Seventeenth Mathematical and Statistical Modeling Workshop for Graduate Students

7 – 15 July 2011
North Carolina State University
Raleigh, NC, USA

Organizers:
Pierre Gremaud, Ilse C.F. Ipsen, Ralph C. Smith
Department of Mathematics
North Carolina State University

This report contains the proceedings of the Industrial Mathematical and Statistical Modeling Workshop for graduate students, held at the Center for Research in Scientific Computation at North Carolina State University (NCSU) in Raleigh, North Carolina, 7 – 15 July 2011.

This was the seventeenth such workshop at NCSU. It brought together 34 graduate students from 26 different universities. Fifteen of the students were from mathematics programs, 18 from statistics, and one from computer science.

The goal of the IMSM workshop is to expose mathematics and statistics students from around the country to: real-world problems from industry and government laboratories; interdisciplinary research involving mathematical, statistical and modeling components; as well as experience in a team approach to problem solving.

On the morning of the first day, industrial and government scientists presented six research problems. Each presenter, together with a specially selected faculty mentor, then guided teams of 4 – 6 students and helped them to discover a solution. In contrast to neat, well-posed academic exercises that are typically found in coursework or textbooks, the workshop problems are challenging real world problems that require the varied expertise and fresh insights of the group for their formulation, solution and interpretation. Each group spent the first eight days of the workshop investigating their project and reported their findings in half-hour public seminars on the final day of the workshop.

The IMSM workshops have been highly successful for the students as well as the presenters and faculty mentors. Often projects lead to new research results and publications. The projects can also serve as a catalyst for future collaborations between project presenter and faculty mentor. More information can be found at

http://www.samsi.info/workshop/2011-industrial-mathstat-modeling-workshop-graduate-students-july-7-15-2011

Sponsors

Statistical and Applied Mathematical Sciences Institute (SAMSI)
Center for Research in Scientific Computation (CRSC)
Department of Mathematics, North Carolina State University

Presenters

Jared Bogacki, ERM Analytics, BB&T
Agustin Calatroni and Herman Mitchell, Rho Inc.
Christopher Kees, US Army Research and Development Center
Laura Potter, Syngenta Biotechnology
Jordan Massad, Sandia National Laboratories
John Peach, MIT Lincoln Laboratory

Faculty mentors

H.T. Banks, NCSU, Mathematics
Cammey Cole Manning, Meredith College
Lea Jenkins, Clemson University
Emily Lei Kang, SAMSI/NCSU, Mathematics
Jeff Scroggs, NCSU, Mathematics
Ralph Smith, NCSU, Mathematics

Student Participants

Tom Ashu, Matthew Avery, Treena Basu, Kanadpriya Basu, Alexandria Bolotskikh, Josh Callaway, Ming-Chun Chen, Yansong Cheng, Rachel Danson, Eric Eager, Michael Fink, Mohan Gopaladesikan, Karl Gregory, Oleksandr Gromenko, Yu Gu, Xiao Han, Fatima Jaafari, Sarah Khasawinah, Liang Kong, Eddy Kwessi, Qi Li, Colleen McCarthy, Sean McNichols, Raymond Mess, Jerome Ouedraogo, Qi Qi, Keri Rehm, Bill Shi, Benjamin Silva, Susan VanderPlas, Anran Wang, Yan Zeng, Yilong Zhang, Wenli Zhou

Projects

- **Robust optimal design of Heliostat arrays for concentrating solar power plants**

  Jordan Massad (Sandia National Laboratories) and Ralph Smith (North Carolina State University)

- **Robot Scientists**

  John Peach (Lincoln Laboratory) and Cammey Cole Manning (Meredith College)

- **Convective flow in porous media**

  Chris Kees (US Army Research & Development Center) and Lea Jenkins (Clemson University)

- **Experimental design and inverse problems in biological modeling**

  Laura Potter (Syngenta Biotechnology) and H.T. Banks (North Carolina State University)

- **Models for predicting indoor pollution from questionnaire and outdoor pollution data**

  Agustin Calatroni and Herman Mitchell (Rho Inc.), and Emily Lei Kang (SAMSI/North Carolina State University)

- **Credit risk quantification**

  Jared Bogacki (ERM Analytics, BB&T) and Jeff Scroggs (North Carolina State University)

Robust Optimal Design of Heliostat Arrays for Concentrating Solar Power Plants

Josh Callaway[1], Oleksandr Gromenko [2], Yu Gu[3], Rachel Hegemann[4], Eddy Kwessi[5], Qi Li[6], Colleen McCarthy [7]

Problem Presenter: Dr. Jordan E. Massad, Sandia National Laboratories
Faculty Mentor: Professor Ralph C. Smith, North Carolina State University

Abstract

Concentrating solar-thermal power plants represent a new generation of power plants that use the sun as a heat source. In central receiver tower configurations, an array of flat mirrors, called heliostats, focuses and concentrates sunlight onto a receiver on the top of a tower. Ultimately, the heat generated from the directed sunlight generates steam, which, in turn, is used in a turbine generator to produce electricity. The amount of sunlight that can be concentrated on a given tower depends on several parameters, such as the annual solar insolation, tower height, and the number, size, and position of the heliostats in the array. The climate, environmental factors, and fabrication tolerances lead to variations and uncertainties in system performance. Determination of a cost-effective array of heliostats that optimizes the annual irradiance on a tower receiver in the presence of uncertainty is sought. Using data collected for the Solar Two Project we were able to build and verify a model of the collection efficiency of a central receiver system. Within the collection efficiency model we considered four main types of loss, including projection loss, blocking and shading, spillage, cosine effect and attenuation. In order to account for specific uncertainties in the physical model, errors from tracking errors, availability of mirrors, reflectance and weather were propagated through the model. The collection efficiency was then optimized over the height of the receiver tower. A cost function was also constructed to approximate the cost of constructing a given central receiver system. Then we were able to model the power-to-cost ratio for increasing tower heights.

1 Introduction and Motivation

1.1 Background and Motivation

The search for new sources of energy is becoming of increasing importance on a global scale. With diminishing natural resources, the focus has shifted to renewable sources of energy, one of which is solar power. Currently solar power provides less than one percent of energy used in the United States [?]. However, there has been promising research in this field and this percentage is expected to increase as new technologies are developed. In June 2011, the Department of Energy offered over $8.5 billion for solar projects to support it's SunShot initiative, that aims to reduce the cost of solar power by 75% by 2020 [?]. If this goal can be accomplished, solar power costs will be competitive with costs of current dominant sources of energy [?].

Within the field of solar power, there are many options for harvesting energy from the sun. One such system is known as concentrating solar power. This mechanism consists of an array of heliostats positioned around a central receiver tower, the top of which contains a receiver surface. The heliostats are oriented to maximize the amount of sunlight reflected onto the receiver. The receiver then uses the concentrated rays to heat up various forms of substrate that can produce steam to power turbines and produce electricity. Figure **??** displays one such system built in California, known as the Solar Two Project [?]. Sandia National Laboratories collected numerous amounts of data on this central receiver system and has provided us with a means to validate our mathematical model.

[1]University of Arkansas for Medical Sciences
[2]Utah State University
[3]Columbia University
[4]University of California Los Angeles
[5]Auburn University
[6]University of Illinois at Urbana Champaign
[7]North Carolina State University

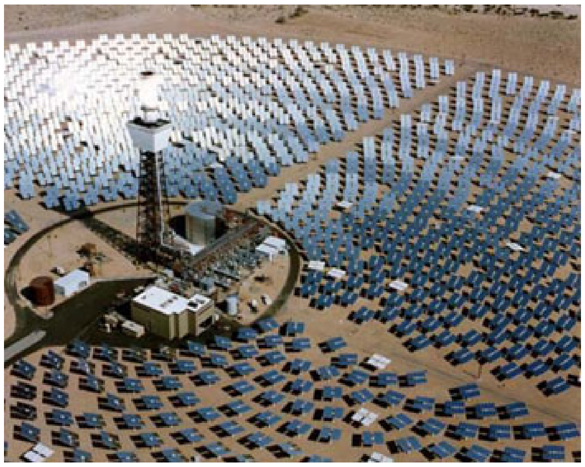


Figure 1: Solar Two Project, Daggett, CA (1995)

The objectives of the Solar Two project were to collect data and learn more about this system in order to increase energy output and minimize cost in future projects [?]. Despite the quality data obtained, this project cost slightly more than $173 million dollars. There is some indication that the efficiency of the system could be increased by varying certain physical parameters. However, changing these parameters can be very costly. Therefore, in our project we set out to model this system mathematically and optimize efficiency over these physical parameters while minimizing cost.

In Section **??** we will discuss our forward model simulating the collection efficiency of central receiver system. This model incorporates losses due to cosine effect, interaction among the neighbors (blocking and shading), spillage, and reflectivity of the mirrors. From this we approximate the collective efficiency of the system. We then consider aspects of uncertainty within the system and input parameters in Section **??**. The cost of the system is estimated in Section **??**. Finally, the optimal ratio of the mirror to tower is determined along with a discussion on optimizing the system in Section **??**. We conclude this report in Section **??**, with a discussion of future work and recommendations for improvements of the model.

2 Forward Model

The main focus of the project is to produce a model of the central receiver system by taking into account the major factors of the physical system. Many of the factors considered were identified in the Solar Two project [?].

Generally, the factors included in the model can be broken into two categories: those that can be controlled and those that cannot. Those that can be controlled include the position of the mirrors, height of the receiver, collection area/size, number of mirrors, size of the individual mirrors, and mirror quality. The factors that cannot be controlled are the sun, weather, and atmospheric conditions. In our model we include all of these variables. Table **??** provides a list and description of the variables incorporated into the model. The flow chart, displayed in Figure **??**, provides a pictorial representation of the input parameters, the calculated sources of loss and the computed output.

2.1 Assumptions

There are a number of assumptions, listed below, made about the physical system in order to model the central receiver system mathematically.

- The dimensions of the mirror elements are constant for each mirror.
- The sun's rays produce a vector that does not spread; i.e., there is no sunspread in the system.

