UNIVERSITY OF CAMBRIDGE
Department of Engineering

Bayesian Harmonic Models for Musical Transcription and Analysis

Author Name: Max Fryer
Supervisor: Simon Godsill
Date: 31/05/2023

I hereby declare that, except where specifically indicated, the work submitted herin is my own original work.

Signed  *date* 31/05/2023

Bayesian Harmonic Models for Musical Transcription and Analysis

Max Fryer
Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ - UK
mf699@cam.ac.uk

Abstract

We describe in this report techniques for multiple pitch estimation and musical transcription based on the Bayesian harmonic model originally developed by Davy, Godsill and Idier[1]. The motivation and structure of a simple model for harmonic resonators that include string and wind instruments is reiterated before a thorough analysis of the Bayesian setting is considered. Unlike previous works that use reversible-jump MCMC methods to generate a suitable posterior distribution for pitch estimation and parameter generation we instead opt for the novel variant FFT-SA MCMC algorithm which uses a coarse estimate generated using STFT methods to seed a generic simulated annealing algorithm. We find that this simple technique is extremely effective for finding the fundamental frequencies present in both monophonic and polyphonic quasi-static recordings and can be combined with binning to produce piano roll transcriptions. Careful consideration is taken to improve the efficiency of large matrix manipulations required for Metropolis-Hastings proposal-acceptance steps. We finally implement a method to convert our estimated transcription into a MIDI representation that can be used by music software plug-ins for further analysis and compare our estimate with the true score used.

I Introduction

"Mathematics and Music, the most sharply contrasted fields of intellectual activity which one can discover, and yet bound together, supporting one another as if they would demonstrate the hidden bond which draws together all activities of our mind, and which also in the revelations of artistic genius leads us to surmise unconscious expressions of a mysteriously active intelligence." [2]

Since the advent of computing, there has been a steady progression in computer-aided techniques for the analysis of music. It is arguably the ultimate problem to work on for the nascent signal-processing researcher, for not only does it draw in methods and ideas from all domains of statistics, but because the results are oftentimes so naturally motivated. Take, for example, the experience of listening to your favourite Britpop band. As you listen, you might gradually focus on what one instrument is playing and begin to hum along. Subconsciously, your auditory system has just achieved effective source separation, cleanly isolating a single line of music in the presence of high levels of correlated noise using a complex abstraction of timbre. Perhaps it is in part because humans are so gifted at music analysis, even if subconsciously, that we take such interest in trying to model and simulate it.

Automatic Music Transcription (AMT) is the process of converting a received acoustic signal into some form of music notation, ether a piano-roll format or a traditional score. The most direct application of this is for recording live performances into a stable, compact, and consistent representation. This is most applicable for genres like jazz, or traditional music where performances are either inherently improvised or passed down by word of mouth. Less directly, conversion into a written medium also allows for other musicians to further analyse and modify the music, either for educational or remix purposes.

Polyphonic AMT is a highly ill-posed problem that includes many possible subtasks. Simultaneous root frequency estimation, time-varying amplitude tracking and inference about instrument-specific structures are all further complicated by the presence of transient sounds and background interference. After some stagnation in improvements on performance tests there has been a recent surge of research in recent years due to the prospects of deep neural network (DNN) approaches, that use large datasets to train models.[3][4][5]

In this work however, we continue with a direct modelling approach, extending the analysis of Bayesian harmonic models to analyse quasi-static musical signals as presented in the works of Davy, Godsill and Idier.[1][6][7] Through this approach we incorporate the rich scaffolding of priors used in western music in tandem with the received signal to compute a fully informed estimate of pitches present in our signal.

The report is organized as follows: in Sec. **I** we introduce some important signal processing and music theory concepts that we use in later analysis. In Sec. **II** we develop a statistical model for our recorded signal that incorporated hierarchical priors into a Gabor-atom structured approach that localises analysis in time and frequency. In Sec. **III** we develop MCMC techniques used to estimate polyphonic root frequencies simultaneously. In Sec. **IV** we apply our technique to the task of musical transcription. In Sec. **V** we conclude.

Musical Pitch

At a fundamental level, instruments can be thought of as a means of producing easily controllable, locally stable sound. In the case of string instruments like guitars, pianos, and violins standing waves are stimulated by plucking, striking, or bowing to create a fundamental tone, along with a series of higher order harmonics. A similar effect is observed in wind instruments like flutes, clarinets and trumpets with vibrating wind columns acting as harmonic resonators. Such a signal can be well described as a harmonic series of sinusoids at multiples of a fundamental frequency (also commonly termed root frequency) resulting in what is perceived by the human auditory system as a musical note. With a few exceptions (particularly percussive), such as cymbals, bass drums, the regularity of the periodicity of individual musical notes means we can often equate the perceived pitch with the fundamental frequency.[8]

Of particular interest is the perception of two signals whose fundamental frequencies have the ratio 2:1 as sounding very similar to the extent that harmonies played octaves apart are considered largely musically equivalent. This is likely due to the harmonics of a fundamental at f_0, at frequencies f_0, $2f_0$, $3f_0$ etc forming a proper superset of the harmonics of a note with fundamental $2f_0$ (or indeed $4f_0$, $8f_0$ corresponding to octave jumps) with higher order frequencies $2f_0$, $4f_0$, $6f_0$ etc. Other notable ratios including 'perfect fifths' with ratio 3:2, and 'major thirds' with ratio 5:4 are perceived as having strong similarities.

For the practical purpose of being able to play in any key without retuning, while approximately maintaining these natural note ratios that we find pleasing, the western music system has adopted an equal tempering scale system. Octaves are divided into twelve equal steps on a logarithmic scale, with each semitone step being $2^{(1/12)} \approx 1.06\times$ larger than its predecessor. We note that $(2^{(1/12)})^7 = 1.498 \approx 3/2$ and $(2^{(1/12)})^4 = 1.260 \approx 5/4$, errors of 1.96 cents and 13.69 respectively (1 cent is 1/100 semitone). On the piano these are ordered further still with the 'white notes' (denoted by C, D, E, F, G, A, B) each being two semitones apart, separated by a 'black note', referenced either by a sharp or a flat (Ab = G#), except for E/F and B/C which are just one semitone apart.

Harmony

Harmony is used to describe how a combination of frequencies are perceived together to form a colour or mood of sound. These combinations, similarly to octaves, are generally translation invariant, and remain similarly perceived when played in different keys. One common classification of these combinations (or chords) is to distinguish between major or minor, with the former being characterized by pitches with simple frequency ratios and therefore common harmonics. For automatic detection of chords, overlapping harmonics pose a challenge since there are multiple possible root frequencies. Some common three and four note chords, along with their harmonic ratios, are outlined in Table.1.[9][10]

Chord	Semitones between Notes	Freq. Ratios
Major	4-3	4:5:6
Minor	3-4	10:12:15
Diminished	3-3	160:192:231
7th	4-3-3	20:25:30:36
Min. 7th	3-4-3	10:12:15:18

Table 1

Short-Time Fourier Transform

In general, for the analysis of music, as in other audio-related applications, we are interested in describing the time-varying energy across different frequency bands. A popular tool for this is the short-time Fourier transform (STFT). Unlike the standard Fourier transform the STFT is defined over successive signal frames, providing time-localized frequency information as opposed to global signal information as a standard Fourier analysis would. We define the frame at time t_0, computed with window w and denoted $s^w_{t_0}(t)$, as

$$s^w_{t_0}(t) = x(t)w(t_0 - t). \tag{1}$$

Windows are symmetric and standard varieties include Hamming, Hanning and Gaussian. Formally, let $\mathbf{x}$ be a discrete-time signal sampled uniformly from a waveform at rate F_sHz. Using an N-point hamming window, $w(n) = 0.54 - 0.46\cos(2\pi n/N)$ for $n = \{0, 1, \ldots, N-1\}$ and an overlap of half a window length, we obtain the discrete STFT

$$X(t,k) = \sum_{n=0}^{N-1} w(n)x(n + t \cdot N/2)\exp\{-j2\pi kn/N\} \tag{2}$$

where $t \in [0 : T-1]$ and k $\in [0 : K]$. T describes the number of frames, $K = N/2$ is the index of the last unique frequency value and so $X(t,k)$ corresponds to the window beginning at time $t \cdot N/(2F_s)$ in seconds and frequency (in Hz) is calculated using

$$f_{coeff}(k) = \frac{k}{N} \cdot F_s. \tag{3}$$

Typical values of $F_s = 44100$ and $N = 4096$ give a window length of 92.8ms, a time resolution of 46.4ms and frequency resolution of 10.8Hz.[11]

When applying the STFT to signals of varying frequencies it's important to note trade-off between time and frequency resolution. The more samples used, the more precise the frequency. A narrow-width window would result in better temporal accuracy, but at the cost of accuracy in the frequency domain. We can visualize the STFT as an intensity plot of the magnitude, $|X(t,f)|$ over time in a spectrogram as shown in Fig. 1.

A useful interpretation of the STFT we use in this paper is as a dot product between $x(t)$ and the windowed complex sinusoid $w(t-\tau)e^{-j2\pi f\tau}$ to form a 'Gabor atom'. This family of elementary time-frequency atoms forms a basis called a *Gabor basis*, a structure we will use extensively for analysis later.[12]

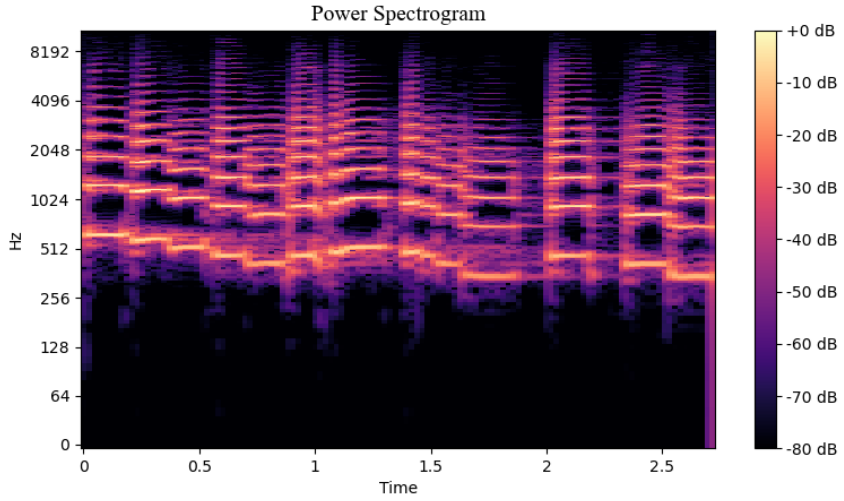


Figure 1: Log-frequency scaled short-time Fourier transform of a trumpet solo extract.

II Bayesian Harmonic Models

In this section we develop the model used in later sections for analysis of musical signals. Beginning with a review of modelling uncertainty that leads into the simplest monophonic (one note) form of the harmonic model, further sections extend definitions to polyphonic (multiple note) modelling in a way that closely mirrors previous works. [13][14][6][7]. Afterwards, priors are defined and a formal method of combining with the likelihood is developed to form a fully Bayesian setting.

Probabilistic Modelling and Likelihood Functions

Random variables model uncertainty in observed physical phenomenon. In the case of audio processing uncertainty could be caused by varying air temperature, speaker imperfections, conversion artifacts, alongside microphone attenuation and frequency response. It quickly becomes impractical to model these all deterministically. Random variables offer a solution, regardless of the cause of deviation or uncertainty. Taking a pure sine wave transmission, we can model the received discrete signal,

$$x(n) = \alpha \sin(2\pi k_0 n + \phi_0) + \epsilon(n) \quad \text{for} \quad n = 1, ..., T \tag{4}$$

where k_0 is the unknown sine frequency, ϕ_0 is the initial phase and α is the signal amplitude. We can model the error, or deviation from the exact model with a random additive-noise variable $\epsilon(n)$.[15] For the sake of generality we model this noise as being independent identically distributed (i.i.d) such that the joint pdf of noise samples is found by the product of the pdfs of each sample, $p(\epsilon(1)\epsilon(2)...\epsilon(T)) = p(\epsilon(1))p(\epsilon(2))...p(\epsilon(T))$. Assuming our model to be an accurate representation of the physical system our noise can be assumed to have zero mean, it is assumed to be a stationary white noise process.

