Poisson-Gap Sampling and FM Reconstruction for Enhancing Resolution and Sensitivity of Protein NMR Data

Supplemental information

Sven G. Hyberts, Koh Takeuchi and Gerhard Wagner*

Comparison of Signal to Peak Noise Ratio

We also explored whether NUS and FM reconstruction affects signal to peak noise. This addresses the question of false positives. We selected ten areas that do not contain peaks **(Fig. S1A**) and measured the signal-to-peak noise for average (**Fig. S1B**) and maximum (**Fig. S1C**) peak-noise values. The larger linear coefficients for FM reconstructed spectra in the Fig. S1B and S1C clearly show the advantage of FM reconstitution over linear sampling or linear prediction in detecting signals above noise. Signal-to average peak-noise ratio for SPS combined with FM reconstruction is ~60% higher than uniform sampling (0.876 vs 0.546).

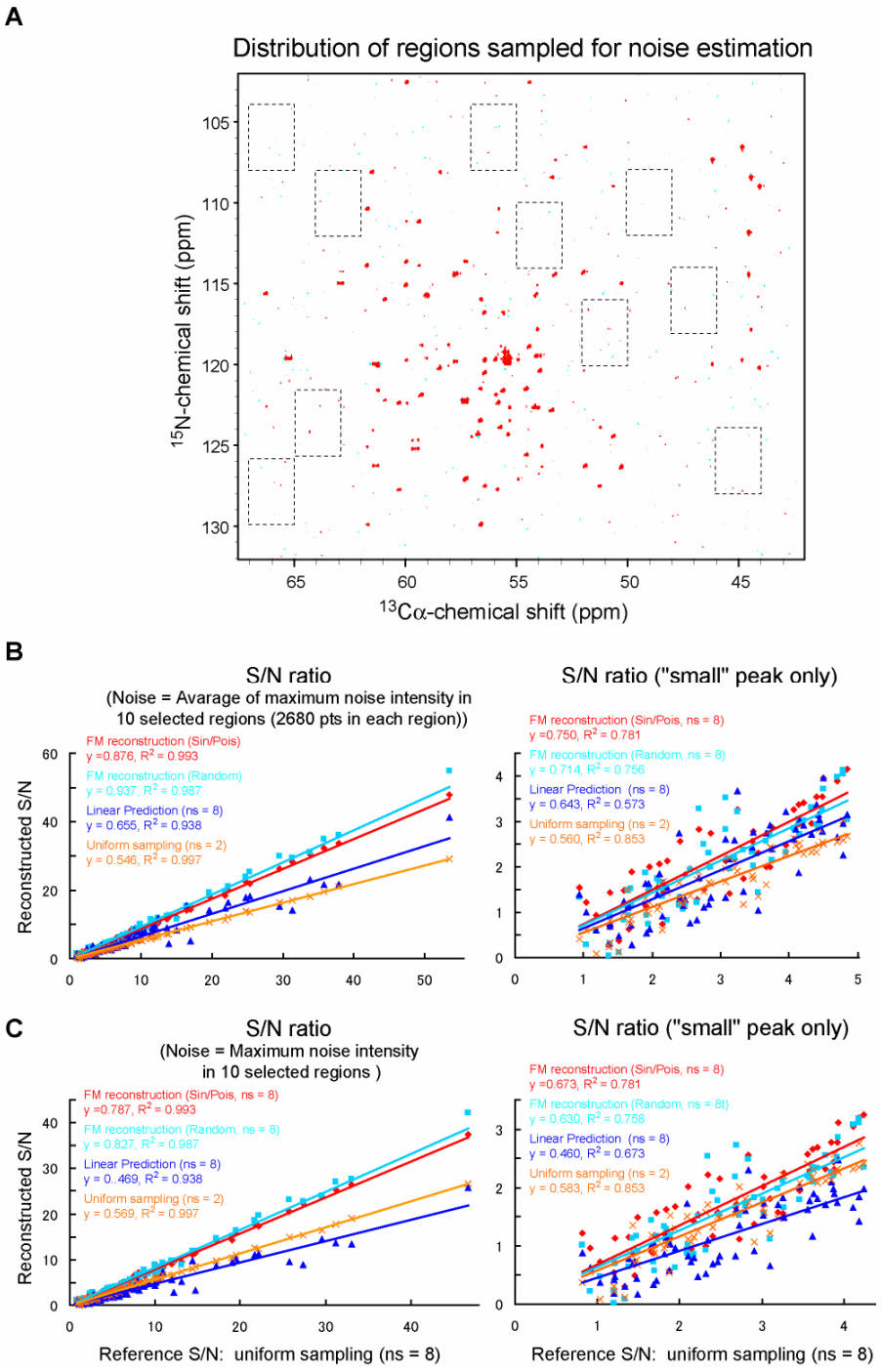

Fig. S1: Comparison of signal-to-peak noise. **A.** Ten areas that do not contain real peaks were selected for measuring the signal-to-peak noise. **B.** Plot of signal-to-average-peak noise versus the same measure in the four-times longer linear experiment as shown in **Fig.2** panel **A5**. The linear coefficient measures the signal-to-average peak noise relative to the long linear experiment with 8 scans per increment. As expected, the coefficient for the short linear experiment with 2 scans per increment is around 0.5 as the S/N is proportional to sqrt(n). **C:** Same as B but the ratio of signal to maximum noise is measured.