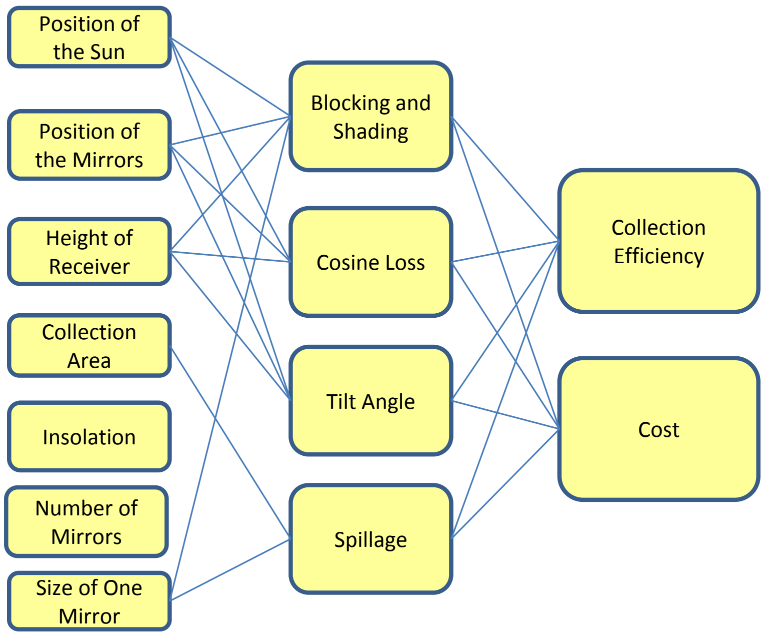


Figure 2: Forward Model

The collection efficiency of the power plant is defined as,

$$\frac{P(t)}{N \cdot A_m \cdot I_0}, \tag{1}$$

where $P(t)$ is how much the plant actually produces, and $N \cdot A_m \cdot I_0$ is the maximum power the plant can produce. Taking into account the assumptions of the system and the input parameters, we approximate the collection efficiency by

$$\frac{P(t)}{N \cdot A_m \cdot I_0} = \frac{1}{N} \sum_{i=1}^{N} F_{\cos}(i;t) \cdot F_{sh,b,sp}(i,t) \cdot F_{\tau}(i;weather) \cdot F_{ref}(i), \tag{2}$$

where N is the number of heliostats in the array, $F_{\cos}(i;t)$ is the efficiency accounting for the cosine effect, and $F_{sh,b,sp}(i,t)$ is efficiency accounting for shading, blocking, and spillage. These losses are described in more detail in Section **??**. The attenuation/transmittance, F_τ, accounts for the loss due to the spread of sunlight traveling from the heliostat to the receiver. The loss due to reflectivity of the mirror is represented by $F_{ref}(i)$. This efficiency can be reduced if there is dust on the mirror, and will depend on the mirror material. It is the product of clean reflectance and cleanliness described in [**?**].

The total power produced by the plant is

$$P(t) = \sum_{i=1}^{N} I_0(weather) \cdot A_m \cdot F_{cos}(i;t) \cdot F_{sh,b,sp}(i;t) \cdot F_\tau \cdot F_{ref}, \tag{3}$$

where I_0 is the direct normal insolation of the sun's incident ray and A_m is the area per heliostat.

Parameter	**Description**
$A(t)$	Azimuth Angle
$\alpha(t)$	Sun's altitude
h	The height of the receiver element
H_{mid}	Distance from the ground to the midpoint of the receiver
$\vec{C}_i = (e_i, n_i, z_i)$	East , North, and Zenith coordinate of the i^{th} heliostat w.r.t. the base of the tower
$\vec{R}_i$	Vector from the center, C_i, of the ith heliostat to the midpoint of the receiver
$\vec{S}_i$	Vector from the center, C_i, of the ith heliostat to the sun
H_m	Height of the mirror element
W_m	Width of the mirror element
A_m	Area of the mirror
I_0	Solar flux

2.2 Loss Mechanisms

There are a number of ways that incoming insolation, I_0, is reduced in projection or before it reaches the receiver. The main loss factors we examined in this project are loss of energy due to the angle of the mirror (cosine effect), loss due to interference among the mirrors (blocking and shading), loss due to the sunlight missing the receiver element (spillage), and loss due to the light moving through the air from the mirror to the receiver (attenuation).

2.2.1 Attenuation

The approximate expression for the atmospheric attenuation is

$$F_\tau(i) = 0.99326 - 0.1046 d_i + 0.01 d_i^2 - 0.002845 d_i^3, \tag{4}$$

for 23-km visibility, and

$$F_\tau(i) = 0.98707 - 0.2748 d_i + 0.03394 d_i^2, \tag{5}$$

for 5-km visibility, where $d_i = \sqrt{h^2 + e_i^2 + n_i^2}$ is a distance from the i^{th} heliostat to the receiver measured in kilometers, [**?**].

2.2.2 Cosine Effect

One of the major sources of loss identified in [**?**] is due to the *cosine effect.* This is the loss in flux from the sun's incident ray, $\vec{S}$, and reflection ray onto the receiver, $\vec{R}$. These two vectors are bisected by the normal of the mirror $\hat{N}$. The incident angle, θ, is defined as the angle from either $\vec{S}$ or $\vec{R}$ to the normal. The cosine loss for a mirror i is defined as

$$F_{cos}(i;t) = \cos(\theta). \tag{6}$$

Note that the cosine of θ decreases as θ gets larger. When $\cos(\theta)$ decreases, flux from the vector $\vec{S}$ is lost. To obtain $\cos(\theta)$, we need to calculate θ using the position of the sun with respect to the base of the tower, $\vec{S}$, and the position of the heliostat, C. From C we calculate the reflection vector, $\vec{R}$.

To calculate the position of the sun, we need three angles: the solar hour angle (ω), the site latitude (ϕ), and the solar declination (δ). The solar hour angle is defined as the horizontal angle from the center of the earth to the sun. The site latitude is the angle created by the line from the zenith (z) at the latitude of the solar power plant to the line, (m), from the center of the earth that forms a perpendicular intersection at the equator with the meridian that runs through the site of the solar power plant. The zenith is the point vertical to the heliostat, and the meridian is the longitudinal line where the solar power plant is located. Once determined, these angles are used to calculate two new angles, solar altitude of the heliostat, (α), and azimuth, A. The azimuth angle is analogous to the solar hour angle in that it forms a horizontal angle from the heliostat to the receiver tower. Figure **??** gives a pictorial representation of these angles.

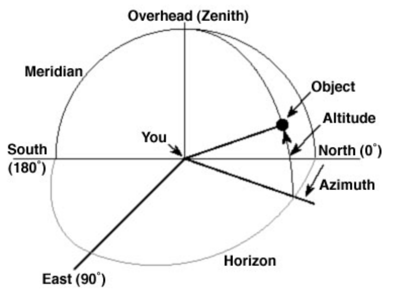


Figure 3: Sun Position Angles

The altitude angle α is defined as

$$\alpha = \sin^{-1}(\sin\phi\sin\delta + \cos\phi\cos\delta\cos\omega) \tag{7}$$

The solar declination δ is described by

$$\delta = 23.45\sin\left(360 \times \frac{284+n}{365}\right), \tag{8}$$

where n is the day number of the year ($n = 1$ is January 1st). Once we have α, we are able to compute the azimuth, A, by,

$$A = \cos^{-1}\left(\frac{\sin\alpha\sin\phi - \sin\delta}{\cos\alpha\cos\phi}\right) + 180. \tag{9}$$

Now we are able to calculate the incident vector, $\vec{S}$, using the relation

$$\vec{S} = (\cos\alpha\sin A, -\cos\alpha\cos A, -\sin\alpha).$$

Using $\vec{C}$, we determine $\vec{R}$, the coordinates of which are defined by,

$$\vec{R} = \vec{T} - \vec{C}. \tag{10}$$

In this case $\vec{T}$ is the vector from the base of the tower to the midpoint of the receiver defined by $(0, 0, H_{mid})$. Now that we have the incident vector $\vec{S}$ and $\vec{R}$ we calculate θ using the relation

$$\theta = \frac{1}{2}\cos^{-1}\left(\vec{S}\cdot\vec{R}\right). \tag{11}$$

When calculating the cosine loss of the entire system we calculate $\cos(\theta)$ for each individual heliostat and then compute the average.

2.2.3 Blocking and Shading

Another significant source of loss is in the interaction among the mirrors in the array and the projection of the light onto the receiver. We consider two forms of interactions between surrounding mirrors: shading and blocking. *Shading* occurs when neighboring mirrors obstruct the light coming into another representative mirror element. Further loss is observed when the light going out of the representative mirror element towards the receiver is blocked by neighbor mirror elements. This is called *blocking*. Finally, there is a possibility that the receiver does not collect all of the light projected from the array. For example, if the mirror orientation is not optimal, some of the light reflected off of the mirror element may not be intercepted by the receiver. This inaccuracy is referred to as *spillage*.

All of these losses depend heavily on the location of the array elements with respect to the sun and the receiver. Figure **??** displays a representative configuration of the sun, receiver and mirror. Each mirror element is defined by its center point, C_i, the vector normal to the plane of the mirror, $\hat{N}_i$, and the height and width of the reflective portion of the array element. In the figure, R_i is the vector from the center of the mirror, (C_i), to the midpoint of the receiver element. Likewise the vector from the center of the mirror to the sun is denoted by S_i. The configuration of the heliostat array assumes that the receiver tower is located at the origin, $(0, 0, h_t + h_h)$, and all of the vectors are in East, North, Zenith or (E,N,Z) Cartesian coordinates. These vectors are used to approximate the losses due to shading, blocking, and spillage for each mirror element.

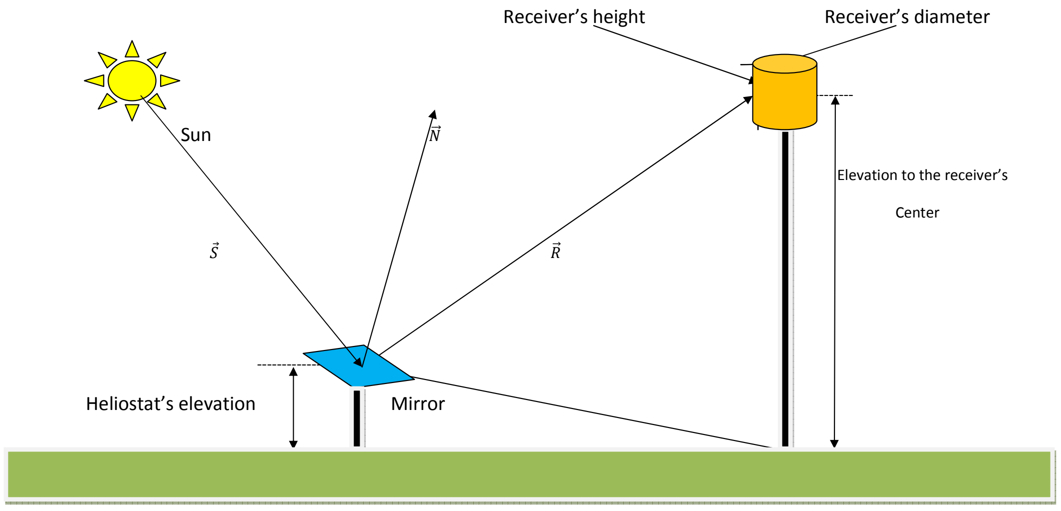


Figure 4: Representation of the mirror, sun, and receiver configuration used to determine the blocking, shading, and spillage of one mirror.

Algorithm Details

Ray tracing is one method that determines the loss of energy due to shading, blocking and spillage. As the name suggests, this method follows a collection of rays from the sun to the elements of the heliostat array. Some of these rays will be reflected; these are then tracked to determine the amount of light that hits the receiver. One large drawback of this type of algorithm is the computational cost.