Denoting the set of unknown parameters, $\boldsymbol{\theta}$ and given the recorded signal $\mathbf{x}$ and the model $f(\boldsymbol{\theta})$, we note that some values of theta are more likely observed than others depending on how close they lie to our model predictions. Alternatively, if we

are given the set of unknown parameters $\boldsymbol{\theta}$ then the received signal can be thought of as a random vector conditional on $\boldsymbol{\theta}$. Using i.i.d Gaussian noise, with diagonal covariance matrix of variance σ_0^2 we can explicitly define the likelihood of our data as the multivariate normal,

$$p(\mathbf{y}|\boldsymbol{\theta}) = (2\pi\sigma_v^2)^{-N/2} \exp\left[-\frac{1}{2\sigma_v^2}\|\mathbf{y} - f(\boldsymbol{\theta})|^2 \right]. \tag{5}$$

Linear Gaussian Model and Maximum Likelihood Estimation

We begin by describing a basic chord model for musical audio. For the moment we consider a short frame of data, in which it can be assumed that note transitions do not occur and that the data is generated from a parametric function with an additive stationary white noise term,

$$x_n = g_n(\theta, e_n). \tag{6}$$

Owing to the close relationship between the higher harmonics created by harmonic resonators and the root frequency, the kth note out of a chord containing K notes can be written as

$$s_{k,t} = \sum_{m=1}^{M} a_m sin(m\omega_k t) + b_i cos(m\omega_k t) \tag{7}$$

For $t \in \{0, ..., N-1\}$. Here $M-1$ is the number of harmonics, or in the more general case, overtone partials present in the note alongside the fundamental partial with angular frequency w_k. The amplitude of the partial can be calculated as $\tan^{-1}(b_i/a_i)$ and the phase, by $\sqrt{a_i^2 + b_i^2}$. For convenience in the discrete domain $w_k \in (0, \pi)$, which corresponds to $\frac{w_k}{2\pi} w_s$. More concisely using vector notation, we model the signal corresponding to a note as

$$\mathbf{x} = \mathbf{G}\boldsymbol{\theta} + \mathbf{e} \tag{8}$$

with basis functions and weights given by,

$$\mathbf{G} = [\mathbf{c}(\omega_1) \quad \mathbf{s}(\omega_1) \quad \cdots \quad \mathbf{c}(\omega_{11}) \quad \mathbf{s}(\omega_{11})], \qquad \boldsymbol{\theta} = \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & \cdots & a_{11} & b_{11} \end{bmatrix}^T.$$

In theory these harmonics extend indefinitely upwards, with the higher harmonics decaying faster after the initial stimulation. However, owing to our 44.1KHz sampling frequency the Nyquist limit suggests that we should restrict our analysis to the first 11 or so harmonics (depending on the note root frequency). Fig.3 suggests that this is sufficient to cover the vast majority of the energy in the signal.

Extending this model to a chord is as straightforward as linearly superposing a number of individual notes,

$$y_t = \sum_{k=1}^{K} s_{k,t} + \nu_t. \tag{9}$$

Armed with our likelihood function (eq .5), which can be thought of as a measure of similarity between model and signal, a natural estimate of the value of theta would be one that maximizes the likelihood,

$$\boldsymbol{\theta}^{ML} = \underset{\boldsymbol{\theta}}{\text{argmax}}[p(\mathbf{x}|\boldsymbol{\theta})] \tag{10}$$

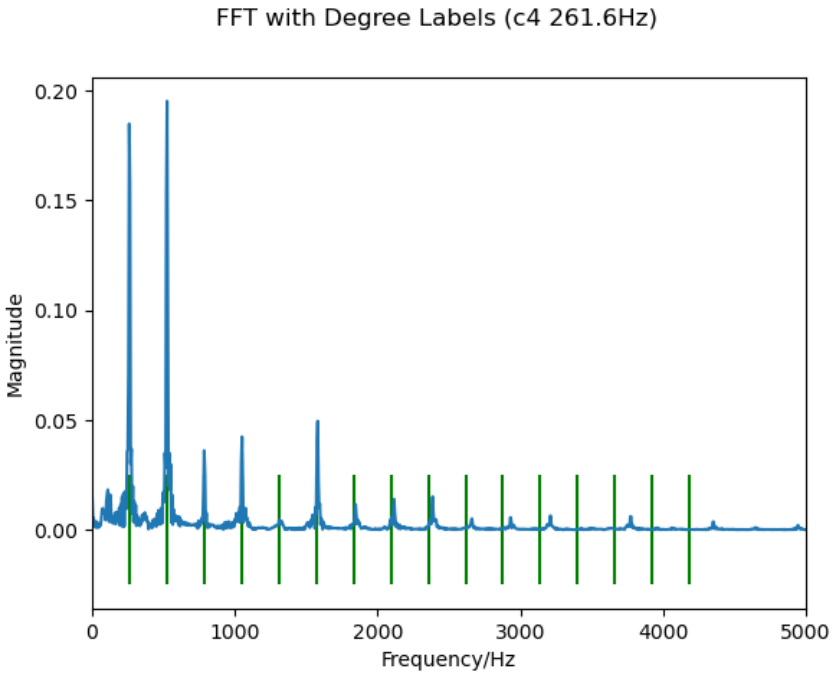


Figure 2: Snapshot of frequency domain of piano note.

For a recording of a single note on the piano (C4, 261.6Hz) we follow a simple adjusted procedure for calculating the maximum likelihood weights for our model. This is the result of a) assuming that 500 samples (roughly 10ms) is a short enough period to consider the recording quasi-stable and b) that the physical processes underlying the decaying weights can be approximated well by linearly interpolating between 20ms points. The results of this procedure are shown in Fig.3.

1. Take samples of length 500 samples ($\sim$ 10ms) around each 1000^{th} sample ($\sim$ 20ms).

2. Calculate maximum likelihood weights up to and including 11 degrees of harmonics. $\boldsymbol{\theta}^{ML} = (G^T G)^{-1} G^T \mathbf{x}$.

3. We then compute all samples using linear interpolation (or triangular weighting functions) before resynthisizing using our model.

For simple, steady harmonic sounds, the above model is close enough to afford more development and analysis, especially if we choose errors and prior assumptions to be Gaussian. However, for any more general analysis this turns out to be an overly simple approach, foremost the assumption of linearly varying weights a and b over time. We therefore parameterize our weights as follows, expanding a and b on a finite Gabor basis localized in time and frequency. A set of 50% overlapped Hanning windows $i \in \{0, ...I-1\}$ is used as a parameterized envelope (see Fig.4),

$$\Phi[t] = \sum_{i=0}^{I} 0.5 - 0.5\cos\left(\frac{2\pi n}{(i+1)M-1}\right), \qquad 0 \leq n \leq (i+1)M-1. \tag{11}$$

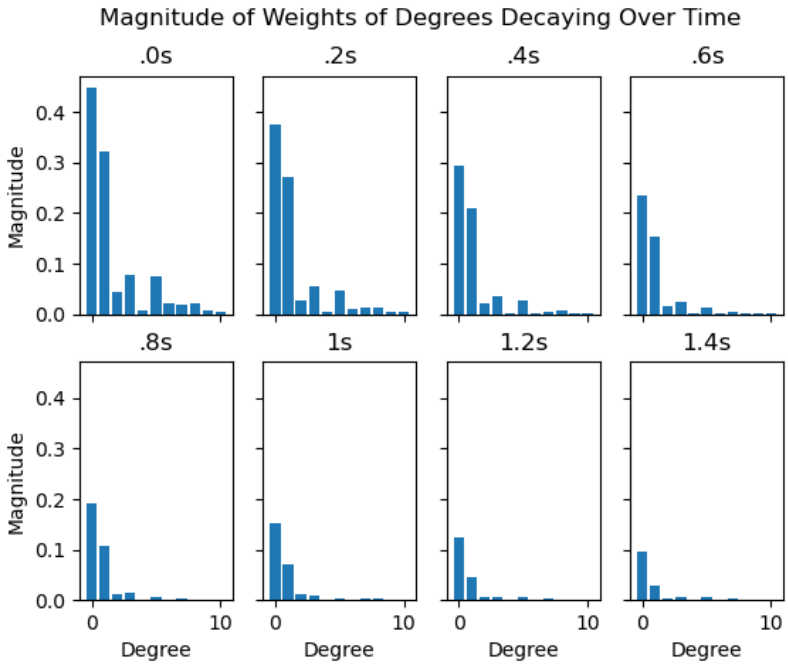


Figure 3: Maximum likelihood weights up to 11th degree harmonic for simple piano note.

Let $g(t)$ be a Gabor atom of length $2N_g + 1$, localized at time t_0 and frequency w_0. The two elements of this atom are

$$g_c(t) = \begin{cases} \Phi[t]\cos(w_0 t) & t = t_0 - N_g, ..., t_0 + N_g \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$g_s(t) = \begin{cases} \Phi[t]\sin(w_0 t) & t = t_0 - N_g, ..., t_0 + N_g \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Our model, comprising K notes (fundamental frequencies), each of which with M partials, over the time-localising I windows, which we use for the remainder of our analysis is then defined as,

$$y[t] = \sum_{k=1}^{K}\sum_{m=1}^{M_k}\sum_{i=0}^{I}\Phi[t - i\Delta_t]\left\{a_{k,m,i}cos(\frac{\omega_{k,m}}{\omega_s}t) + b_{k,m,i}sin(\frac{\omega_{k,m}}{\omega_s}t)\right\} + \nu[t]. \tag{14}$$

Considering the $\mathbf{G}$ matrix notation used in eq .8, this can be visualized as a 'G matrix of G matrices', now referred to as $\mathbf{D}$, multiplied by a series of enveloping Hanning window functions. More tersely, as

$$\mathbf{D} = \mathbf{G}_t^* = \Phi[t] \times [\mathbf{G}_1 \quad \mathbf{G}_2 \cdots \quad \mathbf{G}_K]. \tag{15}$$

The unknown parameters for this model are the amplitudes, β, with length $2R(I+1)$, where $R = \sum_{k=1}^{K} M$ is the total number of partials. We can now compactly write our model in the general linear form as

$$\mathbf{y} = \mathbf{D}\boldsymbol{\beta} + \nu. \tag{16}$$

Including the variance of the noise, σ_v^2 and the frequencies and number of partials the polyphonic Bayesian harmonic model has total unknown parameters $R(2I + 3) + 2$.

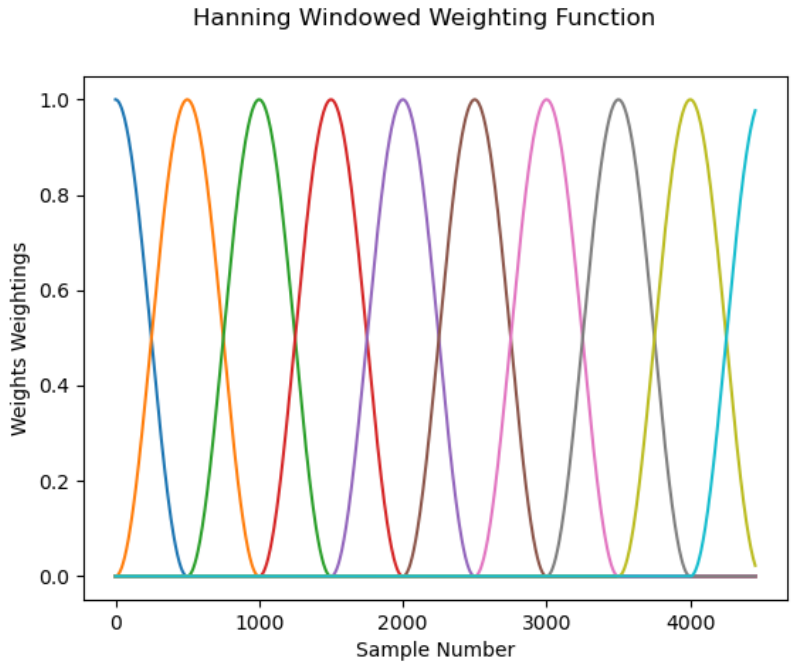


Figure 4: 50% overlapped Hanning windows used to localise analysis in time.

