Physical Limits of Intrabody Signalling

by

Matthew Gray

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1997

Author . . . .
Department of Physics
May 16, 1997

Certified by . . . .
Neil Gershenfeld
Associate Professor
Thesis Supervisor

Accepted by . . . .
June Matthews
Thesis Coordinator

Physical Limits of Intrabody Signalling

by

Matthew Gray

Submitted to the Department of Physics
on May 16, 1997, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Physics

Abstract

The Personal Area Network (PAN) is a wireless system for signalling through the body. In order to make PAN more useful, it is valuable to know how much data can be realistically tranmitted through a PAN channel. The channel capacity of PAN (Personal Area Network) is highly dependent on geometry and quality of amplification. The geometry is modelled as a set of equipotentials, with their capacitive couplings solved for based on the particular geometry. The noise present in PAN is dominated by amplifier noise and external interference. Capacitance noise is negligible. This paper fully implements and explores models to calculate the signal strength as a function of geometry and to estimate noise strength and sources. Resultant calculations of channel capacities are done and directions for improvement are identified.

Thesis Supervisor: Neil Gershenfeld
Title: Associate Professor

Physical Limits of Intrabody Signalling

by

Matthew Gray

Submitted to the Department of Physics
on May 16, 1997, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Physics

Abstract

The Personal Area Network (PAN) is a wireless system for signalling through the body. In order to make PAN more useful, it is valuable to know how much data can be realistically tranmitted through a PAN channel. The channel capacity of PAN (Personal Area Network) is highly dependent on geometry and quality of amplification. The geometry is modelled as a set of equipotentials, with their capacitive couplings solved for based on the particular geometry. The noise present in PAN is dominated by amplifier noise and external interference. Capacitance noise is negligible. This paper fully implements and explores models to calculate the signal strength as a function of geometry and to estimate noise strength and sources. Resultant calculations of channel capacities are done and directions for improvement are identified.

Thesis Supervisor: Neil Gershenfeld
Title: Associate Professor

Acknowledgments

Without the support of my mother, father, Carrie, and all my friends, this would not have happened.

I'm also grateful to Neil and all the students in the Physics and Media Group for making giving me the opportunity to learn so much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis primarily deals with calculation of a channel capacity for the human body in the context of the PAN system as developed by Tom Zimmerman and Neil Gershenfeld. A brief description of the system will be provided here, and a more thorough discussion is in [5] and [4].

PAN, or Personal Area Network, is a system for data transmission through the human body. Each of the transmitter and receiver is configured with two electrodes. One of these electrodes is in a geometry which is more strongly capacitively coupled to the body than to the ground. The other electrode is less strongly coupled to the body and more so to absolute (room) ground. In all points of a PAN transmission, capacitive coupling dominates. In experimental work, typical transmission voltages were fives or tens of volts.

Two typical geometries for PAN are shown in Figure 1-1. In **A**, the first receiver electrode is nearly in contact with the bottom of the foot and the second electrode is nearly in contact with the ground. This provides the desired symmetry breaking about as ideally as possible for the human body. In **B**, the first receiver electrode is in near contact with the body and the second electrode is somewhat further from he body, and hence less well coupled to the body, and with a slightly stronger ground coupling. This configuration is less than ideal (as will be shown in more mathematical rigor later), but is a usable arrangement. For both configurations shown, the transmitter is beneath the foot, as for the receiver in **A**.

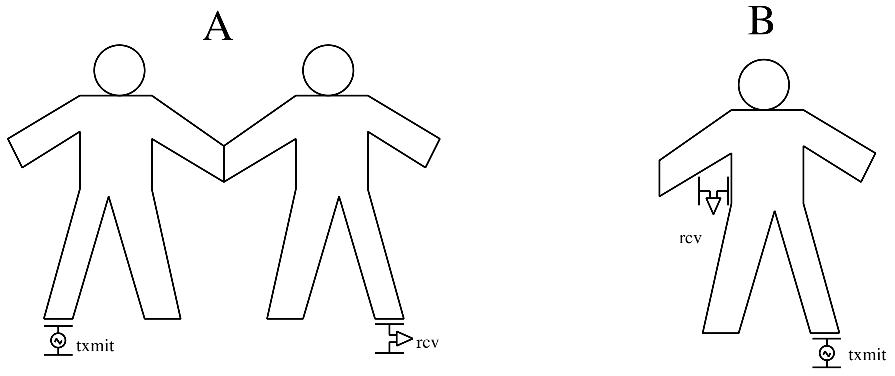

Figure 1-1: Typical PAN Geometries

1.1 Motivation for the PAN system

The PAN system is motivated in the same way as many wireless technologies: It aims to enable communication of a sort otherwise not possible. PAN has a number of advantages, disadvantages and synergies when compared to traditional wireless, such as radio-frequency (RF) or infra-red (IR).

PAN's advantages and disadvantages primarily stem from the fact that it is not a radiative form of communication. Due to this, PAN transmissions are somewhat private. This makes it very poor for long distance communications, but means transmissions between devices on the body will remain private to the body, making eavesdropping substantially more difficult. This also resolves many channel sharing issues, as the number of communicating devices on a body is going to be limited to the dozens at most. This makes it much easier to do channel allocation, as contrasted with a room size, building size, city size or nationwide system which may have to deal with many dozens, hundreds, thousands, or millions of devices. Additionally, PAN can be used in a fairly traditional network model in cooperation with other wireless strategies. By having all devices on the body capable of transmitting via PAN, with individual "router" devices which can retransmit messages off the body via some other mechanism if need be.

1.2 Experimental and Apparatus History

The original PAN prototype was developed by Tom Zimmerman, used on-off-keying (OOK), and ran at 300 baud. A transmitter and receiver were built, and each was a one-off single task device.

A second design and protoboard implementation was done by Chris Turner.[3] This design also used OOK, however each device was a transceiver and the devices ran at 2400 baud. The design was based on a series of gain stages, a pair of quadrature integrators followed by digital demodulation in a PIC microprocessor.

Further development of a modified version of Turner's design was carried out by the author. Both OOK and differential binary phase shift keying (BPSK) were used for encoding, with limited success. This design was built on PCBs, however the design ultimately failed due to noise rejection problems. Attempts to resolve this problem through use of an analog gain control (AGC) were made, but the AGC was was also abandoned in favor of the final design.

A final hardware design by Matthew Reynolds, employing a phase-locked-loop (PLL) for analog demodulation was implemented using a frequency shift keying (FSK) encoding scheme. This design, as of May, 1997, has shown the most promise, and has proven reliable for speeds up to 9600 baud in certain geometries.