Instead of following the trajectory of each ray, we implemented an algorithm proposed by Armstrong [**?**]. Here we sequentially consider the contribution of sunlight to the receiver from each heliostat. In this way we always consider a representative mirror and its kth nearest neighbors. To determine the contribution of the representative mirror each neighboring mirror is first encoded as a polygon in the global coordinate system. In the case of shading, each of the neighboring polygons are projected along their individual sun vectors (S_i) onto the plane defined by the representative mirror element. The union of these projected polygons are the loss of light due to shading,L_S. Note that L_S is a polygon that may have disjoint pieces and/or holes. The loss due to blocking is calculated by projecting the polygon of the neighbors onto the plane of the representative mirror along the vector defined by the center of the representative mirror to the receiver, R_i. The union of these projected polygons is the loss due to the blocking, L_B. The effective shape, M_e, of the light that is projected to the receiver is then the mirror polygon of the representative element, M, minus the union of L_B and L_S.

$$M_e = M - (L_B \cup L_S) \tag{12}$$

Spillage was not addressed in [**?**]. However, we are able to extend the concept of blocking to determine the amount of light obtained by the receiver. In our framework we view the receiver as another object that is blocking the sun's rays. Here the amount of light blocked by the receiver is the amount of light obtained by the receiver directly from the sun. Specifically after the M_e has been determined the plane of the receiver is projected onto the representative mirror element along the vector defined from the center of the mirror

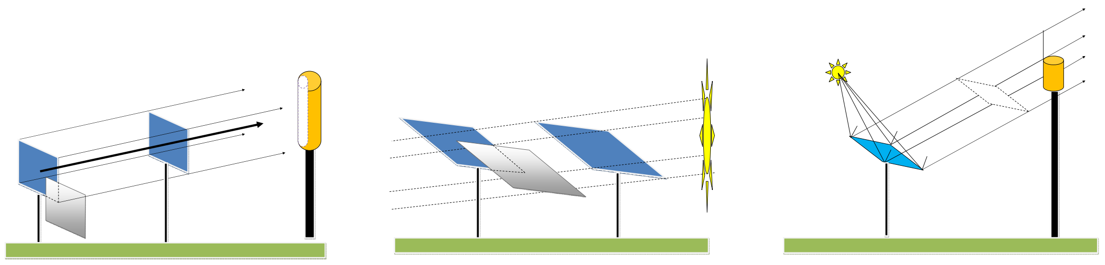


Figure 5: Representations of blocking (left), shading (center) and spillage (right). In each case the polygon of the neighboring mirror , or the receiver, is projected onto the plane of the representative element.

element to the midpoint of the receiver. The final shape of the representative mirror element that reaches the receiver is the intersection of the projection of the receiver and M_e. From this final shape the area of sunlight for the representative element is calculated.

Code Details

The implementation of the shading, blocking, and spillage can be calculated with the function named *BlockingAndShadingAndSpillage*. Table **??** provides a list of the inputs and output and a short description of each variable.

Name	Solar2 Value (m)	Description	Data Type
Input:			
mHeight	6.9	Height of the mirror	scalar
mWidth	6.8	Width of the mirror	scalar
C		Location of each mirror	Matrix -Nx3
rHeight	76.2	Distance from the ground to the of the midpoint of the receiver	scalar
rRadius	2.55	Radius of the receiver	scalar
r_height	6.2	Height of the receiver	scalar
tSunPos	[0, -.88 10^{11}, 1.2 10^{11}]	Position of the sun	Vector - 1x3
nearestNeighborIndex		Lookup table identifying the nearest neighbors for each array element	Matrix - NxN
Output:			
effectiveArea		Approximated area of sunlight obtained by the receiver	Vector- Nx1

Table 1: Input and output parameters for the function *BlockingAndShadingAndSpillage* with description. Note that all coordinate locations are in (E,N,Z) Cartesian coordinates with respect to the bottom of the receiver tower. N refers to the number of array elements.

There is one deviation from [**?**] to note. Instead of using the proposed equations to project the vertices of the neighboring polygon onto the plane of the representative mirror, the 3-D projections were calculated via Equations **??**-**??**. Conceptually, these equations take a point V and move it a distance t along the direction vector, $\hat{W}$. Equation **??** determines the exact distance required to move the point to the plane of the representative mirror, defined by the center of the mirror, C, and the unit normal, $\hat{N}$.

$$t = (-\hat{N} \cdot V - \hat{N} \cdot C)/(\hat{N} \cdot \hat{W}) \tag{13}$$

$$P_x = V_x + \hat{W}_x * t \tag{14}$$

$$P_y = V_y + \hat{W}_y * t \tag{15}$$
$$P_z = V_z + \hat{W}_z * t \tag{16}$$

One downside to the current implementation is the computational cost. When run on the Solar Two data the method took approximately 81 seconds when considering each mirror's eight nearest neighbors. We have identified the built-in MATLAB function *polybool* to be a serious bottle neck in the code. This function is used to compute the union and subtractions of the projected polygons. One way to significantly improve the run time is to use a different polygon clipping algorithm. Further, this code can easily be parallelized by computing the effective area for each array element on a separate node.

2.3 Applying Model to Solar Two Project

Once the forward model was constructed we were able to test it using data from the Solar Two Project. Figure **??** shows the cosine efficiency and collection efficiency of each heliostat for the Solar Two Project. As expected, heliostats positioned north of the receiver and closer to the receiver have higher efficiency than those placed south of or further away from the receiver.

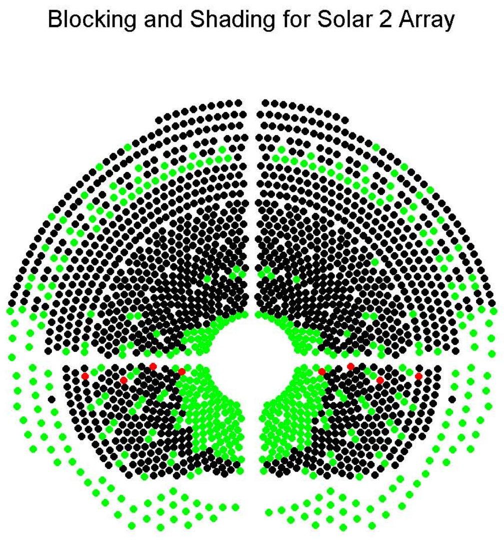


Figure 6: Solar 2 array identifying mirrors that were blocked (black), shaded (red), or neither (green).

3 Uncertainty Propagation

3.1 Pointing Accuracy

In our forward model, we assume that each mirror will adjust its orientation automatically and directly reflect the sunlight to the center of the receiver. Given the direction of the sunlight and the center of the mirror and the receiver, the orientation of the mirror is fixed. However, in reality this is not always accurate. We must add randomness to our model to account for uncertainty from factors such as machine malfunction, wind and so on. We represent this error by the random variable ϵ; if the desired incidence is β_0, we set the incidence to be

$$\beta = \beta_0 + \epsilon.$$

We assume that ϵ has a Gaussian or uniform distribution. The uncertainty in this parameter will affect blocking, shading and spillage.

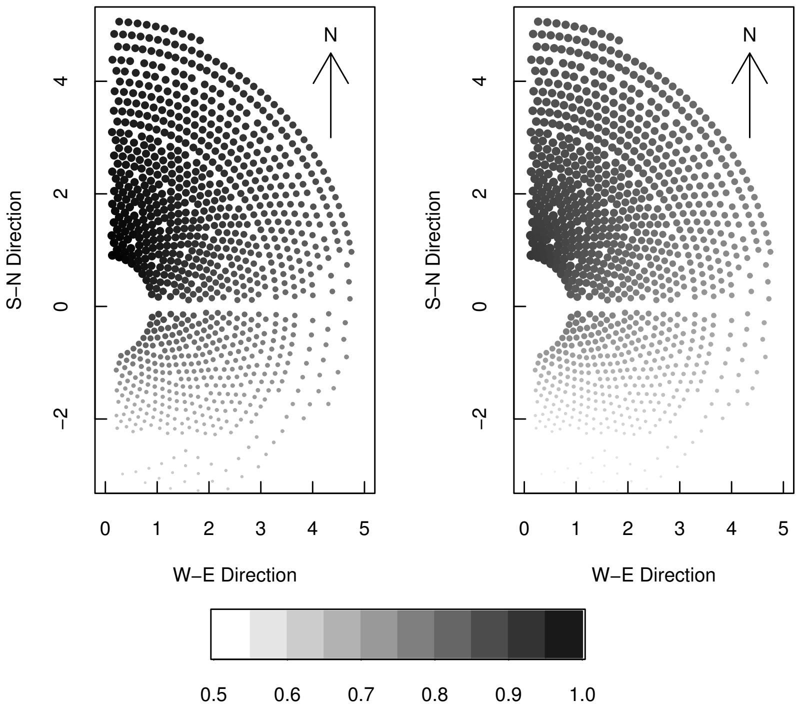


Figure 7: Left panel shows Cosine Efficiency and right panel shows Collection Efficiency of Solar Two Heliostats

Table 2: Cumulative Probabilities

Collection efficiency	Initial system	Optimized system
$\geq 50\%$	100%	100%
$\geq 60\%$	99%	100%
$\geq 65\%$	0	99%
$\geq 70\%$	0	5.6%

3.2 Mirror Quality

Another source of uncertainty is the quality of the mirror. Let N be the total number of mirrors, and assume the reflectance values for each mirror are independent and identically distributed random variables ξ_n, $n = 1, \ldots N$ of a certain distribution. To choose the appropriate distribution, we must take into account that some mirrors may malfunction and lose all ability to reflect sunlight. We account for this by introducing a positive probability, p, that each ξ_n will have zero reflectivity. Based on the data provided from Solar Two, we let $p = 0.1$.

To determine the distribution of the collection efficiency we let the following parameters vary:

α_1, α_2 Two spillage angles, $\alpha_{1,2} \sim N(0, \sigma)$, where σ is chosen to be 0.005 rad and can be adjusted correspondingly. Note that the collection efficiency depends dramatically on the selection of σ.

b availability of mirror $b \sim Bin(1, p)$ where $p = 0.1$ from the data in Solar Two.

ρ reflectivity of a heliostat, $\rho \sim U[0.85, 1]$.

w Weather condition $w \sim Bin(1, p)$, where p is the probability of a clear day.

After generating random parameter values from the above distributions we run the forward model to produce an ensemble of collection efficiencies. Next we calculate the CDF. In our simulations we use 1000 different realizations.

4 Optimization

The next step is to optimize the collection efficiency. There are several ways to optimize the performance of the system. Ideally, we would optimize over the height of the tower, position of the heliostats, size of the heliostats, size of the receiver, and the number of the heliostats. Unfortunately optimization over all these parameters is computationally expensive. Thus, we focus on optimizing the height of the tower for the given array of heliostats in the Solar Two and fix all other parameters.

The collection efficiency as a function of the height of the tower is shown in Figure **??** along with 60% and 90% confidence intervals. We determine that the optimal height of the tower is 140m, which is approximately two times taller than the Solar Two receiver tower. The cumulative distributions of the initial model and the optimized model are shown in Figure **??**.

