Efficient implementation of spatially-varying 3D ultrasound deconvolution

Henry Gomersall, David Hodgson, Richard Prager,
Nick Kingsbury, Graham Treece, Andrew Gee

***Abstract*—There are sometimes occasions when ultrasound beamforming is performed with only a subset of the total data that will eventually be available. The most obvious example is a mechanically-swept (wobbler) probe in which the three-dimensional data block is formed from a set of individual B-scans. In these circumstances, non-blind deconvolution can be used to improve the resolution of the data. Unfortunately, most of these situations involve large blocks of three-dimensional data. Furthermore, the ultrasound blur function varies spatially with distance from the transducer. These two facts make the deconvolution process time-consuming to implement. This paper is about ways of addressing this problem and producing spatially-varying deconvolution of large blocks of three-dimensional data in a matter of seconds. We present two approaches, one based on hardware and the other based on software. We compare the time they each take to achieve similar results and discuss the computational resources and form of blur model that they each require.**

I. Introduction

ULTRASONIC imaging can be modelled as a linear process in which a spatially-varying blur is convolved with a scatterer field that is a property of the material being scanned. Sometimes, we can estimate the blur function with sufficient accuracy that non-blind deconvolution provides a potential means of enhancing the resolution of the data. However, such deconvolution strategies are not always used in practice because they are perceived to be computationally expensive.

This paper explores ways of addressing this problem by using hardware and software techniques to implement rapid deconvolution with a known spatially-varying blur. The spatial variation in the blur means that the deblurring operation cannot be performed entirely in the spatial frequency domain. We explain how this increases the complexity of the required computation and discuss some practical ways of addressing it. We are not presenting a fundamentally new deconvolution algorithm and we do not address the important issue of how accurately it is possible to estimate the spatially-variant blur. This paper is about strategies for efficient implementation of existing techniques.

Section II gives an overview of the linear model of ultrasound imaging that we use and described how Wiener deconvolution can be used to remove a blur that has been applied during the imaging process. Section III introduces the *in vitro* data set that will be used for testing in the experiments. Section IV describes how the conjugate gradients algorithm can be implemented on a graphics processing unit (GPU) to perform spatially-varying deconvolution. Section V presents an alternative approach that runs on a normal PC using pre-computed inverse matrices. Section VI gives our results and section VII presents the conclusions.

University of Cambridge, Department of Engineering, Cambridge CB2 1PZ, UK. email: rwp@eng.cam.ac.uk

Manuscript received .....

II. Wiener Deconvolution

We consider a single A-line of received radio-frequency (RF) ultrasound signal, $q_{RF}\left(\mathbf{r_0}, t\right)$, to be a function of time, t, and the centre, $\mathbf{r_0}$, of the region on the transducer aperture where it is received.

$$q_{RF}\left(\mathbf{r_0}, t\right) = h_{RF}\left(\mathbf{r}, t\right) \underset{\mathbf{r}}{\otimes} f_m\left(\mathbf{r}\right)\Big|_{\mathbf{r}=\mathbf{r_0}} \tag{1}$$

$h_{RF}\left(\mathbf{r}, t\right)$ is the point-spread (or blurring) function of the imaging system. It incorporates the impulse response of the ultrasound transducer in both transmit and receive, as well as describing how a point scatterer contributes to the back-scattered ultrasound field. $f_m(\mathbf{r})$ is the field of scatterers which represent the information that, ideally, we would wish the ultrasound scanner to display. $\mathbf{r} = (x, y, z)$ is a vector to some point in space, where x and y are across the face of the transducer and z is the direction of insonification. The symbol $\otimes$ denotes convolution; in equation 1 it is over $\mathbf{r}$.