Effect of Order Parameters in Linear Prediction

We explored optimal parameters for processing time domain data with linear prediction. In particular, we used different order parameters of the nmrPipe software[1]. Fig. S2A shows a comparison of the same spectral region as in Fig. 2 of the main manuscript. The panel at the right is the same as in Fig. 1A, panel A5 of the main manuscript and is used for comparison. Clearly linear prediction looks best when using an order parameter of 30. (Higher order parameters than 30, however, are no meaningful since there are never more than 30 signals in a single trace). Fig. S2 B and C compare the signal-to-noise ratio with those of the linearly sampled data. Here peaks were picked as described in the main text, and the noise is again the median of 10,000 randomly picked points in the spectrum not including real peaks. The S/N of this spectrum is used as horizontal axis in Fig. S2B and C. Based on this comparison we find that the order parameter 30 yields the best S/N and is used for comparison in the main manuscript. However, small changes of peak positions are evident.

The biggest drawback is that some weak peaks get shifted. Here this is particularly clear for the peak at 115.1 ppm, which is marked with a broken line.

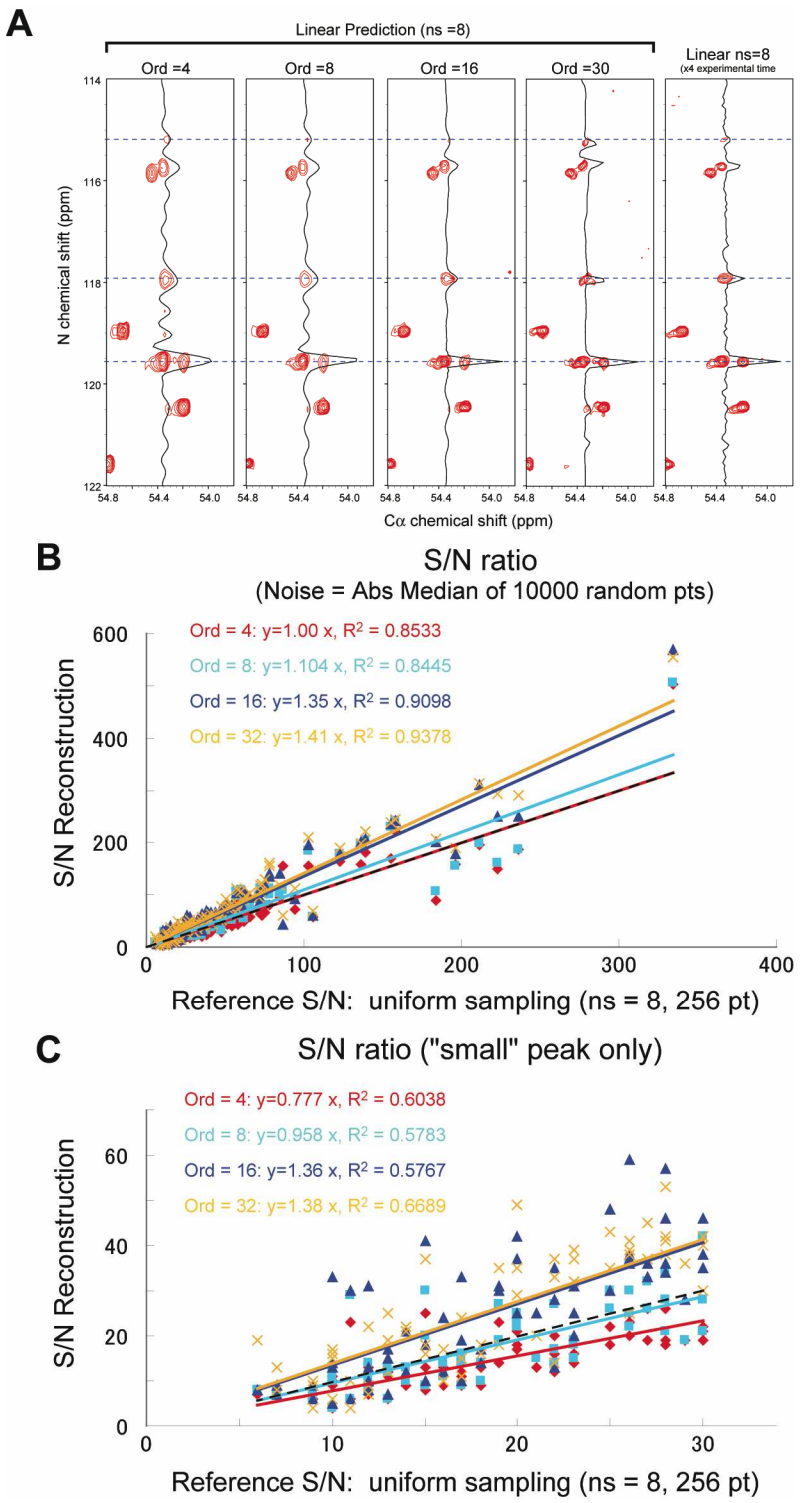

Figure S2: Comparison of spectral quality and S/N of linear prediction with different order parameters. Large order parameter is needed to obtain high S/N with linear prediction, which is not commonly used. However, linear prediction can cause small changes of peak positions, which are clearly seen for the peak at the nitrogen position of 115.1 ppm (top dotted line) and 117.8 ppm (middle dotted line).

Comparison of Resolution between Linear Prediction and NUS/FM Reconstruction

Figure S3 shows a comparison of a section of an NCa experiment recorded linearly over 256 increments and 8 scans per increment with spectra obtained from either the first 64 increments extended with linear prediction routine of the nmrPipe program and an order parameter of 30[1]. The bottom panels show the same spectral regions from spectra obtained with sinusoidally modulated Poisson-gap sampling and processed using FM reconstruction without (left) and with distillation.[2] Linear prediction suffers from significantly lower resolution, which matters for crowded spectral regions as indicated by arrowheads. The distill process can remove artifact significantly.