Chapter 2

Modeling

The primary goal of this thesis is to provide a detailed model for the channel capacity of a PAN channel. This analysis will provide no indication as to what encoding scheme will best achieve the predicted maximum capacity. The experimental work described briefly in section 1.2 addresses some of this, but it seems likely that there are encoding schemes which will be able to more effectively approach the theoretical limit.

In order to evaluate the channel capacity of the body (which is taken to be a Gaussian channel), it is necessary to evaluate:

$$C = \Delta f \log_2 \left(1 + \frac{S}{N}\right) = \Delta f \log_2 \left(1 + \frac{S}{N_0 \Delta f}\right) \frac{\text{bits}}{\text{second}} \tag{2.1}$$

In this equation, C is the channel capacity, Δf is the bandwidth (and the sampling period is $\frac{1}{2\Delta f}$), S is the signal power, N is the noise power, and N_0 is the noise power spectral density.

In order to calculate S, it is necessary to calculate the signal strength at the receiver. This will be highly dependent on geometry and transmission strength. In order to calculate N, a noise estimate must be made, which will be dependent on geometry but is expected to be dominated by amplification noise in the receiver.

The signal calculation will be accomplished by an electrostatic model of the body or bodies with the transmitter, receiver and ground, and a circuit model of the interconnected capacitances. The noise calculation will be done from first principles for the noise in the body and empirical noise estimates for the receiver amplification.

Finally, given these values it will be possible to obtain a reasonable estimate for the channel capacity of the body given practical amplification, as well as an estimate for the absolute upper bound, given ideal amplification.

2.1 Electrostatic Physics Model

In the PAN system, the body is modeled as an equipotential, which has an induced charge caused by the potential on the transmitter electrode. This charge on the body is measurable by the receiver electrodes.

To effectively model this charge for an arbitrary geometry, it is necessary to numerically solve Poisson's equation:

$$\nabla^2 \phi = \frac{\rho}{\epsilon} \tag{2.2}$$

This is accomplished by successive over-relaxation (SOR) of the three-dimensional finite difference approximation of the diffusion equation, of which Poisson's equation is a steady-state solution. Doing this calculation gives the full values of the potential in the vicinity of the objects in question. Given ϕ and the definition of electric field:

$$\vec{E} = -\nabla \phi \tag{2.3}$$

it is possible to calculate the charge on each of the objects

$$\int_s \vec{E} \cdot d\vec{A} = \frac{q}{\epsilon} \tag{2.4}$$

and from that, and the known potential between each pair of objects ($V = \phi_1 - \phi_2$), the capacitance can be calculated:

$$C = \frac{q}{V} \tag{2.5}$$

More details of the electrostatic model used can be found in Chapter 3.

2.1.1 Dipole/Multipole Approximation

It is also possible to model the electrostatics as a dipole. That is, assuming strong coupling between the body transmit electrode and the body, and strong coupling between the ground transmit electrode and the ground, allows the system to be modeled as a charged body with a reflected, and inversely charged, body image in the ground.

Given this dipole, potentials at various positions in the vicinity of the body can be calculated. See Chapter 6 for some comparisons of the dipole approximation results with full SOR calculations.

A full multipole expansion can also be done to better approximate the fields in the vicinity of the body. This calculation is not done in this thesis.

2.2 Circuit Model

Given a capacitively fully interconnected network of six nodes, the circuit is best solved by a Gauss-Jordan matrix solver. The current node equations are used.

As a linear network, all six equations are unnecessary, as any five equations fully specify the sixth. One node must be arbitrarily set to be the reference voltage, 0 for convenience. Additionally, an arbitrary current is run across the transmitter. The linearity of the system can be exploited to solve the system given any transmitter voltage.

2.3 Noise Model

The entry of noise into the intrabody signalling system comes from three sources:

- Interference. Environmental noise, both background, other unintentionally interfering devices, and other intrabody signalling devices all produce a highly variable amount of noise.

- Capacitive, in body noise, as contributed by the actual transmission of the data through the body. Each equivalent capacitor contributes $\frac{1}{2}kT$ noise power per quadratic degree of freedom.

- Amplification noise. Whatever amplification system is used in the receiver affects the total noise in the channel.

Amplification noise is expected to dominate, though clearly external interference has potential to be significant, depending substantially on the encoding scheme selected.

Chapter 3

Electrostatic Physics Model

The goal of the electrostatic physics model is to calculate the capacitances between each of the six nodes: the body, the ground, the two transmitter electrodes and the two receiver electrodes, as shown in Figure 3-1.

All nodes are treated as equipotentials. This is obviously the case for the 4 electrodes. The body's internal impedance is on the order of 10s of ohms, which makes it effectively equipotential. The ground, depending on the specific conditions, will have an internal impedance as well, but remains approximately equipotential.

3.1 Successive Over-Relaxation (SOR)

In order to to numerically solve Poisson's equation (Equation 2.2), it is necessary to notice that its steady state solution is the same as that for diffusion, and can be solved by iteration of the 3D successive over-relaxation equation for a field without free charge:

$$\phi^{n+1}_{j,k,l} = (1-\alpha)\phi^{n}_{j,k,l} + \frac{\alpha}{6}(\phi^{n}_{j+1,k,l} + \phi^{n}_{j,k+1,l} + \phi^{n}_{j,k,l+1} + \phi^{n}_{j-1,k,l} + \phi^{n}_{j,k-1,l} + \phi^{n}_{j,k,l-1}) \quad (3.1)$$

Here α is a relaxation rate not to exceed 2; it is best determined empirically.

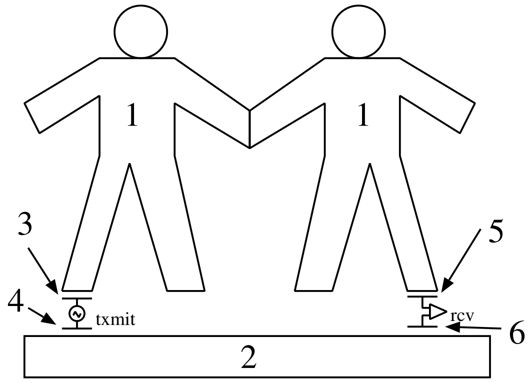

Figure 3-1: Node numberings

3.2 Geometry

While an arbitrarily detailed model of electrodes and a person or persons could be used, for this thesis, people will be modeled as cylinders, spheres, rectangular prisms, or combinations thereof.

A set of configurations, representing expected practical geometries were simulated.

3.3 Capacitance calculation

To determine the capacitance between any two nodes in the system, a different arbitrary potential is assigned to each of the two nodes (ϕ_1 and ϕ_2). Equation 3.1 is then applied until the changes become small.

