Application of Compressive Sensing to Simulated Chemical Spectra

Camille Houferak, Joe Kasper, Joseph J. Radler, Shichao Sun

March 13, 2016

Abstract

Abstract

I. Introduction and Overview

Computational chemistry seeks to construct fine-grained models of chemical systems in order to model and predict experimentally observable quantities in support of experimental research. A major implementation of computational modeling is the construction of idealized chemical spectra, which are essentially the 'fingerprints' of a chemical species in a system of interest. Traditionally, spectra have been generated experimentally, but recently have been generated from models of molecular electronic structure which comprises a major field of study in the computational chemistry community. However, generating predictive spectra from model and simulation can be a difficult undertaking, as the number of degrees of freedom for a so-called *full configuration* solution increases exponentially (or factorially!) for each additional electron found in the system. Full configuration calculations are ideal for capturing *all* of the physics of a system, but come at great computational expense as every interaction between each particle must be accounted for in the Hamiltonian and the second-order differential *Schrödinger Equation* solved numerically. If atomic nuclear dynamics are also modeled, this increases the complexity even further as each solution for each time step. Consequently, approximations and calculation schemes that minimize the complexity and computational cost of a model are highly valued in the field. Accordingly, techniques for reducing or removing the necessity of small time-steps and long timescales are also of great utility, as dynamical simulations examining nuclear motion often require picosecond timescales to capture the physics of interest whereas models examining only electronic dynamics require only femtosecond timescales.

Things are not so bleak, however, as there is an abundance of mathematical techniques for reducing the scale of such problems. This study focuses on one such technique called

Compressed Sensing (CS) that is used in data analysis to reduce the necessary sampling rate. CS does this by assuming a high degree of sparsity in a signal and makes use of the L^1 norm minimization technique to locate the vector of signal transform coefficients that is the most sparse. As CS is frequently applied to time-frequency analysis problems, we apply it to modeled chemical spectra which heavily feature Fourier analysis for signal resolution. This particular investigation focuses on *Optical Absorption Spectroscopy* – an electronic process in molecular systems where electrons are excited to higher discrete quantum states upon absorption of incident photons.

Modeling absorption spectra computationally presents a unique challenge since the propagation of electronic states through time is the fundamental process behind the generation of experimentally observable spectra. Several schemes exist for performing calculations that capture electron dynamics, however here we focus only on simulated spectra obtained through a *Real-Time Time-Dependent Density Functional Theory* (RT-TDDFT) scheme which solves electron dynamics by self-consistently solving a set of pseudoeigenvalue equations of electronic wavefunctions for each iteration in time, then solving an electron density functional equation based on those wavefunctions. The calculations are performed while holding the nuclear positions fixed (using the *Born-Oppenheimer approximation*) in order to reduce the number of degrees of freedom in the system of interest. As one can imagine, this can become prohibitively expensive for large systems with many electrons or large numbers of time steps; therefore we seek to find the best CS sampling rate to reduce the number of necessary time steps to achieve satisfactory optical absorption spectral resolution for molecular systems.

This investigation divides the examination of compressed sensing into two parts. *Part One* compares the output of the Fast Fourier Transform (FFT), Discrete Sine Transform (DST), and Discrete Cosine Transform (DCT) when applied within a CS algorithm to an idealized, known linear combination of sinusoidal functions representing our absorption signal. *Part Two* explores the application of CS utilizing the SPGL_1 L^1 norm minimization algorithm to our simulated spectra for water obtained by RT-TDDFT calculations. All code is implemented in MATLAB (see *Appendix B.*)

II. Theoretical Background

A. An Introduction to Optical Absorption Spectroscopy

B. Time-Dependent Density Functional Theory (TDDFT)

C. Compressive Sensing

D. Applications to Chemical Spectroscopy

III. Algorithm Implementation and Development

Part One – Known Ideal Signal

(1.) Create idealized signal

(2.) Choose sampling rate

(3.) Generate Discrete Sine Transformation (DST) matrix

(4.) Solve the norm minimization equation with Basis Pursuit Denoising (BPDN)

