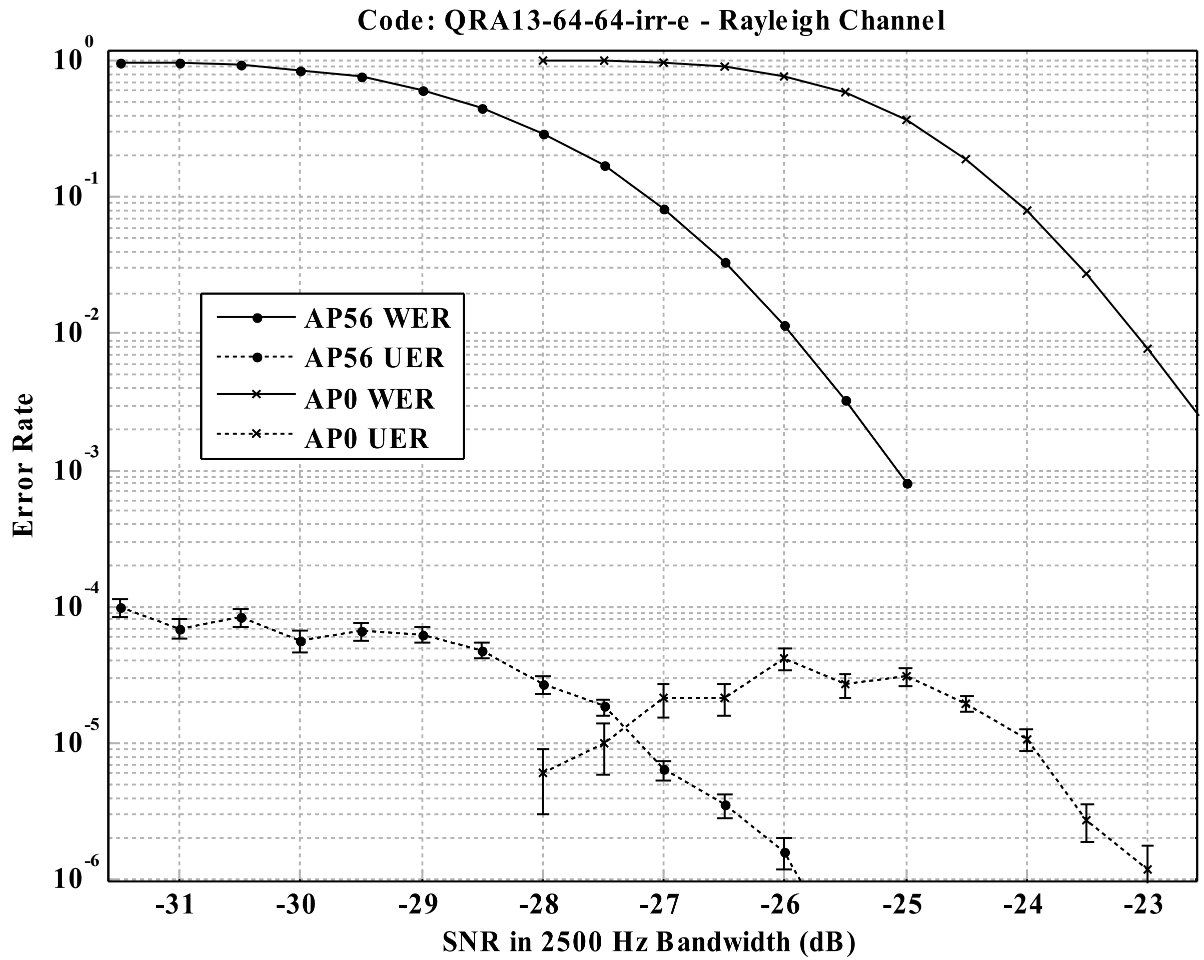
- Underlined fields fed to the MAP decoder as “a priori” info as the QSO proceeds to the end
- 1 field → ~28 bit AP - 2 fields → ~56 bit AP - 3 fields → 72 bit AP