Bayesian Framework

The most powerful result of Bayesian statistical theory is the ability to 'reverse the conditioning' and incorporate prior knowledge about the distribution of random variables alongside reasoning from data to deduce the *posterior distribution*. Bayes' Rule is as follows,

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{x})}.$$

In the context of our model, the likelihood, $p(\mathbf{x}|\boldsymbol{\theta})$, is as described in eq.(5) and the prior information, $\boldsymbol{\theta}$, is the vector of unknown parameters $[\boldsymbol{\beta}, \sigma_v^2, \boldsymbol{\omega}, \mathbf{M}, K]^T$. We choose to structure the priors hierarchically such that each term is more pleasant to understand and model, as a product of simpler priors,

$$\begin{aligned} p(\boldsymbol{\beta}, \sigma_v^2, \boldsymbol{\omega}, \mathbf{M}, K) =& p(\boldsymbol{\beta}|\sigma_v^2, \boldsymbol{\omega}, \mathbf{M}, K) \\ & \times p(\boldsymbol{\omega}|\mathbf{M}, K)p(\mathbf{M}|K) \\ & \times p(K)p(\sigma_v^2) \end{aligned} \tag{17}$$

which we detail below.
a) The prior distribution on the $2R(I+1)$ weights vector given the other variables $p(\boldsymbol{\beta}|\sigma_v^2, \boldsymbol{\omega}, \mathbf{M}, K)$ is selected as a zero-mean, multivariate Gaussian with diagonal covariance matrix $(\sigma_v^2/\zeta^2)\mathbf{I}$, where ζ^2 can be interpreted as a signal-to-noise ratio (SNR).[6] As pointed out in Davy et Al. choosing such a covariance matrix is necessary for maximising the effectiveness of MCMC techniques described later.[1] As well, it is not clear that there should be a natural covariance between weights. While the amplitudes seem to decay as the order increases, because the phase information is held in the combination of $g_c[t]$ and $g_s[t]$, $\beta_{k,m,i}$ is as likely to be positive as negative. The hyper-parameter ζ^2 used to define the prior on $\boldsymbol{\beta}$ is chosen as an inverted

gamma distribution with small parameters,

$$p(\zeta^2) = IG(\alpha_\zeta, \beta_\zeta) \propto \frac{e^{\beta_\zeta/\zeta^2}}{\zeta^{2(\alpha_\zeta+1)}} \tag{18}$$

and is shown in Fig.5.
b) Second is the prior distribution for root frequencies, given the number of partials per note and the number of notes, $p(\boldsymbol{\omega}|\mathbf{M}, K)$. As in [1] a uniform distribution is chosen for $p(\boldsymbol{\omega})$ over the whole frequency range $[0, w_s/2M_k]$ for each note $k \in \{1, ..., K\}$. This is to mitigate aliasing issues by limiting the root frequencies such that even the highest order partials remain less than the Nyquist limit. A more instrument specific approach could be taken, for example modelling the non-linearity of piano partials more accurately using

$$\omega_{k,m} = m\omega_{k,1}\sqrt{\frac{1+m^2B}{1+B}} \tag{19}$$

but for generality we use the already described perfect harmonic model in eq .7.[15]
c) The prior over parameter M describes the number of partials for a given note. Again following previous works, we choose a distribution that encourages enough partials to capture the energy of the relevant signal but which discourages so many partials as to undercut the fundamental frequency and begin modelling low frequency noise. Both become a problem for signal reconstruction and signal separation. A balanced solution that reflects these preferences has been found to be a Poisson distribution for each M_k,

$$p(M_k = m|\Lambda_k) = e^{-\Lambda_k}\frac{\Lambda_k^m}{m!}, \quad \text{for } k = 1, \ldots, K \tag{20}$$

where the prior Λ_k is gamma distributed. This is equivalent to the single prior,

$$p(M_k = m) \propto \frac{\Gamma(m+\alpha_\Lambda)}{\Gamma(\alpha_\Lambda)m!}(\beta_\Lambda + 1)^{-m} \tag{21}$$

where $\Lambda(\cdot)$ is the Gamma function.
d) The noise prior $p(\sigma_v^2)$ is an inverted Gamma distribution, encouraging solutions that have a small residual energy.

$$p(\sigma_v^2) \propto (\sigma_v^2)^{-\psi_0/2-1}e^{-2/\mu_0\sigma_v^2}, \tag{22}$$

where ψ_0 and μ_0 are small.

Expanding the $\boldsymbol{\theta}$ of unknowns into the explicit variables discussed, the posterior distribution, defined as

$$\begin{aligned} p(\beta, \sigma_v^2, \omega, \mathbf{M}, K|\mathbf{y}) \propto & p(\mathbf{y}|\beta, \sigma_v^2, \omega, \mathbf{M}, K) \\ & p(\beta, \sigma_v^2, \omega, \mathbf{M}, K), \end{aligned} \tag{23}$$

becomes the central feature of our analysis. Ideally the problem of overfitting that is common with similar ML approaches is mitigated by the use of priors that penalize too many partials. Our objective, still to find the note parameters, can then by approached by finding the maximum posterior or MAP estimate,

$$(\hat{\boldsymbol{M}}, \hat{K}) = \underset{(\mathbf{M},K)}{\text{argmax}} p(\mathbf{M}, K|\mathbf{y}) \tag{24}$$

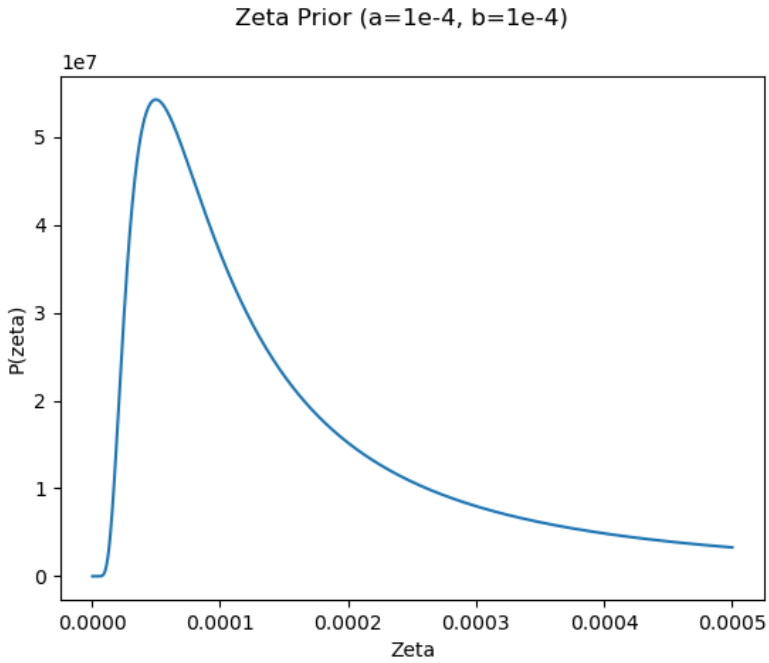


Figure 5: Prior over ζ^2 used define a diagonal covariance matrix $\boldsymbol{\beta}$ in relation to noise variance σ_v^2.

where

$$p(\mathbf{M}, K|\mathbf{y}) = \int p(\beta, \sigma_v^2, \omega, \mathbf{M}, K|\mathbf{y}). \tag{25}$$

After identifying these parameters, the weights, $\hat{\boldsymbol{\beta}}$ for a particular model are calculated using a minimum mean squared error (MMSE) estimator

$$\hat{\boldsymbol{\beta}} = \int \boldsymbol{\beta} p(\boldsymbol{\beta}, \sigma_v^2, \omega|y, \hat{\boldsymbol{M}}, \hat{K}) d\boldsymbol{\beta} d\sigma_v^2 d\omega. \tag{26}$$

Various methods are used to estimate this posterior, most of them utilizing an MCMC architecture. Since the ordering of the notes is non-unique there is the non-trivial issue of label switching, that is there exist differently-ordered vectors of root frequencies that contain the same frequencies but in different orders. Davy and Godsill developed the reverse jump MCMC algorithm to deal with this issue, using 'birth' and 'death' processes to fit the correct number of notes and partials. We use a more limited but simpler approach that we present later.[16][1] We can therefore estimate the posterior distribution $p(\boldsymbol{\beta}, \sigma_v^2, \boldsymbol{\omega}, \mathbf{M}, K)$, noting that the integrals involving $p(\boldsymbol{\omega}, \mathbf{M}, \mathbf{K}|\mathbf{y})$, $p(\sigma_v^2|\boldsymbol{\omega}, \mathbf{M}, K, \mathbf{y})$ and $p(\boldsymbol{\beta}|\sigma_v^2\boldsymbol{\omega}, \mathbf{M}, K, \mathbf{y})$ can all be computed analytically.

Standard computations lead to

$$\begin{aligned} p(\boldsymbol{\omega}, \mathbf{M}, K|\mathbf{y}) \propto & (\gamma_0 + \mathbf{y}^t \mathbf{P} \mathbf{y})^{(-N+v_0)/2} \det(\mathbf{S})^{1/2} p(\boldsymbol{\omega}|\mathbf{M}, K) \\ & \times \exp\left(\sum \Lambda_k\right) \frac{\Lambda'^k}{K!} \prod_{k=1}^{K} \frac{\Lambda_k^{M_k}}{M_k!} \end{aligned} \tag{27}$$

where $\mathbf{P} = \mathbf{I} - \mathbf{D}\mathbf{S}\mathbf{D}^t$ is an N-Dimensional square matrix, requiring the matrix inversion of the $2R(I+1)$ square matrix $\mathbf{S} = [\mathbf{D}^t\mathbf{D} + (1/\zeta^2)\mathbf{I}]^{-1}$. In a similar fashion

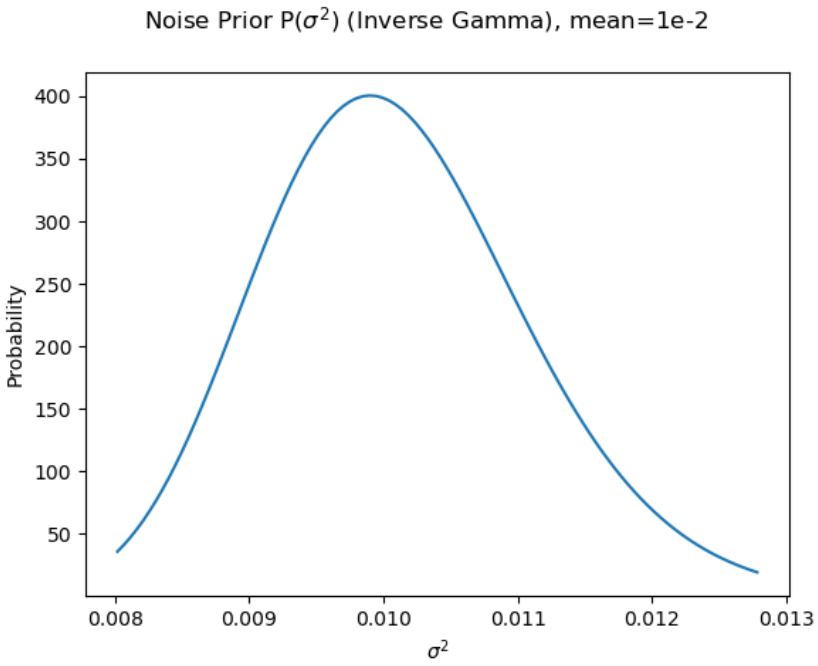


Figure 6: Inverse Gamma prior over signal noise.

the posterior distribution of σ_v^2 conditioned of the other parameters is an inverse gamma distribution,

$$\begin{aligned} p(\sigma_v^2|\boldsymbol{\omega}, \mathbf{M}, K, \mathbf{y}) =& \mathcal{IG}\Big(\frac{N+v_0}{2}, \frac{\gamma_0 + \mathbf{y}^t\mathbf{P}\mathbf{y}}{2}\Big) \\ &\propto \exp\Big(-\frac{\gamma_0 + \mathbf{y}^t\mathbf{P}\mathbf{y}}{2\sigma_v^2}\Big)\sigma_v^{(-N+v_0)/2} \end{aligned} \tag{28}$$

and the posterior distribution of $\boldsymbol{\beta}$ conditioned on the other parameters is the multinomial Gaussian

$$p(\boldsymbol{\beta}|\sigma_v^2\boldsymbol{\omega}, \mathbf{M}, K, \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}, \sigma_v^2\mathbf{S}) \tag{29}$$

with mean $\boldsymbol{\mu} = \mathbf{S}\mathbf{D}^t\mathbf{y}$ and covariance matrix $\sigma_v^2\mathbf{S}$.