In order to represent this signal efficiently, we use the analytic forms, $\hat{q}_{RF}$ and $\hat{h}_{RF}$.

$$\hat{q}_{RF}\left(\mathbf{r_0}, t\right) = q_{RF}\left(\mathbf{r_0}, t\right) + j\mathcal{H}_t\left\{q_{RF}(\mathbf{r_0}, t)\right\} \tag{2}$$

$$\hat{h}_{RF}\left(\mathbf{r}, t\right) = h_{RF}\left(\mathbf{r}, t\right) + j\mathcal{H}_t\left\{h_{RF}(\mathbf{r}, t)\right\} \tag{3}$$

where $\mathcal{H}_t\left\{\cdot\right\}$ denotes the Hilbert transform. These functions can be multiplied by a sinusoid at the probe centre frequency to give a slowly varying complex envelope that can be down-sampled. Drawing on the analysis and assumptions from [1], it can be shown that the time trace at lateral position x and elevational position y is given by:

$$q\left(x, y, t\right) = \int h\left(x, y, z, t\right) \underset{x}{\otimes} \underset{y}{\otimes} f\left(x, y, z\right) \mathrm{d}z \tag{4}$$

$$\text{where} \quad q\left(x, y, t\right) = \hat{q}_{RF}\left(x, y, 0, t\right) e^{-j\omega_0 t}$$

$$h\left(x, y, z, t\right) = \hat{h}_{RF}\left(x, y, z, t\right) e^{j(2k_0 z - \omega_0 t)}$$

$$\text{and} \quad f\left(x, y, z\right) = f_m\left(x, y, z\right) e^{-2jk_0 z}$$

ω_0 is the centre frequency of the probe and k_0 is the corresponding spatial frequency in the material being scanned.

Since this is a linear equation, for discretised, real-world signals, it can be formulated in matrix-vector notation as

$$\mathbf{q} = H\mathbf{f} + \mathbf{n} \tag{5}$$

where H encompasses the h term, $\mathbf{q}$ is analogous to q and $\mathbf{f}$ is the vector of scatterers relating to f. We also introduce measurement noise, $\mathbf{n}$, which we assume to be normally distributed with zero mean and variance σ_n^2.

We assume elements of $\mathbf{f}$ are independent with a prior distribution that is complex, zero-mean and Gaussian with covariance c_f. We can thus calculate $\widehat{\mathbf{f}}$ which is a maximum *a posteriori* estimate of $\mathbf{f}$ using a Wiener filter [2],

$$\widehat{\mathbf{f}} = \left(H^{\mathrm{H}}H + \eta I\right)^{-1} H^{\mathrm{H}}\mathbf{q}. \tag{6}$$

where $\eta = \sigma_n^2/c_f$ and I is the identity matrix. $\left(H^{\mathrm{H}}H + \eta I\right)$ is too large to be stored or for its inverse to be explicitly computed as a whole. Despite this, it is simple to show that the matrix is Hermitian positive definite and so we can compute our estimate, $\widehat{\mathbf{f}}$, using the conjugate gradients algorithm

III. The test problem

The purpose of this paper is to study techniques for efficient implementation of spatially-varying deconvolution on three-dimensional ultrasound data of a realistic size. We have chosen to work on a rectangular block of analytic RF data with dimensions $96 \times 144 \times 584$. This is big enough to bring out most of the issues relating to practical problems and densely sampled data. The depth direction, in which the blur function varies, has dimension 584. It is worth noting that most clinical data sets are not regularly sampled but recorded in a fan-shaped sweep. This can often be accommodated using the same algorithms that are discussed in this paper, provided an appropriate transformation is applied to the assumed blur function.

Our test data block is a scan of an ultrasound phantom manufactured by the Department of Medical Physics at the University of Wisconsin, Madison, Wisconsin, USA [3]. It consists of a number of spheres with different levels of attenuation and back-scatter and enables geometric distortion and blurring to be easily seen. In particular, the data was acquired with a single focal depth, roughly half way down the block, in both the elevational and lateral directions. This results in clear distortions away from the focus that the deconvolution can correct. The benefit of similar deconvolution in a state-of-the-art machine will be dependent on whether the compromises in the beam-forming are greater than the error with which it is possible to model the blurring function. This paper is about implementation efficiency, so we do not discuss this important issue further [4].

The blur implicit in the beamforming process is modelled in three-dimensions using the Field II program [5]. An estimate of the noise-to-signal power ratio is computed in advance by spectral ratio techniques and made available to both the algorithms under test. For clarity and concision, we discuss the algorithms without considering edge effects. To facilitate this, we make the data tend to an average value at the elevational and lateral edges using a cosine window.