4.1 Cost Function

The cost function of the solar power plant can be defined by

$$\text{Total Cost} = h_t \cdot C_t(h_t) + N(A_m \cdot C_{mirror} + C_{install}), \tag{17}$$

where h_t is tower height, $C_t(h_t)$ is the cost function of tower height, N is the number of heliostats, A_m is the area in square meters of each individual heliostat, C_{mirror} is the cost per square meter for each heliostat mirror, and $C_{install}$ is the cost of installing one heliostat into the ground. From [**?**], the cost function of tower height is defined as,

$$C_t(h_t) = e^{.1775 + .004352 h_t + .000007561 h_t^2}. \tag{18}$$

From this we can see that the cost of the tower increases in an exponential manner as the height of the tower increases. This is illustrated in Figure **??**. We assume that it costs \$20,000 to install one heliostat with a 95 m^2 mirror area [**?**]. Of the total cost, approximately \$5,000 corresponds to the mirror components. Therefore, we estimate that the cost to install non-mirror components for one heliostat is \$15,000, while the cost per unit mirror area is $52\$/m^2$. The height of the receiver centerline of the Solar Two tower is 76.2 meters, while the absorber height is 6.2 meters [**?**]. Therefore,

$$h_t = \frac{6.2}{2} + 76.2 = 79.3\ m. \tag{19}$$

Essentially, the output of the solar power plant increases as the height of the tower and the number of heliostats increases. However, since the cost of the tower increases in an exponential fashion as tower height increase, this causes the total cost of the solar power plant to increase faster than the power produced with all other variables held fixed. Basically, the power produced cannot keep up with the cost of producing it, so at some point the benefit of the power does not outweigh the cost of plant installation and upkeep. In other words, the power-cost-ratio decreases as the height of the tower increases with all else held fixed. This relationship is described by the equation,

$$PCR = \frac{P(t)}{TotalCost}, \tag{20}$$

and is visualized in Figure **??**. Our goal is to maximize power output while minimizing cost by optimizing this ratio over our parameters.

5 Conclusion

In this project we were able to develop two models for collection efficiency, the main difference being in the calculations of blocking, shading and spillage. When run with the parameter values from the Solar Two data our results for both models were similar to those measured in the Solar Two Project. Within one of our models, we developed an algorithm to simulate blocking, shading and spillage, and the efficiency of these loss mechanisms. Once we had the forward model, we considered the uncertainty associated with many of the model inputs. Then we obtained distributions for these parameters and propagated the uncertainty to use in our robust optimization of the tower height. In addition to our forward model, we obtained a function to calculate the total cost of the plant. We then obtained a power-to-cost ratio for the power plant and developed a curve as the tower height increases. This revealed that the power does not increase as quickly as the cost when we increase the height of the tower. Thus, as the tower height increases, it becomes non beneficial to increase the tower height. In order to optimize this power-to-cost ratio, we need to increase the power. We can accomplish this by several various measures. For instance, we could change the number of mirrors, mirror position, mirror size, or various combinations of these.

In our results we were able to optimize over the ratio of the receiver and mirror height. We found, through our estimated cost function, that there are cost limits on the height of the tower. As the height of the tower increases, so does the cost. Keeping the monetary concerns in mind, finding an optimal array for this system cannot only consider the height of the tower. Instead, future efforts could examine the number, size and location of mirrors.

References

[1] Power from the Sun. http://powerfromthesun.net/Book/chapter10/chapter10.html.

[2] IEA-Solar Power and Chemical Energy Systems: Task III: Solar Technology and Application, ATS H15. Solar Paces Technical Report: Catalog of Solar Heliostats (No. III - 1/00: p. 15). 2000 June.

[3] J.E. Pacheco, et al.: Final Test and Evaluation Results from the Solar Two Project, Sandia Technical Reports, SAND2002-0120 (2002 January)

[4] Battleson, K.W.(1981), ”Solar Power Tower Design Guide: Solar Thermal Central Receiver Power Systems, A Source of Electricity and/or Process Heat,” Sandia National Labs Report SAND81-8005, April.

[5] Sterns Roger Engineering Company (1979), ”Tower Cost Data for Central Receiver Studies,” Sandia National Labs Report SAND78 -8185, June.

[6] Department of Energy. http://apps1.eere.energy.gov/news/daily.cfm/hp_news_id=308. June 2011.

[7] National Atlas. http://nationalatlas.gov/articles/people/a_energy.html#three.

[8] Biggs, F. and C. M. Vittitoe,(1979), The HELIOS Model for the Optical Behavior of Reflecting Solar Concentrators,” Sandia National Labs Report SAND76-0347, March.

[9] Peter Armstrong. An annotated algorithm for the shading and blocking computations of a field of heliostats arranged in a grid layout. Technical report, Tietronix Software Inc., 2009.

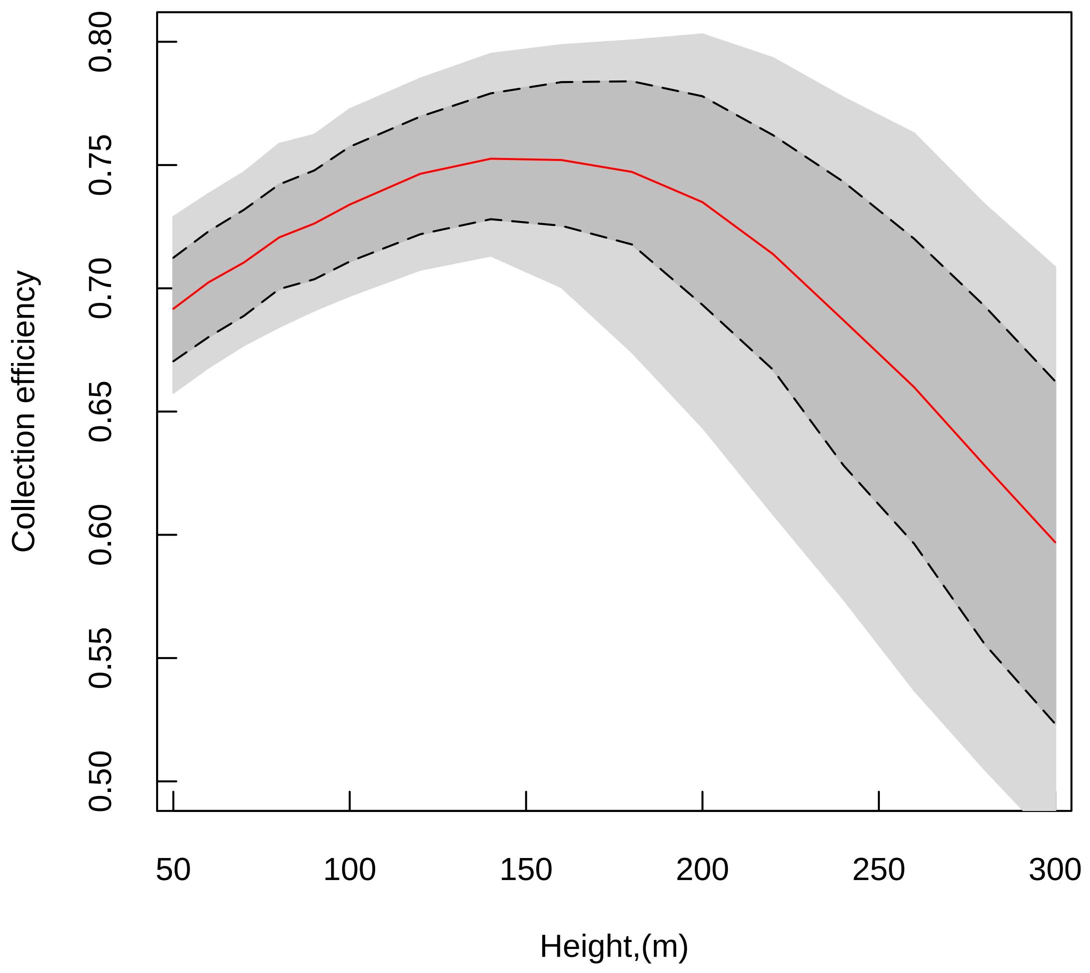


Figure 8: The solid red line represents the collection efficiency as a function of the tower height. Dark gray and light gray regions represent 60% and 90% respectively. Dashed lines are added for convenience.

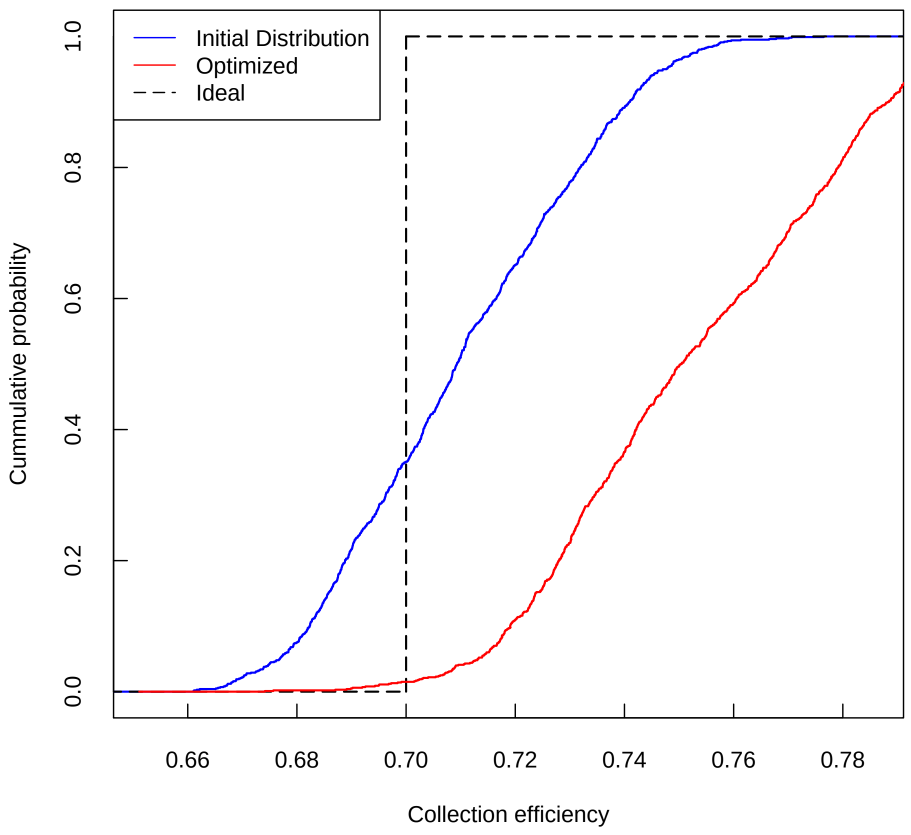


Figure 9: Cumulative distribution functions, blue line is CDF of the system before optimization, red line is CDF of the system after optimization and dashed line is desired CDF.