III Markov Chain Monte Carlo Techniques for Pitch Estimation

Markov chain Monte Carlo algorithms are a class of methods for simulating stochastic processes. Generally speaking, the aim of these approaches is to output a chain of samples whose probability distribution asymptotically approaches a target distribution $\pi(\boldsymbol{\theta})$. The two most common flavours are the Gibbs sampler and the Metropolis-Hastings (MH) sampler. Metropolis-Hastings sampling is particularly useful when distributions are only known up to a constant of proportionality as in our posterior expression (as with many Bayesian inference problems). Although simulating independent samples of the process may intractable for integrals of the form,

$$I(\theta) = E_{\theta|x}[f(\theta)] = \int_{\theta} f(\theta)\pi(\theta|x)d\theta, \tag{30}$$

it is possible to simulate dependent samples, forming a irreducible Markov Chain whose stationary distribution is our distribution of interest, $\pi(\boldsymbol{\theta})$. Metropolis Hastings guarantees such chains, providing a consistent method for even the most complex stochastic process. We proceed by averaging the function over the series,

$$\hat{I}_N(\theta) = \frac{1}{N}\sum_{n=1}^{N} f(\theta^{(n)}) \rightarrow E_{\theta|x}[f(\theta)], \tag{31}$$

a process which converges to the target distribution $\pi(\boldsymbol{\theta})$ which in our case is the posterior. Gibbs sampling on the other hand is particularly useful in cases where the joint distribution of the variables is known, but sampling from it directly is challenging due to high-dimensionality or complex dependencies. Instead, samples are formed iteratively from the conditional distributions of each variable, given all the other variables in the distribution.

The basic application of MCMC algorithms that we use is to explore the space of possible values of $p(\boldsymbol{\omega}, \mathbf{M}, K|\mathbf{y})$ in a semi-random way, but with a tendency towards more likely proposals. After many iterations, the sampler will have converged to some maximum in the space which it is then unlikely to move away from. Samples taken from this series then are distributed according to the target distribution. At each iteration a proposal is selected via some semi-random proposal distribution, $\boldsymbol{\theta}^*(\boldsymbol{\theta})$. The probability of this value is compared to the probability of the current value and is accepted with a probability calculated according to the acceptance ratio,

$$\alpha(x, y) = \min\left(\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}, 1\right), \tag{32}$$

where $q(x, y)$ is the probability of proposing y from x and $\pi(x)$ is the likelihood of x. Either the proposed value is rejected and a new proposal is calculated, or it is accepted and the algorithm continues, with the next iteration starting from the new value of theta.

The first algorithm presented is a generic Metropolis-Hastings algorithm. $\boldsymbol{\theta}$ is sampled one at a time according to the proposal distribution $q(\boldsymbol{\theta})$ to form the proposed candidate, $\boldsymbol{\theta}^*$. This is randomly accepted or rejected according to the acceptance probability, $\alpha(x, y)$.

After an initial phase the chain begins to produce samples $\{..., \tilde{\boldsymbol{\theta}}_j^{(l)}, \tilde{\boldsymbol{\theta}}_j^{(l+1)}, \tilde{\boldsymbol{\theta}}_j^{(l+2)}, ...\}$ distributed according to the marginal posterior, $p(\boldsymbol{\omega}, \mathbf{M}, K|\mathbf{y})$.

Algorithm 1 Generic Metropolis-Hastings (MH) Algorithm

for $j = 1 \Rightarrow N$ **do**
  `Simulate y from` $q(x^{(j)}, \cdot)$
  `Simulate u from` $U(0, 1)$
  if u $\leq \alpha(x^{(j)}, y)$ **then**
    `Set` $x^{(j+1)} = y$
  else
    `Set` $x^{(j+1)} = x^{(j)}$
  end if
end for
`Return` $[x^{(1)}, x^{(2)}, \cdots, x^{(N)}]$

An important algorithmic decision that must be made when using this methodology is how to propose new candidate values. The difference between a good choice and a bad choice can be orders of magnitude in compute time or iterations required to form a suitably converged chain. A good choice of proposal distribution must be simple enough that it can be directly sampled from but general enough that it can explore the function space thoroughly in a reasonable number of iterations. It also must be easily extendable to multiple frequencies, as will become more important later. The most common options are Gaussian distributions and uniform distributions.

Gaussian Random Walk (GRW) Methods

We begin with simplest case, a monophonic (single-instrument) recording, using a Gaussian proposal distribution,

$$p(w_{n+1}) = \frac{1}{\sqrt{2\pi\beta^2}} \exp\left(-\frac{1}{2\beta_v^2}(w_{n+1} - w_n)^2 \right). \tag{33}$$

Using the posterior we try to find the root frequency present in our 'toy' recording, an E5 (659.25 Hz) played on the piano.

Owing to its pleasant computational qualities, a Gaussian distribution for the proposal is a natural starting point. It is a symmetrical distribution p(x,y) = p(y,x) so that at each iteration when calculating the acceptance probability a(x,y) the proposal probability terms cancel, and the expression reduces to a ratio of probability terms,

$$\alpha(x, y) = \min\left(\frac{\pi(y)}{\pi(x)}, 1 \right). \tag{34}$$

For single tones this method is effective and after some tuning of the s.d parameter β the chain quickly converges, identifying the correct root frequency within 20 iterations (see Fig.7).

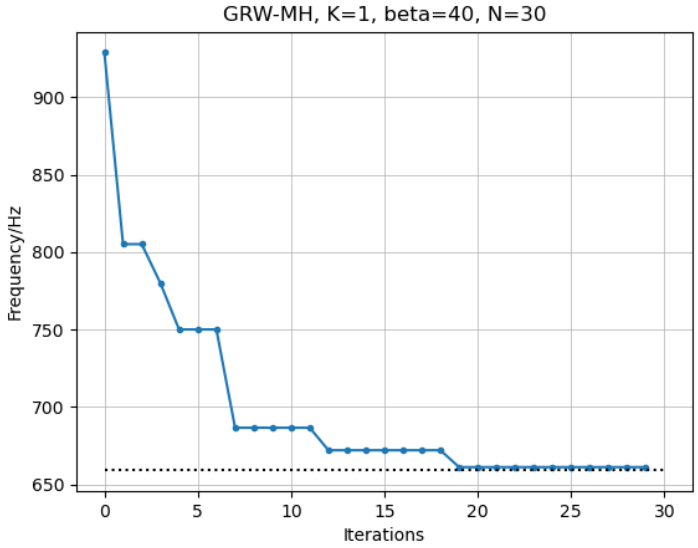
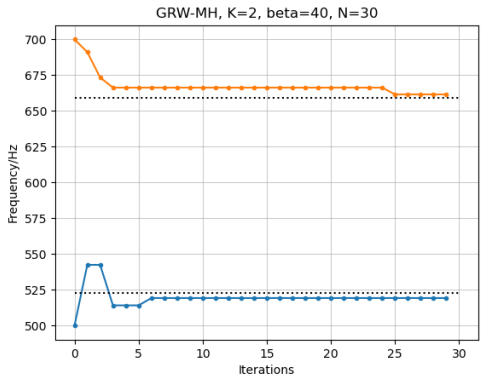
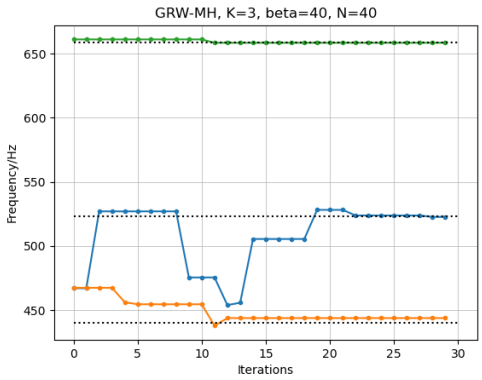


Figure 7: 30 iterations of a GRW-MH algorithm for a single recorded E5 piano note with target frequency indicated by dotted line.

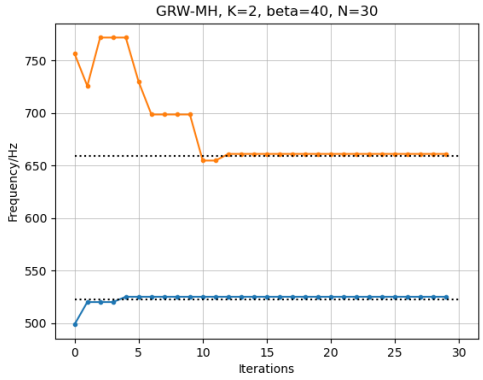
Extending this to multiple notes (K>1) we apply exactly the same methodology. A proposal-acceptance step is taken for each frequency in turn, for each iteration of the algorithm. At every stage, the most recently updated $\boldsymbol{\theta}$ is used. For K=2 the graphs show consistent convergence. For K=3 however, the probability that the initial placement of the frequency estimates causes the algorithm to mischaracterise recording increases. We discuss this in the SA section.



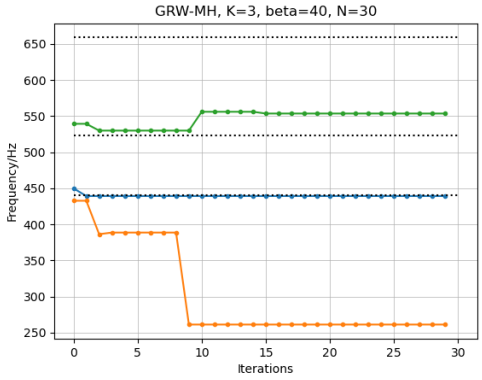
(a) K=2, trial 1.



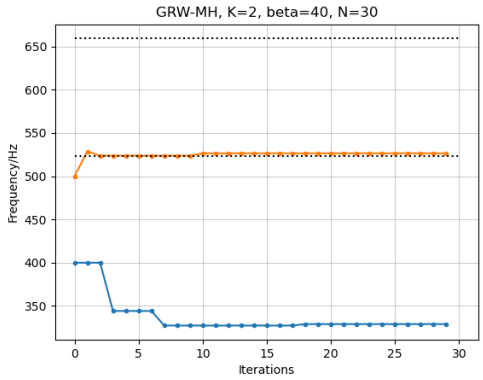
(b) K=3, trial 1.



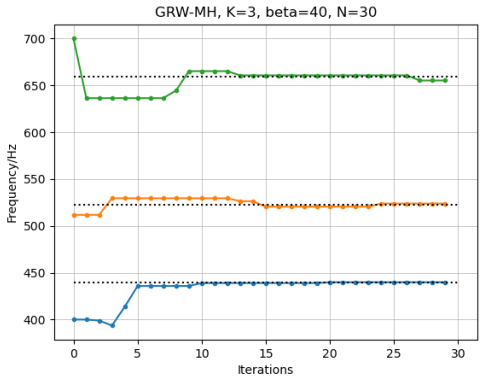
(c) K=2, trial 2.



(d) K=3, trial 2.



(e) K=2, trial 3.



(f) K=3, trial 3.

Figure 8: 3 trials showing the effectiveness of GRW-MH techniques for simple polyphonic recordings.

Simulated Annealing (SA) Methods

One of the drawbacks of GRW-MH is the very fact that it uniformly proposes new values to add to the chain. If the initial proposal frequency is close to the actual frequency present in the sample then a small value of beta is sufficient to explore the function space around the maximum and produce a sequence that converges quickly and precisely. The issue becomes more clear when the initial proposal is not close

to the frequency present in the sample. In this case the small beta Gaussian jumps do not make consistent progress in the relatively flat function space (see Fig. 9) and the number of iterations required to form a suitably converged chain increases in a highly non-linear fashion.

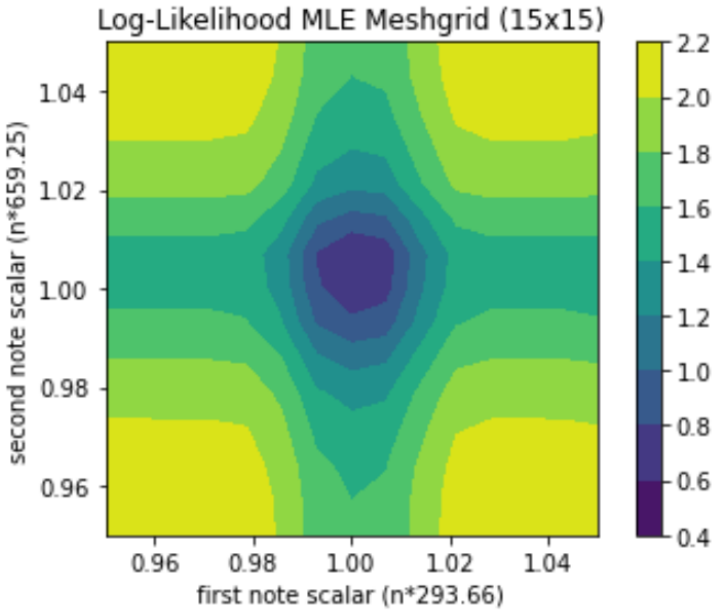


Figure 9: Meshgrid of marginal posterior probability over values of $[w_0, w_1]$ close to true values. Probability changes smoothly 2% either side of global maximum, but flattens out considerably further away.