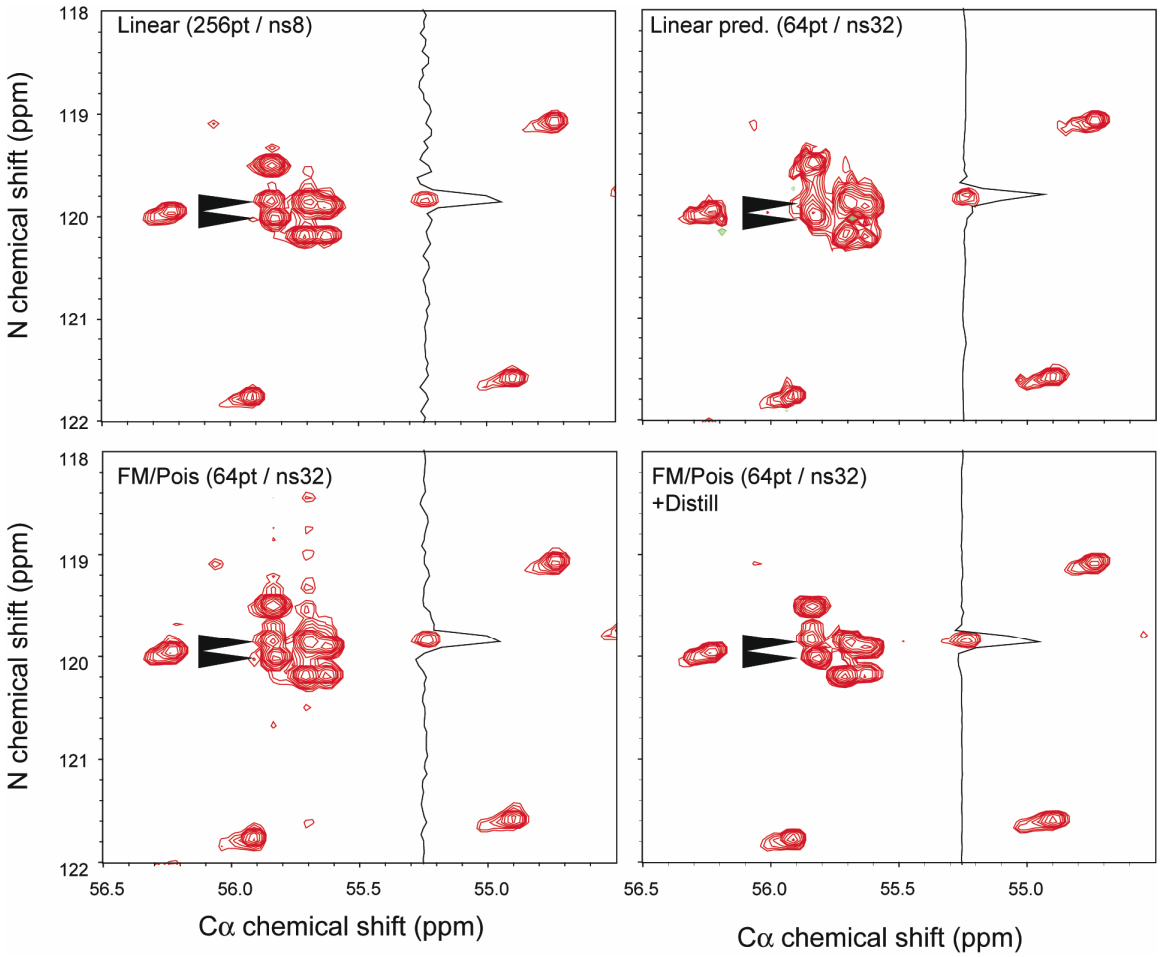

Figure S3: Comparison of the resolution between linearly sampled data with 256 time points and 8 scans per increment **(top left**), linear prediction using the first 64 data points with 32 scans per increment, predicted to 256 points, and processed with an order parameter of 30 (**top right**), non-modulated Poisson gap sampling of 64 increments followed by FM reconstruction[2] without (**bottom left)** and with distillation (**bottom right**).

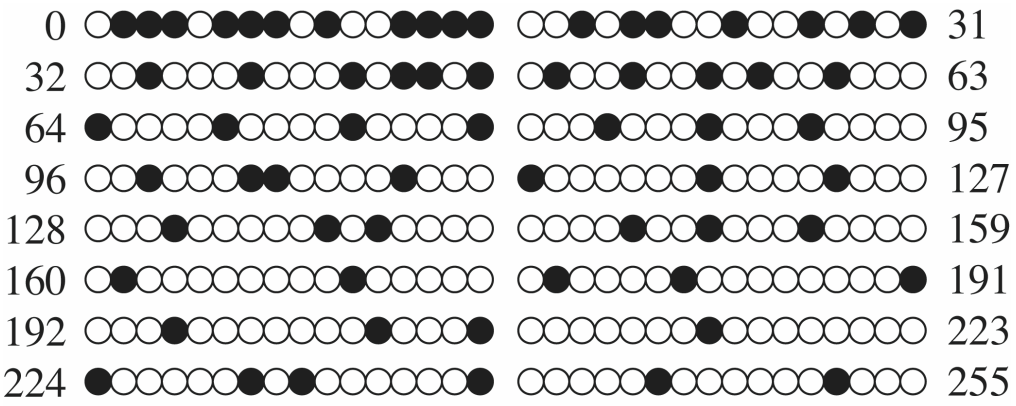

Figure S4: Graphical representation of the sinusoidal Poisson-Gap sampling Schedule used in the manuscript. Dark circles represent obtained data, unfilled circles non-obtained data. The gap in the middle of the line is for visualization aid only and groups the circles in groups of 16.

C-program for calculating sinusoidal weighted Poisson-gap sampling schedules.

The sampling schedule for Poisson-gap sampling can be calculated with the following C-program.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

// generate random Poisson-distributed numbers as given
// by Donald E. Knuth (1969). Seminumerical Algorithms.
// The Art of Computer Programming, Volume 2. Addison Wesley

int   poisson ( double lmbd )
{
      double     L = exp( -lmbd );
      int        k = 0;
      double     p = 1;

      do {
            double     u = drand48();
            p *= u;
            k += 1;
      } while ( p >= L );