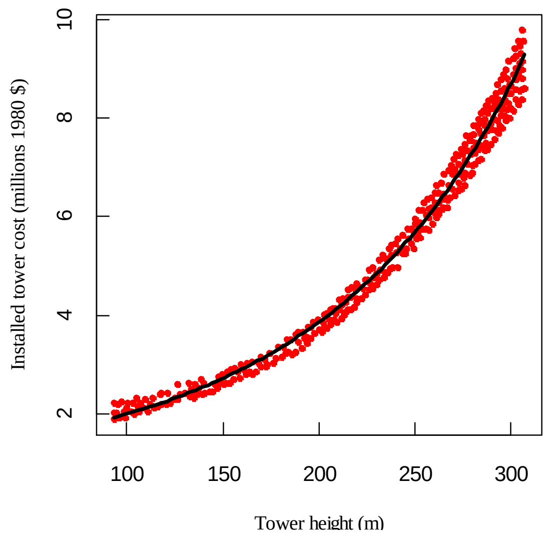


Figure 10: Installed Tower Cost vs. Tower Height. The red points represent observations from the cost curve in Chapter 10 of [?]. ([?],[?])

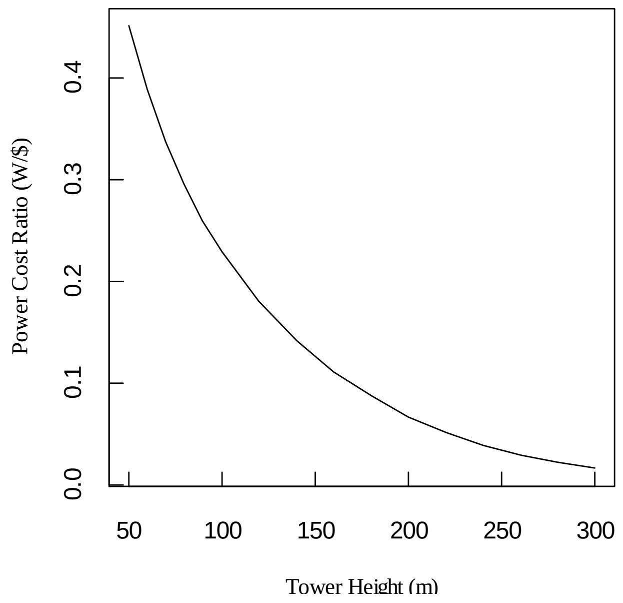


Figure 11: Power to Cost Ratio vs. Tower Height. The ratio is measured in Watts per dollar of input into the system for the tower, mirrors, and mirror installation [?],[?]

.

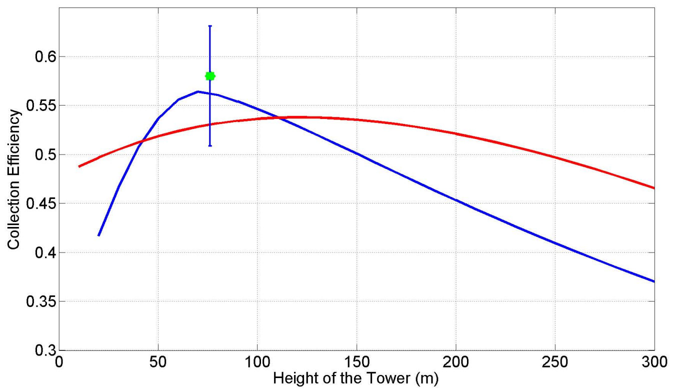


Figure 12: Collection Efficiency vs. Tower Height. The red line shows the effective model, the blue line shows the forward model, the green point is the measured value and the vertical blue bar represents the 95% confidence interval.

PROBLEM 2: ROBOT SCIENTISTS

Tom Ashu[1], Michael Fink[2], Mohan Gopaladesikan[3], Karl Gregory[4], Fatima Jaafari[5], Qi Qi[6]

Problem Presenter: John Peach
MIT Lincoln Laboratory

Faculty Mentor: Cammey Cole Manning
Meredith College

Abstract

In this paper we study and analyze symbolic regression using Genetic Programming (GP) as a tool. We successfully model simulated data from Salustowicz function using Eureqa and MATLAB's GPTIPS toolbox. Next we study a classification problem on a diabetes dataset of Native American woman from Arizona. In search of finding optimal setting for the algorithm, we varied different parameters like the population size, number of generations, the maximum tree depth, the maximum mutation tree depth and assessed the performance in terms of the complexity of the resulting models as well as their accuracy, sensitivity and specificity when predicting a binary response. We find that the number of generations does not have much effect on the accuracy, and that the fitness function to evaluate our solution could be altered based on the importance we gave to sensitivity versus specificity. In this way GP is malleable, as the fitness function can be altered to guide the algorithm toward a model with specific traits. Lastly we compared the performance of neural networks and principal component analysis with GP, finding the latter to perform just as well in terms of accuracy.

1 Introduction and Motivation

Traditional modeling has relied upon human insight to provide functional forms for observed phenomena. Once the scientists supplies a computer with these forms, the computer estimates the parameters of the specified functions. Now, the possibility of letting the computer itself seek out functional forms as well as estimate parameters is being explored. Dr. Hod Lipson of Cornell has developed a software called Eureqa which employs Genetic Programming to accomplish this[2]. By generating a population of candidate functions and selectively breeding them across several iterations (generations), the algorithm, it is hoped, will stumble upon the correct functional form.

Thus, the only inputs for Genetic Programming, as it is called, are data and a set of variables and usable operations, such as +, - , * , / , and sine. The Eureqa software, given data describing the movement of a double pendulum, correctly "derived" Newton's second law of motion and the law of conservation of momentum in 30 hours [4]. It is hoped that as-yet-unknown laws can be distilled from experimental data using the Genetic Programming approach. However, if a law is unknown, it cannot be determined whether the functional form chosen by the algorithm is correct. Further inquiry into the reliability of Genetic Programming is required before its implementation in new areas for which true functional forms are unknown.

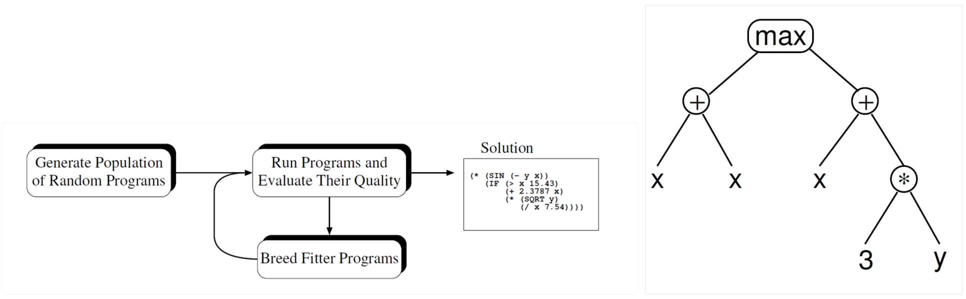
A short introduction to Genetic Programming is given in Section 2. Then the Eureqa software, as well as a MATLAB (The Math Works, Inc., Natick, Massachusetts) package called GPTIPS are used to explore the capabilities and performance of Genetic Programming. In Section 3, the two programs are put to the test on data generated from different functions. In Section 4, the two programs are used on data from an undisclosed function. Then, in Section 5 GPTIPS is implemented in the analysis of a real dataset in which the governing functions relating the predictors to the responses are unknown. The dataset describes a set of patients, and

[1]Kent State University
[2]Binghamton University
[3]Purdue University
[4]Texas A&M University
[5]Binghamton University
[6]University of Missouri

the response is a binary variable indicating the presence of diabetes. The effects of various parametric settings of the algorithm on the resulting model are then explored using the MIT Lincoln Laboratory Grid in Section 6. Lastly, in Sections 7 and 8 the performance of the conventional techniques of principal component analysis and neural networks is compared with that of Genetic Programming.

2 Primer on Genetic Programming

In this section we give a brief, concise and not so rigorous introduction to Genetic Programming (GP) adapted from *A Field Guide to Genetic Programming* [1]. GP is an evolutionary methodology inspired by biological evolution to find computer programs that perform a user-defined task. It is a specialization of Genetic Programming where each individual is a computer program. The algorithm is used to optimize a population of computer programs according to a fitness landscape determined by means of a fitness function, which is a measure of the error between a generated output and the desired output. The schematic below depicts the working of a GP.


Generate Population of Random Programs
Run Programs and Evaluate Their Quality
Solution
Breed Fitter Programs
max
+
+
x
x
x
*
3
y

```
(* (SIN (- y x))
   (IF (> x 15.43)
       (+ 2.3787 x)
       (* (SQRT y)
          (/ x 7.54))))
```

Figure 1: Flowchart of Genetic Programming

Every program that a GP handles is represented in the form of a tree. In other words a GP has a convenient tree representation, where the internal nodes of the tree represent basic operations like +, - , $\times$, / and sine, and the leaf nodes(terminals) are occupied by the variables and constants. Figure 1 illustrates the tree representation of the function $max(x + x, x + 3 * y)$ as it is handled in a GP. It must be understood that the set of basic operations is not limited to the ones enumerated above and that it could be extended to a wide variety of mathematical functions.

Two key genetic operations performed in GP are crossover and mutation, both of which are stochastic. These are formally defined as

- Crossover: The creation of a child program by combining randomly chosen parts from two selected parent programs.
- Mutation: The modification of a child program by altering a randomly chosen part of it.

A pictorial representation is found in Figure 2. Some parameters which can be controlled are:

1. *Maximum tree depth* (MTD): The maximum number of edges which must be traversed from any node to the root node.
2. *Maximum mutation tree depth* (MMTD): The maximum depth of any tree appended at the mutation point of another tree.
3. *Population size*: The number of randomly generated functions with which the algorithm begins.
4. *Number of generations*: The number of times the algorithm creates new populations by crossing and mutating the population in the previous step.

The algorithm below explains GP more clearly.

```
Randomly create an initial population of programs from the available primitives
repeat
  Execute each program and ascertain its fitness.
  Select a few programs from the population with a probability based on fitness to participate in
  genetic operations.
  Create new program by applying genetic operations with specified probabilities.
until an acceptable solution is found or other stopping condition is met
return the best-so-far individual.
```

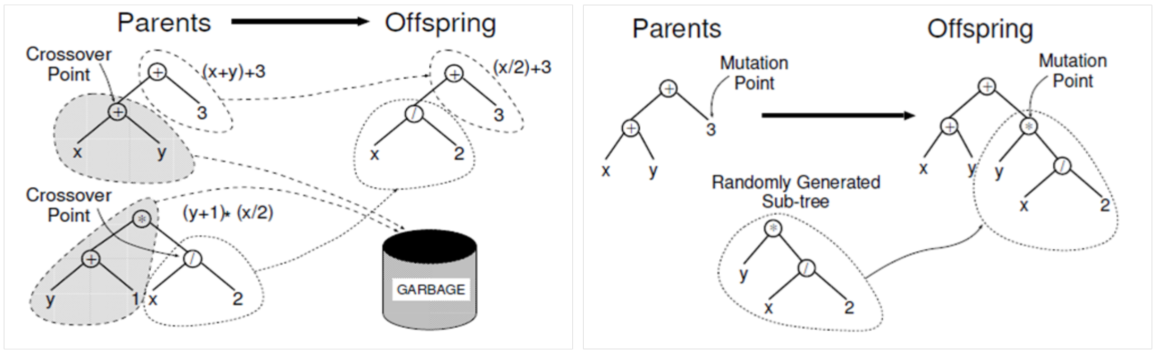


Figure 2: Illustration of Genetic Operations

3 Trial Runs on Known Functions

Data were simulated according to two different functions in order to assess Eureqa's function-finding capability [3].