We experiment with choosing a higher value of beta, so that proposal jumps in function space can move more easily from the flat regions far away from the true frequency towards the region around the true frequency (where MH steps make more consistent progress). This works to the extent that there is a larger range of initial values of frequency that begin chains that converge consistently to the desired value. However, because of the larger beta, once estimate has converged to region around true frequency it is then more unlikely to propose the small change in $\boldsymbol{\theta}$ required to accurately estimate frequency compared to small beta. This is more of an issue in the lower register when the difference in frequency between notes is smaller but is still somewhat an issue in the higher octaves.

One possible solution is to propose GRW-MH samples with a beta value that decreases over the course of samples,

$$p(w_{n+1}) = \frac{1}{\sqrt{2\pi\beta^2}} \exp\left(-\frac{1}{2\beta_v^2}(w_{n+1} - w_n)^2 \right) \exp\left(\frac{-2i}{N} \right). \tag{35}$$

This approach, named simulated annealing (due to the similarities with cooling metals) begins by taking large jumps in function space, able to move from flat regions far from the true frequency. As the Gaussian proposals 'cool' the scale of the proposal jumps from previous values decreases and the algorithm is able to get more fine tuned accuracy when compared with using a large static value of beta.

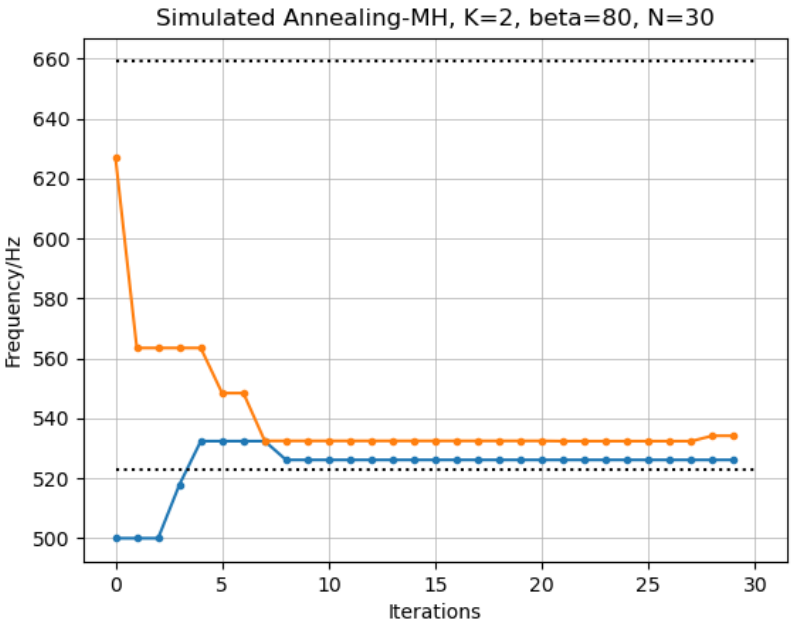


Figure 10: An example of how the simulated annealing algorithm can converge incorrectly.

The primary issue that motivates the more consistently initiated FFT-SA algorithm described later is highlighted in Fig.10. Using semi-random proposal frequencies, the first few steps of proposal-acceptance have a chance of both converging on the same note. Once the samples from chains have fallen into the region around one of the notes present in the recording, the higher value of the marginal posterior lead to large acceptance ratios and both chains converge to the same target frequency. This can be thought of as a similar paradigm to maximisation algorithms converging to local maxima rather than the global maximum. While there are techniques to mitigate this issue, mostly via proposal distributions that are able to escape unlucky initiation, as with function maximisation algorithms there is no one-hit solution that works consistently.

This is especially problematic with SA methods which thus far have performed on par with or better than GRW methods. By reducing the probability of large-variance proposals through use of a cooling-coefficient, $\exp(\frac{-2i}{N})$, we limit the chance for the chain to escape the marginal posterior local maxima that it initially locates.

Later we propose a method that calculates the residual after each of the k frequencies is estimated, subtracting the corresponding estimated note model from the signal before proceeding with the next estimate. The advantage of this approach is that it is then much less likely for multiple chains to locate the same root frequency.

Alternating Uniform Gaussian (AUG) Methods

Another approach experimented with is to alternate between a uniform guess (piano notes C0 to C6) and GRW-MH techniques discussed already. This is another way of dealing with the ineffectiveness of the GRW-MH algorithm when proposals are initiated too far from frequencies present and marginal posterior function space is

too flat for consistent progress, or from deterring more than one chain converging on the same root frequency. After brief experimentation this method was found to be too inconsistent as there are too many possible notes to choose from. The resulting number of uniform proposals required before choosing correctly is too large for it to be a viable choice, a problem which grows exponentially when K>1.

Fast Fourier Transform Seeded GRW-MH

A new method is proposed to consistently find root frequencies. We note that all previous methods are highly dependant on the specific signal being analysed – a note near the bottom of the piano is very unlikely to be converged to by all but the most pleasant initial proposal regardless of the GRW variation. The FFT is a very computationally-light method for finding the coarse power spectrum of a signal. Combining the coarse estimate found using an FFT with the more computationally heavy but precise simulated annealing MH algorithm, the FFT-seeded simulated annealing algorithm is presented,

Algorithm 2 FFT-SA Algorithm

for $k = 1, ..., K$ **do**
- Calculate $\mathcal{F}(\mathbf{x}) = [X_0, X_1, ..., X_t]$, FFT of samples
- Find frequency w_0 corresponding to maximum value in $[X_0, X_1, ..., X_t]$

for $i = 1, ..., N$ **do**
- Simulate y from $q(x^{(j)}, \cdot)$
- Simulate u from $U(0, 1)$

if u $\leq \alpha(x^{(j)}, y)$ **then**
`Set` $x^{(j+1)} = y$
else
`Set` $x^{(j+1)} = x^{(j)}$
end if
`Return` $[x_k^{(1)}, x_k^{(2)}, \cdots, x_k^{(N)}]$
`Set` $w_0 = x_k^{(N)}$
end for
- Using w_0 calculate MAP estimate of weights using $\boldsymbol{\mu} = \mathbf{SD}^t\mathbf{y}$
- Calculate signal residual **x***
- `Set` $\mathbf{x} = \mathbf{x}^*$

end for
`Return` $[x_0^{(N)}, ..., x_K^{(N)}]$

After taking a FFT to locate a suitable initial frequency estimate, we use a simulated annealing GRW-MH algorithm to converge to the target density. Using this MCMC-derived estimate of the most prominent root frequency we calculate the MAP estimate of the weights, $\boldsymbol{\beta}_k$, for the partials associated with this root using eq.29. The next step is to subtract this modelled note from the signal, leaving a residual present. This process is repeated for K notes, the final residual of which is the error between the true signal and the sum of our resynthisized modelled notes.

We apply this algorithm to a recording of a two-note chord played on the piano.

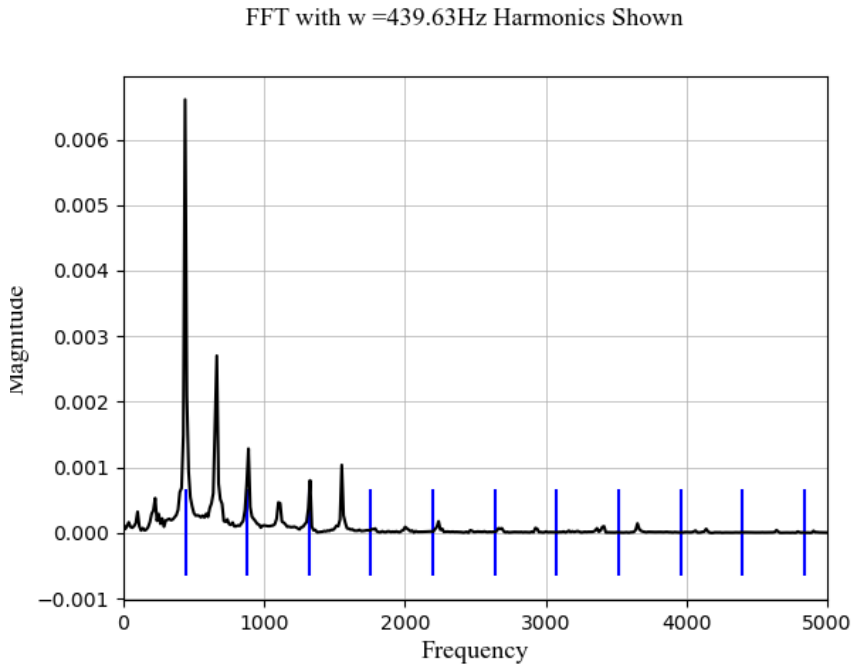


Figure 11: Graph showing the first note and its harmonics overlaying the original signal.

In the first round of FFT-SW, $k = 1$, the most significant frequency is correctly identified as 439.63Hz. The modelled harmonics of this root frequency are shown in blue in Fig.11.

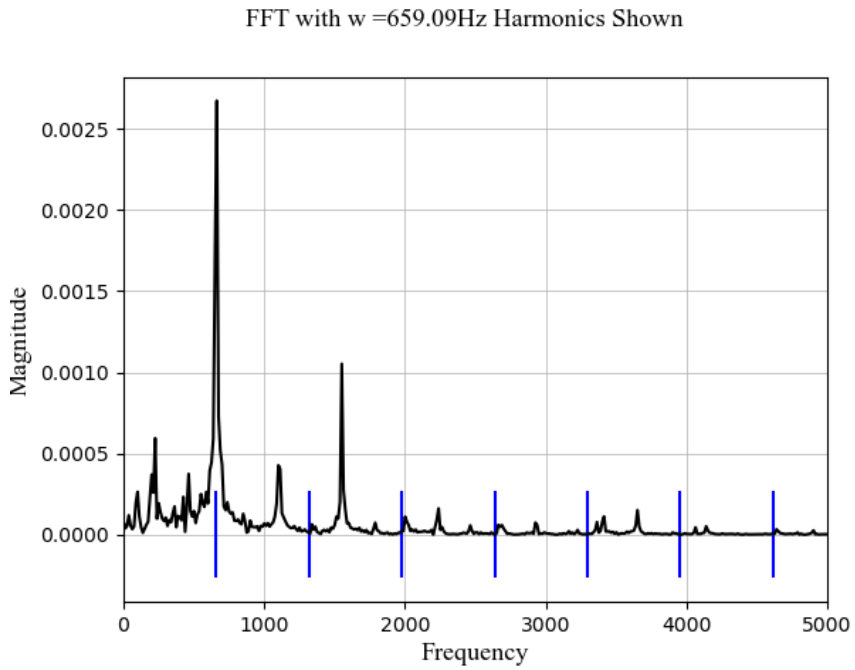


Figure 12: Graph showing the second note and its harmonics overlaying the residuals from the first estimated note.

In the second round of FFT-SW, $k = 2$, having computed the residual signal using the estimate from the first round, we repeat the steps of identifying the most

prominent frequency (via combination of FFT and SA), calculating the MAP estimate of weights associated with the partials, and subtracting it from signal to calculate the residual error after the $K = 2$ modelled notes.