IV. Conjugate gradients algorithm on a GPU

The conjugate gradients algorithm [6], [7] finds the least mean-squared error solution for $\mathbf{z}$ in

$$A\mathbf{z} - \mathbf{b} = 0 \tag{7}$$

provided that A is positive definite. Comparing equation 7 to equation 6 for Wiener deconvolution, it is apparent that we can substitute $A = \left(H^{\mathrm{H}}H + \eta I\right)$ and $\mathbf{b} = H^{\mathrm{H}}\mathbf{q}$.

The vector $\mathbf{f}_0$ is set to an initial estimate of the deconvolved data, in our case simply the analytic RF data $\mathbf{q}$. Simple Jacobi preconditioning incorporated using a diagonal matrix, M, with values set to the value of η plus the average energy in the blur function, H, at each corresponding depth. $\mathbf{f}_n$ is the estimate of the the scatterer field in the nth iteration of the algorithm, $\mathbf{s}$ is a working vector generally known as the residual, $\mathbf{z}$ and $\mathbf{p}$ are further working vectors, and i_{max} is the number of iterations of the conjugate gradient algorithm required. The conjugate gradients algorithm is performed as follows:

1 $\mathbf{b} = H^{\mathrm{H}}\mathbf{q}$
2 $A = \left(H^{\mathrm{H}}H + \eta I\right)$
3 $\mathbf{f}_0 = \mathbf{q}$
4 $\mathbf{s}_0 = \mathbf{b} - A\mathbf{f}_0$
5 $\mathbf{z}_0 = M^{-1}\mathbf{s}_0$
6 $\mathbf{p}_0 = \mathbf{z}_0$
7 **for** $n = 1 : i_{max}$
8 $\quad \alpha = \dfrac{\mathbf{z}_{n-1}^{\mathrm{H}}\mathbf{s}_{n-1}}{(A\mathbf{p}_{n-1})^{\mathrm{H}}\mathbf{p}_{n-1}}$
9 $\quad \mathbf{f}_n = \mathbf{f}_{n-1} + \alpha\mathbf{p}_{n-1}$
10 $\quad \mathbf{s}_n = \mathbf{s}_{n-1} - \alpha A\mathbf{p}_{n-1}$
11 $\quad \mathbf{z}_n = M^{-1}\mathbf{s}_n$
12 $\quad \beta = \dfrac{\mathbf{z}_n^{\mathrm{H}}\mathbf{s}_n}{\mathbf{z}_{n-1}^{\mathrm{H}}\mathbf{s}_{n-1}}$
13 $\quad \mathbf{p}_n = \mathbf{z}_n + \beta_n\mathbf{p}_{n-1}$
14 **end**

We have developed an efficient implementation of this algorithm in the CUDA[1] language that runs on Nvidia[2] graphics cards. Run on an Nvidia C1060 GPU, with a predominantly single-precision implementation, each iteration of this algorithm (lines 7–14) takes 358 ms (averaged over 50 iterations). In this, 93.5% of the time is spent calculating the spatially-varying forward blur operation, $A\mathbf{p}_{n-1}$, that is used in lines 8 and 10. We will therefore focus our discussion on the implementation of this operation.

Figure 1 shows the form of the ultrasound data block. If the blur was spatially invariant (subject to the consideration of edge effects), we could Fourier transform the data in all three dimensions, multiply by the frequency domain representation of the blur, and transform the result back to the spatial domain to get the answer. As the blur only varies with depth, we can still use this strategy for the two other dimensions. The overall strategy is therefore:

1) Perform a two-dimensional Fourier transform on the planes of data at each depth (see the example labelled in figure 1). This takes 50.7 ms.
2) Multiply each column (see figure 1) of the resulting data by a different 584×584 complex matrix to perform the

[1]Compute Unified Device Architecture
[2]Nvidia Inc, 2701 San Tomas Expressway, Santa Clara, USA