3.1 Salustowics Function

The one- and two-dimensional Salustowics functions are:

$$\begin{aligned} f(x) &= e^{-x}x^3\sin(x)\cos(x)\left(\sin^2(x)\cos(x)-1\right) \\ f(x_1,x_2) &= e^{-x_1}x_1^3(x_2-5)\sin(x_1)\cos(x_1)\left(\sin^2(x_1)\cos(x_1)-1\right) \end{aligned}$$

For the one-dimensional case, the function was evaluated at 1001 equally spaced datapoints on the domain $[-1, 15]$, and for the two-dimensional case, the function was evaluated on a 30×30 grid in $[0, 10] \times [0, 10]$. Two datasets were created containing the function value along with the values of its input variable(s). For both functions, an additional column in the dataset was created which added a random realization from the normal distribution with mean 0 and variance 1/16 to the true function value.

When no random noise was added, Eureqa found the exact function in about 12 minutes using combinations of the functions $+, -, *, /$, sine, cosine, and exponential. When the Gaussian noise was added, the Eureqa estimate of the function after 15 minutes was

$$\tilde{f}(x) = e^{-x}\left(0.146479 - 0.588727x^3\sin(1.98139x)\right)$$

Figure 3 contains plots of $f(x)$ against x and $f(x)+u$, where $u \sim N(0, \frac{1}{16})$ with the Eureqa-fitted functions overlaid.

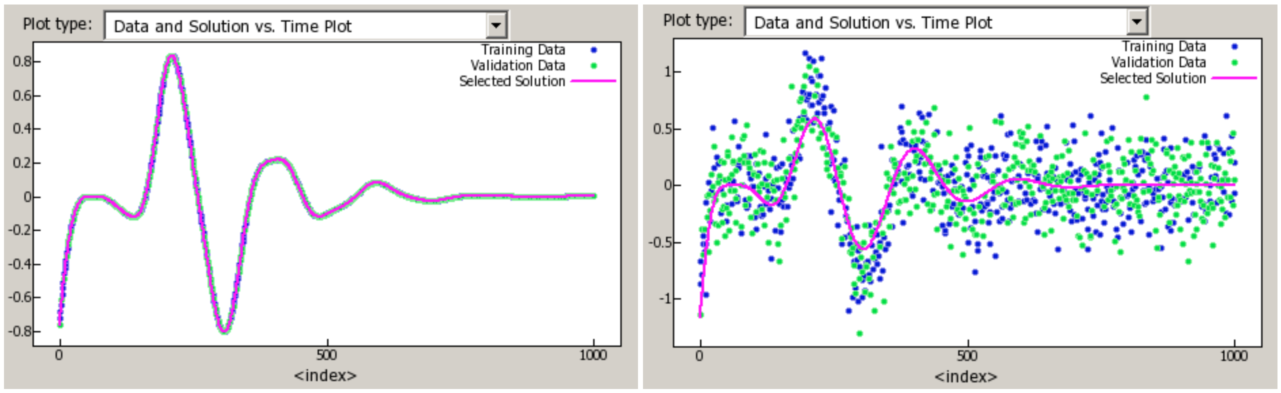
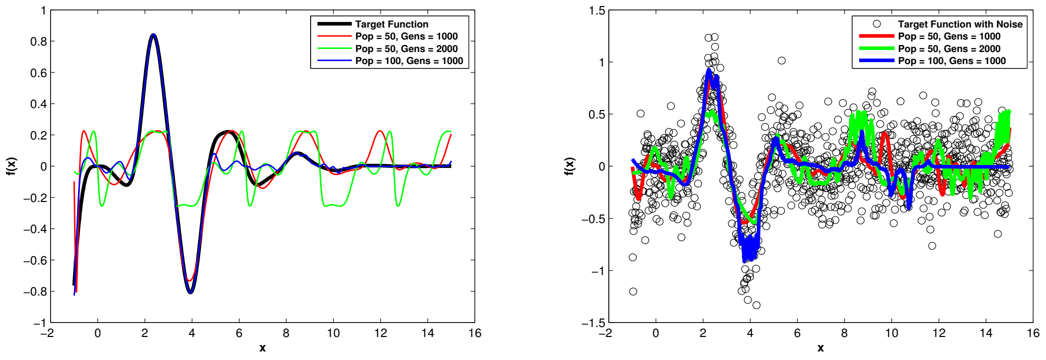


Figure 3: One-dimensional Salustowics function in Eureqa

Genetic programming was implemented in GPTIPS with different population sizes and numbers of generations. The algorithm was run on a subset of the observations called the training dataset, which comprised 66% of the total observations. From the plots in Figure 4, it appears that the population size has a significant effect on how well the chosen function fits the data. The graph in the left panel displays the attempts of GPTIPS to fit a function to the data in the absence of any noise, while the graph on the right displays the function with Gaussian noise added; the three GPTIPS-estimated functions are overlaid.



.

Figure 4: One-Dimensional Salustowics function with GPTIPS

When the data for the two-dimensional function was analyzed in Eureqa, the best function after 2 hours and 53 minutes was

$$\begin{aligned}\tilde{f}(x_1, x_2) &= (30.06x_1\sin(2.00x_1) + 1.23x_1x_2\sin(2.00x_1)\sin(-33.86x_1) \\ &\quad -6.08x_1\sin(2.00x_1)\sin(-33.86x_1) - \frac{6.08x_1x_2\sin(2.00x_1)}{(\frac{26.49}{x_1} + e^{(0.85x_1)})}\end{aligned} \quad (1)$$

The algorithm could potentially have improved its estimate if it were given more time; in the last 30 minutes it had made slight improvements. In the left panel of Figure 5, the true surface as well as that of the estimated function (1). They are similar in shape, with some discrepancies. The right panel displays a scatterplot of the function with Gaussian noise ($\sigma^2 = 1/16$) added, and the surface is the function estimated by Eureqa after running for about 44 minutes (it had remained stagnant after about 40 minutes).

3.2 Kotancheck Function:

In an effort to reconstruct the Kotanchek function whose expression is given by

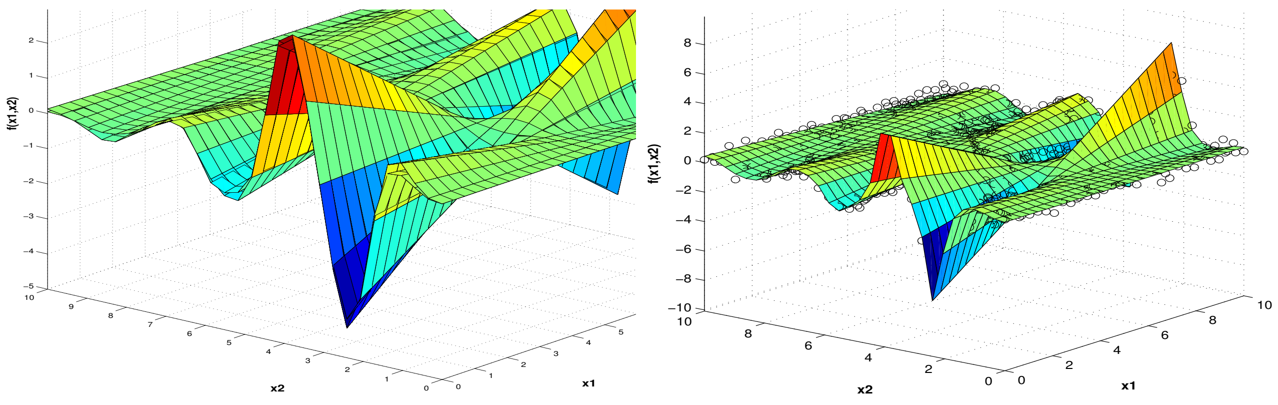


Figure 5: Two-Dimensional Salustowics function estimated with Eureqa

$$f(x) = \frac{e^{-(x1-1)^2}}{(x2-2.5)^2 + 3.2}$$

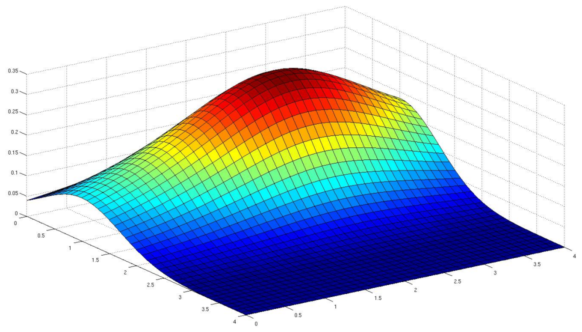


Figure 6: $f(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{(x_2-2.5)^2+3.2}$

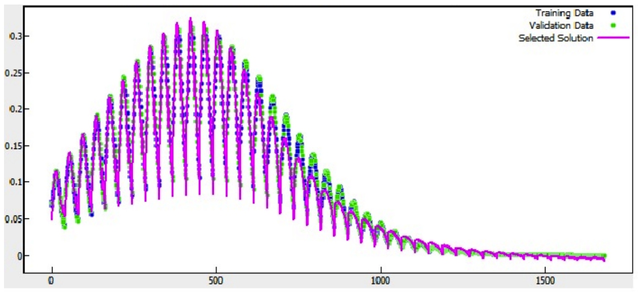
we have generated a random sample of 1800 tuples from the set [0,4]x[0,4] along with the images of these values under this function, and that is considered the response variable for the data set. The curve is plotted in Figure 6. To achieve the task, we have used two techniques that use Genetic Programming to fit the generated data: Eureqa and GPTIPS. We will investigate the results from each software.
As shown from the fitting on the left of Figure 7, Eureqa did fit the dataset with a relatively small computational time. It has produced the following function:

$$\frac{2.48 + 1.48x_1 + 2.25x_1(1.2x_2)^{0.404}}{39.4 + e^{x_2} + (1.2x_1)^{5.37} - 12.7x_2} - 0.00902$$

GPTIPS produced the following function given that it uses the mean square error fitness function:

$$\frac{0.4956}{e^{0.23e^{x1}}} - 0.0008\big({x_2}^2 - x_2 - x_1 - \frac{1.89}{e^{e^{x_1}}}\big)^2 - \frac{0.1987e^{x_1}}{x_2 - x_1 + e^{x_1}} + 0.2189$$

Looking at Figure 8, it seems like GPTIPS does well on predicting the function values.



