2005

Solving an MRI spin relaxometry problem with parallel computing

Alan Kaminsky

Luke McOmber

Solving an MRI Spin Relaxometry Problem with Parallel Computing

Alan Kaminsky
Department of Computer Science
Rochester Institute of Technology
ark@cs.rit.edu

Luke McOmber
Department of Computer Science
Rochester Institute of Technology
lam1101@cs.rit.edu

June 27, 2005

Abstract

MRI spin relaxometry is the process of recovering the spin density spectrum from the time samples of the spin signal for each pixel of a magnetic resonance image. Since healthy tissue exhibits different spin relaxation rates from diseased tissue, MRI spin relaxometry potentially has utility for diagnosing disease. However, recovering the spin relaxation rates involves solving an inverse problem which requires substantial computation. The computation's running time can be reduced by processing the pixels in parallel on a parallel computer. A parallel program for solving the MRI spin relaxometry problem, SRSolve, was implemented in Java with MPI, its running time was measured on a 32-processor cluster parallel computer, and its performance was compared to the CONTIN program. CONTIN required about 44 sec on the average to solve one pixel and about 3600 sec to solve an entire 64×64-pixel test image (with 2,597 unmasked pixels) on the parallel computer. SRSolve required 3.04 sec on the average to solve one pixel and 263 sec to solve the entire image on the parallel computer.

1 Magnetic Resonance Imaging

Hornak [1] describes **magnetic resonance imaging**.

> Magnetic Resonance Imaging (MRI) is an imaging technique used by radiologists to diagnose disease in the human body. Radiologists are continually looking for ways to extract more information from magnetic resonance images and hence make a better diagnosis. It has been suggested that relaxometry, or the distribution of MRI spin-lattice relaxation rate (R_1) and the spin-spin relaxation rate (R_2) of the tissues may have diagnostic utility. [2] This diagnostic utility is based on R_1 and R_2 being a measure of the mobility of molecules, which is useful as the mobility of molecules in a tissue changes with disease state.

An MRI scanner takes images of two-dimensional slices through the object being scanned, such as a human body. By taking a series of images at different locations, a three-dimensional

picture can be constructed. To image one slice, the MRI scanner sends magnetic pulses through the object and captures an **imaging sequence**. That is, each slice's data consists of M images taken at a sequence of M time values t, with each image comprising $R \times C$ pixels. A certain pixel in a certain image measures the MRI **signal** at the pixel's location for the image's corresponding time t.

An MRI scanner can measure different kinds of signals. For this study we will use the **inversion recovery sequence**. In this case the ideal value of a pixel's signal S at a given time t is

$$S(t) = \rho(1 - 2\mathrm{e}^{-xt}) \tag{1}$$

where ρ is the spin's **density**, x is the spin's **relaxation rate** in sec^{-1}, and t is the time in sec. (We have simplified the notation a bit from [1].) Figure 1 is a plot of S versus t for ρ = 1000 and $x = 0.5$.

Figure 1: S versus t — single spin

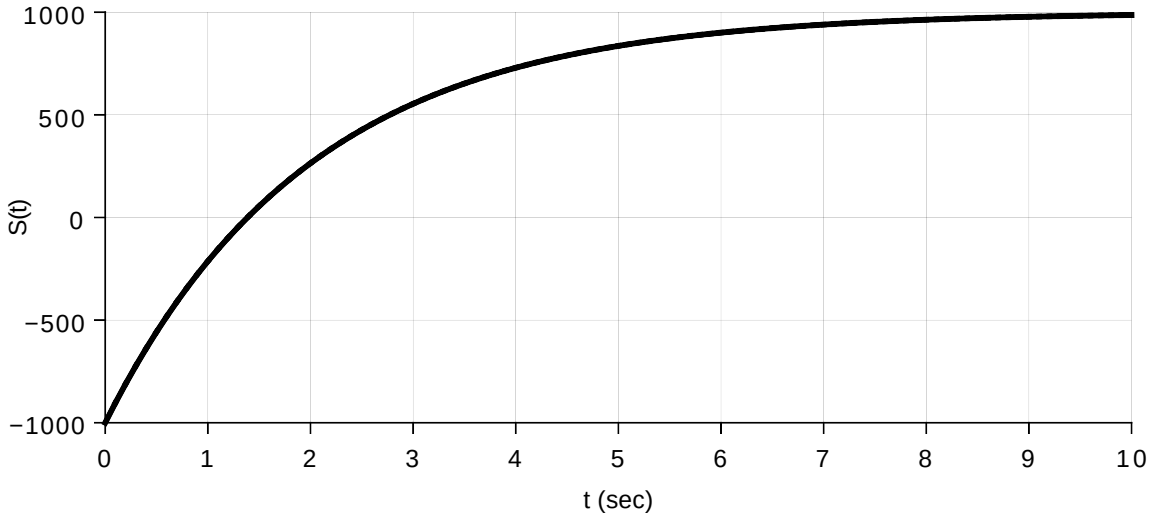
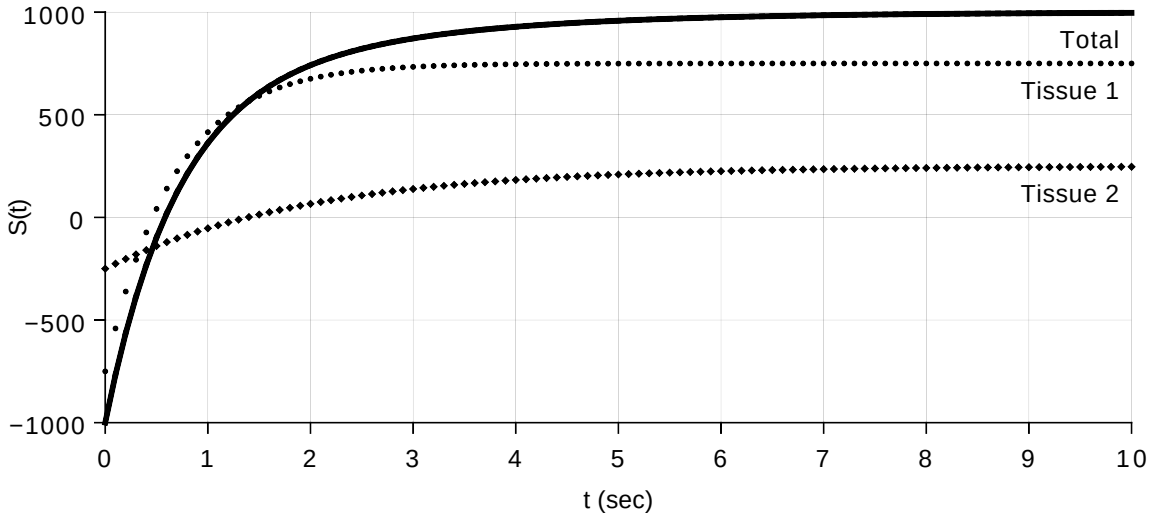


Figure 1 shows the signal that would result if there were only one spin. Actually, each pixel in the image represents several kinds of tissue, and each kind of tissue has its own separate spin density and relaxation rate. So the ideal value of a pixel's signal S at a given time t is a summation of the signals from the separate kinds of tissue.

$$S(t) = \sum_j \rho_j(1 - 2\mathrm{e}^{-x_j t}) \tag{2}$$

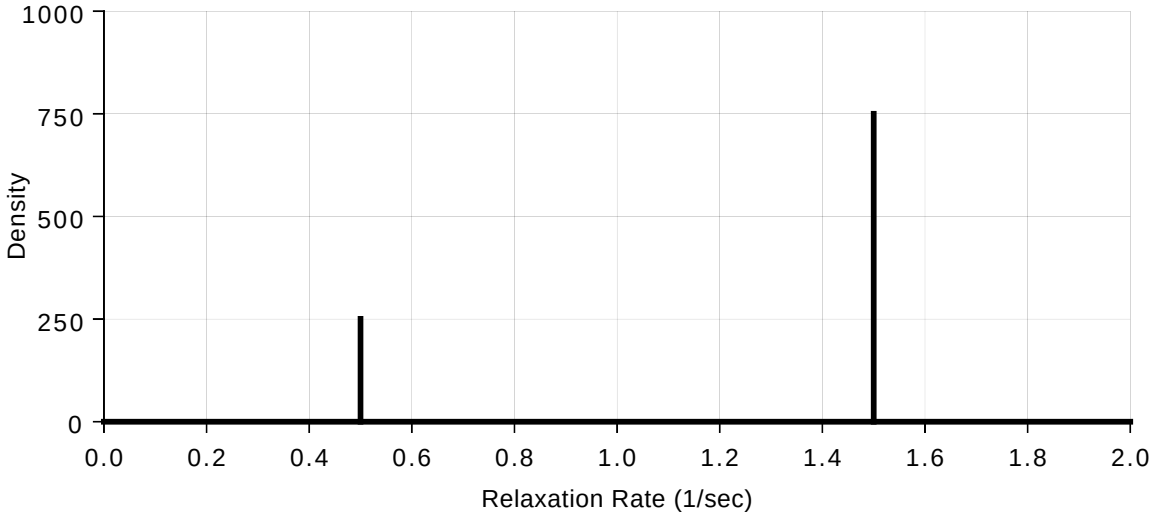
where the index j ranges over all the tissues, ρ_j is the spin density for tissue j, x_j is the spin relaxation rate for tissue j in sec^{-1}, and t is the time in sec. Figure 2 is a plot of S versus t for two kinds of tissues. Tissue 1 has $\rho_1 = 750$ and $x_1 = 1.5$. Tissue 2 has $\rho_2 = 250$ and x_2 = 0.5.