Given the potentials $\phi_{j,k,l}$, the resultant charge implied can be calculated using the finite difference approximation of Equation 2.4:

$$\frac{q}{\epsilon} = \sum_{\text{node}} 6\phi_{j,k,l} - \phi_{j+1,k,l} - \phi_{j,k+1,l} - \phi_{j,k,l+1} - \phi_{j,k,l-1} - \phi_{j,k-1,l} - \phi_{j-1,k,l} \tag{3.2}$$

The charges for the pair of nodes in questions, q_1 and q_2, with the assigned potentials can be used to arrive at an inter-node capacitance:

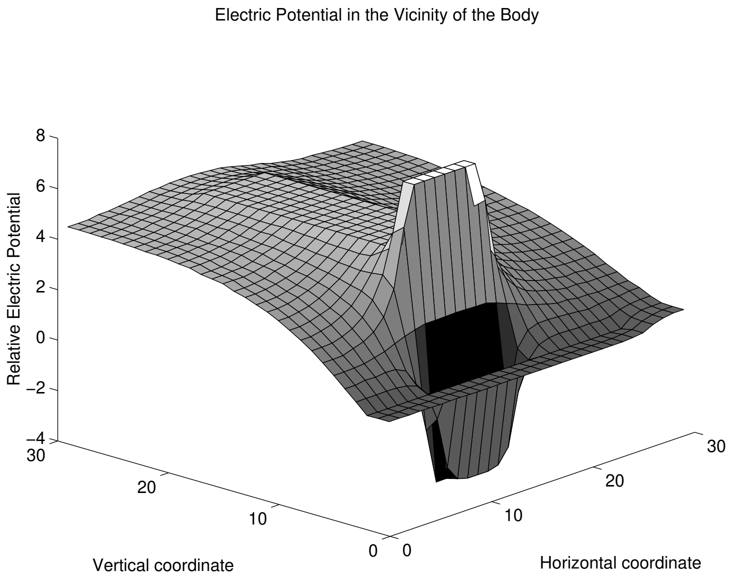

Figure 3-2: Electric Potential in the Vicinity of the Body with PAN Transmitter

$$C_{12} = \frac{q_2}{\phi_1} = \frac{q_1}{\phi_2} \tag{3.3}$$

3.4 Implementation

The implementation was done in C and can be found in Appendix A. The current version is somewhat more memory intensive than would be ideal. Each "point" is internally represented as a float, which is 4 bytes on most operating systems. The algorithms could likely be redone in integer math with substantial savings in both efficiency and memory usage, with some loss of precision.

In order to model systems on the scale of a person (2 meters), at a resolution down to typical electrode separations (centimeters), it is necessary to use a grid of greater than 200 points per axis, which yields a memory usage of over 32 megabytes.

A sample of the total resultant field appears in figure 3.4. The body is a vertical cylinder above the parallel transmitter plates, which are directly above a ground plate. In the figure, the transmitter potentials (still the largest features of the plot) have been truncated for clarity of visual display.

Chapter 4

Circuit Model

Once a set of capacitance values for the particular geometry are calculated, a circuit model is utilized to calculate the signal strengths at the receiver.

4.1 Circuit model

This fully interconnected network of nodes (as shown in Figure 4-1) is solved using node equations. At each node (except the transmitter pair) the total current must be zero. This yields the system of equations in Equation 4.2. All 6 node equations are unnecessary, so one is arbitrarily dropped (and hence it becomes the reference potential).

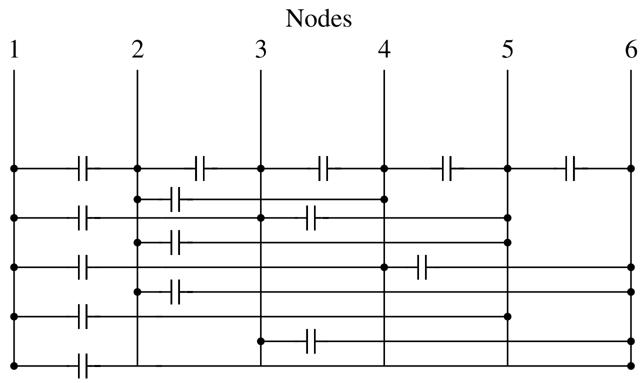

Figure 4-1: Circuit model of PAN using node numberings from Figure 3-1

In this case we are ignoring any resistive or inductive components of the network, but these can be reasonably assumed to not substantially impact the overall node-to-node impedances.

4.2 Calculation of Signal Strength

To solve this network, the following system of equations are solved simultaneously using a Gauss-Jordan solver, a modified version of one found in [2]. Z_{ab} is the impedance between the pair of nodes a and b.

$$\alpha_i = \sum_{y=0, y\neq i}^{6} -\frac{1}{Z_{iy}} \tag{4.1}$$

$$\begin{pmatrix} \alpha_1 & \frac{1}{Z_{12}} & \frac{1}{Z_{13}} & \frac{1}{Z_{14}} & \frac{1}{Z_{15}} \\ \frac{1}{Z_{12}} & \alpha_2 & \frac{1}{Z_{23}} & \frac{1}{Z_{24}} & \frac{1}{Z_{25}} \\ \frac{1}{Z_{13}} & \frac{1}{Z_{23}} & \alpha_3 & \frac{1}{Z_{34}} & \frac{1}{Z_{35}} \\ \frac{1}{Z_{14}} & \frac{1}{Z_{24}} & \frac{1}{Z_{34}} & \alpha_4 & \frac{1}{Z_{45}} \\ \frac{1}{Z_{15}} & \frac{1}{Z_{25}} & \frac{1}{Z_{35}} & \frac{1}{Z_{45}} & \alpha_5 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -I \\ +I \\ 0 \end{pmatrix} \tag{4.2}$$

Here, I is an arbitrary transmitter current. Node 6 is set arbitrarily to 0, and the other nodes will be relative to it.

The signal strength will be proportional to the square of the voltage difference between the receiving electrodes. The actual value calculated will be arbitrary, as the currents in the transmitter were chosen arbitrarily in Equation 4.2. Values for the voltage difference at the transmitter will have also been calculated, and due to the system being a linear network, the output voltage for any arbitrary input voltage can be calculated.

Chapter 5

Noise Model

As described in Section 2.3, the three sources of noise for an intrabody signalling system are external interference, thermal system noise, and amplification noise. The first of these is highly dependent on the particular environment in which it is used. Used on a body with multiple transmitters, the external interference would be drastically higher than an isolated receiver far from other radiative systems. The thermal and amplification noise are easier to quantitatively characterize.