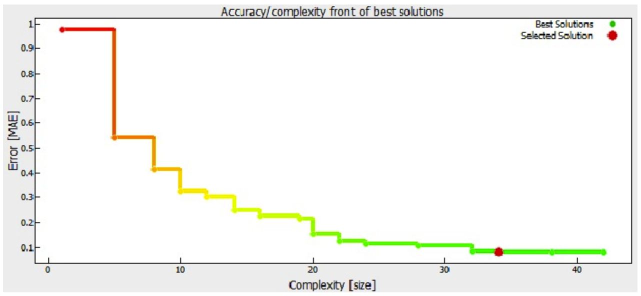


Figure 7: Output from Eureqa

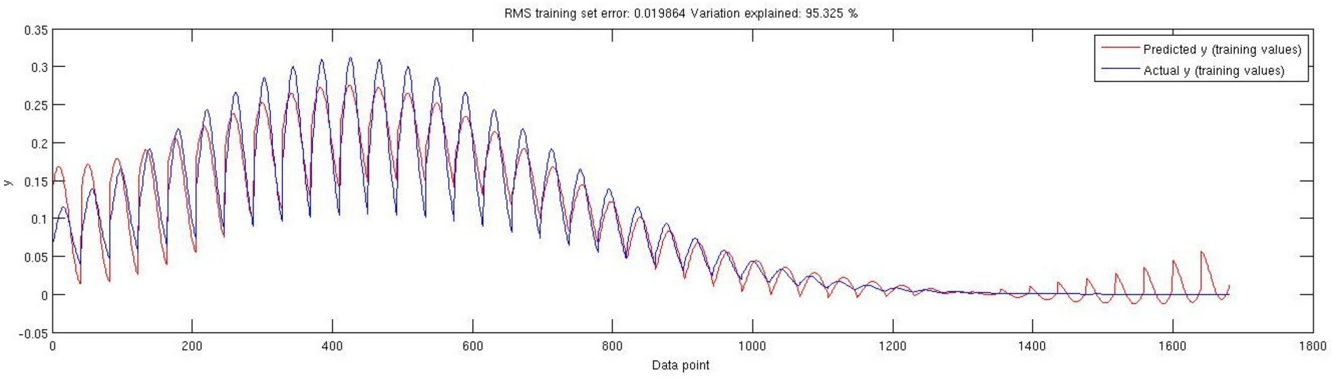


Figure 8: GPTIPS function fit to the Kotanchek function data

4 Mystery Function

A data set was provided from which we were to determine the functional relation between the three variables. We used Eureqa and MATLAB's GPTIPS to analyze them. Surprisingly, we found the function that evolved from both these programs had little or no dependence on the second variable, r.

In GPTIPS we varied various parameters of our program like maximum tree depth, maximum mutation tree depth, population and number of generations, only to find that there is no marked difference in the fitting or the fitness function value. We included many functions like +, -, $\times$, rdivide, plot, square, sine, and exponential. We found that our fit was averaging well with the observed data but did not match exactly. We got a very close estimate of the real data within and root mean square error of about 0.22. All of them also had very comparable fitness. Different runs yielded different functions. Some examples are the following, which are plotted in Figure 9:

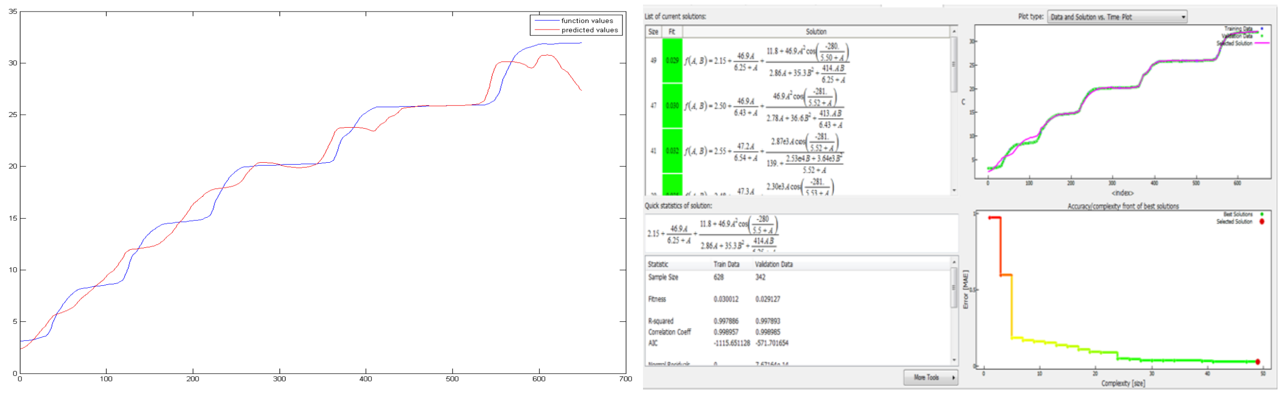
- $\theta = 0.7633t - \frac{147.3}{t+4.558} + 34.56$
- $\theta = \frac{3540t}{\left(\frac{1.932}{t}+11.92\right)(2t+15.48)\left(cos(cos(t))-\frac{r}{t-18.29}+1.852\right)}$

Interestingly, Eureqa gives a much closer curve and the function is complicated with around 49 nodes and a fitness of 0.029. One suggested equation is

$$\theta = 2.15 + \frac{46.9t}{6.25+t} + \frac{11.8 + 46.9t^2 cos(\frac{-280}{5.50+t})}{}2.86t + 35.3r^2 + \frac{414tr}{6.25+r}$$

4.1 Swinging Atwood Machine Data

After the mystery function data was demystified as data from the Swinging Atwood Machine satisfying the ODE



List of current solutions:

Size	Fit	Solution
49	0.029	$f(A,B)=2.15+\frac{46.9A}{6.25+A}+\frac{11.8+46.9A^2\cos\left(\frac{-280.}{5.50+A}\right)}{2.86A+35.3B^2+\frac{414.AB}{6.25+A}}$
47	0.030	$f(A,B)=2.50+\frac{46.9A}{6.43+A}+\frac{46.9A^2\cos\left(\frac{-281.}{5.52+A}\right)}{2.78A+36.6B^2+\frac{413.AB}{6.43+A}}$
41	0.032	$f(A,B)=2.55+\frac{47.2A}{6.54+A}+\frac{2.87e3A\cos\left(\frac{-281.}{5.52+A}\right)}{139.+\frac{2.53e4B+3.64e3B^2}{5.52+A}}$

Quick statistics of solution:

$$2.15+\frac{46.9A}{6.25+A}+\frac{11.8+46.9A^2\cos\left(\frac{-280}{5.5+A}\right)}{2.86A+35.3B^2+\frac{414AB}{}}$$

Statistic	Train Data	Validation Data
Sample Size	628	342
Fitness	0.030012	0.029127
R-squared	0.997886	0.997893
Correlation Coeff	0.998957	0.998985
AIC	-1115.651128	-571.701654

Figure 9: Plots of function values versus predicted values using GPTIPS and Eureqa

$$(\mu+1)\frac{d^2r}{dt} - r\left(\frac{d\theta}{dt}\right) + g\left(\mu - cos(\theta)\right) = 0$$

we suggested the following equation to Eureqa

$$\frac{d^2r}{dt^2} = f\left(r, \theta, \frac{d\theta}{dt}\right)$$

Our building blocks included only constants, addition, multiplication, subtraction, division and cosine, which all appear in the equation. Eureqa predicted a form of the PDE with mean absolute error of 0.007 in under two minutes. Here is the predicted partial differential equation

$$\frac{d^2r}{dt^2} = 0.334r\left(\frac{d\theta}{dt}\right)^2 + 3.26cos(\theta) - 6.53$$

Several runs of Eureqa produced similar results with the same parameters.

Next we suggested the following equation to Eureqa

$$\frac{d^2r}{dt^2} = f\left(r, \theta, \frac{d^2\theta}{dt^2}, \frac{d\theta}{dt}, \frac{dr}{dt}\right) \tag{2}$$

Our building blocks now included functions that don't appear in the PDE. Eureqa still predicted a form of the PDE with a mean absolute error of 0.005

Here is the predicted PDE:

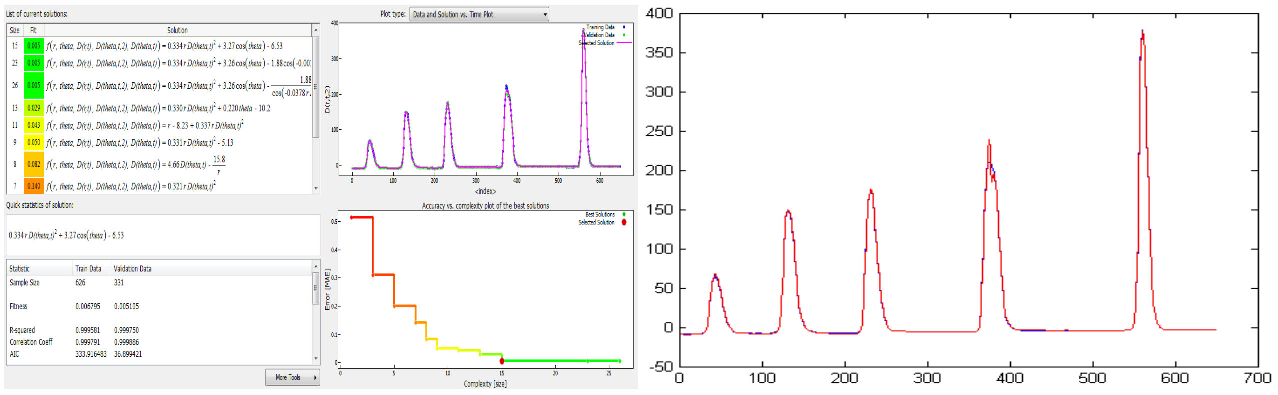
$$\frac{d^2r}{dt^2} = 0.334r\left(\frac{d\theta}{dt}\right)^2 + 3.26cos(\theta) - 6.53$$

just as the case in which we suggested a function of $r, \theta, \frac{d\theta}{dt}$ only.

MATLAB GPTIPS produces similar results. A picture of one of our runs is given in Figure 10. Both Eureqa and MATLAB GPTIPS rightly predicted the form of the PDE when we suggested that the data models a PDE and fail when we suggested a functional relationship.

5 Diabetes Data

The diabetes dataset decribed in Table 1 contains records for 768 patients on which 9 variables were measured [5]. This dataset serves as a testing ground for exploring the behavior of the algorithm when its various settings are altered.