QRA Decoder with AP ↔ JT65



- QRA(12,63) code with same parameters/modulation/sync. pattern of JT65
- Rayleigh channel – sync. losses not included
- Decode always with info received from the channel (unlike the JT65 deep-search)

QRA Decoder UER Performance



- Undetected Error Rate (UER) improved through design of a QRA(13,64) code.
- 13th symbol is a CRC-6 check computed from the 12 information symbols
- The CRC-6 symbol is not sent through the channel (punctured code)
- The resulting code is still a QRA(12,63) with much better UER (< 10^-4)

IV. The QRA64 mode

QRA64

- New mode(s) for WSJT-X
- Based on a irregular QRA(12,63) code with the same rate/symbol set of the RS code used in JT65:
  - 12 information symbols (each 6 bit long)
  - 51 parity check symbols (codeword length = 63 symbols)
  - Actually a punctured QRA(13,64) code over GF(64) with CRC-6
- 21 symbols synchronization pattern made by three 7x7 Costas arrays (Tnx Joe Taylor – K1JT) – 1.9 dB sync. energy gain over JT65
- Submodes A, B, C, D, E to handle Doppler spreads up to microwaves
- QRA encoder/decoder (me - IV3NWV)
- Sync algorithms/WSJT-X integration/twistles and bells (Joe – K1JT)
- > 3 dB coding gain over JT65 (with no AP knowledge)
- < -28 dB SNR threshold at 50% copy exploiting AP on CQ calls

QRA64 - 10 GHz EME On-Air Tests

- Made by Charlie Suckling G3WDG and Rex Moncur VK7MO during July/August 2016
- Tests made with the 1.7.0 WSJT-X development version
- Lot of wav files recorded from real EME QSOs
- Tested Doppler spreads from ~0 Hz and up to 100 Hz
- QRA64A, B, C, D, E modes and JT4F mode recordings to evaluate differences, benefits or disadvantages
- Performance compared using SNR degradation feature of WSJT-X:

  1) Degrade wav files SNR until messages are no more decoded

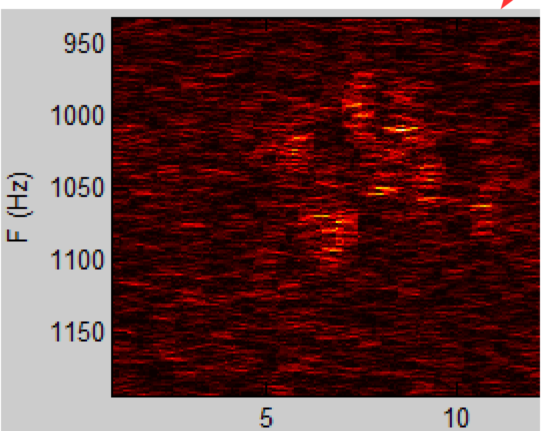
  2) The higher the SNR degradation, the better the performance
- Very useful to understand how to handle fast-fading conditions:

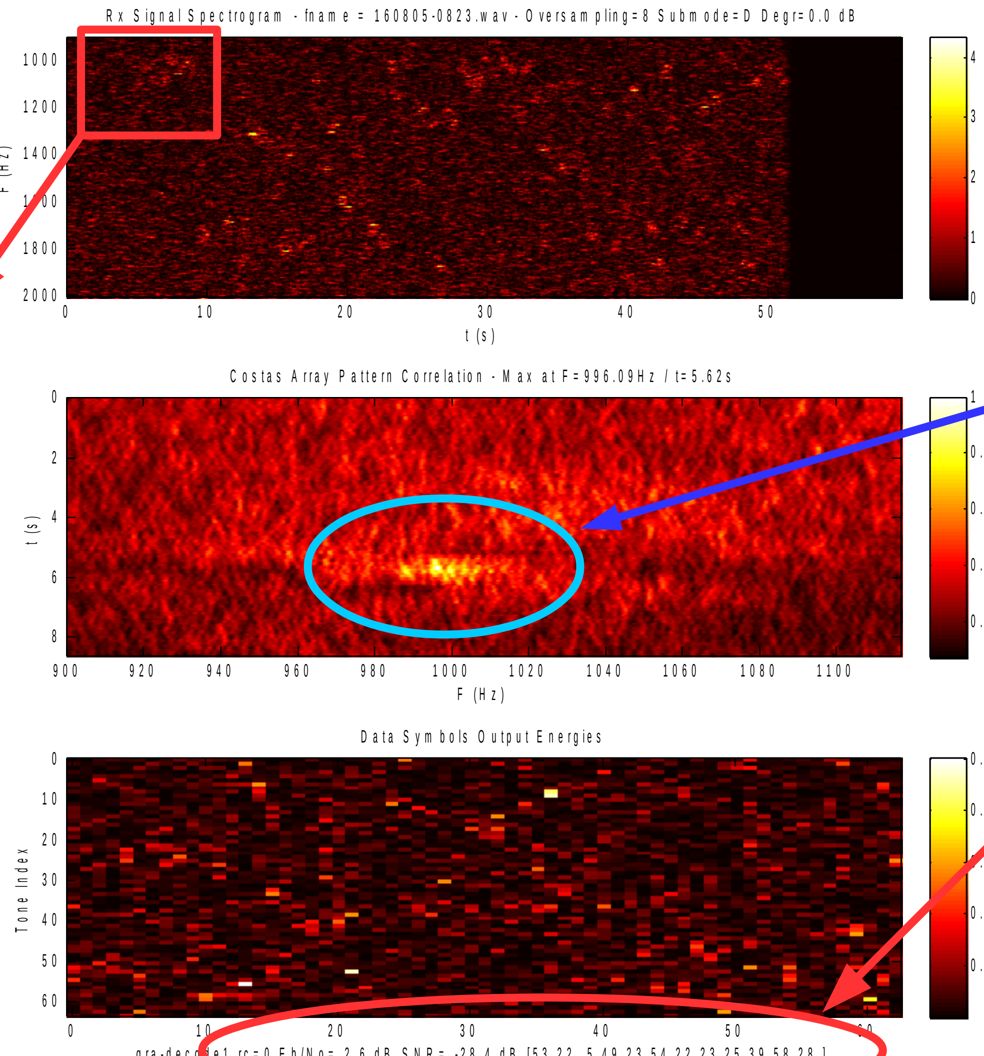
  QRA64 gains ~6 dB over JT4 when proper fast-fading likelihoods metric is used

QRA64D EME Tests (G3WDG ↔ VK7MO)

10 GHz - 100 Hz Doppler Spread – No Fast-Fading Metric

Detail of first Costas array.
Signal looks strong…





Correlation peak of the Costas arrays not really evident (many sec. peaks)

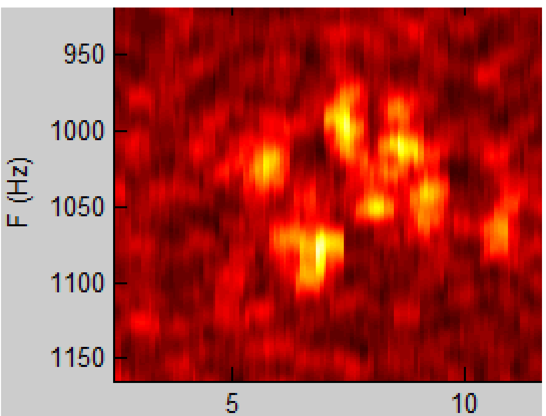
The QRA decoder shows a poor SNR (-28.4 dB). Small SNR degradation is tolerated (~2 dB).