5.1 Interference

Estimating interference magnitudes is necessarily an approximate task. The possibility of an intentionally interfering source yields possible noise values with essentially no upper limit, so this discussion will focus on non-intentional sources of interference.

Transmitters other than the one trying to be listened to are an obvious source of interference. The magnitude of this noise could be expected to be roughly equal to the intended received signal strength.

Other electronic devices are a likely source of interference as well. Typical values measured on an electrode near a body (a typical PAN configuration) in the lab (admittedly a noisy electrical environment) in general did not exceed 50 $\frac{\text{nV}}{\sqrt{Hz}}$.

5.2 Thermal Noise

Each effective capacitance has associated with it an effective noise current (i_n) in parallel with a power of $\frac{1}{2}kT$. Each of these noise currents is zero centered and has a root mean square amplitude dependent on the actual capacitance. Equation 4.2 is then solved for each of the noise sources independently. Each of the noise currents across the pair of receiver electrodes is then RMS summed:

$$i_{n_{tot}} = \sqrt{\sum_j (i_{n_j})^2} \tag{5.1}$$

This final noise current can then be used with the receiver impedance to get an mean square voltage amplitude.

The resultant thermal noise is confirmed to be as expected, very small. A typical value for this mean square voltage is 10^{-18}V^2.

5.3 Amplification Noise

The source noise generated by a feedback amplifier, from [1], is a function of some noise parameters of the op-amp, and the values of the feedback network. In Equation 5.2, e_A^2 is the noise voltage squared per hertz, e_n is the voltage noise from the amplifier, and i_n is the current noise. e_A, for an inverting amplifier, is calculated as:

$$\begin{aligned} i_A^2 &= i_n^2 + \tfrac{4kT}{R_2} \\ e_A^2 &= e_n^2 + R_1^2 i_A^2 \end{aligned} \tag{5.2}$$

Using values for a typical low noise op-amp ($i_n = .01\text{pA}/\sqrt{\text{Hz}}, e_n = 30\text{nV}/\sqrt{\text{Hz}}$) at an inverting gain of 10, the total input referred noise is approximately 30 $\frac{\text{nV}}{\sqrt{\text{Hz}}}$. This represents roughly as much noise power per hertz as there is thermal noise total. In practice, a higher noise op-amp was used which had a total input referred noise voltage of approximately 150 $\frac{\text{nV}}{\sqrt{\text{Hz}}}$.

Chapter 6

Results

6.1 Fields Around the Body

6.1.1 Dipole Analysis

In the dipole simplification, the person is represented by a point of charge q, at a distance of h from the oppositely charged ground image. For large distances $d >> h$, the field falls off as d^3, as with any dipole. In the near field, however, the field of the dipole is better evaluated as the field of the two distinct charges. The receiver is considered to be a distance d from the body, where $d << h$. The two cases of the receiver being placed between the body an the ground image (Equation 6.1) and perpendicular to the dipole moment, near the body (Equation 6.2 are shown.

$$|\vec{E}| = q(\frac{1}{d^2} - \frac{1}{(h-d)^2}) \tag{6.1}$$

$$|\vec{E}| = q\sqrt{\frac{1}{d^4} + \frac{1}{s^4}} \tag{6.2}$$

In both cases, the receiver is optimally positioned as closely to the body charge as possible.

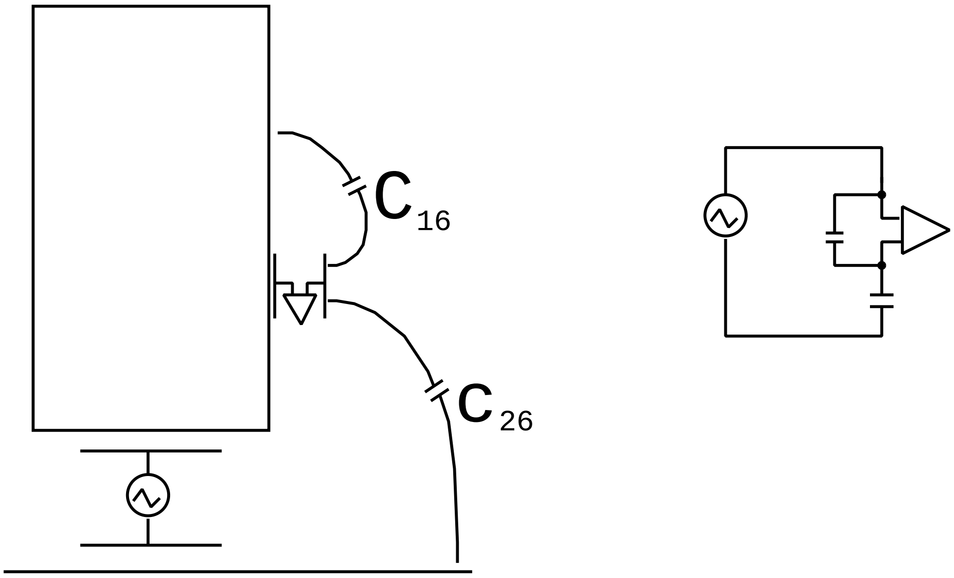

Figure 6-1: Simplified View of PAN

6.1.2 Full Simulation

A PAN receiver essentially measures the modulation of the electric field in the vicinity of the body. As the electric field lines are necessarily perpendicular to the surface of the body (a conductor), the best geometries place one electrode close to the body and the other the maximal reasonable distance away from the body perpendicular to the bodies surface.

The essential tradeoff for signal strength lies in the relative coupling of the ground electrode to the ground, and to the body. When the electrodes are directly between the body and ground, as in the receiver in configuration **A** in Figure 1-1, the ground electrode is better coupled to the body than it is to the ground. In geometries such as those in configuration **B**, however, the coupling of the ground electrode to ground (C_{26}) gets weaker and weaker as it is moved up the body, while the ground electrode to body coupling (C_{16}) stays nearly the same.

In the simplified version of PAN shown in Figure 6.1.2, the relationship between changes in height of the receiver and the values of C_{16} and C_{26}, the above mentioned capacitances, and V_r, the coefficient corresponding to the number of millivolts at the

Height	$V_r(\frac{\text{mV}}{\text{V}})$	C_{16} **(pF)**	C_{26} **(pF)**
10 cm	7.37	3.66	6.13
20 cm	7.72	3.99	5.45
30 cm	7.87	4.29	4.90
40 cm	7.87	4.58	4.44
50 cm	7.76	4.83	4.04
60 cm	7.58	5.06	3.70
70 cm	7.35	5.26	3.40
80 cm	7.11	5.43	3.13
90 cm	6.84	5.57	2.90
100 cm	6.59	5.66	2.69
110 cm	6.33	5.72	2.50
120 cm	6.10	5.73	2.32
130 cm	5.87	5.69	2.17
140 cm	5.66	5.59	2.02
150 cm	5.55	5.23	1.80
160 cm	5.43	4.84	1.58
170 cm	5.33	4.42	1.38

Table 6.1: Simulation results at various receiver heights

receiver per volt at the transmitter, are fundamental to understanding the geometry constraints of PAN.