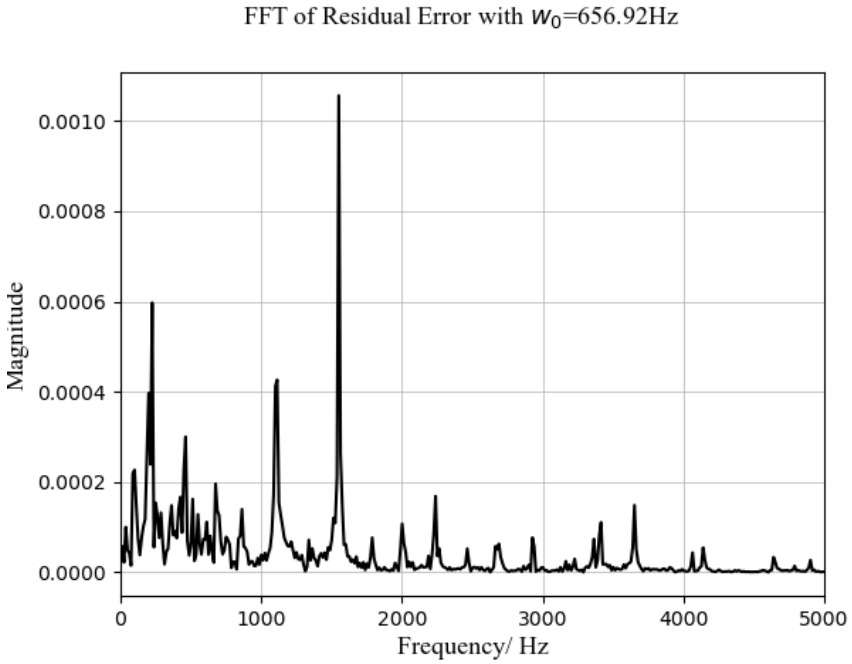


Figure 13: Residual Error after two steps of FFT-SA have been run.

Examining the final residual we note that to a large extent the energy of the signal has been captured by the two modelled notes. Taking the sum of each sample's amplitude squared as a metric for energy, $\sum_T x_t^2$, we present in Table.2 the cumulative energy captured by the FFT-SA algorithm for the first 4 samples from Fig.17 . Using this relatively simple method we are able to capture on average over 90% of the energy present in the signal.

End of Round 1	+ End of Round 2	+ Residual Error
69.0%	90.1%	100%
52.3%	93.6%	100%
80.0%	96.7%	100%
83.4%	89.4%	100%

Table 2: Accumulated signal energy capture of FFT-SA algorithm for polyphonic (K=2) recorded chord.

Efficient Marginal Probability Computations

The total time taken to analyse short audio clips using the methods described above is highly dependent on the speed of each iteration of the MCMC algorithm used. For example, a typical target root frequency estimation (as in Fig.8) might take 30 or more proposal-acceptance steps to suitably converge. If we want to analyse a 10s audio clip, resolved using 0.08s bins, it would require in total $125 \times 30 = 3750$ iterations of MCMC. If we are to improve the efficiency of the whole algorithm we should start with the proposal-acceptance steps.[17]

For each acceptance step, the marginal posterior (eq .27) of the proposed and the current estimates of $\boldsymbol{\theta}$ are compared to calculate the acceptance probability ratio. Given the simple form of the simulated annealing proposal (eq .35) it is in calculating this ratio that all the complexity lies. We note that efficient implementations only require one computation of the target distribution for each MH step, as the value of the most recent sample is stored and reused.

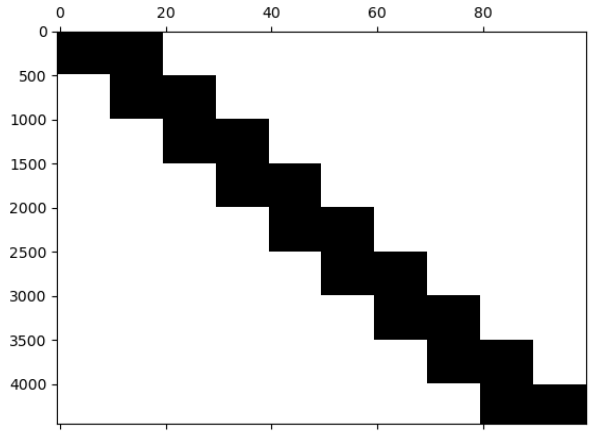


Figure 14: Spyplot showing the values in $\mathbf{D}$ that can have non-zero values. Owing to the enveloping Hanning functions it is only the values along the diagonal that are relevant. Elsewhere the Hanning functions ensure a value of 0.

One step in calculating the marginal posterior is computing $\mathbf{D}$, the matrix of enveloped basis functions used to model notes in a time-localised way. As is shown in Fig.14, $\mathbf{D}$ has a very specific structure. Because all non-zero blocks in $\mathbf{D}$ are located along the diagonal the efficiency of calculating it can immediately be improved by excluding calculations of the off-diagonal elements, harmonics that will ultimately be multiplied by a zero-value enveloping function.

We introduce a subscript notation to manipulate blocks of amplitudes/parameters in a more concise way. The component of the signal $\mathbf{y}$ relating to note k can then be written as $\mathbf{y}_k = \mathbf{D}(\boldsymbol{\theta}_k)\boldsymbol{\beta}_k$, where $\mathbf{D}(\boldsymbol{\theta}_k)$ is the $N \times R_k$ sub-matrix of $\mathbf{D}$.

Next, for the calculation of $\mathbf{P} = \mathbf{I} - \mathbf{DSD}^T$ we note that the matrix multiplication of each block is in fact independent of all the other blocks. Again, we can improve the efficiency by calculating the sub-matrix multiplications and then combining them after the fact,

$$\mathbf{P}(\boldsymbol{\theta}_k) = \mathbf{I}_N - \mathbf{D}(\boldsymbol{\theta}_k)\mathbf{S}(\boldsymbol{\theta}_k)\mathbf{D}(\boldsymbol{\theta}_k)^T \tag{36}$$

Next, we follow the implementation described by [17] for the efficient calculation of $\mathbf{y}^T\mathbf{P}\mathbf{y}$. This uses a Cholesky decomposition to form a lower triangular matrix that can then be used to solve a triangular system of the form Cv = u for v. The complexity of this is calculated as $O(R^3 + NR^2)$. While the actual complexity of calculating D^TD may well be less than this (see eq.36), the order of calculating $\mathbf{y}^T\mathbf{P}\mathbf{y}$ is still $O(R^3)$.

Considering the harmonic model presented in **II**, the size of each block/note is $R_k = 2M(I+1)$. Typical values of $I = 15$, $M = 15$ and therefore $R = 480$. For $K = 3$ notes, $R = 1440$. Calculating each of the sub-blocks individually as opposed to calculating for $\mathbf{D}$ in one go grants significant improvements in efficiency. the overall complexity is $\mathcal{O}((\sum_{k=1}^{K} M_k)[2(I+1)]^3)$ instead of $\mathcal{O}(([\sum_{k=1}^{K} M_k)2(I+1)]^3)$, in theory a gain of around 2000.

GPU Acceleration

The computational power available for signal processing has significantly increased since the original proposal of Bayesian harmonic models. However, while number of cores and clock speed have improved, for vector manipulations a larger improvement still can be gained with GPU acceleration.

Briefly, this is because of the fundamental differences between GPUs and CPUs. CPUs excel at managing a diverse set of inputs and outputs, not optimized for either a small number of very computationally complex tasks or a large number of computationally simple tasks. The architecture of a GPU is different. Made up from much larger numbers of much less powerful cores, they excel at parallel processing. As such, they are exceptionally well-optimized for performing a small set of relatively simple computations over an extensive collection of data.[18]

Nvidia have developed an easily implemented API to access GPU acceleration for `numpy` functions, an example of which (code segment for calculating $\mathbf{P}$, the N-dimensional square matrix used in computing marginal posterior) can be found in the appendix.[19] We use this, and similar functions to replace all matrix multiplications performed by proposal-acceptance steps.

For a 0.1s (4410 samples) length audio signal, the average time (over 30 samples) taken for each stage of the calculation of the marginal posterior before algorithmic efficiency and GPU acceleration and after is as follows,

Sub-Process	Initial Duration/(s)	Sped Up Duration/(s)
D Matrix	1.375	0.123
S Matrix	0.684	0.320
P Matrix	1.281	0.187
Marginal Posterior	0.141	0.060
Total	3.481	0.690

Table 3

IV Automatic Music Transcription (AMT)

Using the FFT-SA method of identifying key frequencies present in short samples we investigate frame-based techniques to estimate how frequencies present change over time. A plot of this can be thought of as a 'root frequency spectrogram' or an 'F0 spectrogram'. This is more often called a 'piano roll', owing to it's similarity to the perforated paper rolls used in self-playing pianolas and is the first step towards full musical-score transcription.

AMT can be divided into several subtasks for our consideration that include multi-pitch (or multiple-F0) estimation, note onset/offset detection, loudness estimation, and tempo and beat estimation. [20]

Even with a relatively precise estimate of notes over time, full transcription to the level of a skilled musician is still a considerable challenge, requiring high-level heuristic rules. This is immediately clear upon consideration of how varied a single note can be. In terms of note pitch and length, a violin playing softly with vibrato might well be abstracted similarly to a trumpet playing a fanfare, though the signals from each would look wildly different. On top of the problems outlined already, to produce an output in the form of sheet music, additional issues would need to be addressed, such as typesetting, fingering, articulation as well as estimation of dynamics.[21][22][23]

This level of detailed heuristic information is not available from the MCMC pitch-estimation algorithm presented and full signal-to-score transcription is, for this report, out of scope (see [24]). Instead, we showcase what can be achieved using simple modifications to the algorithm discussed already, outlining a method that takes in as its input a monophonic recording and outputs a MIDI representation of the score, that can be used for further analysis and modified using standard music software packages.

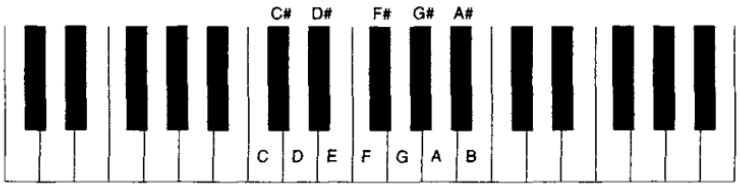


Figure 15: Piano notes with octave labelled. Failing true signal-to-score AMT, this method aims to estimate which notes are being played and when, similar to an original pianola-style piano roll.

The basic methodology is as follows. i) Take a recording of length T and cut into short frames for analysis. ii) Applying the FFT-SA algorithm, find an estimate for the root frequencies present. iii) Combining frequency estimate information from frames, form a root frequency/time representation of the recording. iv) Using a simple log-minimum distance heuristic estimate the note being played (see fig15). v) Splice these estimates into larger-scale notes and convert to MIDI. vi) Use standard music software packages for automatic transcription/ further analysis. This approach is summarised in Algorithm 5.

To start, we append three piano notes (E5, C5, A4) together into a continuous recording and implement steps and, as a proof of concept, plot the root fre-

quency/time graph shown in Fig.16. For note estimation we follow the FFT-SA algorithm with K=1, M=15, beta=5 and N=30; that is, having identified the coarse frequency estimate using the STFT we then use 30 iterations of MCMC to find a more accurate estimate. Each point in Fig.16 represents the estimate for one frame. As expected when K=1, the FFT-SA pitch estimation algorithm is very effective.

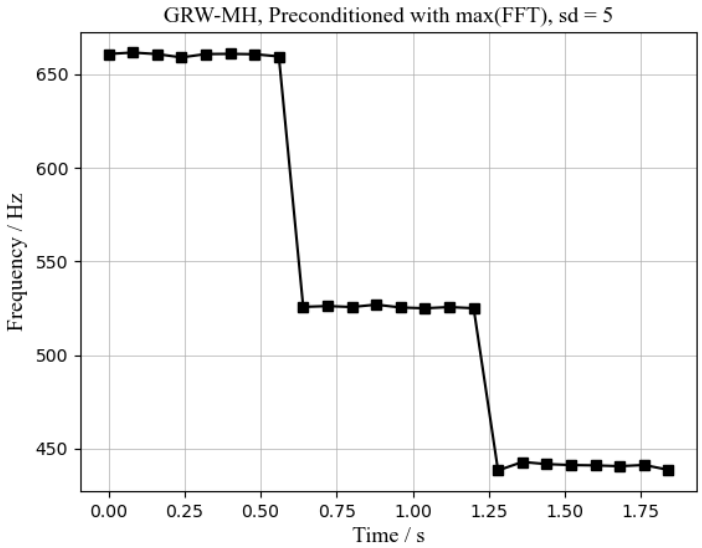


Figure 16: FFT-SA algorithm applied to recorded monophonic descending notes using 0.08s bins.

Next the analysis is applied to a polyphonic (K=2) test case, Fig.17. The process is the same as in Fig.16 with K=2. However, with more than one frequency estimate at each frame, it's no longer trivial to tie point estimates together into notes. For calculating note onset/note offset timings, which is needed for MIDI conversion later, we require knowledge of note structures present in the recording. To implement this, at each frame we assess if the previous frame contains any of the same notes located in the current frame. If so, the already existing note is incremented in length by the frame length, t=0.08s. If not, we instead create a new note object with the new frequency of length t.

