Efficient denoising algorithms for large experimental datasets and their applications in Fourier transform ion cyclotron resonance mass spectrometry. Supporting Information

L. Chiron[1] M.A. van Agthoven[2] B. Kieffer[1] C. Rolando[2] M-A. Delsuc[1]

1 Institut de Génétique et de Biologie Moléculaire et Cellulaire, Institut National de la Santé et de la Recherche, U596; Centre National de la Recherche Scientifique, Unité Mixte de Recherche 7104; Université de Strasbourg, 67404 Illkirch-Graffenstaden, France

2 Miniaturisation pour la Synthèse, l'Analyse et la Protéomique, Centre National de la Recherche Scientifique, Unité de Service et de Recherche 3290, and Protéomique, Modifications Post-traductionnelles et Glycobiologie, Université de Lille 1 Sciences et Technologies, 59655 Villeneuve d'Ascq Cedex, France Glycobiologie, IFR 147, Université de Lille 1 Sciences et Technologies, 59655 Villeneuve d'Ascq Cedex, France

1 Algorithms

The algorithms for **rQRd** and **urQRd** are given here in formal representation.

Algorithm S1 rQRd

given a time series X, rank K and order M, returns $\tilde{X}$ a denoised approximation of X

Require: $X, K, M \quad K \le M \le length(X)/2$

Require: Function RANDOM : $n, p \mapsto \Omega$ ▷ Ω a $\mathcal{N}(0,1)$ $n \times p$ matrix

Require: Function QR : $A \mapsto Q, R$ ▷ the QR decomposition of A

$L \leftarrow \text{LENGTH}(X)$

$N \leftarrow L - M + 1$

for $i \leftarrow 1, M \quad j \leftarrow 1, N$ **do**

  $H_{ij} \leftarrow X_{i+j-1}$ ▷ H is a $M \times N$ matrix

end for

$\Omega \leftarrow \text{RANDOM}(N, K)$

$Y \leftarrow H\Omega$

$(Q, R) \leftarrow \text{QR}(Y)$

$\tilde{H} \leftarrow QQ^*H$

for $l \leftarrow 1, L$ **do**

  $\tilde{X}_l \leftarrow \langle H_{ij} \rangle_{i+j=l+1}$

end for

return $\tilde{X}$

The largest objects stored in memory are the matrices H and $\tilde{H}$. This represents a memory burden proportional to $O(MN) \lesssim O(L^2)$.

The slowest step is the computation of $\tilde{H} = QQ^*H$ in $O(KMN)$ while the computation of $\tilde{X}$ is in $O(LM)$. This results in a theoretical time dependence in $O(KMN + LM)$ The initial computation of Y is also non-negligible, but in all cases the computation of the QR decomposition seems to be negligible in our implementation.

Algorithm S2 Fast Hankel Matrix product

Require: Function $\mathcal{F} : f_i \mapsto F_j$, ▷ compute F_j the Digital Fourier Transform of f_i

function $\textsc{Fhv}(X, V)$

given a time series X, and a vector V, returns the result of the matrix product of H by V, where H is the Hankel matrix constructed from X as in Algorithm S1

$L \leftarrow \textsc{length}(X)$

$N \leftarrow \textsc{length}(V)$

$W \leftarrow \{\underbrace{0, \ldots, 0}_{M-1 \text{ values}}, V_N, V_{N-1}, \ldots, V_1\}$ ▷ so that length of W is L

$X' \leftarrow \mathcal{F}(X)$

$W' \leftarrow \mathcal{F}(W)$

$S' \leftarrow \{X'_1 W'_1, \ldots, X'_L W'_L\}$

$S \leftarrow \mathcal{F}^{-1}(S')$

$R \leftarrow \{S_1, \ldots, S_{L-N}\}$

return R

end function

function $\textsc{Fhm}(X, A)$

given a time series X, and a matrix A, returns the result of the matrix product of H by A, where H is the Hankel matrix constructed from X as in Algorithm S1

$N, P \leftarrow \textsc{Shape}(A)$

for $p \leftarrow 1, P$ **do**

$A^{(p)} \leftarrow \{A_{1,p}, \ldots, A_{N,p}\}$

$B^{(p)} \leftarrow \textsc{Fhv}(X, A^{(p)})$

end for

return matrix B where $B_{i,j} = B_j^{(i)}$ ▷ B is a $M \times P$ matrix

end function

Function $\textsc{Fhv}()$ is in $O(L \log(L))$ and function $\textsc{Fhm}()$ is in $O(NL \log(L))$.