The results of a full simulation with various receiver heights appear in Table 6.1.2. The simulation was done with a 170cm high, 40cm radius cylinder approximating a person. The transmit electrodes were beneath the person and the receive were beside them at various heights.

As seen in Figure 6.1.2, receiver voltage changes linearly with height. The curving of the line at the beginning and end of graph are due to edge effects near the persons feet and head. Specifically, near the feet, both receive electrodes become very highly ground coupled. This decreses the necassary break in symmetry at the receiver, and causes a decrease in signal strength.

A foot to foot configuration is far better, with a value of V_r on the order of $50\frac{\text{mV}}{\text{V}}$.

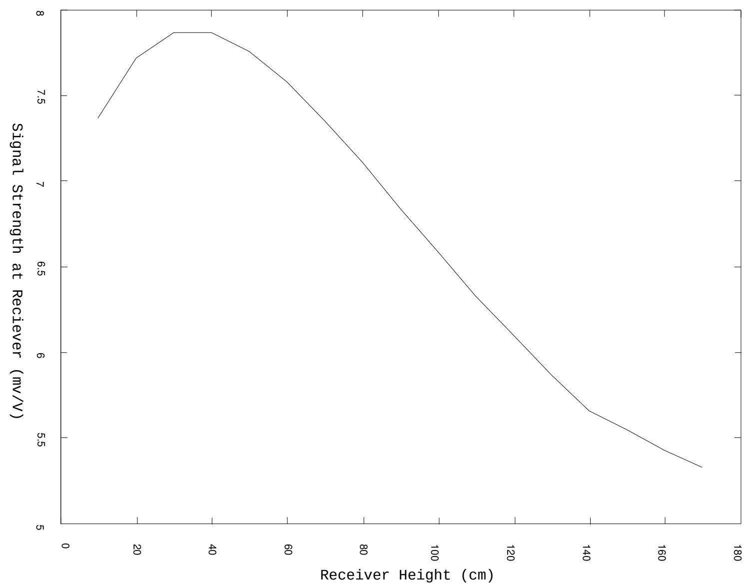

Figure 6-2: Signal Strength vs. Receiver Height

6.2 Noise Characterization

As anticipated, the noise is dominated by amplifier noise, with possible substantial contributions from external interference. Thermal noise is sufficiently small it is possibly unmeasurable in practice.

The amplifier noise depends substantially on the quality of the op-amp itself. The noise density from the amplifier varies from 30 to 150 $\frac{\mathrm{nV}}{\sqrt{Hz}}$ for reasonable op-amps. External interference, even in a noisy lab environment typically remains below 50 $\frac{\mathrm{nV}}{\sqrt{Hz}}$. Therefore, the total noise at the receiver is between 30 $\frac{\mathrm{nV}}{\sqrt{Hz}}$ (low noise op-amp and negligible interference) and 200 $\frac{\mathrm{nV}}{\sqrt{Hz}}$ (generic op-amp and electrically noisy environment).

6.3 Bandwidth

The bandwidth of the electric field around the human body is fundamentally limited by the effective low pass filter caused by the capacitance between the receiver and ground. This limit however, is very high due to the miniscule capacitances involved, and allows for a bandwidth of many megahertz.

This bandwidth, however, is not realistic for PAN, because as the frequency increases, the body begins to exhibit antenna-like properties and radiate, which is in direct conflict with the goal of PAN. In practice, the bandwidth is limited by the bandwidth of a filter on the front end of the receiver which is typically a few kilohertz.

In practice, the bandwidth could likely be usefully pushed up into the tens or perhaps hundreds of kilohertz. Above that, pickup of high frequency interference is likely to become a serious problem as well as the problem of the body behaving as an antenna.

6.4 Channel Capacity

The channel capacity, as seen in Equation 2.1, and repeated here, can now be calculated.

$$C = \Delta f \log_2 \left(1 + \frac{S}{N}\right) = \Delta f \log_2 \left(1 + \frac{S}{N_0 \Delta f}\right) \frac{\text{bits}}{\text{second}} \tag{6.3}$$

Using the estimated values for noise, and assuming a transmitter voltage on the order of 10 V, a channel capacity can be calculated for a foot-to-foot configuration or as a function of receiver height. For the height dependent case, the channel capacity decreases roughly linearly with height.

Using the most current experimental hardware (not a low noise op-amp) with a front end low pass filter to limit the bandwidth to 1.2 kHz, and an interference estimate of white noise at 50 $\frac{\text{nV}}{\sqrt{Hz}}$, the channel capacity is roughly 30.8 kilobits per second (kbps). A low noise op-amp would increase this to roughly 33.9 kbps. The thermal noise has almost no impact – less than 10 bits per second decrease relative to an ideal system without thermal noise. Using a wider bandwidth is the most effective for increasing channel capacity. Spreading the signal to a 10 kHz band, with a low noise op-amp yields a channel capacity of 252 kbps. At 100kHz, the capacity is in excess of 2000 kbps.

6.5 Conclusion

As expected, foot-to-foot configurations far outperform other configurations. Also, amplifier noise and interference are the primary sources of noise in the system. Finally, channel capacities are capable of being substantially boosted by improvements in coding, but more substantial gains can be made by increasing the bandwidth over which the PAN signal is spread.

Appendix A

Source Code

The code for the PAN modeler follows.

A.1 man Page

```
NAME
        panmodel - model PAN system


SYNOPSIS
        panmodel [-v slice] [-n size] [-x] -a alpha


DESCRIPTION
        panmodel takes a geometry description on standard
        input and reports its output on standard error.  It
        will graphically display its progress if the -x flag
        is used.


        The geometry description is a list of points and which
        component they are part of.  One point is specified per
        line as:
```

```
x y z component

x, y, and z are integer coordinates from 1 to
size. The
component value is one of the following, depending on
which component the point is part of:

1       Body
2       Ground
3       Transmit body electrode
4       Transmit ground electrode
5       Receive body electrode
6       Receive ground electrode

If a single point is listed as part of multiple components,
the component listed last is used.
```

OPTIONS

-v slice

Graphically view a two dimensional slice of the model at x = slice during the 3D SOR portion of the calculation. This option is required when using the -x option. Currently slices in x are the only ones supported.