(5.) Check that error is small

(6.) Perform signal reconstruction from CS coefficients c

(7.) Normalize and Plot Reconstructed Signal

Part Two – Simulated TDDFT Water Spectra

(1.) Read in RT-TDDFT Simulated dipole files for each spatial component

Our RT-TDDFT simulation produces three data arrays corresponding to time-series of dipole moment orientations in each of the three spatial dimensions of the form

$$\mathbf{p}_w = \begin{pmatrix} (p_w)_x(t_s) \\ (p_w)_y(t_s) \\ (p_w)_z(t_s) \end{pmatrix}$$

for $w = x, y, z$ and $0 \leq s \leq 50000$.

(2.) Process for FFT

For the traditional FFT process, we first define a damping function

$$f(\tau) = e^{-(t-t_0)/\tau}$$

where τ is a our damping constant, and we damp the time series by applying $f(\tau)$ in the following way

$$p_i^{damped} = p_i \times f(\tau).$$

At this point, setting $\tau = 150$ for a 50000 time-step simulation, the function is damped such that only the first 500 or so time iterations are relevant, and the remainder may be truncated. Applying the Fourier Transform to the damped dipole time series

$$\alpha_{ij}(\omega) = \frac{1}{\kappa_i} \int_0^\infty e^{i\omega t}[p_j(t) - p_j(t_0)]dt$$

yields the polarizability tensor α such that each component α_{ij} (for $i, j = x, y, z$) allows us to calculate the photon absorption cross-section for the molecule given by the equation

$$\sigma(\omega) = \frac{4\pi\omega}{3c} \Im \sum_{i=x,y,z} \alpha_{ii}(\omega)$$

where we see the absorption cross section is simply the imaginary part of the trace of the polarizability tensor. The frequency spectrum is plotted for σ as a function of frequency ω.

(3.) Process for CS

In contrast, the Compressive Sensing algorithm uses a different process that does not require damping. The dipole moments and polarizability tensor are constructed similarly to the FFT scheme

IV. Computational Results

A. Ideal Data Cases

In order to test the validity of our CS algorithm for data reconstruction, we first applied it to idealized data sets represented by known linear combinations of sinusoidal functions. Having knowledge of the exact "signal" *a priori* allows us to determine the sampling rates and discrete transforms that result in the best approximations under ideal conditions.

B. Simulated Optical Absorption Spectra Cases

V. Summary and Conclusions

References

[1]

[2]