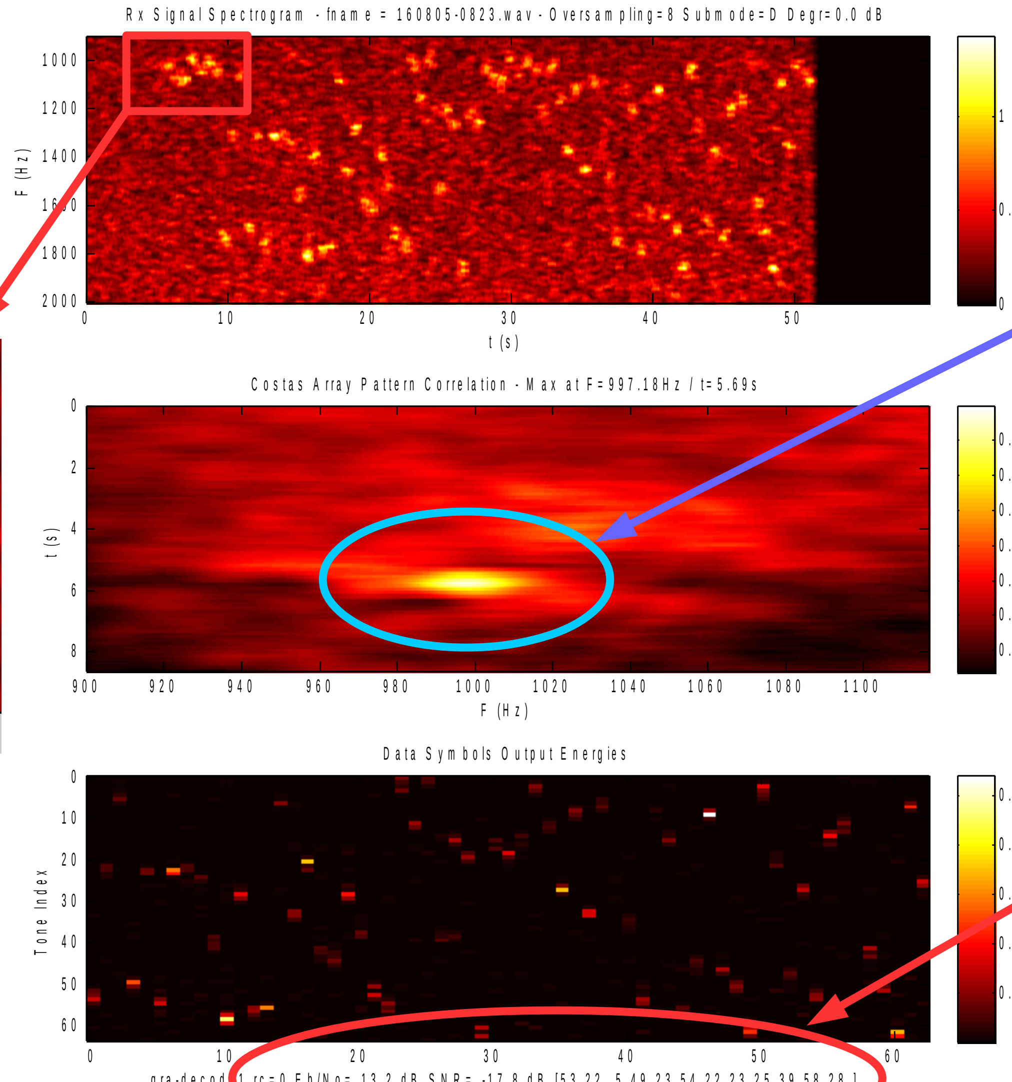
Performance similar to JT4F

QRA64D with Fast-Fading Metric

Same file as before – 10 GHz/100 Hz Doppler Spread – Fast-Fading Likelihoods Processing

Symbols likelihoods now peak clearly out of noise





More evident sync correlation peak

Estimated SNR is much higher (~ -18 dB)