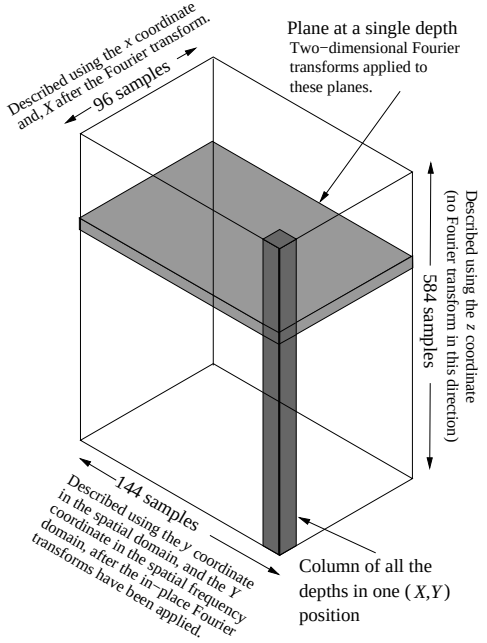


Fig. 1. The block of three dimensional ultrasound data showing the number of samples in the complex analytic representation and the coordinates used in the spatial and spatial-frequency domains.

spatially varying blur in the depth direction. This takes 234 ms.

3) Perform inverse Fourier transforms on all the planes of data as in (1) above. This takes another 50.7 ms.

There are thus $144 \times 96 = 13824$ matrices containing a frequency domain representation of the blur function. Although they are formally 584×584 complex elements, they are both Hermitian and sparse, with most of the non-zero elements close to the leading diagonal. When stored as double-precision sparse matrices in Matlab[3] this takes up about 2.2 Gbytes. By using single precision and only storing the values on and above the leading diagonal (see figure 2) we can reduce this to 0.76 Gbytes. In this form, the 13824 blurring matrices are stored in the global memory of the GPU.

The data block is also stored in the GPU global memory. In order to enable the Fourier transforms to be performed efficiently, it is stored so that planes of data at each depth are contiguous. This means that the values relating to each "column of data at one position" (see figure 1) are spread out in memory and less efficient to read. There are 292 ($= 584/2$) *threads* in each CUDA *thread block*. After the Fourier transforms, each thread block is responsible for multiplying one matrix by one column of the data. First, all the threads in the block load the input column into shared memory, then each thread works out the dot product required for two of the elements of the answer, lastly they copy the answers back to global memory. This leaves the data correctly positioned for the inverse Fourier transforms. When working out the dot products it is important that the small values, away from the leading diagonal elements, are accumulated first. If values are added up in the wrong order it greatly increases the rounding errors.

[3]The Mathworks, 3 Apple Hill Drive, Natick, MA, USA

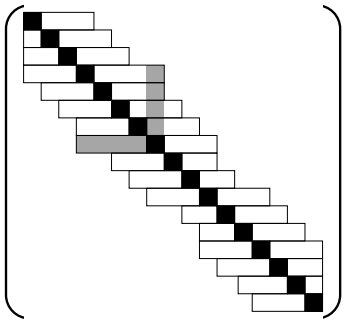


Fig. 2. Structure of the blurring matrix for a single column in the data block. The matrix is Hermitian so it is only necessary to store the number on and above the leading diagonal. Numbers below the leading diagonal can be calculated from the complex conjugates of the corresponding numbers above; see the shaded regions on the diagram.

As far as the remaining 6.5% of the time is concerned, the computation is performed with one thread for each value in the data and the threads are grouped to permit efficient coalesced access to global memory. Furthermore, a tree-structured *parallel reduction* strategy is used instead of a loop in all the dot-product summations. Single precision arithmetic is not sufficient for calculating the factors α and β (in lines 8 and 15 of the algorithm), so this part of the calculation is performed in double precision.

V. Precomputed direct inversion

By considering the form of equation 6, an alternative method for computing $\widehat{\mathbf{f}}$ becomes apparent. It is easier to study the equation by presenting the forward operation, which we rewrite here for the reader's convenience.

$$\left(H^{\mathrm{H}}H + \eta I\right)\widehat{\mathbf{f}} = H^{\mathrm{H}}\mathbf{q}. \tag{8}$$

We now write this equation as an integral equation of four variables in the continuous domain, analogous to equation 4. This allows us to present the inherently multi-dimension nature of the problem in a more obvious manner. We firstly define $h'(x,y,z,t)$ to be the complex conjugate of $h(x,y,z,t)$ and $\widehat{f}_m(x,y,z)$ to be the estimated field of scatterers, giving:

$$\int \left[h'(x,y,z,t) \underset{x}{\otimes}\underset{y}{\otimes} \int h(x,y,z',t) \underset{x}{\otimes}\underset{y}{\otimes} \widehat{f}_m(x,y,z')\,\mathrm{d}z' \right] \mathrm{d}t + \eta \widehat{f}_m(x,y,z) = \int h'(x,y,z,t)\, q(x,y,t)\,\mathrm{d}t.$$