-n size

Specify a grid size. Currently only cubic grids are supported. Default is 50.

-x

Enable X windows viewing of the 3D SOR process. Requires the -v option accompany it.

```
    -a alpha
            Set the value for alpha.  Values above 2 are unstable.

BUGS
    The X support is mediocre.  It may not work under some conditions,
    and its colormap handling is poor.

    Doesn't provide reasonable defaults for alpha or slice.

    Fails ungracefully if input geometry is incompatible with specified
    size.
```

A.2 main.c: Pan Simulator

```c
/*      3d SOR solver
        by Matthew Gray, <mkgray@mit.edu> */

#include "main.h"
#include "sor.h"
#include "xgraphics.h"

#include <unistd.h>
#include <stdio.h>
#include <math.h>
#include <string.h>

fpt     *Data;
char    *realData;
int     *Boundary;
int       *Objects;
int c, i, j, k,ct;
double alpha;
fpt ch;
```

```
int viewslice;
char useX=0;
char dump=0;
int nobj;

extern NPTS;

void clear_data_and_boundaries();
void read_boundaries_file();
void set_boundaries(int o1, int o2);
fpt run_simulation(int f, int s);
void gaussj(float a[7][7], int n, float b[7][2], int m);

void
main(int argc, char **argv){
        extern char *optarg;
        extern int optind;
        int o1, o2, o3, o4;
        fpt cap, thermalnoise, tmp;
        fpt capmat[7][7];
        fpt solveme[6][6], solvemeo[6][6];
        fpt rhs[6][2];
        int node, ct;
        int FREQ = 250000;

        /* Process the command line options */

        NPTS = 50;
        while((c = getopt(argc, argv, "a:v:n:f:xd")) != EOF)
          switch (c) {
          case 'a':
            alpha = (double) strtod(optarg, NULL);
            break;
          case 'v':
            viewslice = (int) atoi(optarg);
            break;
```

```
  case 'n':
    NPTS = (int) atoi(optarg);
    break;
  case 'd':
    dump = 1;
    break;
  case 'f':
    FREQ = (int) atoi(optarg);
    break;
  case 'x':
    useX = 1;
    break;
  }
fprintf(stderr, "Alpha is %f.\n", alpha);


/* Allocate memory for field solver*/
fprintf(stderr, "Allocating memory for a %d cubic grid\n", NPTS);
Data = malloc(NPTS*NPTS*NPTS*sizeof(fpt));
realData = malloc(NPTS*NPTS*sizeof(char));
Boundary = malloc(NPTS*NPTS*NPTS*sizeof(int));
Objects = malloc(NPTS*NPTS*NPTS*sizeof(int));

if(useX)
        initialize_graphics();
fprintf(stderr, "Initialized graphics\n");
clear_data_and_boundaries();
read_boundaries_file();

/* First, do the field solution for each pair of objects */

for(o1=1; o1 <=nobj; o1++){
        for(o2=(o1+1);o2<=nobj;o2++) {
                if(o1 == o2)
                        continue;
                fprintf(stderr, "Starting simulation between objects %d and %d.\n", o1, o2);
```

```
                set_boundaries(o1, o2);
                cap = run_simulation(o1, o2);
                capmat[o1][o2]=cap;
        }
}

/* Ok, now we do the circuit half of the problem */

/* First, clear the matrix */
for(o1=1;o1<7;o1++){
  for(o2=1;o2<7;o2++){
    solvemeo[o1][o2] = 0.0;
    rhs[o1][1] = 0.0;
    if(o1>=o2){
      fprintf(stdout, "------ ");
    }
    else{
      fprintf(stdout, "%6.4f ", capmat[o1][o2]*1000000000);
    }
  }
  fprintf(stdout, "\n");
}

/* Now, set the total current to each node to 0 */
for(o1=1;o1<6;o1++){
  for(o2=1;o2<6;o2++){
    if(o1>o2){
      solvemeo[o1][o2] = FREQ*capmat[o2][o1];
    }
    else if (o2>o1){
      solvemeo[o1][o2] = FREQ*capmat[o1][o2];
    }
  }
}

for(o1=1;o1<6;o1++){
```

```
  solvemeo[o1][o1] =  0;
  for(o2=1;o2<7;o2++){
    if(o1 != o2){
      if(o1>o2){
            solvemeo[o1][o1] -= FREQ*capmat[o2][o1];
      }
      else{
            solvemeo[o1][o1] -= FREQ*capmat[o1][o2];
      }
    }
  }
}

for(o1=1;o1<6;o1++){
  for(o2=1;o2<6;o2++){
    solveme[o1][o2] = solvemeo[o1][o2];
  }
}

/* Add the transmitter current */
rhs[4][1] = .00100;
rhs[3][1] = -.00100;

gaussj(solveme, 5, rhs, 1);



fprintf(stderr, "---------------------------------------\n");
for(o1=1;o1<6;o1++){
  for(o2=1;o2<6;o2++){
    fprintf(stdout, "%4.4f ", solveme[o1][o2]);
  }
  fprintf(stdout, "      %4.10f\n", rhs[o1][1]);
}

fprintf(stdout, "Receiver voltage per transmission volt:  %f\n",
            rhs[5][1]/(rhs[3][1]-rhs[4][1]));
```

```
/* Noise propagation*/
thermalnoise=0.0;
for(o3=1;o3<6;o3++){
  for(o4=1;o4<6;o4++){
    if(o4>o3){
      for(o1=1;o1<6;o1++){
            rhs[o1][1] = 0.0;
            for(o2=1;o2<6;o2++){
              solveme[o1][o2] = solvemeo[o1][o2];
            }
      }
      rhs[o4][1] = .01;
      rhs[o3][1] = -.01;
      gaussj(solveme, 5, rhs, 1);
      tmp = ((4.14e-21)/capmat[o3][o4])*rhs[5][1]/(rhs[o3][1]-rhs[o4][1]);
      thermalnoise += (tmp*tmp);
    }
  }
}
fprintf(stdout, "Thermal noise in V^2:  %.12f x 10^-9\n", thermalnoise*1000000000);

for(o1=1;o1<6;o1++){
  for(o2=1;o2<6;o2++){
    solveme[o1][o2] = solvemeo[o1][o2];
  }
}

/* Add the transmitter current */
rhs[5][1] = 1.0;

gaussj(solveme, 5, rhs, 1);

fprintf(stdout, "Impedance at amplifier is %f\n", rhs[5][1]);

exit(0);
```

```
}

fpt run_simulation(int firstob, int secondob) {
        int ct;
        fpt stab, cap;
        static lastsim = 0;

        stab = 1.0;
        ct = 0;
        if(firstob != lastsim){
          lastsim = firstob;
        while((10000*stab/(NPTS*NPTS*NPTS) > 1) || (ct <=3)){
          ct++;
          stab = relax(alpha, Data, Boundary);
          if(useX){
            i = viewslice;
            for(j = 0; j < NPTS; j++)
              for(k=0;k<NPTS;k++) {
                    Dd(realData, j, k) = (int) F(Data, i, j, k);
              }

            update_image((char **) realData);
          }
          fprintf(stderr, ".");
          if(ct%25 == 0) {
            fprintf(stderr, "%d (%f)\n", ct, stab);
          }
        }
        fprintf(stderr, "\n");
        if(dump){
          fprintf(stdout, "Dump of potential map between %d and %d\n",
                   firstob, secondob);
          for(i = 0; i < NPTS; i++) {
            for(j=0;j<NPTS;j++) {
              for(k=0;k<NPTS;k++){
                    fprintf(stdout, "%d ", (int) F(Data, i, j, k));
```

```
          }
          fprintf(stdout, "\n");
        }
        fprintf(stdout, "\n");
      }
    }
    }

    ch = charge(Data, Objects, secondob);
    fprintf(stderr, "Charge of object 2 is %f\n", ch);

    cap = fabs(8.854e-12*(ch/CHARGE_UP));
    ch = charge(Data, Objects, firstob);
    fprintf(stderr, "Charge of object 1 is %f\n", ch);

    fprintf(stderr, "Intraobject capacitance is %f\n", cap);

    return cap;
}


void
clear_data_and_boundaries() {
  int i,j,k;

  for (i = 0; i < NPTS; ++i)
    for (j = 0; j < NPTS; ++j) {
      for (k = 0; k < NPTS; ++k) {
        F(Data, i, j, k) = 0;
        F(Boundary, i, j, k) = -1;
        F(Objects, i, j, k) = -1;
      }
    }
}

void
```

```
set_boundaries(int obj1, int obj2){
  int i,j,k;

  for (i = 0; i < NPTS; ++i)
    for (j = 0; j < NPTS; ++j) {
      for (k = 0; k < NPTS; ++k) {
            if(F(Objects, i, j, k) == obj1)
                        F(Boundary, i, j, k) = CHARGE_UP;
            else if(F(Objects, i, j, k) > 0)
                        F(Boundary, i, j, k) = 0;
            else
                        F(Boundary, i, j, k) = -1;
            }
            }
}

void
read_boundaries_file() {
/*
            Ok.  The boundaries file gives the locations of the equipotentials
            for the body(s), the ground, the txmit body plate, the txmit ground
            plate, the recv body plate, and the recv ground plate
            The format of the file is as follows:

            xcoord ycoord zcoord part

            where, xcoord, ycoord, and zcoord are integer coordinates
            and part is one of:

            1           Body
            2           Ground
            3           txmit body
            4           txmit ground
            5           recv body
            6           recv ground
*/
```

```c
	char readme[100];
	char *sp;
	char *sp2;
	char *sp3;
	char *sp4;
	int i, j, k;
	int val, obj;
	int tmpobj=0;
	nobj = 0;

  while(fgets(readme, 100, stdin)){
	sp = (char *) strchr(readme, (int) ' ');
	*sp = (char) '\0';
	sp2= (char *) strchr((char *) sp+1, (int) ' ');
	*sp2 = '\0';
	sp3= (char *) strchr((char *) sp2+1, (int) ' ');
	*sp3 = '\0';
	i = atoi(readme);
	j = atoi(sp+1);
	k = atoi(sp2+1);
	obj = atoi(sp3+1);
	F(Objects, i, j, k) = obj;
	nobj = (nobj > obj) ? nobj:obj;
  }
}
```