Figure 2: S versus t — two discrete spins



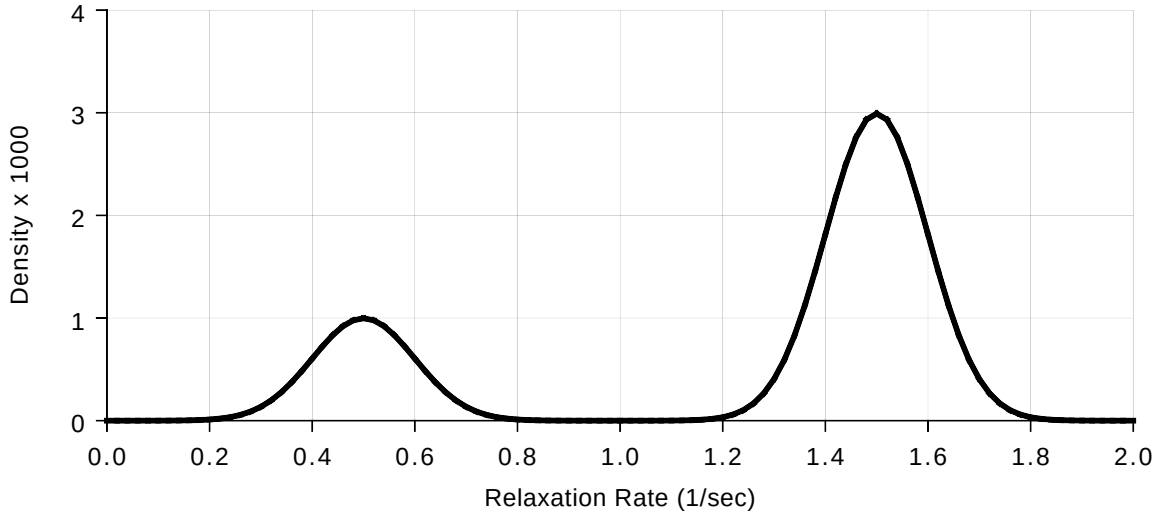
In Equation (2), both the density ρ_j and the relaxation rate x_j were expressed in terms of the index j. Alternatively, we can express them in terms of each other. Specifically, we can treat the density as a function of the relaxation rate, $\rho(x)$. If there are two tissues, for example, $\rho(x)$ will be 0 everwhere except at two discrete values of x.

Figure 3: Spin density versus relaxation rate — two discrete spins



In a real MRI scan, however, we would not expect the tissues' spin relaxation rates to be so sharply defined. Instead, we would expect to see a range of relaxation rates clustered around some central value or values. In other words, we would expect ρ to be a *continuous* function of x instead of a discrete function. We will refer to $\rho(x)$ as the **spin density spectrum** or **spin spectrum** for short.

Figure 4: Spin spectrum

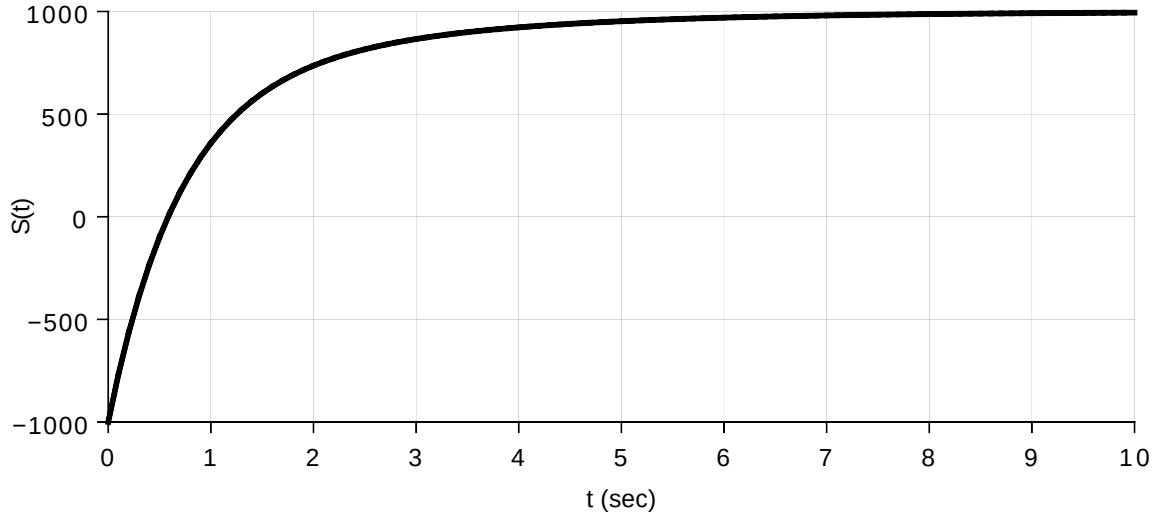


Consequently, the formula for the spin signal becomes a continuous integral instead of a discrete summation — compare Equation (3) to Equation (2).

$$S(t) = \int \rho(x)(1 - 2\mathrm{e}^{-xt})\mathrm{d}x \tag{3}$$

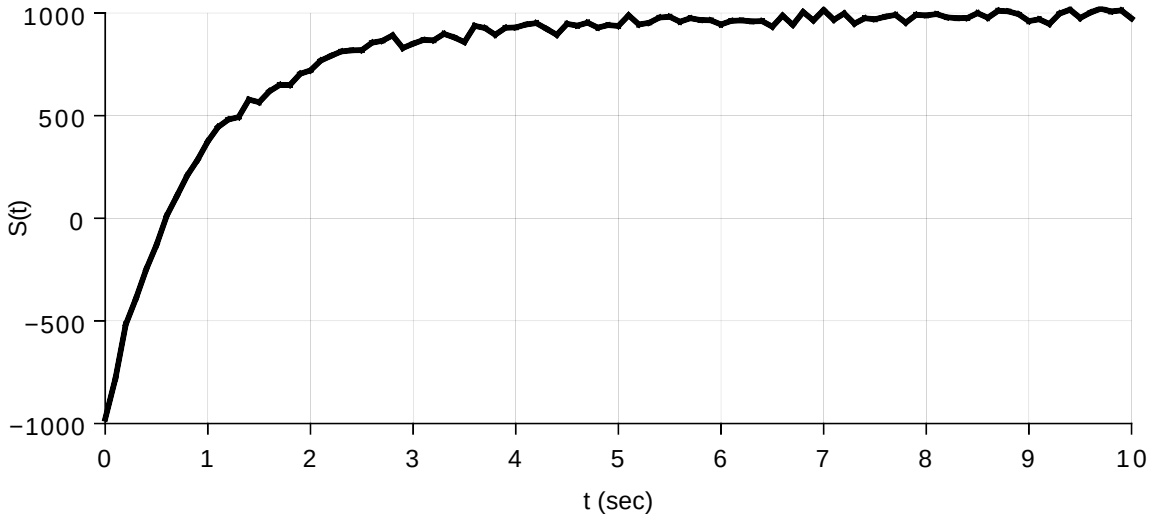
Figure 5 is a plot of $S(t)$ for the spin spectrum in Figure 4.

Figure 5: S versus t — continuous spins



Equation (3) is the ideal MRI signal function. In reality, the MRI scanner's measurements at each value of t are subject to **measurement errors**. The measurement errors show up as random "noise" added to the ideal signal function. So here, finally, is what the signal might look like for one pixel of one image from an MRI scan.

Figure 6: S versus t — continuous spins with noise



The **spin relaxometry analysis** problem is to take a noisy MRI signal like the one in Figure 6 and derive the spin spectrum which generated the signal. This is known as an **inverse problem** — we are going "backwards" from the final output $S(t)$ to the original input $\rho(x)$. Furthermore, we have to do this for *every* pixel of the MRI image, since each pixel represents different tissues with different spin spectra.

A radiologist would examine the pixels' spin spectra to diagnose disease. Healthy tissue would have peaks in the spin spectrum at certain spin relaxation rates; diseased tissue would have peaks in the spin spectrum at different spin relaxation rates. We are not going to address the problem of *interpreting* the spin spectra. We are just going to *analyze* the MRI signals to produce the spin spectra.

2 Input Data Files

This section describes the input data files for the MRI spin relaxometry analysis problem. There are a **times file**, a number of **image files**, and a **mask file**.

2.1 Times File

The times file is a plain text file giving the image dimensions in pixels and the list of discrete time values at which the images were taken. Each line of the times file consists of fields separated by white space.

The first line of the times file has two fields, the number of pixel rows in the image R (integer) and the number of pixel columns in the image C (integer).

The rest of the times file consists of M lines. The first field on each line gives t_i, the time (sec, real number) at which image i was taken, $1 \leq i \leq M$. The second field on each line gives the name of the image file for image i.

2.2 Image File

Each image file is a binary file containing the pixel values for a certain image i taken at time t_i. The file contains $R \times C$ pixel values $S_{rc}(t_i)$, where r is the pixel's row in the range $0 \leq r \leq R-1$ and c is the pixel's column in the range $0 \leq c \leq C-1$. The pixel values are stored in row major order — first the pixels in the topmost row from left to right, then the pixels in the second row from left to right, and so on.

Each pixel value is stored in binary as a two-byte integer in the range $-32768 \leq S_{rc}(t_i) \leq 32767$. The first two pixels ($S_{00}$ and S_{01}) do not contain actual data, but instead are set to the values $S_{00} = 0$ and $S_{01} = 1$. These are used to determine the byte order or **endianness** with which the pixel values are stored. If the first four bytes of the image file are 0, 0, 0, 1, then the pixel values are stored in **big-endian order** — most significant byte first. If the first four bytes of the image file are 0, 0, 1, 0, then the pixel values are stored in **little-endian order** — least significant byte first.[1]

2.3 Mask File

The mask file is a binary file containing $R \times C$ pixel mask values M_{rc}. M_{rc} is 1 if pixel (r, c) has data to be analyzed. M_{rc} is 0 if pixel (r, c) is not to be analyzed. Masking out certain pixels, such as pixels near the edges of the image that are not part of the object being scanned, reduces the time needed to do the analysis.

Each mask value is stored in binary as a two-byte integer. The first pixel (M_{00}) does not contain actual mask data, but instead is set to the value $M_{00} = 1$. This is used to determine the mask values' byte order. If the first two bytes of the mask file are 0, 1, then the mask values are stored in big-endian order. If the first two bytes of the mask file are 1, 0, then the mask values are stored in little-endian order.

2.4 Example

Figure 7 depicts the input data for an MRI spin relaxometry analysis problem supplied by Hornak. There are $M = 64$ images of a two-dimensional slice through someone's brain, each image consisting of $R = 64$ rows and $C = 64$ columns. The black pixels around the edge of each image are pixels that have been masked out. The gray pixels in the center of each image represent the signal values $S_{rc}(t_i)$, with darker gray representing smaller values and lighter gray representing larger values. The time value t_i is listed under each image.