Algorithm 3 Signal-to-MIDI AMT

1: Map recording to $t = 0.08s$ frames.
2: Apply FFT-SA Algorithm to each frame independently to locate vector of root frequencies, **F0**.
3: Using closest-note heuristic, map **F0** to note estimate.
4: Predict note onset/offset by splicing together point estimates.
5: Using MIDI api, convert piano roll estimate into MIDI estimate.

Now with some confidence that this method is effective at converting studio-clean piano recordings to MIDI, we apply it to a more realistic scenario. A recording of the famous opening bars of Debussy's Syrinx from YouTube is converted into .mp3 for analysis.[25] Fig.18 shows the raw sample estimates along with the note predictions.

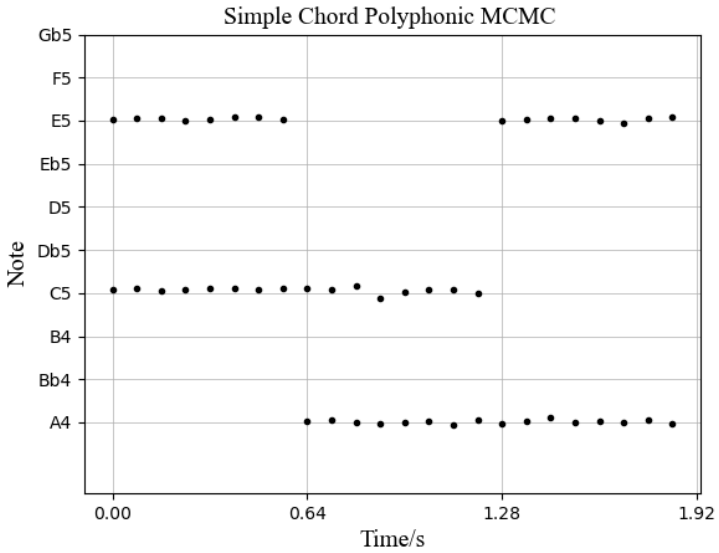


Figure 17: FFT-SA applied to polyphonic (K=2) recording.

One aspect that hasn't been discussed is how FFT-SA deals with musical rests. Almost all music has periods of silence that need to be gauged and modelled if we are to attempt transcription. An example of this is at the beginning of the recording in Fig.18 before the performance has begun, though the explanation will be the same for intentional rests throughout the recording. In the absence of notes, the only frequencies present are from low frequency noise. The FFT-SA algorithm finds that the frequency that minimizes the residual is almost zero, such that the 'harmonics' account for as much of the noise as possible. The result is that the estimated frequency is consistently less than 10Hz, equivalent to an octave below the lowest note on a full-size piano and can be safely understood by any labelling algorithm as a rest.

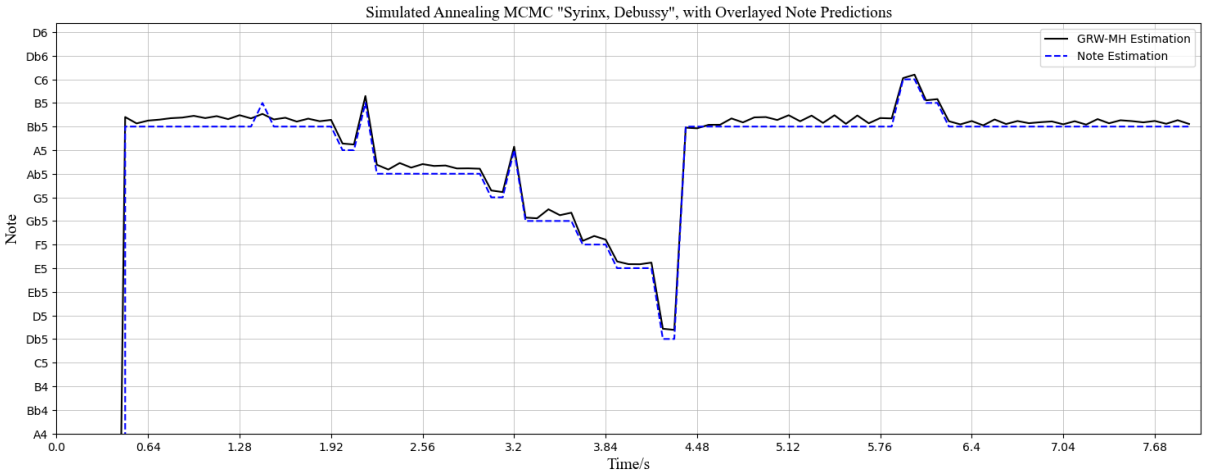


Figure 18: FFT-SA applied to the opening bars of Debussy's Syrinx, L. 129 played on flute with note estimations overlayed.

The full AMT process is outlined in Fig.20, from the original score, to a recorded signal of a real performance (along with all the heuristic decisions taken by the performer), to the MIDI abstraction isolated from the recording using the FFT-SA method, to a fully typeset score representation of the MIDI file generated using free music software (MuseScore 4). The only additional information used to produce the score is the original marked tempo, the key signature, and the time signature, none of which influence the note pitches or lengths.

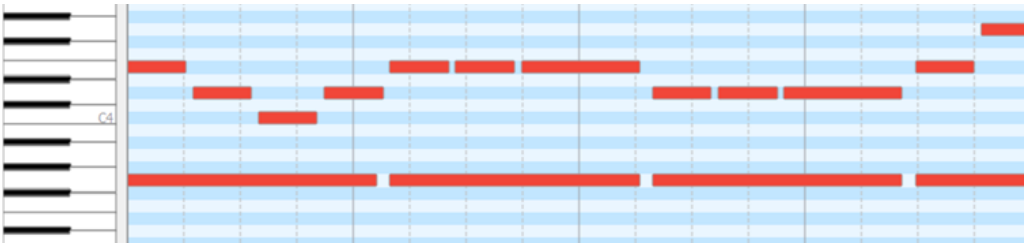


Figure 19: Extract from MIDI transcription of 'Mary had a little lamb'.

Discussion

The presented method achieves a high quality of MIDI transcription (shown in Fig.20(iii)). Note onsets are accurately represented and to a high degree, the pitch is correct. For K=2, we note that in none of the frames do both pitch estimates converge to the same note, highlighting the advantages of the residual-recompute approach compared with typical MCMC multi-pitch estimators.

When resynthesized using typical MIDI instruments in MuseScore 4, not only are the synthesized recordings similar to the original recordings that they're extracted from, they are often almost indistinguishable. From this heuristic measure of success, the approach outlined in this work has successfully transcribed a recorded signal into an abstracted written form that can be reproduced by another musician to sound highly like the original.

We note some of the limitations of this implementation. To start, the flexibility with regards to K is limited. For example, in orchestral music, it is common that instruments have many bars of rests in a row, maybe playing less than half the time. With a static number of notes K, it is unclear how this approach would model instruments coming in and out so frequently. In future work we might implement a dynamic K such that the model might be able to detect the number of notes present in a particular frame automatically.

Secondly, as it stands there is no consideration of instrument classification in our method as presented. For automated polyphonic MIDI transcription, performance would be enhanced if the algorithm was able to detect which instrument each note belonged too based on timbre alone and separate lines for transcription. Since, for each k in K we calculate the MAP estimates of the harmonic amplitudes (for residual calculation), in theory we could further process our collection of $\boldsymbol{\beta}_k$ to extract timbre and alongside knowledge of larger structures (notes, phrases etc) estimate instrument lines. This would approach the parallel task of source separation, and also aid any signal-to-score transcription attempts.

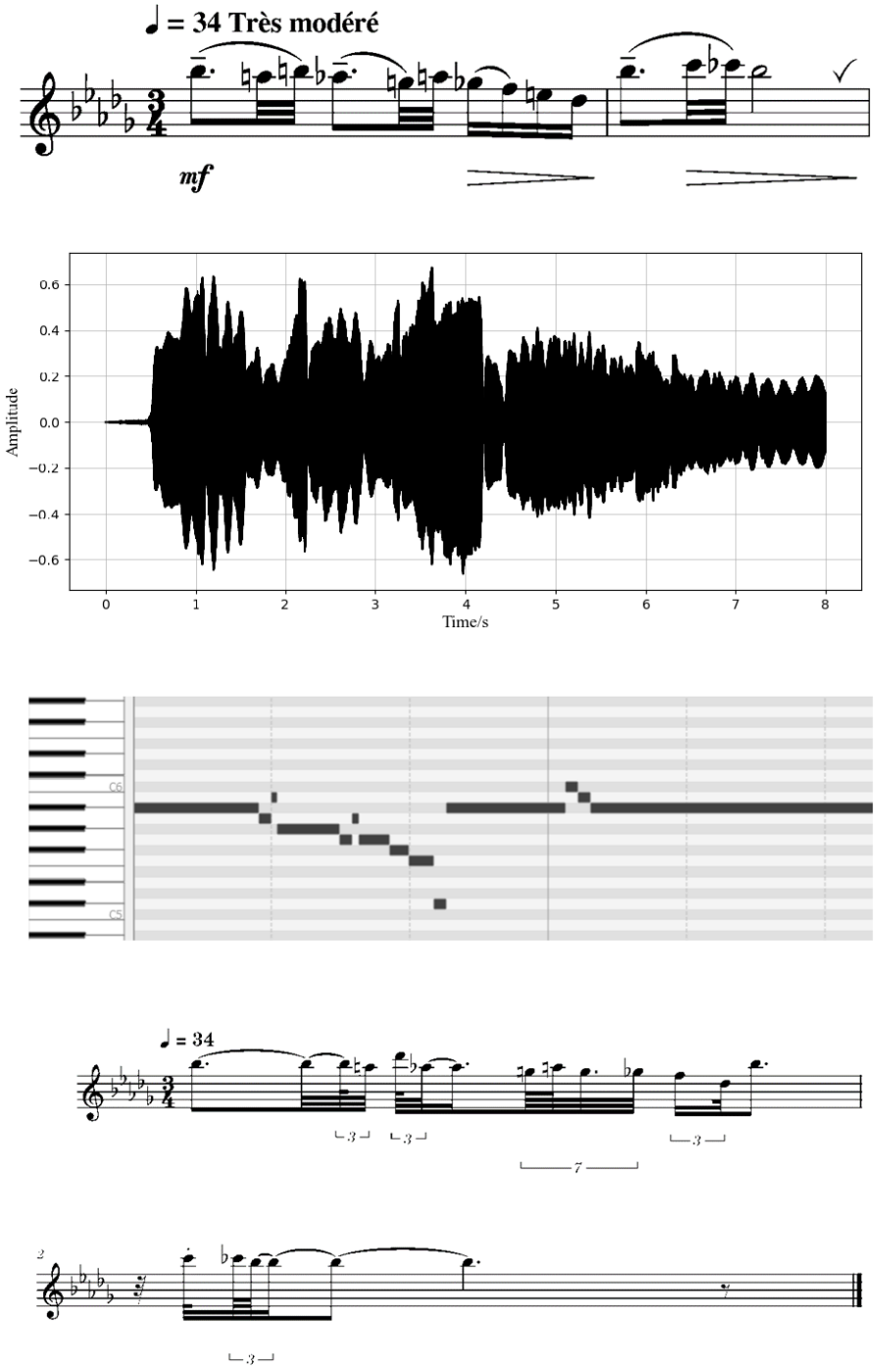


Figure 20: The functional stages of our automatic musical transcription algorithm.(i) We start with the ground-truth score (first two bars of Debussy's Syrinx analysed throughout) which is then played and recorded (ii). Using the signal-to-MIDI methodology described, we produce a MIDI representation of the signal (iii). This is then processed into a full score automatically using MuseScore (iv).

As a final step, in the functional stages diagram (Fig. 20) we use a free MIDI-to-score conversion tool (MuseScore 4) to convert our successfully transcribed MIDI score into more performable sheet music. Without any of the semantic information about the recording, this final score does not clearly reflect the intent of the original score and a musician would be hard-pressed to recreate a performance that sounds like that originally used for the analysis.