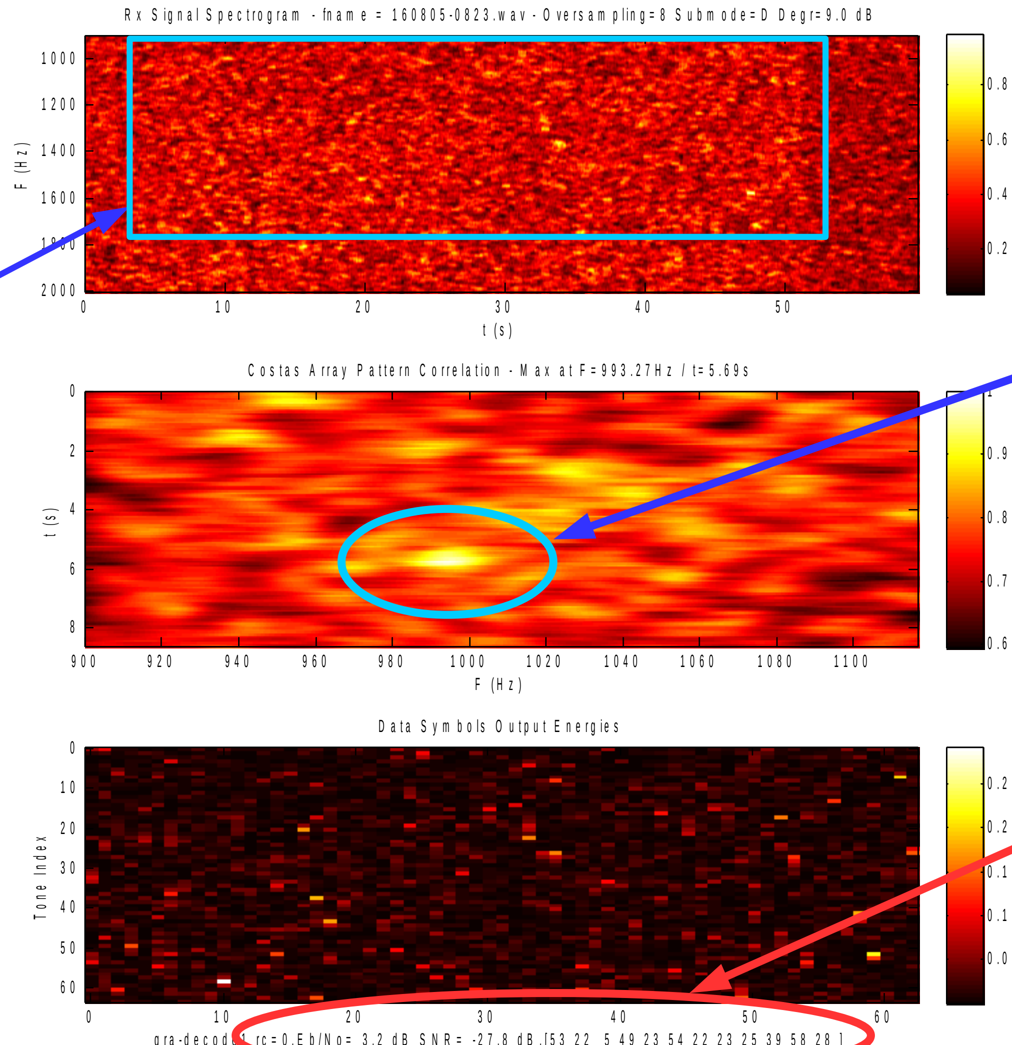
<u>Large QRA decoder noise margin</u>

QRA64 with Fast-Fading Metric

10 GHz/100 Hz Doppler Spread - SNR of original file degraded by 9 dB

Signal is hardly visible on a waterfall

Sync correlation peak still evident



Estimated SNR ~ -28 dB. Successful decoding.

~6 dB gain over JT4F

Fast-Fading Metric recovers almost all the losses a single matched filter decoder exhibits

QRA Codes Software Availability

- General purpose QRA encoding/decoding software with AP features stable and available as Open Source (GPL License) for Windows and Linux platforms here:
  - http://github.com/microtelecom/qracodes

    (not yet fully documented but evaluation tools included)

- Integration into WSJT-X to be completed with fast-fading metric/freq. drift compensation
  - Use JTSDK and WSJT-X software repository for WSJT-X specific developments.

Acknowledgments

Thanks to:

- Leif Asbrink - SM5BSZ for fruitful discussions and suggestions about EME QSOs issues
- Joe Taylor - K1JT for all the work he has done to integrate the QRA codes into WSJT-X and for his new sync algorithm innovations in QRA64
- Andrea Montefusco – IW0HDV for his help porting the QRA codes software to Linux platforms
- Charlie Suckling – G3WDG & Rex Moncur - VK7MO for their useful support with 10 GHz EME tests

...and thank you all for your attention

73

Nico Palermo, IV3NWV