[1] Note that a Java program cannot use class java.io.DataOutputStream to read the pixel values, since that class always uses big-endian order.

Figure 7: Example input data

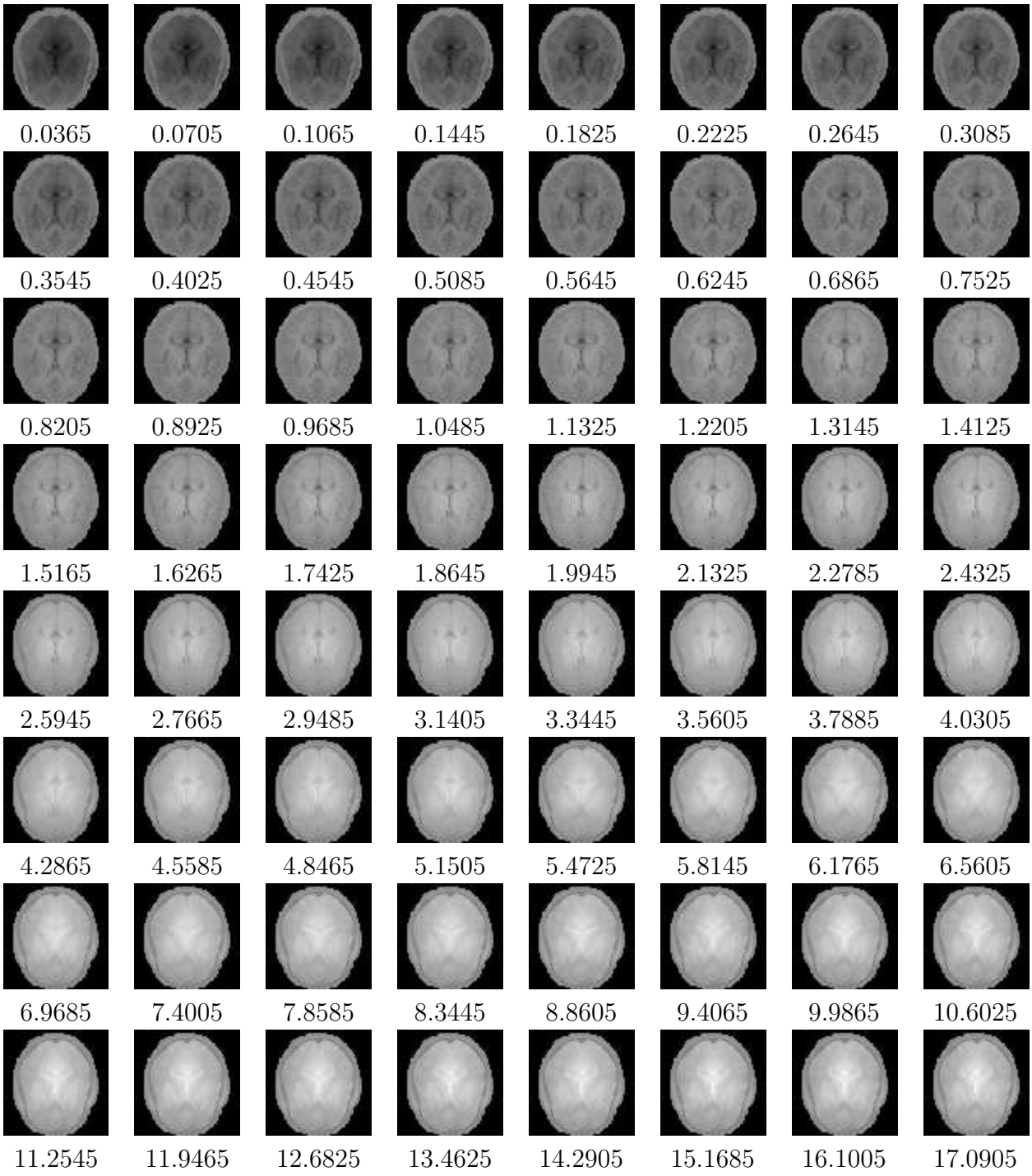
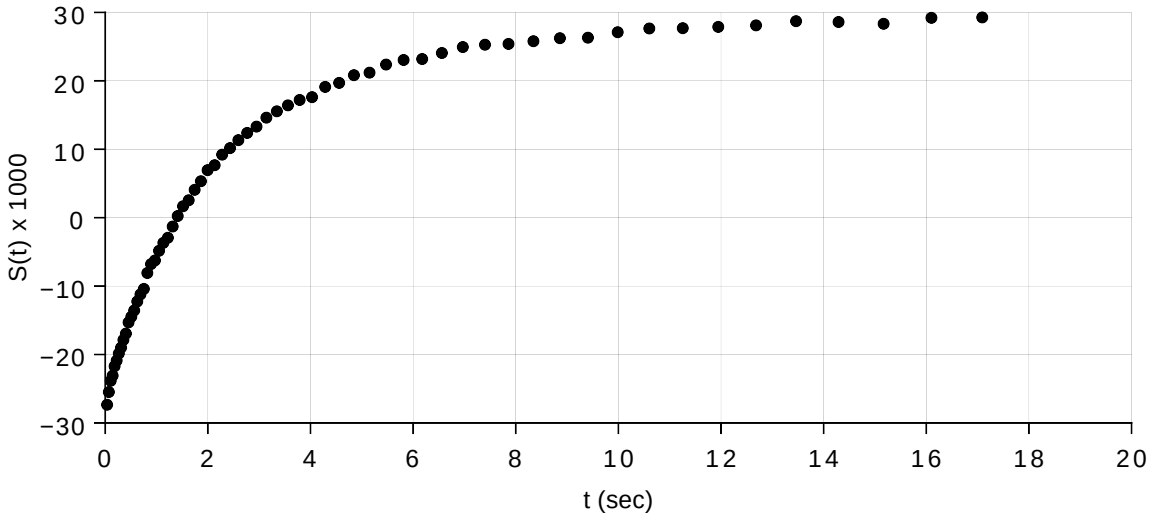


Figure 8 focuses on a single pixel and plots $S(t)$ for pixel $(32, 32)$.

Figure 8: S versus t — pixel $(32, 32)$



3 Solving the Inverse Problem

Hornak had been using the CONTIN program, a Fortran program written by Provencher in 1982, to solve the MRI spin relaxometry inverse problem. CONTIN is a "general purpose constrained regularization program for inverting noisy linear algebraic and integral equations" [3, 4]. For the 64×64-pixel test image in Section 2, which has 2,597 unmasked pixels, Bak, Hornak, and Schaller [1] reported that CONTIN required over 55 hours to solve the whole image using one processor of a cluster parallel computer — about 76 seconds per pixel on the average. They investigated running multiple copies of CONTIN simultaneously to reduce the time required to solve the image, and they reported a running time of less than an hour using all 32 processors of the cluster parallel computer — about 44 seconds per pixel on the average. (They noted, but did not have an explanation for, the fact that the program required less time per pixel when running on 32 processors than when running on one processor.)

However, their approach treated CONTIN as a "black box" and only added parallelism "outside the black box" — that is, CONTIN itself did not use a parallel algorithm, rather multiple copies of CONTIN's sequential algorithm were run simultaneously on different processors. We hypothesized that we could achieve a further reduction in the running time by putting parallelism "inside the black box" — that is, by implementing a new program that was designed using parallel algorithms in the first place.

We investigated two methods for solving the inverse problem. The first was the Backus-Gilbert method [5, 6, 7, 8]. We were initially attracted to the Backus-Gilbert method because it had the potential for significant speedups with a parallel implementation. However, prototype investigations showed that the Backus-Gilbert method did not recover the spin spectrum with adequate resolution. Specifically, an input signal derived from a spin spectrum with sharp spikes (like Figure 3) resulted in an output spin spectrum with broad, flattened humps,

and no adjustment of the Backus-Gilbert method's parameters resulted in an output spin spectrum that looked anything like the original spin spectrum.

We then investigated the constrained linear regularization method [9, 10], which is the method that CONTIN uses (although CONTIN is more general). Constrained linear regularization gave much better results; it was able to recover an output spin spectrum that closely matched a known original spin spectrum.

We wrote a Java program, SRSolve, to solve the MRI spin relaxometry inverse problem for all pixels of an image on a cluster parallel computer. To write SRSolve, we had to address two issues: how to perform the constrained linear regularization, and how to determine the value of the linear regularization method's tradeoff parameter. These issues are discussed in Sections 4 and 5, respectively.

4 Performing the Constrained Linear Regularization

4.1 Linear Regularization

Linear regularization, also known as **Tikhonov regularization** [10], seeks to solve the following inverse problem. An output function $g(y)$ is defined in terms of an input function $f(x)$ and a response kernel $r(x, y)$ as:

$$g(y) = \int f(x)r(x, y)\mathrm{d}x \tag{4}$$

The continuous integral is approximated as a discrete summation:

$$g(y_i) = \sum_{j=1}^{N} f(x_j)r(x_j, y_i),\ i = 1, 2, \ldots M \tag{5}$$

Then given the M values of y_i, the N values of x_j, the measured samples of the output function $g(y_i)$ (which typically include measurement errors), and the response kernel values $r(x_j, y_i)$, the inverse problem is to find the samples of the input function $f(x_j)$ that yield the given output function samples.

Linear regularization begins by finding a least-squares solution; that is, by minimizing the χ^2 statistic:

$$\chi^2 = \sum_{i=1}^{M} \left[\frac{g_i - \sum_{j=1}^{N} r_{ij} f_j}{\sigma} \right]^2 \tag{6}$$

where we have defined $g_i \equiv g(y_i)$, $f_j \equiv f(x_j)$, $r_{ij} \equiv r(x_j, y_i)$, and σ is the standard deviation of the measurement error in the output function samples. Minimizing Equation (6) is equivalent to finding the vector $\mathbf{f}$ that is the best (least-squares) solution to the matrix equation

$$\mathbf{A} \cdot \mathbf{f} = \mathbf{g} \tag{7}$$

where $\mathbf{A}$ is an M-row-by-N-column matrix of the values r_{ij}/σ, $\mathbf{f}$ is an N-element vector of the values f_j, and $\mathbf{g}$ is an M-element vector of the values g_i/σ.