The $\textsc{Fhm}()$ function can be further optimized by allowing the vector S' computed in $\textsc{Fhv}()$ to be stored between each call, rather than recomputed.

Algorithm S3 urQRd

given a time series X, rank K and order M, returns $\tilde{X}$ a denoised approximation of X

Require: $X, K, M \quad K \le M \le length(X)/2$
Require: Function $\textsc{Random} : n, p \mapsto \Omega$ ▷ a $\sim \mathcal{N}(0,1)$ $n \times p$ matrix
Require: Function $\mathrm{QR} : A \mapsto Q, R$ ▷ the QR decomposition of A
Require: Function $\textsc{Fhv} : H, M, X \mapsto Y$
Require: Function $\textsc{Fhm} : H, M, A \mapsto B$

$L \leftarrow \textsc{length}(X)$
$N \leftarrow L - M + 1$
$\Omega \leftarrow \textsc{Random}(N, K)$
$Y \leftarrow \textsc{Fhm}(X, \Omega)$
$(Q, R) \leftarrow \mathrm{QR}(Y)$
$U \leftarrow \left[\ \textsc{Fhm}(X, Q^*)\right]^*$
for $k \leftarrow 1, K$ **do**
  $Q^{(k)} \leftarrow \{Q_{1,k}, \ldots, Q_{M,k}\}$
  $U'^{(k)} \leftarrow \{U_{k,N}, U_{k,N-1}, \ldots, U_{k,1}\}$
  $W^{(k)} \leftarrow \{\ \underbrace{0, \ldots, 0}_{N-1 \text{ values}}, Q_1^{(k)}, \ldots, Q_M^{(k)}, \underbrace{0, \ldots, 0}_{N-1 \text{ values}}\ \}$ ▷ $W^{(k)}$ are of length $L + N - 1$
  $Z^{(k)} \leftarrow \textsc{Fhv}(W^{(k)}, U'^{(k)})$ ▷ $Z^{(k)}$ are of length L
end for
$Z \leftarrow \sum_{k=1}^{K} Z^{(k)}$
for $l \leftarrow 1, L$ **do**

$$\tilde{X}_l \leftarrow \alpha_l Z_l \quad \text{with } \alpha_l = \begin{cases} 1/l & 1 \le l \le M \\ 1/M & M < l < N \\ 1/(L - l + 1) & N \le l \le L \end{cases}$$

end for
return $\tilde{X}$

The largest objects stored in memory are the matrices Y, Q and U. This represents a total memory burden proportional to $O(KL)$.

The slowest step is the loop on K for the computation of $\tilde{X}$ and its processing time is proportional to $O(KL\log(L))$. The computation of Y and of U are also non-negligible, but in all cases the computation of the QR decomposition seems to be negligible in our implementation.

2 Robustness against varying signal distortion

A synthetic dataset was used to test the robustness of **rQRd** relatively to various types of noise. A noise-free signal containing 20 random lines with intensities ranging from 1 to 20, is created and perturbed with a random process. In all cases, the random series is stationary with a Gaussian distribution, zero mean, and white Fourier Transform. It is obtained from the `numpy.random` library. For each realization, the perturbation level was chosen so that the apparent noise in the Fourier spectrum is approximately of the same intensity level. **rQRd** analysis is performed with $K = 50$.

Signal modifications are as follows:

- *additive noise* : a random signal is added the noise-free dataset. This is the case explicitly considered in the theoretical section.
- *scintillation noise* : the amplitude and the frequency of each signal component are subject to random variation of their value.
- *sampling noise* : each point of the series used to sample the theoretical signal is displaced by a random amount.
- *missing points* : some randomly chosen points of the signal series are set at 0.0

In all cases, the noise level is such that the SNR of noisy dataset is around 0 dB; except for the sampling case, where the SNR is 3 dB. All details can be found in the code deposited on the web site urqrd.igbmc.fr.