Note that in this equation, z' is a dummy variable. Changing the order of the integration yields:

$$\int \left[\int h'(x,y,z,t) \underset{x}{\otimes}\underset{y}{\otimes} h(x,y,z',t)\,\mathrm{d}t \right] \underset{x}{\otimes}\underset{y}{\otimes} \widehat{f}_m(x,y,z')\,\mathrm{d}z' + \eta \widehat{f}_m(x,y,z) = \int h'(x,y,z,t)\, q(x,y,t)\,\mathrm{d}t$$

So, defining

$$g(x,y,z,z') = \int h'(x,y,z,t) \underset{x}{\otimes}\underset{y}{\otimes} h(x,y,z',t)\,\mathrm{d}t,$$

we can write down an integral equation form for equation 6:

$$\int g(x,y,z,z') \underset{x}{\otimes} \underset{y}{\otimes} \widehat{f}_m(x,y,z')\,\mathrm{d}z' + \eta \widehat{f}_m(x,y,z) = \int h'(x,y,z,t)\, q(x,y,t)\,\mathrm{d}t \quad (9)$$

It is apparent that by the convolution theorem, we can immediately remove the two inner convolution integrals of the first term on the left hand side. Defining $G(\omega_x,\omega_y,z,z') = \mathcal{F}_x\mathcal{F}_y[g(x,y,z,z')]$ and $\widehat{F}_m(\omega_x,\omega_y,z) = \mathcal{F}_x\mathcal{F}_y\left[\widehat{f}_m(x,y,z)\right]$ where $\mathcal{F}_x$ and $\mathcal{F}_y$ denote the Fourier transform in the x and y directions respectively, we can say

$$\int [G(\omega_x,\omega_y,z,z') + \eta\delta(z-z')]\,\widehat{F}_m(\omega_x,\omega_y,z')\,\mathrm{d}z' = \mathcal{F}_x\mathcal{F}_y\left[\int h'(x,y,z,t)\, q(x,y,t)\,\mathrm{d}t\right] \quad (10)$$

where $\delta(z)$ is the Dirac delta function.

At this point, it is easiest to think, once again, in discrete matrix vector form. The important thing to realise about the left hand side of equation 10 is that we now have an integral in terms of only z'. This means that in the discrete matrix-vector formulation, the time-consuming part of the calculation reduces to the process of inverting a 584×584 matrix for each column of data in the x–y plane, and these matrices can be precomputed and stored.

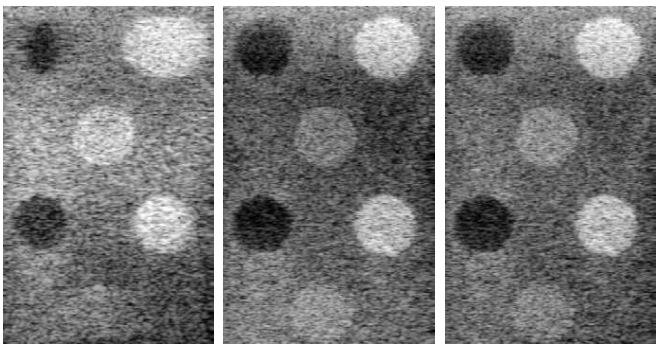
We define $\widehat{\mathbf{F}}_{X,Y}$ to be the subvector of $\widehat{\mathbf{F}}$ for the column of data (see figure 1) in the spatial frequency domain at position (X,Y). Similarly, we define $\mathbf{B}_{X,Y}$ as the subvector for the column of data at position (X,Y) with entries described by the right hand side of equation 10. For each corresponding pair of $\widehat{\mathbf{F}}_{X,Y}$ and $\mathbf{B}_{X,Y}$ we define a matrix $D_{X,Y}$, using the left side of equation 10, that describes the mapping from $\widehat{\mathbf{F}}_{X,Y}$ to $\mathbf{B}_{X,Y}$. Hence:

$$\widehat{\mathbf{F}}_{X,Y} = D_{X,Y}^{-1}\mathbf{B}_{X,Y}$$

In our test problem, each $D_{X,Y}$ matrix is 584×584. Once all the 13824 D^{-1} matrices have been computed, the calculation is similar in form to a single forward pass of the blurring operation, $A\mathbf{p}$ that we discussed in the previous section for the conjugate gradients algorithm. However, it is slightly less convenient to perform this operation on a graphics card as the $D_{X,Y}^{-1}$ matrices are less sparse than the matrices that make up A. It takes over 4.2 Gbytes to store all the $D_{X,Y}^{-1}$ matrices, even using single precision and dense packing and this is too much to fit on currently available GPU hardware. Furthermore, we can get reasonable performance using the parallelism available on a standard multi-processor PC.