Solving just Equation (7) by itself would yield a solution that faithfully matched the output vector $\mathbf{g}$, including all the random fluctuations due to measurement error. This tends to yield a solution vector $\mathbf{f}$ with random fluctuations also. However, we typically assume that the true input vector does not have random fluctuations, and we want to recover a solution vector with the random fluctuations "smoothed out."

Linear regularization imposes smoothness by making a chosen derivative of the solution as close to zero as possible for every point in the solution. For the MRI spin relaxometry problem, we chose to use second-order regularization (which is what Provencher recommends for problems of this sort [3]); that is, to make the second derivative of the solution as close to zero as possible. The formula

$$f'_j = -f_j + f_{j+1} \tag{8}$$

is a discrete approximation that is proportional to the first derivative of f_j, and the formula

$$f''_j = -f'_j + f'_{j+1} = f_j - 2f_{j+1} + f_{j+2} \tag{9}$$

is a discrete approximation that is proportional to the second derivative of f_j. If we then minimize the quantity

$$\zeta^2 = \sum_{i=1}^{M} \left[0 - \sum_{j=1}^{N-2} (f_j - 2f_{j+1} + f_{j+2}) \right]^2 \tag{10}$$

we will get a smoothed solution where every three-point "patch" of consecutive values (f_j, f_{j+1}, f_{j+2}) is close to a straight line. Minimizing Equation (10) is equivalent to finding the vector $\mathbf{f}$ that is the best (least-squares) solution to the matrix equation

$$\mathbf{B} \cdot \mathbf{f} = \mathbf{0} \tag{11}$$

where $\mathbf{B}$ is an $(N-2)$-row-by-N-column matrix with all elements 0 except $B_{ii} = 1, B_{i(i+1)} = -2, B_{i(i+2)} = 1, i = 1, 2, \ldots N-2$. For example, for $N = 6$, the matrix $\mathbf{B}$ is

$$\mathbf{B} = \begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \end{pmatrix} \tag{12}$$

We now have two criteria for solving the inverse problem using linear regularization: to get the best possible *agreement* between the actual output vector $\mathbf{g}$ and the output reconstructed from the solution vector $\mathbf{f}$ by minimizing χ^2 in Equation (6), and to get the best possible *smoothness* in the solution vector $\mathbf{f}$ by minimizing ζ^2 in Equation (10). However, these criteria conflict. Getting the best possible agreement yields a solution that is not smooth (it contains random fluctuations due to measurement error), and getting the best possible smoothness yields a solution that does not agree (the output vector reconstructed from the solution vector is nowhere near the actual output vector). Therefore, we must effect a *tradeoff* between agreement and smoothness by minimizing the combined metric ξ^2:

$$\xi^2 = \chi^2 + \lambda^2 \zeta^2 \tag{13}$$

where λ is the **tradeoff parameter**. If λ is small, the first term will dominate, and the solution will agree but will not be smooth. If λ is large, the second term will dominate, and the solution will be smooth but will not agree. (Section 5 will discuss how to choose an appropriate value for λ.) Minimizing Equation (13) is equivalent to finding the vector $\mathbf{f}$ that is the best (least-squares) solution to the matrix equation

$$\begin{pmatrix} \mathbf{A} \\ \lambda\mathbf{B} \end{pmatrix} \cdot \mathbf{f} = \begin{pmatrix} \mathbf{g} \\ \mathbf{0} \end{pmatrix} \tag{14}$$

The left hand side matrix has $M + N - 2$ rows and N columns, with the first M rows the same as the matrix $\mathbf{A}$ and the remaining rows the same as λ times the matrix $\mathbf{B}$. The right hand side vector has $M + N - 2$ elements, with the first M elements the same as the vector $\mathbf{g}$ and the remaining elements 0. ξ^2 is the squared norm of the error for this solution, that is, the square of the Euclidean norm (Euclidean distance) between the left hand side and the right hand side of Equation (14).

4.2 Measurement Error

The elements of the matrix $\mathbf{A}$ and the vector $\mathbf{g}$ are scaled by the measurement error standard deviation σ. If σ is not known, we will not be able to solve the inverse problem. However, note that in Equation (14), if σ is changed by some amount, λ can also be changed by some amount such that both sides of the equation are scaled by the same factor; then the equation will yield the same solution as before. This means we don't actually need to know σ to solve the inverse problem. We can simply take σ to be a convenient value, say $\sigma = 1$. We still have to pick the proper value of λ, as Section 5 will discuss.

4.3 Formulas for MRI Spin Relaxometry

For the MRI spin relaxometry problem, the quantities in the linear regularization formulas in the previous section are defined as follows.

- M = number of time samples.
- $y_i = t_i$, the i-th time value, $i = 1, 2, \ldots M$.
- $g_i = S(t_i)$, the i-th spin signal value for the pixel being analyzed, $i = 1, 2, \ldots M$.
- N = number of spin spectrum samples.
- $x_j = x_j$, the j-th spin relaxation rate value, $j = 1, 2, \ldots N$.
- $f_j = \rho(x_j)$, the j-th spin density value for the pixel being analyzed, $j = 1, 2, \ldots N$.
- $r_{ij} = 1 - 2\mathrm{e}^{-x_j t_i}$, the spin relaxation function evaluated at spin relaxation rate x_j and time t_i.
- We assume σ is unknown, and we use $\sigma = 1$.

Note that M, y_i, and g_i come from the MRI input data set, N and x_j are inputs specified by the user, r_{ij} are calculated from the inputs, and f_j are outputs.

4.4 Constrained Linear Regularization

Several methods can be used to solve Equation (14), including the method of normal equations and the method of singular value decomposition [14]. To get the absolute minimum least-squares solution, these methods are perfectly happy to pick any values for the elements of the solution vector $\mathbf{f}$, including positive or negative values.

However, for many inverse problems, the nature of the problem imposes **constraints** on the solution. For example, each element of the solution vector must lie within a certain range, or each element of the solution vector must equal or exceed a certain lower bound. When solving such problems using linear regularization, doing a **constrained linear regularization** that incorporates the constraints into the solution often yields much more accurate solutions than unconstrained linear regularization.

For the MRI spin relaxometry problem, the solution vector elements — the spin densities — are by definition nonnegative: $\rho(x) \geq 0$. We therefore seek a solution that minimizes Equation (14) while constraining the solution vector to be nonnegative — a **nonnegative least squares** solution.

Lawson and Hanson [9] have published an algorithm, NNLS, that finds a nonnegative least squares solution to a matrix equation like Equation (14). Lawson and Hanson have also provided a public-domain Fortran code for NNLS in the Netlib Repository [15]. We translated the NNLS Fortran code into Java and used it to do constrained linear regularization in our SRSolve program. Provencher's CONTIN program also uses the NNLS algorithm [3].

5 Picking the Tradeoff Parameter

The only issue left is to decide how to pick the appropriate value for the tradeoff parameter λ in the constrained linear regularization method. Since the value of λ has a profound effect on the inverse problem's solution (as will be shown below), picking the right value of λ is important.

5.1 Effect of λ on the Solution

To explore the effect of λ on the solution, we wrote a prototype program that created a known spin spectrum $\rho(x)$; generated the MRI input signal $S(t)$ corresponding to $\rho(x)$; added random noise to the input signal to simulate measurement error, the noise having a normal distribution with zero mean and a specified standard deviation σ; and performed a constrained linear regularization on the noisy input signal, using a specified λ, to calculate the output spin spectrum $\hat{\rho}(x)$. The program then plotted $\rho(x)$ and $\hat{\rho}(x)$ for comparison and displayed the root-mean-square error between the two.

Figures 10, 11, and 12 show the prototype program's spin spectrum plot for different values of λ. The original spin spectrum $\rho(x)$, with two peaks centered at $x = 0.5$ and $x = 1.5$, is plotted in red. The output spin spectrum $\hat{\rho}(x)$ is plotted in black. First we tried a wee, little λ of 1×10^{-6}, but the output was too spiky. Next we tried a great, big λ of 1×10^{0}, but the output was too smooth.

Figure 10: Original and output spin spectra — $\lambda = 1 \times 10^{-6}$

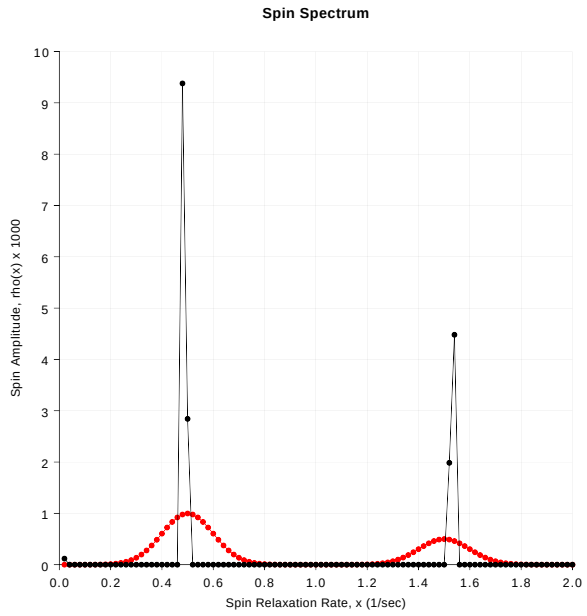
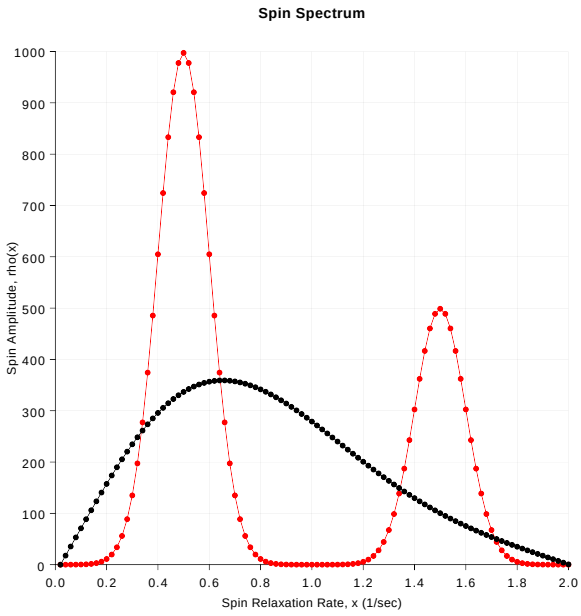


Figure 11: Original and output spin spectra — $\lambda = 1 \times 10^{0}$



Finally we tried a middle-sized λ of 3.63×10^{-3}, and then the output was just right. More precisely, this is the value of λ that gives the smallest root-mean-square error between $\rho(x)$ and $\hat{\rho}(x)$ for this test case. Note that, although it is close, $\hat{\rho}(x)$ does not precisely match $\rho(x)$. This is due to the noise (measurement error) in the input signal. With less noise (smaller σ), the inverse procedure recovers the original spin spectrum more accurately.