      return( k-1 );
}

int   main ( int argc, char** argv )
{

      float s = atof( argv[1] );  // seed value e.g. 1.0
      int   p = atoi( argv[2] );  // sampled # of indices e.g 64
      int   z = atoi( argv[3] );  // total # of indices e.g. 256
      int   i;          // Fourier grid index, e.g. 1 through 256
      int   k;          // generated gap size
      int   n;          // temporary # indices
```

```
      int   *v;        // temporary storage vector
      int   j;         // wrking variable
      float ld = (float) z / (float) p;  // establish 1/fraction
      float adj = 2.0*(ld-1);     // initial guess of adjustment

      srand48( s );

      v = ( int* ) malloc( z*sizeof( z ) );

      do {
            i = 0; n = 0;
            while ( i < z ) {
                  v[n] = i;
                  i += 1;
k =poisson(adj*sin((float)(i+0.5)/(float)(z+1)*1.5707963268) );
                  i += k;
                  n += 1;
                  }
            if ( n > p ) adj *= 1.02;  // too many pts created
            if ( n < p ) adj /= 1.02;  // too few pts created

      } while ( n != p );   // if not at first, try, try again

      for ( j = 0 ; j < p ; k++ ) printf( "%d\n", v[j] );

}
```

References

(1) Delaglio, F.; Grzesiek, S.; Vuister, G. W.; Zhu, G.; Pfeifer, J.; Bax, A. *J Biomol NMR* **1995**, *6*, 277-93.
(2) Hyberts, S. G.; Frueh, D. P.; Arthanari, H.; Wagner, G. *J Biomol NMR* **2009**.