Appendix A MATLAB Functions and Implementations

function subsection

Appendix B MATLAB Code

First, we have the script *compressed_sensing* which manipulates the idealized case from *Part One*:

```
%Insert Matlab Code Here.  Whitespace is retained within {Verbatim}
%Compressed Sensing
%AMATH 582 Final Project
%J. Kasper, C. Houferak, J. Radler, S. Sun

%% Step 1: Known Signal
clc;
clear;
N = 2000; %total time length (ENTIRE signal)
t2 = linspace(0,1,N+1); t=t2(1:N);

S = 2*sin(10*t*(2*pi))+2*sin(7*t*(2*pi))+3*cos(2*t*(2*pi)); %idealized signal

n=400; %number of points to sample
samp = S(1:n)';

figure(1)
subplot(2,1,1)
plot(t,S)
xlabel('Time')
ylabel('Signal Amplitude')
title('Idealized Signal')

subplot(2,1,2)
plot(t(1:n),samp)
xlabel('Time')
ylabel('Signal Amplitude')
title('Short Sample')

deltaOmega = t(N)-t(1); %frequency difference

%generate Discrete Sine matrix
```

```
for j = 1:n
    for k = 1:N
        F(j,k) = (2/pi)*deltaOmega*sin(j*t(k));
    end
end

% % generate Discrete Cosine matrix
% for j = 1:n
%     for k = 1:N
%         F(j,k) = (2/pi)*deltaOmega*cos(j*t(k));
%     end
% end

%solve the equation F*x = samp
sigma = 0.05;
opts = spgSetParms('verbosity',0);         % Turn off the SPGL1 log output
x = spg_bpdn(F, samp, sigma, opts);

%check if the error is small
norm(samp-F*x)

%reconstruct the signal from CS coefficients
csSignal = zeros(size(t));
for i = 1:N
    csSignal = csSignal + sin(deltaOmega*t*i)*x(i);
end

figure(2)
plot(t,csSignal)
xlabel('Time')
ylabel('Signal Amplitude')
title('Compressed Sensing Signal')


x = x/max(abs(x)); %normalize

St2 = abs(fftshift(fft(S)));
St = St2(N/2+1:N)/max(St2); %take the last half (positive frequencies)

sampt2 = abs(fftshift(fft(samp)));
sampt = sampt2(n/2:n)/max(sampt2);
```

```
%frequency domain for CS signal
for i = 1:length(t)
    freq(i) = i/(2*pi);
end

%frequency domain for entire signal
for i = 1:length(St)
    freq2(i) = i-1;
end

%frequency domain for Fourier transformed sample
for i = 1:length(sampt)
    freq3(i) = (N/n)*(i-1);
end


figure(3)
hold on
subplot(3,1,1), plot(freq2,St);
title('Ideal Frequency Spectrum')
xlabel('Frequency (Hz)')
ylabel('Normalized Intensity')
axis([0 100 -.2 1])
subplot(3,1,2), plot(freq,x)
title('Compressed Sensing Frequency Spectrum')
xlabel('Frequency (Hz)')
ylabel('Normalized Intensity')
axis([0 100 -.2 1])
subplot(3,1,3), plot(freq3,sampt)
title('Fourier Transform Frequency Spectrum')
xlabel('Frequency (Hz)')
ylabel('Normalized Intensity')
axis([0 100 -.2 1])
hold off
```

Next we have the code for applying CS to simulated optical absorption spectra of water calculated with RT-TDDFT:

```
x=csvread('h2o_x_RealTime_Dipole.csv',1,0);
y=csvread('h2o_y_RealTime_Dipole.csv',1,0);
```

```
z=csvread('h2o_z_RealTime_Dipole.csv',1,0);
% read the data of dipole moment


tic
nn=6000;             % number of data points (long edge of F matrix)
t=x(1:nn,1);         % time
m=fix(nn/5);           %number of used data points  (short edge of F matrix)

T=t(nn);            %simulated simulation time. actual simulation time will be T/5.
dt=t(2)-t(1);        % time step
w=(pi/T)*[0:nn-1];   % frequency
kick=0.0001;         % amplitude of electric field E




pxt=x(1:m,3);
pyt=y(1:m,4);
pzt=z(1:m,5);
% pick the data of angular momentum

damp_const=150;
damp = exp(-(t(1:m)-t(1))/damp_const);


px=pxt-pxt(1);
py=pyt-pyt(1);
pz=pzt-pzt(1);
% subtract mean value

px=px.*damp;
py=py.*damp;
pz=pz.*damp;

% damp

%change to atomic unit
px=0.393456*px;
py=0.393456*py;
pz=0.393456*py;
```

```
 %%
% do L1 optimization
opts = spgSetParms('verbosity',0);
FF=idst(eye(nn,nn));
F=FF(1:m,:);

ax  = spg_bp(F,px,opts);

ay  = spg_bp(F,py,opts);

az  = spg_bp(F,pz,opts);
%%

sigma=(4*pi/(3*137*kick))*w'.*(ax+ay+az);  % sigma=blabla*(axx+ayy+azz)


w=27.2114*w;                               %change frequency w into eq
number=find(w<40);                         % confine the w to the frequency interval of we
np=length(number);
figure (2)
plot(w(1:np),sigma(1:np))
toc
```

Appendix C Custom MATLAB Functions

Custom MATLAB Function 1