Figure 12: Original and output spin spectra — $\lambda = 3.63 \times 10^{-3}$

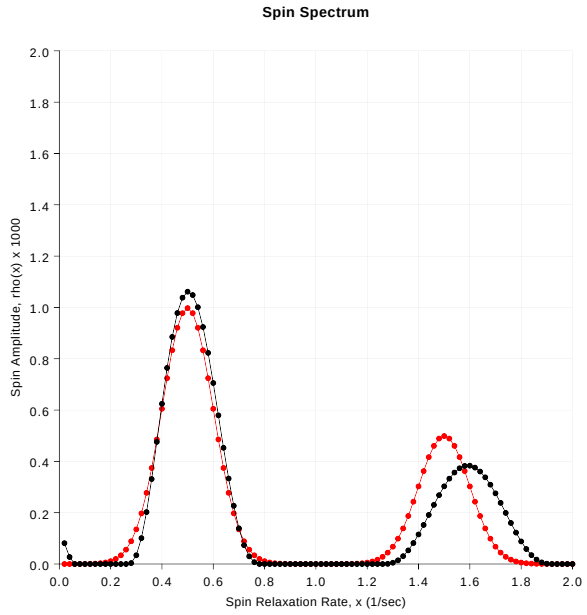
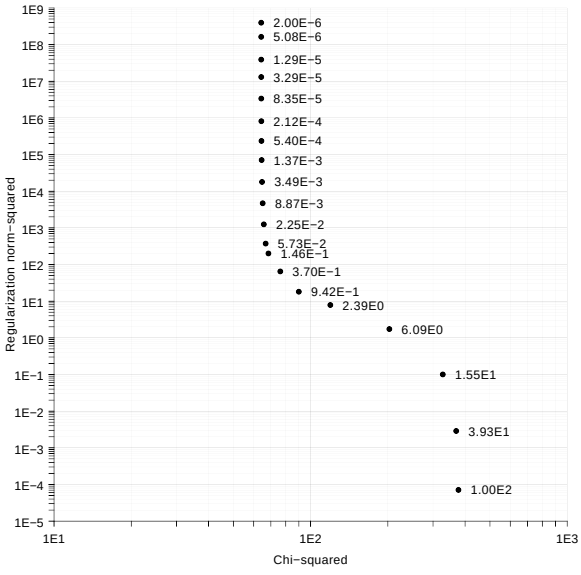


Figure 13: L-curve



5.2 Picking λ Using the L-Curve

It is easy to choose an appropriate value of λ when we have the original spin spectrum for comparison. However, when solving a real MRI spin relaxometry problem the original spin spectrum is not known, and we need some other way of choosing λ.

One way is to use the so-called **L-curve** [10]: Make a log-log plot of the squared error in the regularization term ζ^2 (Equation (10)) versus the squared error in the solution term χ^2 (Equation (6)) for various values of λ. This is typically supposed to be an L-shaped curve: As λ increases from a small value, the curve starts out moving downward vertically, then bends over and moves to the right horizontally. Pick the value of λ that falls at the "knee" where the curve bends over. This point represents a balance between minimizing ζ^2 (as far down vertically as possible) and minimizing χ^2 (as far to the left horizontally as possible).

Figure 13 shows the L-curve for the same test case as Figures 10–12. The value of λ is shown next to each point. The plot illustrates the difficulties with using the L-curve to pick λ. The curve is not very L-shaped. While a human being could locate a sort of knee on the curve somewhere between $\lambda = 3.70 \times 10^{-1}$ and $\lambda = 9.42 \times 10^{-1}$, it is difficult to see how a computer program could find the knee of the curve automatically. Finally, the knee of the curve yields too high a value for λ. As we have seen, λ should be 3.63×10^{-3} to get the most accurate solution for this test case — two orders of magnitude smaller than λ at the knee of the curve. Because of these difficulties we decided not to pick λ using the L-curve in the SRSolve program.

5.3 Picking λ Using the χ^2 Statistic

Press et al. [14] recommend using the χ^2 statistic, Equation (6), to choose λ. Since χ^2 is based on M input values, each subject to random measurement errors, χ^2 is a random variable having the chi-squared distribution with M degrees of freedom. The chi-squared distribution with M degrees of freedom has a mean of M and a variance of $2M$. Therefore, the expected value of χ^2 is M.

If λ is chosen to be a very small value, the inverse problem solution tends to yield a reconstructed output function which closely follows all the random fluctuations in the actual output function; in other words, $\chi^2 \approx 0$. Yet we know χ^2 is "supposed" to be M. This indicates the solution is not smooth enough. As λ increases, the solution becomes smoother, the reconstructed output function matches the actual function less closely, and χ^2 increases. Pick the value of λ that makes $\chi^2 = M$.

However, two problems prevent picking λ this way for the MRI spin relaxometry analysis problem. First, χ^2 depends on the measurement error standard deviation σ, but we are assuming we don't know σ. Second, the reconstructed spin signal never closely follows the random fluctuations in the actual measured spin signal, even if λ is a very small value. This is due to the nature of the response kernel for this particular problem, namely the spin relaxation function (Equation (1)). The sum of several spin relaxation functions is always a smooth function that is not capable of following random fluctuations in the measured spin signal. Because of these difficulties we decided not to pick λ using the χ^2 statistic in the SRSolve program.

5.4 Picking λ Using the F Statistic

Provencher recommends still another method for choosing λ, which he uses in the CONTIN program [3], and which we adopted in the SRSolve program. This method is appropriate when σ is not known, as we are assuming.

First, solve the inverse problem using $\lambda = \lambda_0$, a very small value for the tradeoff parameter, such that the solution is effectively unregularized. We used $\lambda_0 = 1 \times 10^{-6}$. Let ξ_0^2 be the squared norm of the error for this unregularized solution. Assuming the measurement errors in the input signal are normally distributed with zero mean and (unknown) standard deviation σ, ξ_0^2 is a random variable having the chi-squared distribution with $R - K$ degrees of freedom, where R is the number of rows in the matrix being solved in Equation (14), $R = M + N - 2$, and K is the number of parameters found in the solution.

K measures the "information content" in the solution, that is, the number of "meaningful" solution parameter values the inverse procedure can find. The CONTIN program calculates K based on a singular value decomposition of a certain matrix calculated during the course of the program. Since the SRSolve program does not do this singular value decomposition, the SRSolve program instead calculates K by examining the output spin spectrum for $\lambda = \lambda_0$. The SRSolve program finds $\hat{\rho}_{MAX}$, the highest peak in the spin spectrum, counts the number of $\hat{\rho}$ values greater than or equal to 10% of $\hat{\rho}_{MAX}$, and uses this count as K. Setting a threshold of 10% of $\hat{\rho}_{MAX}$ eliminates spurious small nonzero $\hat{\rho}$ values from the count. For example, in Figure 10, $K = 4$; the inverse procedure was able to find only four meaningful $\hat{\rho}$ values, the other values being zero or close to zero.

Now, suppose we solve the inverse problem using some value of λ greater than λ_0. Let ξ^2 be the squared norm of the error for this solution. Consider the difference $\xi^2 - \xi_0^2$, which is a random variable having the chi-squared distribution with K degrees of freedom. The variance of the random variable ξ_0^2 characterizes the variation in the solution due to measurement errors in the input signal. The variance of the random variable $\xi^2 - \xi_0^2$ characterizes the variation in the solution due to increasing the tradeoff parameter from λ_0 to λ. Provencher's recommendation for choosing the tradeoff parameter is to pick λ such that the variation in the solution due to measurement errors in the input signal is the same as the variation in the solution due to increasing the tradeoff parameter.

To determine whether the variations are equal, we use the **F test**, a statistical test which tests whether two chi-squared random variables have the same variance. Since $\xi^2 - \xi_0^2$ is a random variable having the chi-squared distribution with K degrees of freedom and ξ_0^2 is a random variable having the chi-squared distribution with $R - K$ degrees of freedom, the statistic

$$F = \frac{(\xi^2 - \xi_0^2)/K}{\xi_0^2/(R-K)} = \frac{\xi^2 - \xi_0^2}{\xi_0^2} \cdot \frac{R-K}{K} \tag{15}$$

is a random variable having the F distribution with K and $R - K$ degrees of freedom. The significance of F is given by

$$Q = I\left(\frac{R-K}{2}, \frac{K}{2}, \frac{R-K}{R-K+KF}\right) \tag{16}$$

where $I(\cdot)$ is the incomplete beta function,

$$I(a,b,x) = \frac{\int_0^x t^{a-1}(1-t)^{b-1}\mathrm{d}t}{\int_0^\infty t^{a-1}(1-t)^{b-1}\mathrm{d}t} \tag{17}$$

Q is the probability that F is as large as it is by random chance if the null hypothesis, that the variances are the same, is true. As λ increases from λ_0, ξ^2 increases, F increases, and Q decreases from 1 towards 0. Provencher recommends picking the value of λ that makes $Q = 0.5$.

6 Spin Relaxometry Analysis Programs

We divided the MRI spin relaxometry analysis problem between two programs written in Java.

The SRSolve program, a noninteractive parallel program, reads an MRI input data set, calculates all the pixels' spin spectra, and stores the results in an output file or files. SRSolve uses mpiJava [11], a Java interface to the standard Message Passing Interface (MPI) middleware [12].

The SRView program, an interactive GUI-based program, reads the output files and displays the analysis results.

For purposes of speedup measurements we also wrote the SRSolveSeq program. This program does exactly the same thing as SRSolve, except SRSolveSeq is a sequential program with all parallel code omitted.

The next sections describe these programs as well as the output file format.