This highlights how much more complex transcription into accurate sheet music is compared to transcription into MIDI. Rather than a literal mapping of pitches and times, signal-to-score transcription can more be thought of as a method of recording intent. Stylistic choices, musical context, as well as range of highly abstracted features inform human transcription, none of which we provide through a simple MIDI file. Similarly complex acoustic and musicological models must be understood by a music transcription system if it is to accurately record the intent of a performance. [20]

Even if methods developed do not entirely automate the process of musical transcription, we note that there is still considerable value in speeding up the process. Semi-automatic approaches ask how a human user could assist with some aspects of AMT for more satisfactory results. Certain sub-tasks such as precise note onset detection, instrument identification and heuristic typesetting quantisation are all necessary for accurate transcription but have proven difficult to estimate algorithmically. If we could combine manual estimates of these with accurate computer-aided note estimation, as well as an automated coarse estimate of note onset/offset we could produce more accurate transcription results than full AMT achieved in a much shorter time than fully manual transcription.

A promising avenue for future work is the use of deep neural networks (DNNs) to convert from MIDI to score. This class of machine learning algorithms use perceptrons to learn statistical patterns, and are well-suited to the type of abstract, high-level statistical classification that classical deterministic algorithmic approaches struggle to contend with. While successful implementations of convolutional neural networks (CNNs) to solve individual subtasks in AMT exist, to the best of our knowledge approaches combining Bayesian harmonic modelling with deep neural networks for signal-to-score AMT have not yet been examined.[4][26]

V Conclusion

In this work, we presented a method for multiple-pitch estimation and note tracking using Bayesian harmonic models which we demonstrate by converting a real-world monophonic flute recording from an mp3 file into an accurate MIDI representation that can be used either for further analysis or as the starting point for semi-automatic/assisted music transcription. We proposed a novel algorithm for multiple-pitch estimation and showed that for $K \leq 3$ notes it performs well, and when combined with GPU-acceleration and algorithmic improvements, can be made to run quickly on a laptop. Furthermore, we proposed two possible extensions to this body of work that would lead to improved results, developing a DNN approach to infer the statistical conversion from MIDI to score, and designing more purposeful semi-automated workflows.

References

[1] Manuel Davy, Simon Godsill, and Jérôme Idier. "Bayesian analysis of polyphonic western tonal music". In: *The Journal of the Acoustical Society of America* 119.4 (Apr. 2006), pp. 2498–2517. ISSN: 0001-4966. DOI: 10.1121/1.2168548. eprint: https://pubs.aip.org/asa/jasa/article-pdf/119/4/2498/14840741/2498_1_online.pdf. URL: https://doi.org/10.1121/1.2168548.

[2] H. Helmholtz. "Vortage und Reden". In: vol. 1. Brauschwig, 1884, p. 82.

[3] Kilian Schulze-Forster et al. "Unsupervised Audio Source Separation Using Differentiable Parametric Source Models". In: (Jan. 2022).

[4] Helena Cuesta, Brian McFee, and Emilia Gómez. "Multiple F0 Estimation in Vocal Ensembles using Convolutional Neural Networks". In: (Sept. 2020).

[5] Jonathan Sleep. *Automatic Music Transcription with Convolutional Neural Networks using Intuitive Filter Shapes*. Sept. 2017.

[6] M. Davy and S. J. Godsill. "Bayesian harmonic models for musical signal analysis". In: *Seventh Valencia International Meeting, Bayesian Statistics 7* (2002).

[7] S. J. Godsill and M. Davy. "Bayesian harmonic models for musical pitch estimation and analysis". In: *International Conference on Acoustics, Speech, and Signal Processing* (2002).

[8] R. Schumacher M. McIntyre and J. Woodhouse. "On the oscillations of musical instruments". In: *Journal of Acoustics* 74 (1983), pp. 1325–1345.

[9] Jan Wild. "The computation behind consonance and dissonance". In: *Interdisciplinary Science Reviews* 27.4 (2002), pp. 299–302.

[10] Hermann L. F. Helmholtz. *On the Sensations of Tone as a Physiological Basis for the Theory of Music*. 3rd edition. Cambridge University Press, 1875.

[11] A. Klapuri M. Muller D. P. W. Ellis. "Signal Processing for Music Analysis". In: *IEEE Journal of Selected Topics in Signal Processing* 5.6 (2011), pp. 1088–1110.

[12] Hans Feichtinger and Franz Luef. "Gabor Analysis and Algorithms". In: Nov. 2015, pp. 575–579. ISBN: 978-3-540-70528-4. DOI: 10.1007/978-3-540-70529-1_354.

[13] S. J. Godsill P. Walmsley and P. J. Rayner. "Multidimensional optimisation of Harmonic Signals". In: *European Signal Processing Conference, EURASIP* (1998).

[14] S. J. Godsill P. Walmsley and P. J. Rayner. "Polyphonic pitch tracking using joint Bayesian estimation of multiple frame parameters". In: *Workshop on Applications of Signal Processing to Audio and Acoustics* (1999).

[15] P. Peeling A. Taylan Cemgil Simon J. Godsill and N. Whiteley. "Bayesian Statistical Methods for Audio and Music Processing". In: *Oxford Handbook of Bayesian Techniques* (2008), pp. 1–45.

[16] C. Andrieu and A. Doucet. "Joint BAyesian detection and estimation of noisy sinusoids via reversible jump MCMC". In: *IEEE Trans. Signal Process* 47.10 (1999), pp. 2667–2676.

[17] M. Davy and J. Idier. "Fast MCMC computations for the estimation of sparse processes from noisy observations". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2. 2004, pp. ii–1041. DOI: 10.1109/ICASSP.2004.1326439.

[18] *GPU Acceleration for High-Performance Computing*. May 2023. URL: https://www.weka.io/learn/ai-ml/gpu-acceleration/.

[19] *Numba for CUDA GPUs*. May 2023. URL: https://numba.readthedocs.io/en/stable/cuda/index.html.

[20] Emmanouil Benetos et al. "Automatic music transcription: Challenges and future directions". In: *Journal of Intelligent Information Systems* 41 (Dec. 2013). DOI: 10.1007/s10844-013-0258-3.

[21] A. Sterian. "Music transcription systems: From sound to symbol". In: *Proc. of AAAI-2000 Workshop on AI and Music* (2000).

[22] K. Kashino. "Organization of hierarchical perceptual sounds: Music scene analysis with autonomous processing modules and a quantitative information integration mechanism". In: *Proc. of IJCAI* (1995), pp. 158–164.

[23] M. Goto and Masataka Goto. "A real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals". In: *Speech communication* 43.4 (2004), pp. 311–329. ISSN: 0167-6393.

[24] Anssi Klapuri and Manuel Davy. *Signal Processing Methods for Music Transcription*. Jan. 2006. ISBN: 978-0-387-30667-4. DOI: 10.1007/0-387-32845-9.

[25] *Debussy: Syrinx for solo flute (Emmanuel Pahud)*. May 2018. URL: https://www.youtube.com/watch?v=RNjroFNi7mA.

[26] Alexandre Défossez. *Hybrid Spectrogram and Waveform Source Separation*. 2022. arXiv: 2111.03600 [eess.AS].

Appendix

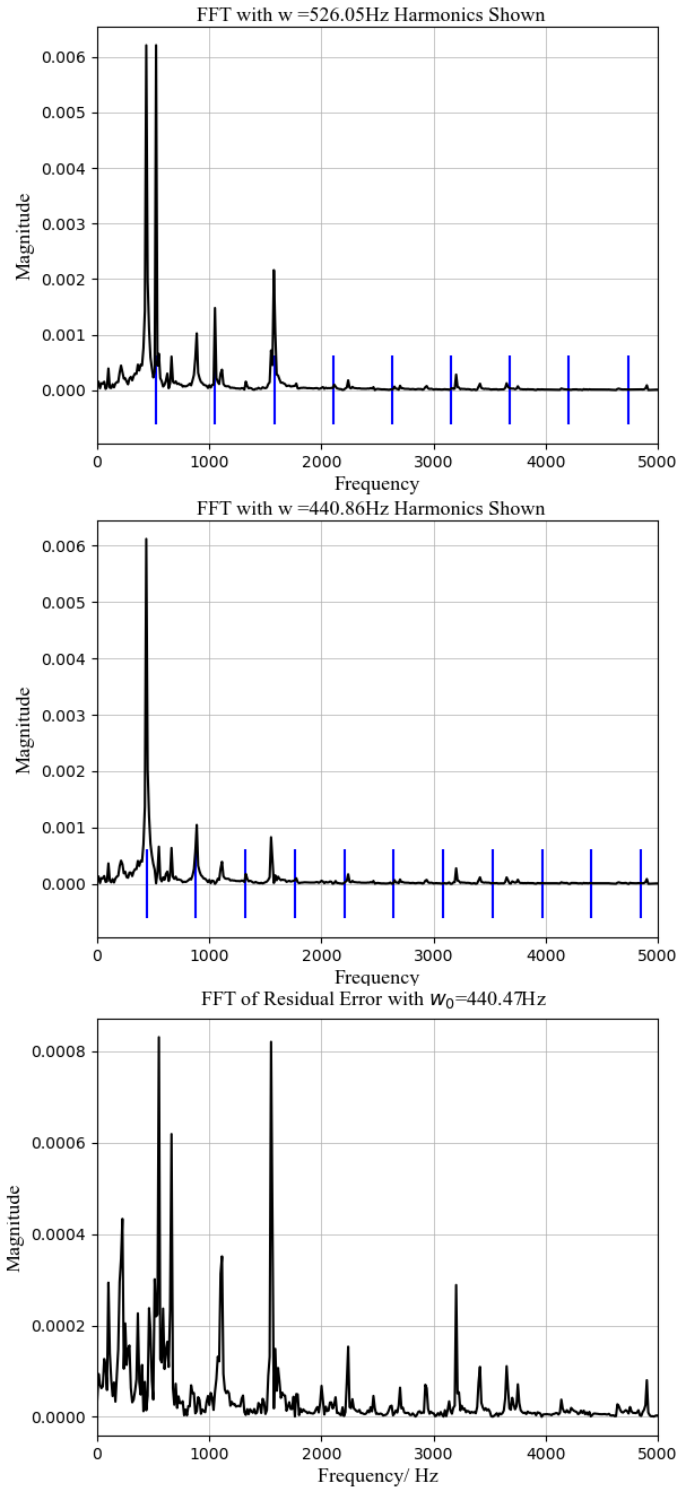


Figure 21: The functional stages of our automatic musical transcription algorithm.(i) We start with the ground-truth score (first two bars of Debussy's

```
import numpy as np
```

```
    from numba import vectorize

    N = 4410

    def GenerateP(D,S):
        # Standard Python matrix multplication
        return np.eye(self.N) - self.D @ self.S @ self.D.T

    # Initialize GPU ufunc container
    @vectorize(["float32(float32,float32)"], target = 'cuda')

    def GeneratePwithGPU(D,S):
        # GPU-accelerated Python matrix multplication
        return np.eye(self.N) - self.D @ self.S @ self.D.T
```

What follows is an example extract from a MIDI representation of "Mary had a little lamb" :

```
    Track 0:
    MetaMessage('time_signature', numerator=4, denominator=4,
    clocks_per_click=24, notated_32nd_notes_per_beat=8,
    time=0)
    MetaMessage('key_signature', key='C', time=0)
    MetaMessage('end_of_track', time=1)
    Track 1:
    control_change channel=0 control=91 value=58 time=0
    control_change channel=0 control=10 value=69 time=0
    control_change channel=0 control=0 value=0 time=0
    control_change channel=0 control=32 value=0 time=0
    program_change channel=0 program=24 time=0
    note_on channel=0 note=64 velocity=72 time=0
    note_on channel=0 note=55 velocity=70 time=0
    note_on channel=0 note=64 velocity=0 time=231
    note_on channel=0 note=62 velocity=72 time=25
    note_on channel=0 note=62 velocity=0 time=231
    ...
    note_on channel=0 note=64 velocity=0 time=231
    note_on channel=0 note=62 velocity=70 time=25
    note_on channel=0 note=55 velocity=0 time=206
    note_on channel=0 note=62 velocity=0 time=25
    note_on channel=0 note=60 velocity=73 time=25
    note_on channel=0 note=52 velocity=72 time=0
    note_on channel=0 note=60 velocity=0 time=974
    note_on channel=0 note=52 velocity=0 time=0
    MetaMessage('end_of_track', time=1)
```

Risk Assessment

This project was conducted entirely in office settings and no physical experiments were done. As such, risks were minimal and required no further mitigation.