A.3 sor.c: Successive Over-Relaxation Solver

```c
/* Successive Over–Relaxation solver */

#include "main.h"
#include "xgraphics.h"
```

```
#include "sor.h"

extern NPTS;

fpt
relax(double alpha, fpt *field, int *bound) {
  int i, j, k;
  fpt tmp, change, new, foo;
  change = 0;
          for (i = 1; i < NPTS−1; ++i) {
            for (j = 1; j < NPTS−1; ++j) {
              for (k = 1; k < NPTS−1; ++k) {
                if(F(bound,i,j,k) == −1) {
                      tmp = F(field,i,j,k);
                      new =
                        (((1.0−alpha)*
                          ((fpt) F(field, i, j, k))));
                      foo = ((fpt) F(field,i,j−1,k));
                      foo+= ((fpt) F(field,i−1,j,k));
                      foo+= ((fpt) F(field,i+1,j,k));
                      foo+=((fpt) F(field,i,j+1,k));
                      foo+=((fpt) F(field,i,j,k−1));
                      foo+=((fpt) F(field,i,j,k+1));
                      foo/=6;
                      foo*=alpha;
                      new += foo;
                      (F(field,i,j,k)) = new;
                      tmp = (F(field,i,j,k)−tmp);
                      change += (tmp*tmp);
                }
                else {
                                F(field, i, j, k) = (fpt) F(bound, i, j, k);
                      }
              }
            }
          }
```

```
        for(i=0; i< NPTS; ++i)
          for(j=0;j < NPTS; ++j){
            F(field, i, j, 0) = F(field, i, j, 1);
            F(field, i, 0, j) = F(field, i, 1, j);
            F(field, 0, i, j) = F(field, 1, i, j);
            F(field, i, j, NPTS-1) = F(field, i, j, NPTS-2);
            F(field, i, NPTS-1, j) = F(field, i, NPTS-2, j);
            F(field, NPTS-1, i, j) = F(field, NPTS-2, i, j);
          }
        return change;
}

fpt
charge(fpt *field, int *bound, int objnum) {
  int i,j,k;
  fpt chg=0.0;

    for (i = 1; i < NPTS-1; ++i)
        for (j = 1; j < NPTS-1; ++j)
            for (k = 1; k < NPTS-1; ++k) {
              if(F(bound,i,j,k) == objnum){
                    chg+= (F(field,i,j,k) - F(field,i,j,k+1));
                    chg+= (F(field,i,j,k) - F(field,i,j,k-1));
                    chg+= (F(field,i,j,k) - F(field,i+1,j,k));
                    chg+= (F(field,i,j,k) - F(field,i-1,j,k));
                    chg+= (F(field,i,j,k) - F(field,i,j-1,k));
                    chg+= (F(field,i,j,k) - F(field,i,j+1,k));
              }
            }
    return chg;
}
```