6.1 SRSolve

The command for running the SRSolve program is:

java SRSolve x_L x_U N λ_L λ_U L *times mask output description*

The command line arguments are:

- x_L, x_U, N — The program calculates the spin densities for N values of the spin relaxation rate x spaced linearly from x_L to x_U inclusive.
- λ_L, λ_U, L — The program calculates the solution for L values of the regularization tradeoff parameter λ spaced logarithmically from λ_L to λ_U inclusive.
- *times, mask* — The name of the times file and the name of the mask file of the MRI input data set.
- *output* — The name of the output file in which to store the analysis results.
- *description* — A description to be included in the output file.

For our testing with the MRI input data set of Section 2, we used $x_L = 0.02$, $x_U = 2.0$, $N = 100$, $\lambda_L =$ 1e–3, $\lambda_U =$ 1e2, and $L = 16$.

Like all MPI-based parallel programs, the SRSolve program is executed by K processes simultaneously, each process running on a different processor of a cluster parallel computer. Each process is assigned a different **rank** from 0 to $K - 1$.

Every process reads the times file and mask file to determine the names of the input image files and to determine which pixels are unmasked and need to be analyzed. Each process takes a different subset of the pixels to analyze; the pixels are divided equally among the processes. Each process reads the spin signal values for its subset of the pixels from the input image files. For every pixel assigned to the process and for every λ value specified on the command line, each process computes the solution $\hat{\rho}(x)$ using the constrained linear regularization (nonnegative least squares) algorithm as described in Section 4. The process also computes the F statistic and its significance Q for that solution as described in Section 5.4. Each process stores all its results in a file named "*output_rank*", where *output* was specified on the command line and *rank* is the process's rank. Thus, a typical SRSolve run yields a group of output files named "*output*_0", "*output*_1", "*output*_2", and so on.

Since there is no communication between processes during a run, the SRSolve program should exhibit near-ideal speedups as the number of processors increases. Also, since the program is performing essentially the same calculations for each pixel and the pixels are divided equally among the processes, the SRSolve program should exhibit good load balancing. Timing measurements for SRSolve are given in Section 7.

For the best performance, the SRSolve program's input and output files should be stored on the processors' local disks, rather than, say, a network file server. This means that the MRI input data set must be replicated onto each processor's local disk.

Note that the SRSolve program does not pick a specific value for λ, rather it computes solutions for a range of λ values specified on the command line. This allows the user to pick the desired value of λ (within the specified range) while examining the analysis results with the SRView program. The SRView program can also determine the recommended value of λ which makes $Q = 0.5$.

6.2 Output File Format

The contents of the MRI output file are as follows. All elements are written in binary format using java.io.DataOutput, which writes multi-byte quantities in big-endian order (most significant byte first). An `int` value occupies 4 bytes. A `float` value occupies 4 bytes.

- Description (UTF-8 string; from the SRSolve command line)
- Number of rows in the image, R (`int`)
- Number of columns in the image, C (`int`)
- Number of time values, M (`int`)
- Time values, t_i (M `float`s)
- Number of spin relaxation rate values, N (`int`)

- Spin relaxation rate values, x_j (N `floats`)
- Number of tradeoff parameter values, L (`int`)
- Tradeoff parameter values, λ_k (L `floats`)
- Number of pixels in this file, P (`int`)
- Pixel indexes[2] in this file (P `ints`)
- For each pixel:
  - Spin signal values, S_i (M `floats`)
  - For each tradeoff parameter value:
    - F statistic (`float`)
    - Significance of F statistic, Q (`float`)
    - Spin density values, ρ_j (N `floats`)

For the MRI test data set of Section 2, with $M = 64$, $N = 100$, $L = 16$, and $P = 2{,}597$, the output file occupies 16.8 megabytes.

6.3 SRView

The command for running the SRView program is:

java SRView *outputfile* [*outputfile* . . .]

The command line arguments are the names of the output files (one or more) generated by one run of the SRSolve program.

The SRView program can display three different views of the analysis results. Initially, it displays the spin signal view (Figure 14). On the left is an image of one of the MRI input files. The Zoom control below the image sets the image magnification factor from 1 to 8. A certain pixel in the image is selected, as specified by the Pixel Row and Column controls below the image. The selected pixel is highlighted with a purple border. Clicking the left mouse button on a pixel in the image also selects that pixel. Clicking the right mouse button on the image displays a popup menu for saving the image in a PNG file.

On the right is a plot of the spin signal for the selected pixel. The input data points are plotted in red. The reconstructed spin signal, corresponding to the spin spectrum solution calculated by the SRSolve program, is plotted in black. Below the spin signal plot is a slider for selecting a time value. The selected time value is shown as a red vertical line on the plot. The selected time value determines which MRI input file is displayed in the image on the left. Clicking the right mouse button on the plot displays a popup menu for saving the plot in a PNG file or a PostScript file.

[2]The pixel index for the pixel at row r, column c is $C \cdot r + c$.

Figure 14: SRView program — spin signal view

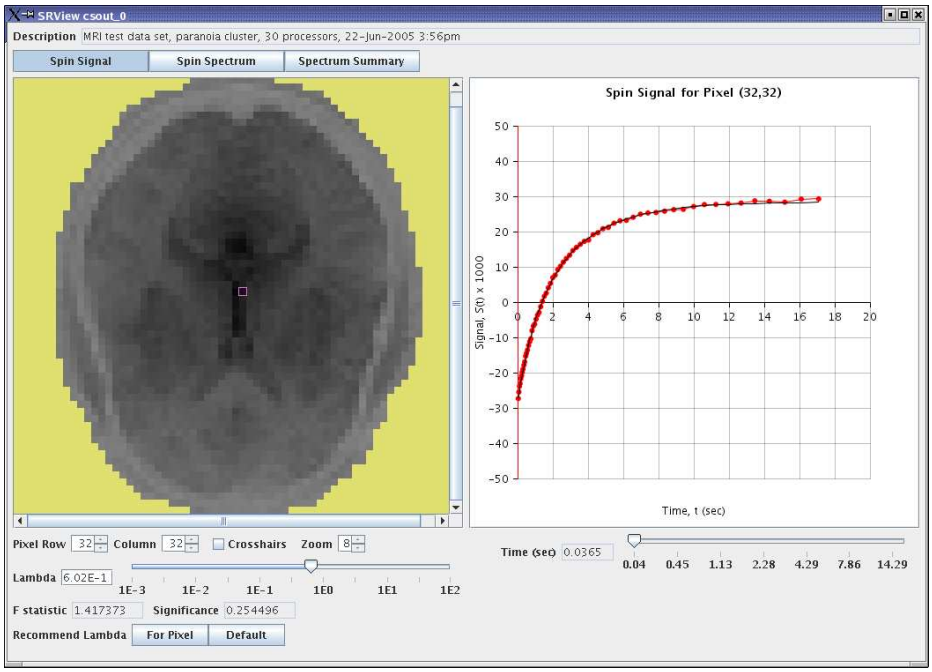
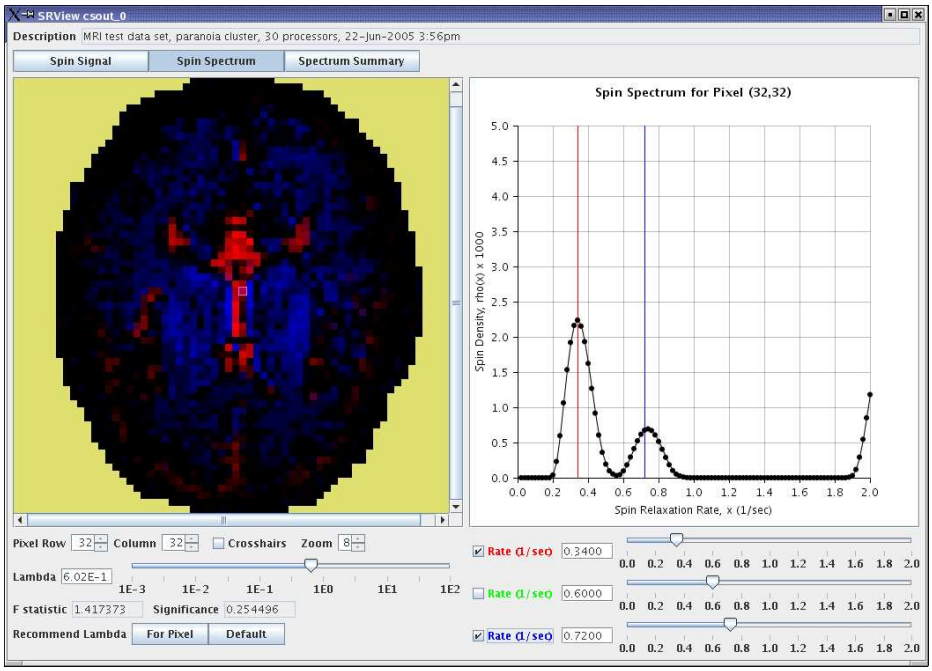


Figure 15: SRView program — spin spectrum view



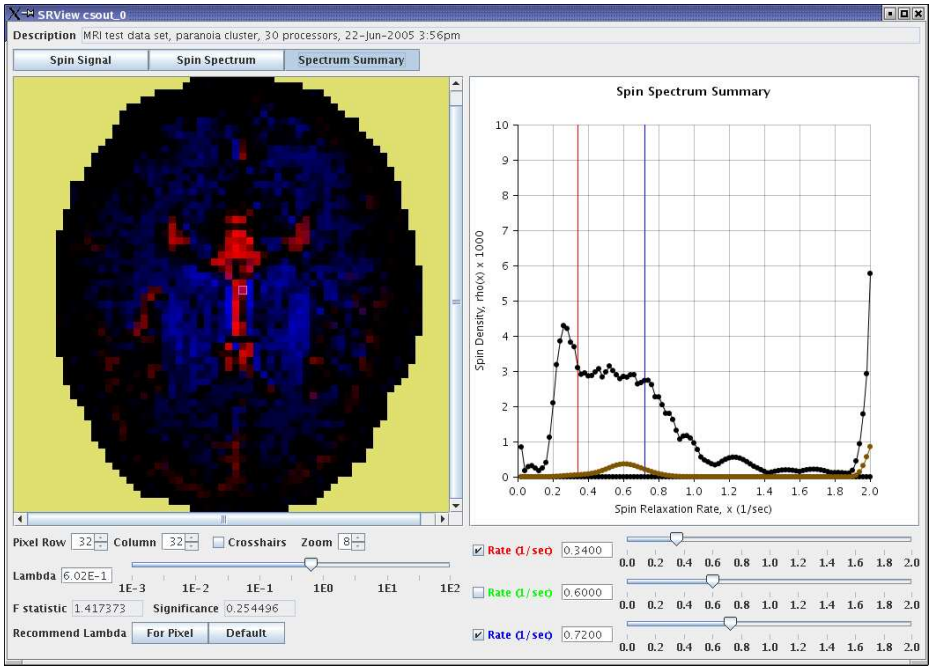
Clicking the Spin Spectrum button at the top of the window switches to the spin spectrum view (Figure 15). On the right is a plot of the calculated spin spectrum for the selected pixel. Below the plot are three sliders for choosing three spin relaxation rate markers, the red rate marker, the green rate marker, and the blue rate marker. A spin relaxation rate value may also be typed directly into one of the text boxes. A checkbox turns each marker on and off.