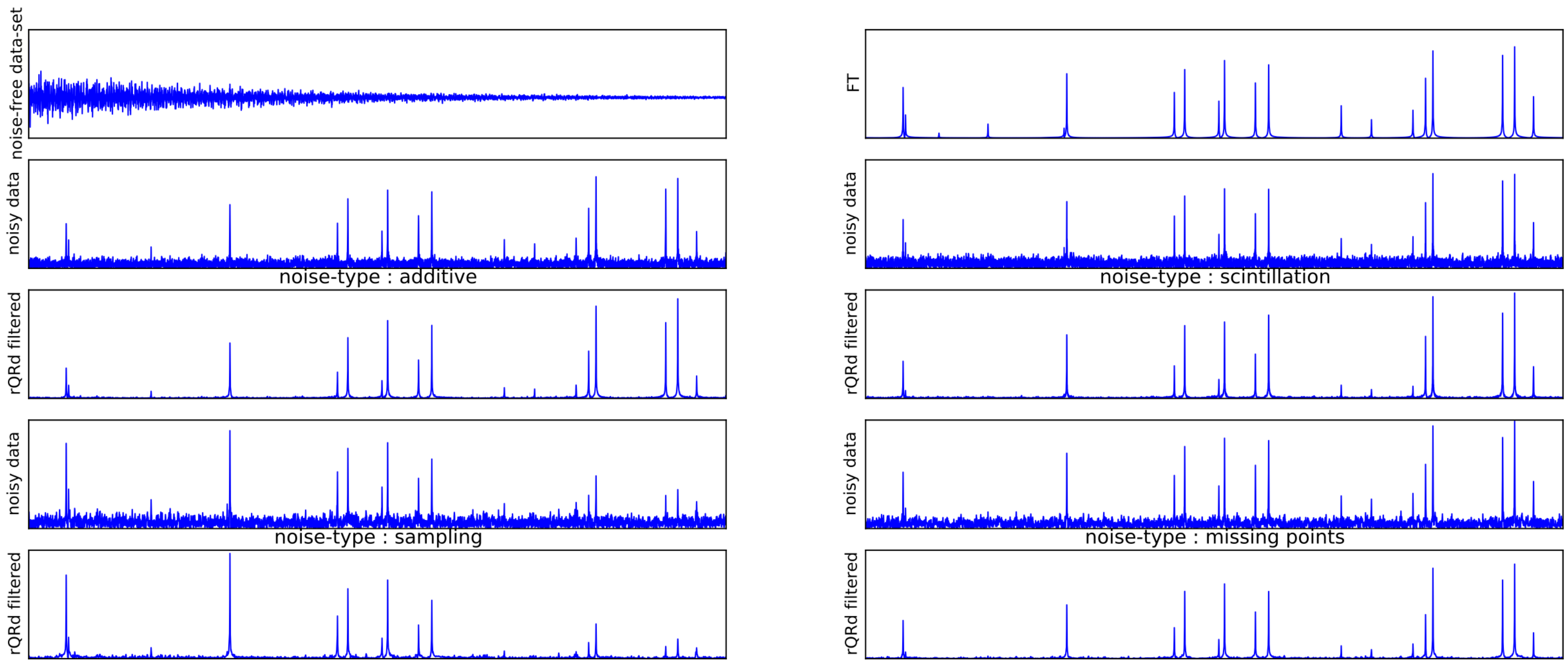

Figure S4: **rQRd** efficiency for various noise types on a synthetic datasets.
Topline the noise-free temporal signal and its Fourier transform.
Left column *second and third row* additive noise *forth and fifth row* sampling noise
Right column *second and third row* scintillation noise *forth and fifth row* missing point

3 Code and Data Deposition

3.1 Data Deposition

The data has been deposited on the site urqrd.igbmc.fr

3.2 Code Deposition

The code has been deposited on the site urqrd.igbmc.fr