VI. Results

Table I shows the time taken to implement the spatially-varying deconvolution in the two ways that are the focus of this paper, compared to an implementation of the conjugate gradients algorithm in Matlab. Figure 3 shows a slice through the middle of the data block before and after each deconvolution presented using envelope detection and logarithmic compression as would be normal for the display of a B-scan.



(a) Before (b) After 50 iterations of the conjugate gradients implementation (c) After use of the pre-computed inverses implementation

Fig. 3. Images showing a central slice through the data block before and after deconvolution. The strips down the edges of the images where the data is windowed to mid-grey to avoid edge effects are not shown.

Implementation method	Seconds
Fifty iterations of the conjugate gradients algorithm in Matlab on an Intel Core i7 at 2.67GHz	454.8
Fifty iterations of the conjugate gradients algorithm in CUDA on an Nvidia C1060	22.6
Application of pre-computed inverse matrices in Matlab on a quad-core Intel Core i7 at 2.67GHz	5.4

TABLE I
TIMINGS OF THE THREE IMPLEMENTATIONS OF WIENER DECONVOLUTION. ALL TIMES ARE THE AVERAGES OF 40 RUNS.

The time for the conjugate gradient algorithm is approximately proportional to the number of iterations for which it is run. We chose 50 iterations as this achieves a result of similar quality to the pre-computed inverse technique (see figure 3).

In the CUDA implementation, the rate-determining operation is the process of accessing the many large blur matrices that are stored in slow global memory. The main problem with the direct solution based on pre-computed inverse matrices is the size of these matrices, however they are still within the capability of a normal PC to manipulate. In both implementations, there is a requirement for an efficient implementation of the Fourier transform in two dimension.

The reduction in accuracy caused by the use of single precision arithmetic and storage in most of the CUDA implementation is not noticable in the resulting ultrasound images and creates a mean error of 0.00016%, and a maximum error of 0.45% which occurs for an output value that is very low (0.00027%) compared to the mean.

VII. Conclusions

We have demonstrated two ways of increasing the speed with which a spatially-varying Wiener deconvolution can be implemented on a large three-dimensional ultrasound data block. One approach is based on GPU hardware, the other involves directly inverting the 13824 blur matrices in advance.

There is scope for further speed improvements by using more than one GPU or a CPU with more than four cores. If the size of the $D_{X,Y}^{-1}$ matrices cannot be reduced but, nevertheless, it is desired to implement the pre-computed inverses approach

using CUDA, then a multi-GPU approach would provide a solution by splitting up the problem in the depth direction and spreading the matrices across the memory of several cards.

REFERENCES

[1] J. Ng, R. Prager, N. Kingsbury, G. Treece, and A. Gee, "Modeling ultrasound imaging as a linear shift-variant system," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 53, no. 3, pp. 549–563, 2006.

[2] ——, "Wavelet restoration of medical pulse-echo ultrasound images in an EM framework," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 54, no. 3, pp. 550–568, 2007.

[3] J. M. Kofler Jr. and E. L. Madsen, "Improved method for determining resolution zones in ultrasound phantoms with spherical simulated lesions," *Ultrasound in Medicine and Biology*, vol. 27, no. 12, pp. 1667–1676, 2001.

[4] H.-C. Shin, R. Prager, J. Ng, H. Gomersall, N. Kingsbury, G. Treece, and A. Gee, "Sensitivity to point-spread function parameters in medical ultrasound image deconvolution," *Ultrasonics*, vol. 49, no. 3, pp. 344–357, 2009.

[5] J. A. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultr asound transducers," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 39, pp. 262–267, 1992.

[6] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[7] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics, 1997.