When a marker is turned on, the marker's spin relaxation rate value is shown as a vertical line on the plot in the corresponding color. Also, each pixel in the image on the left depicts the spin density value corresponding to the marker's spin relaxation rate value, with higher spin densities resulting in brighter colors. Thus, one can instantly see in the image which pixels have high spin densities at a certain spin relaxation rate. Up to three different spin relaxation rates can be depicted simultaneously using the three colored markers.

In the lower left part of the window are controls for selecting the regularization tradeoff parameter. λ may be chosen manually by moving the slider or typing a value into the text box. The F statistic of the solution for the selected pixel for the chosen value of λ, along with its significance Q, are also displayed. Clicking the Recommended Lambda — For Pixel button sets λ to be the recommended value for the selected pixel, namely the value that makes $Q = 0.5$. Clicking the Recommended Lambda — Default button sets λ to be the median of the recommended values for all the pixels.

Clicking the Spectrum Summary button at the top of the window switches to the spectrum summary view (Figure 16). This is similar to the spin spectrum view, except the plot shows a summary for all the pixels instead of just the selected pixel. The uppermost curve on the plot (in black) gives the maximum spin density value over all the pixels versus the spin relaxation rate. The middle curve on the plot (in brown) gives the average spin density value over all the pixels versus the spin relaxation rate. The lowermost curve on the plot (in black) gives the minimum spin density value over all the pixels versus the spin relaxation rate. Peaks in the maximum spin density curve show spin relaxation rate values for which at least one pixel has a high spin density; to discover which pixels these are, position a marker on the peak and examine the image. Peaks in the average spin density curve show spin relaxation rate values for which many pixels have a high spin density.

Figure 16: SRView program — spectrum summary view



The SRView program calculates the values for the spin spectrum plots and the intensities for the spin spectrum images by interpolating in the tables of data stored in the SRSolve output files. To calculate a spin density value ρ as a function of spin relaxation rate x and tradeoff parameter λ, SRView uses bilinear interpolation, with a linear scale for x and a logarithmic scale for λ. Similarly, SRView calculates the recommended λ value for a pixel using inverse interpolation in the table of Q versus λ values to $Q = 0.5$, again using a logarithmic scale for λ.

6.4 Sample Results

Figure 17 shows the calculated spin spectrum, and Figure 18 shows the original spin signal and reconstructed spin signal, for pixel (32,32) in the test image of Section 2. These are plotted using the pixel's recommended λ value of 2.23×10^{-1}.

Figure 17: Spin spectrum for pixel (32,32)

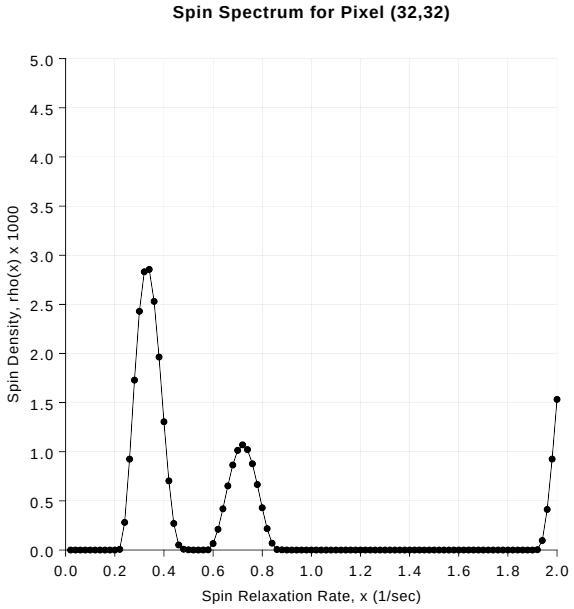


Figure 18: Spin signals for pixel (32,32)

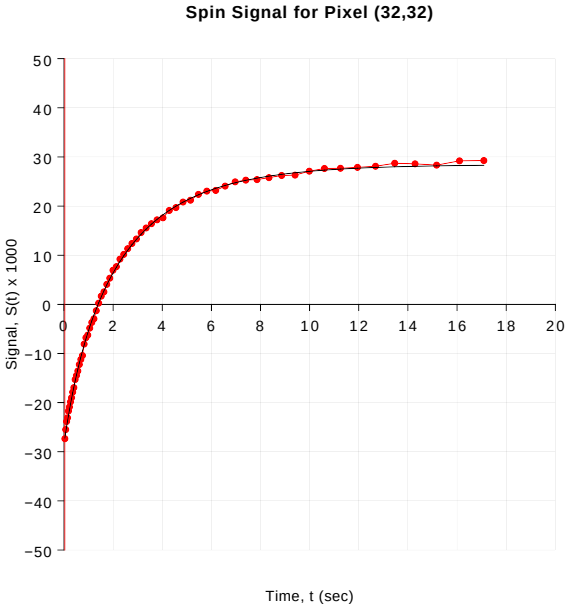


Figure 19 shows the calculated spin spectrum, and Figure 20 shows the original spin signal and reconstructed spin signal, for pixel (32,20). These are plotted using the pixel's recommended λ value of 3.98×10^{-1}.

Figure 21 shows the calculated spin spectrum, and Figure 22 shows the original spin signal and reconstructed spin signal, for pixel (47,13) — one of the poorer-quality signals. These are plotted using the pixel's recommended λ value of 3.04×10^{0}.

Figure 19: Spin spectrum for pixel (32,20)

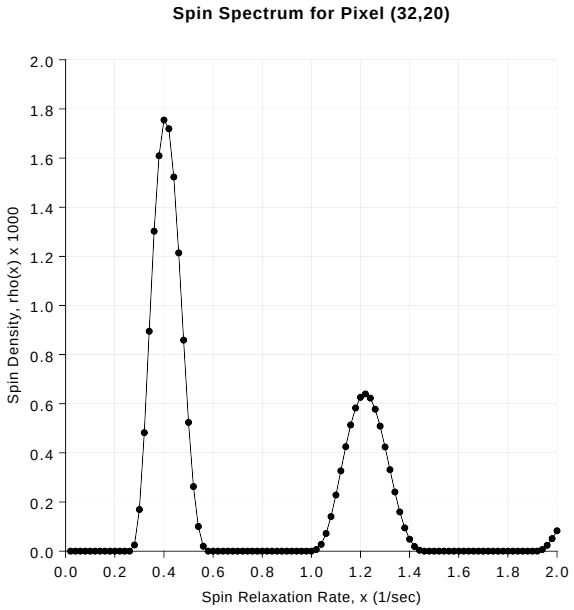


Figure 20: Spin signals for pixel (32,20)

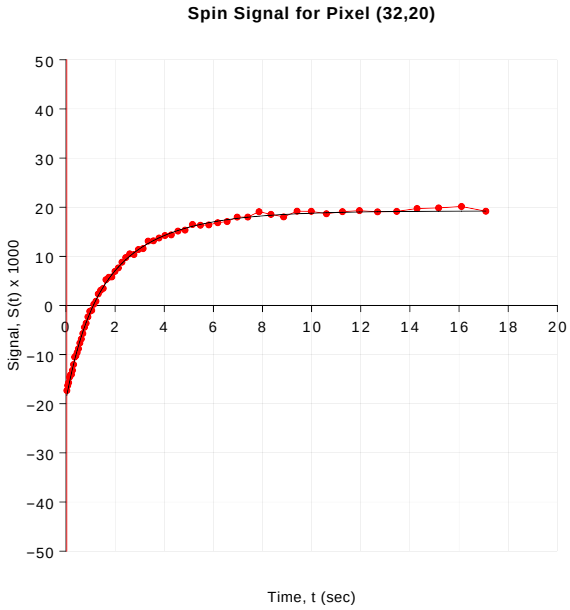


Figure 21: Spin spectrum for pixel (47,13)

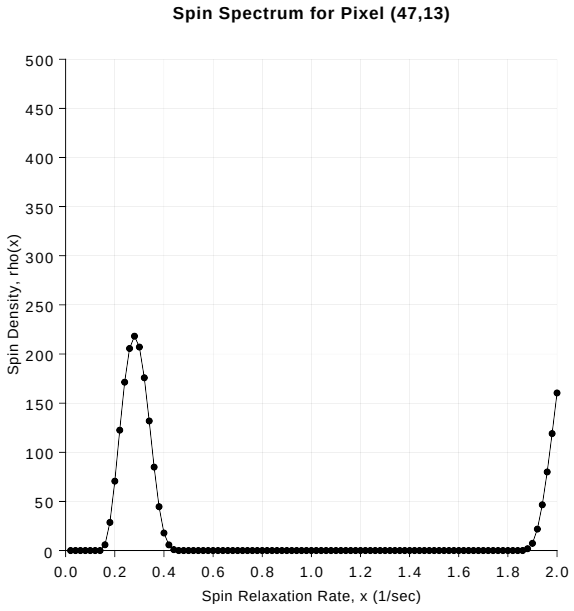
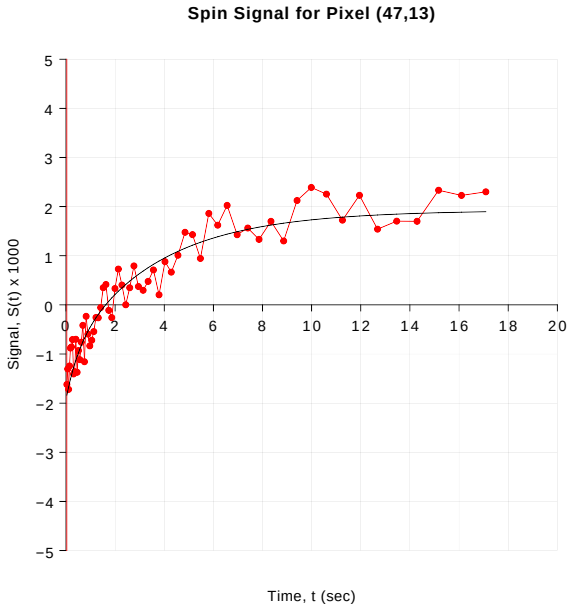