A.4 gaussj.c: Gauss Jordan Maxtrix Solver

```
/* Slightly modified version of the stock numerical recipes
```

```
Gauss Jordan maxtrix solver */

#include <math.h>
#define NRANSI
#include "nrutil.h"
#define SWAP(a,b) {temp=(a);(a)=(b);(b)=temp;}
#include <stdio.h>

void gaussj(float a[6][6], int n, float b[6][2], int m)
{
	int *indxc,*indxr,*ipiv;
	int i,icol,irow,j,k,l,ll;
	float big,dum,pivinv,temp;

	indxc=ivector(1,n);
	indxr=ivector(1,n);
	ipiv=ivector(1,n);
	for (j=1;j<=n;j++) ipiv[j]=0;
	for (i=1;i<=n;i++) {
		big=0.0;
		for (j=1;j<=n;j++)
			if (ipiv[j] != 1)
				for (k=1;k<=n;k++) {
					if (ipiv[k] == 0) {
						if (fabs(a[j][k]) >= big) {
							big=fabs(a[j][k]);
							irow=j;
							icol=k;
						}
					} else if (ipiv[k] > 1) nrerror("gaussj: Singular Matrix-1");
				}
		++(ipiv[icol]);
		if (irow != icol) {
			for (l=1;l<=n;l++) SWAP(a[irow][l],a[icol][l])
			for (l=1;l<=m;l++) SWAP(b[irow][l],b[icol][l])
		}
```

```
			indxr[i]=irow;
			indxc[i]=icol;
			if (a[icol][icol] == 0.0) nrerror("gaussj:  Singular Matrix-2");
			pivinv=1.0/a[icol][icol];
			a[icol][icol]=1.0;
			for (l=1;l<=n;l++) a[icol][l] *= pivinv;
			for (l=1;l<=m;l++) b[icol][l] *= pivinv;
			for (ll=1;ll<=n;ll++)
				if (ll != icol) {
					dum=a[ll][icol];
					a[ll][icol]=0.0;
					for (l=1;l<=n;l++) a[ll][l] -= a[icol][l]*dum;
					for (l=1;l<=m;l++) b[ll][l] -= b[icol][l]*dum;
				}
		}
		for (l=n;l>=1;l--) {
			if (indxr[l] != indxc[l])
				for (k=1;k<=n;k++)
					SWAP(a[k][indxr[l]],a[k][indxc[l]]);
		}
		free_ivector(ipiv,1,n);
		free_ivector(indxr,1,n);
		free_ivector(indxc,1,n);
}
#undef SWAP
#undef NRANSI
```

A.5 xgraphics.c: Graphics routines

```
/* X graphics routines to display the slice */
/* This code currently does not work, but is included because
		the errors should be relatively minor and have not yet
		been found */

#include "xgraphics.h"
```

```
static Display  *D;
static Visual   *V;
static Colormap C;
int     S,Loop,Point;
static Window   W;
static GC       Gc,GcRev;
static XColor   Colors[NCOLORS];
static XSetWindowAttributes Attr;
static XVisualInfo Info;


float    x,y,r,z;
int      i,j;
double alpha;
char readme[100];
char * sp;
char *sp2;
int val;
int SIZE, WIDTH, HEIGHT, NPTS;

void
initialize_graphics() {

  /* Set up the basics */

  SIZE = WIDTH = HEIGHT = NPTS;
  D = XOpenDisplay("");
  S = DefaultScreen(D);
  if (XMatchVisualInfo(D,S,DEPTH,PseudoColor,&Info) == 0) {
    printf("Display can not handle %d bit pseudo color\n",DEPTH);
    return;
  }
  V = Info.visual;


  /* Allocate some colors */
```

```
  for (i = 0; i < NCOLORS; ++i) {
    Colors[i].pixel = i;
    Colors[i].red = sqrt(i)*16*255;
    Colors[i].green = sqrt(i)*16*255;
    Colors[i].blue = sqrt(i)*16*255;
    Colors[i].flags = DoRed | DoGreen | DoBlue;
  }

  C = XCreateColormap(D,RootWindow(D,S),V,AllocAll);
  if(C == NULL)
    fprintf(stderr, "Damnit\n");
  XStoreColors(D,C,Colors,NCOLORS);
  Attr.colormap = C;
  Attr.background_pixel = WhitePixel(D,S);
  Attr.border_pixel = BlackPixel(D,S);
  W = XCreateWindow(D,DefaultRootWindow (D),0,0,WIDTH,HEIGHT,100,
                    DEPTH,InputOutput,V,CWColormap|CWBackPixel|CWBorderPixel,&Attr);
  XStoreName(D, W, "ximage output");
  XMapRaised(D, W);
  Gc = XCreateGC (D, W, 0L, (XGCValues *) 0);
}

void
update_image (char **data) {
        static XImage   *I;
        static int created = 0;

        if(!created){
                I = XCreateImage(D, V, DEPTH, ZPixmap, 0, data, WIDTH, HEIGHT, DEPTH, 0);
        }

        XPutImage(D,W,Gc,I,0,0,0,0,NPTS,NPTS);
}

int get_mouse_x() {
```

```
  Window *rr, *cr;
  int *rx, *ry;
  int *wx, *wy;
  unsigned int *mask;
  XQueryPointer(D,W,rr,cr,rx,ry,wx,wy,mask);
  return *wx;
}
```

A.6 main.h: Main Header File

```
#ifndef MAIN_H
#define MAIN_H

#define CHARGE_UP 250
typedef float fpt;

#endif
```

A.7 sor.h: Header file for sor.c

```
#ifndef SOR_H
#define SOR_H
#include "main.h"

#define F(v, a, b, c) *((v) + ((a)*NPTS*NPTS) + ((b)*NPTS) + (c))
#define Dd(v, a, b) *(v + (a*NPTS) + b)

fpt relax(double a, fpt *f, int *b);
fpt charge(fpt *f, int *b, int o);

#endif
```

A.8 xgraphics.h: Header file for xgraphics.c

```
/* My graphics routines... */
```

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include <stdlib.h>

#define NCOLORS 256
#define DEPTH 8

void initialize_graphics();
void update_image(char **d);
```

Bibliography

[1] Horowitz and Hill. *Art of Electronics, Second Edition*. Cambridge University Press, 1989.

[2] Vetterling Press, Teukolsky. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[3] Chris Turner. PAN: A circuit analysis. Advanced undergraduate project, Massachussetts Institute of Technology, 1996.

[4] Tom Zimmerman. Personal area networks (PAN): Near-field intra-body communication. Master's project, Massachussetts Institute of Technology, 1995.

[5] Tom Zimmerman. Personal area networks: Near-field intrabody communication. *IBM Systems Journal*, 35, 1996.