Figure 22: Spin signals for pixel (47,13)



7 Running Time Measurements

We measured the SRSolve program's running time on a 32-processor cluster parallel computer using the MRI input data set of Section 2 with this command line:

java SRSolve 0.02 2.0 100 1e-3 1e2 16 *times mask output description*

This is the same cluster parallel computer that Bak, Hornak, and Schaller [1] used to measure the CONTIN program's running time. Each processor in the cluster was a Sun Microsystems 440 MHz UltraSPARC-IIi CPU with 256 MB of main memory. We measured the parallel SRSolve program's running time when running on K = 1, 2, 3, 4, 5, 10, 15, 20, 25, and 30 processors.[3] As a baseline for speedup and efficiency, we also measured the sequential SRSolveSeq program's running time on one processor. We repeated each measurement five times, recording the shortest time taken by any process and the longest time taken by any process in the program run. Table 1 lists the average of the five longest-process running times for each value of K, along with the speedup and efficiency relative to the sequential program's running time. (Appendix A has the complete list of measurements.)

Table 1: SRSolve running time

K	Time (sec)	Speedup	Efficiency
seq	6686		
1	7197	0.929	0.929
2	3624	1.84	0.922
3	2514	2.66	0.887
4	1879	3.56	0.890
5	1641	4.07	0.815
10	759	8.81	0.881
15	508	13.2	0.877
20	389	17.2	0.859
25	311	21.5	0.860
30	263	25.4	0.847

Figures 23 and 24 plot the speedup and efficiency versus number of processors K. SRSolve's performance was within 18.5% of the ideal as the number of processors scaled up from 1 to 30. For every program run, the shortest running time for any process was within 2.1% of the largest running time for any process, evincing good load balance.

Out of curiosity, we also measured the sequential SRSolveSeq program's running time on Prof. Kaminsky's laptop computer, an Intel 1.6 GHz Pentium CPU with 512 MB of main memory. The average of five program runs was 662 sec.

[3] We did not go all the way to K = 32 processors because one processor was down while we were doing our measurements.

Figure 23: SRSolve speedup versus K

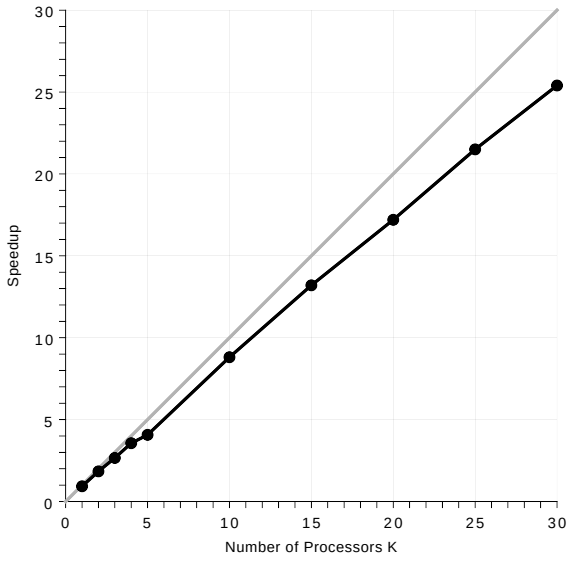
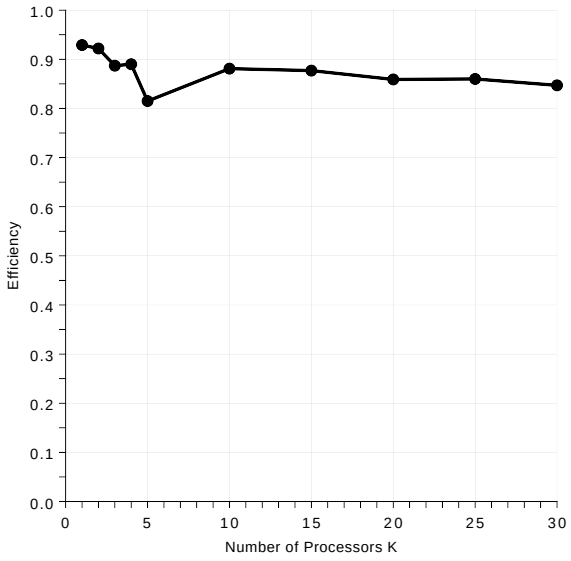


Figure 24: SRSolve efficiency versus K



8 Conclusion

We implemented SRSolve, a Java program for spin relaxometry analysis of MRI images designed to run on a cluster parallel computer. We measured SRSolve's running time and compared it to the previously reported running time for the CONTIN program. We also implemented SRView, a Java program for examining SRSolve's results.

Comparing timing measurements, the average running time per pixel was:

- CONTIN, cluster parallel computer, 32 processors — 44.4 sec
- SRSolve, cluster parallel computer, 30 processors — 3.04 sec
- SRSolveSeq, cluster parallel computer, 1 processor — 2.57 sec
- SRSolveSeq, laptop computer, 1 processor — 0.255 sec

Thus, the laptop computer was about 10 times faster than each processor of the cluster parallel computer. The very latest CPUs are about twice as fast as the laptop computer. Thus, the SRSolve program running on a cluster parallel computer with the latest CPUs might be expected to require 3.04 sec $\div$ 20 = 0.152 sec per pixel. A spin relaxometry analysis of a 512×512-pixel image, with 64 time values and the same proportion of unmasked pixels as the test image, solving for 100 spin relaxation rate values and 16 tradeoff parameter values as in our tests, might be expected to require 789 seconds (13.2 minutes) on 32 processors, or 395 seconds (6.6 minutes) on 64 processors. Of course, solving for fewer spin relaxation rate values or fewer tradeoff parameter values would reduce the running time.

Future research directions include measuring SRSolve's performance on larger images; measuring SRSolve's performance on cluster parallel computers with more and faster processors; and reimplementing SRSolve using shared memory parallel programming techniques

in addition to message passing parallel programming techniques, to take advantage of cluster parallel computers with hyperthreaded or multicore processors and reduce the running time still further.

An electronic version of this technical report, a Java Archive (JAR) file with the source code and documentation for the Java programs, and the test MRI input data set are available at `http://www.cs.rit.edu/~ark/sr/`.

References

[1] Andrew P. Bak, Joseph P. Hornak, and Nan C. Schaller. From impractical to practical: Solving an MRI problem using parallelism. In *Rochester Institute of Technology B. Thomas Golisano College of Computing and Information Sciences 2005 Conference on Computing and Information Sciences*, January 2005. `https://ritdml.rit.edu/dspace/bitstream/1850/423/1/V6+ExtendedAbstract-BakHornakSchaller.pdf`. Retrieved May 12, 2005.

[2] L. M. Fletcher, J. B. Barsotti, and J. P. Hornak. A multispectral analysis of brain tissues. *Magnetic Resonance in Medicine*, 29:623–630, 1993.

[3] Stephen W. Provencher. A constrained regularization method for inverting data represented by linear algebraic or integral equations. *Computer Physics Communications*, 27:213–227, 1982.

[4] Stephen W. Provencher. CONTIN: A general purpose constrained regularization program for inverting noisy linear algebraic and integral equations. *Computer Physics Communications*, 27:229–242, 1982.

[5] G. E. Backus and F. Gilbert. The resolving power of gross earth data. *Geophysical Journal of the Royal Astronomical Society*, 16:169–205, 1968.

[6] G. E. Backus and F. Gilbert. Uniqueness in the inversion of inaccurate gross earth data. *Philosophical Transactions of the Royal Society of London A*, 266:123–192, 1970.

[7] R. L. Parker. Understanding inverse theory. *Annual Review of Earth and Planetary Science*, 5:35–64, 1977.

[8] T. J. Loredo and R. I. Epstein. Analyzing gamma-ray burst spectral data. *Astrophysical Journal*, 336:896–919, 1989.

[9] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.

[10] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier Academic Press, 2005.

[11] mpiJava Home Page. `http://aspen.ucs.indiana.edu/pss/HPJava/mpiJava.html`, May 2000. Retrieved May 12, 2005.

[12] The Message Passing Interface (MPI) Standard. `http://www-unix.mcs.anl.gov/mpi/`. Retrieved May 12, 2005.

[13] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, 1992.

[15] Netlib Repository. `http://www.netlib.org/`. Retrieved May 25, 2005.

A Running Time Measurements

The first block of measurements are for the SRSolveSeq program running on one processor. The other blocks of measurements are for the SRSolve program running on K processors.

	Time (msec)			Time (msec)	
K	Minimum	Maximum	K	Minimum	Maximum
seq	6688890		10	726467	726859
	6690417			766738	766975
	6680008			771247	771402
	6722614			769422	769897
	6686025			758925	759255
1	7296683		15	510147	517034
	7276342			508653	511962
	7058111			488214	489043
	7263911			509214	511299
	7092146			508908	509655
2	3678937	3679353	20	385372	386554
	3435312	3435611		385542	386373
	3669116	3669822		384913	386710
	3670622	3671337		386920	388093
	3667815	3667889		386716	395094
3	2376117	2376348	25	314681	321467
	2552599	2552821		315315	316218
	2559185	2559945		315425	317828
	2537814	2537840		298492	299812
	2543983	2544378		297783	299352
4	1811779	1811970	30	251262	253203
	1893643	1893920		268941	270224
	1901654	1902181		268337	269157
	1896436	1896936		251033	253369
	1891083	1891265		268707	269707
5	1459988	1461046			
	1462753	1464123			
	1747170	1748055			
	1759096	1759184			
	1771496	1771685			