List of current solutions:

Size	Fit	Solution
15	0.005	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 0.334\,r\,D(theta,t)^2 + 3.27\cos(theta) - 6.53$
23	0.005	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 0.334\,r\,D(theta,t)^2 + 3.26\cos(theta) - 1.88\cos(-0.00$
26	0.005	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 0.334\,r\,D(theta,t)^2 + 3.26\cos(theta) - \frac{1.88}{\cos(-0.0378\,r}$
13	0.029	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 0.330\,r\,D(theta,t)^2 + 0.220\,theta - 10.2$
11	0.043	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = r - 8.23 + 0.337\,r\,D(theta,t)^2$
9	0.050	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 0.331\,r\,D(theta,t)^2 - 5.13$
8	0.082	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 4.66\,D(theta,t) - \frac{15.8}{r}$
7	0.140	$f(r, theta, D(r,t), D(theta,t,2), D(theta,t)) = 0.321\,r\,D(theta,t)^2$

Quick statistics of solution:

$0.334\,r\,D(theta,t)^2 + 3.27\cos(theta) - 6.53$

Statistic	Train Data	Validation Data
Sample Size	626	331
Fitness	0.006795	0.005105
R-squared	0.999581	0.999750
Correlation Coeff	0.999791	0.999886
AIC	333.916483	36.899421

Figure 10: Left: Eureqa estimating $\frac{d^2r}{dt^2}$. Right: MATLAB estimating $\frac{d^2r}{dt^2}$

Variable	Description
x_1	Number of times pregnant
x_2	Glucose concentration
x_3	Diastolic blood pressure (mm Hg)
x_4	Tricep skin fold thickness (mm)
x_5	2-Hour serum insulin (mu U/ml)
x_6	Body mass index
x_7	Diabetes pedigree function
x_8	Age
y	Indicator of diabetes (1= diabetic, 0 = not diabetic)

Table 1: Description of diabetes data

In an initial investigation, the effects of changes in four of its parameters were explored. For each of the four chosen parameters, a high and a low value were established; then the algorithm was run three times at every feasible combination of the high and low settings.[7] The four parameters were:

1. *Maximum tree depth* (MTD): Low = 3, High = 12
2. *Maximum mutation tree depth* (MMTD): Low = 2, High = 7
3. *Population size*: Low = 50, High = 200
4. *Number of generations*: Low = 1,000, High=2,000

In order to assess the performance of a predictive function, it is common, when working with a large dataset, to fit the model only on a fraction of the observations, and then to test its performance on the remaining observations. The former fraction is called the training set, as it is used to build or "train" the model, and the latter is called the test set, as it is used to evaluate its performance. Such an approach guards against over-fitting, the case in which the model conforms too closely to the observed sample that it misrepresents the underlying population.

In the following analysis, the 768 records of the diabetes dataset were randomly permuted a single time, and then the first 507 of the observations were designated as the training set. The random permutation was done to scramble any inherent ordering that may have been present in the original dataset. Once these 507 observations were chosen, they were used as the training dataset in all of the analyses. In a typical analysis, the training set would be randomly re-chosen everytime the analysis was run, but in the present study it is necessary to isolate the variability in results that arises from modifying the Genetic Programming parameters; if the training set were randomly selected anew at the start of each run, algorithmic modifications would be confounded with random changes in the training dataset.

[7]Infeasible combinations were those in which the maximum mutation tree depth exceeded the maximum tree depth; a tree exceeding the maximum tree depth could not be appended to another tree.

The functions made available to the algorithm for the following runs were the arithmetic functions, $+$, $-$, $\times$, $/$, and tanh.

MTD	MMTD	Pop Size	Gens	Time	Num Nodes	Accuracy	Sensitivity	Specificity
L	L	L	L	30.00	7.00	0.7714	0.4940	0.9007
L	L	L	H	72.67	7.00	0.7710	0.4643	0.9157
L	L	H	L	187.67	7.00	0.7666	0.4879	0.8970
H	L	L	L	185.33	334.33	0.7676	0.4739	0.9045
H	L	L	H	961.33	837.67	0.7497	0.4819	0.8762
H	H	L	L	205.00	342.67	0.7676	0.5341	0.8762
L	L	H	H	326.67	7.00	0.7663	0.5060	0.8876
H	L	H	L	786.33	389.67	0.7548	0.4779	0.8839
H	L	H	H	2754.67	541.33	0.7612	0.4940	0.8858
H	H	L	H	607.67	496.67	0.7561	0.4739	0.8876
H	H	H	L	632.00	284.67	0.7497	0.5140	0.8596
H	H	H	H	3906.67	507	0.7433	0.4538	0.8784

Table 2: Results under 12 algorithmic settings on diabetes data

Table 2 summarizes the results. The first four columns indicate which parameters were set to high (H) and low (L) values. As the algorithm was run three times under each setting, the statistics in the table are averages across the three runs. Five characteristics of are displayed: the mean duration of the algorithms from start to finish, the mean number of nodes of the three chosen models, and the mean accuracy, sensitivity and specificity acheived by the three chosen models under the specified settings.

Accuracy is simply the proportion of correct predictions—that is, the proportion of patients in the test data set for which the model correctly predicted a 1 (patient diabetic) or a 0 (patient not diabetic). Sensitivity is the rate at which 1s are correctly predicted as 1s, which are called true positives (TP) and specificity is the rate at which 0s are correctly predicted as 0s, which are called true negatives (TN).

It is notable that the formula chosen by the genetic program does not yield only 1s and 0s but a range of values in the neighborhood of 1 and 0. Under conventional methods for predicting a binary response, like logistic regression, the predictions are constrained to lie between zero and one. In this case, it is not so; though the algorithm was furnished with the hyperbolic tangent function (which is sigmoidal), with which it could easily constrain its predictions to lie between zero and one, it did not always do so. Therefore, simply rounding the raw predicted values to the nearest integer will not always yield a prediction vector of 0s and 1s. To produce the accuracy, sensitivity, and specificity statistics on the test dataset, the predicted values were set equal to 1 if they exceeded .5 and set equal to 0 otherwise; this is intuitively appealing, as it equates the raw predictions to the closer of the values 0 and 1.

The following function was generated at the LLLL setting. It has 7 nodes and it acheives an accuracy of 0.7739, a sensitivity of 0.5060, and a specificity of 0.8989:

$$\hat{y} = I\left(.0001553(x_1 + x_6)(x_2 - 13.25271) - .2574 > .5\right)$$

where I returns a 1 if the condition is true and a 0 if it is false. The prediction is thus set to 1 when the GPTIPS expression exceeds .5.

5.1 Trying Different Fitness Functions

The metric for evaluating the fitness of each function during the algorithm in all of these runs was the Euclidean distance between the predicted vector and the observed vector. However, since the reponse vector in the diabetes data contains 0s and 1s, and the goal is merely to predict whether a patient has diabetes, it seems for logical that fitness should be a function of correct and incorrect guesses.

Thus, rather than training the algorithm by favoring functions that acheive a low Euclidean distance between the observed and predicted vectors, the fitness function will now be defined in terms of accuracy, sensitivity, and specificity. To obtain a model acheiving the maximum accuracy (proportion of correct predictions), the fitness function is set to the proportion of incorrect predictions, which the algorithm tries to minimize.

Whereas in the previous runs predicted values were set equal to 1 whenever the raw prediction was greater than .5 and to 0 otherwise, it was decided that the raw predictions should firstly be rescaled to lie between zero and one and then the rescaled values which were greater than .5 would be set equal to 1 and those less than .5 would be set equal to 0, in this way:

```
pred_scale = (pred_raw - min(pred_raw)) / range(pred_raw)
if (pred_scale > .5)  pred_final = 1  ;  else pred_final = 0
```

Recall that the two possible errors are to predict a 0 when the true value is 1 or a 1 when the true value is 0. If one of these errors has more severe consequences than the other, a model can be chosen which accordingly maximizes sensitivity (the rate of TPs) or specificity (the rate of TNs). Table 3 shows results from runs under three different parametric settings; in each, three fitness functions are used. The first is the proportion of wrongly predicted responses, the second favors functions acheiving high specificity, and the third favors functions acheiving a high specificity. In order for the fitness function to favor specificity over sensitivity, a cost function was defined which could place different weights on the two types of predicted errors—false positives and false negatives. The rate of false positives is 1 minus the rate of true positives, that is 1 minus the sensitivity, and likewise the rate of false negatives is 1 minus the specificity. Thus, one can associate different costs to FPs and FNs which reflect the gravity of either mistake.

MTD	MMTD	Pop Size	Gens	Fitness	Time	Num Nodes	Accuracy	Sensitivity	Specificity
L	L	L	H	Misspec.	92.00	7.00	0.7535	0.4418	0.8989
				5× Sens.	87.33	6.33	0.5900	0.8996	0.4457
				5× Spec.	89.33	7.00	0.7356	0.1968	0.9869
H	H	L	L	Misspec.	149.58	109.67	0.7535	0.4659	0.8876
				5× Sens.	130.84	140.33	0.5722	0.9196	0.4101
				5× Spec.	129.26	126.00	0.7177	0.2209	0.9494
H	L	H	L	Misspec.	669.07	259 .00	0.7548	0.3976	0.9213
				5× Sens.	717.64	98.33	0.5913	0.8876	0.4533
				5× Spec.	626.54	90.66	0.7420	0.2811	0.9569

Table 3: Runs in which accuracy, sensitivity, or specificity was targeted under three different settings.

6 Large-Scale Study on The MIT Lincoln Laboratory Grid

The earlier brief investigation into how Genetic Programming responds to changes in its parameters yielded the following observations. Firstly, models with very few nodes performed as well as, and even better in some cases, than highly complex models with a high number of nodes. A higher maximum tree depth led to greater complexity, and when the maximum tree depth was at 3, almost all of the chosen models had 7 nodes—which is the greated complexity possible. The algorithm, given room to grow, will grow, it seems.

The availability of the MIT Lincoln Laboratory Grid made it possible to allocate large sets of algorithm runs to 64 different processors. Thus, algorithmic parameters could be changed incrementally within a range, and the algorithm run at each level in order to see in detail it affected the output.

6.1 Maximum Tree Depth and Maximum Mutation Tree Depth

Firstly, given the apparent importance of maximum tree depth (MTD) and maximum mutation tree depth (MMTD) to the complexity of the chosen model, it was of interest to identify, if possible, an optimal ratio of the two parameters. Thus, for a fixed population size, 200, and a fixed number of generations, 1000, models were fit to the diabetes data for MTD in $\{3, 4, \ldots, 12\}$ and MMTD from $\{2, \ldots, \text{MMTD}\}$.

The results are plotted in Figure 11. The plots to the right and beneath are profiles of the 3-dimensional plot, showing the number of nodes against the maximum tree depth and the number of nodes against the maximum mutation tree depth, respectively. Note that model complexity and the maximum tree depth have a strong positive relationship, appearing to be exponential. For a fixed maximum tree depth, the maximum mutation tree depth does not appear to greatly influence the number of nodes. The apparent correlation between the maximum mutation tree depth and the number of nodes in the lower panel is spurious; the maximum tree depth values always exceed the maximum mutation tree depth values, so as the maximum mutation tree depth increases, the apparent accompanying increase in the number of nodes is only a consequence of the higher maximum tree depth. Note also that in the 3-dimensional plot, for each level of maximum tree depth, the plotted points do not exhibit any notable pattern across the level of maximum mutation tree depth. Therefore, no recommendation for a ratio of maximum tree depth to maximum mutation tree depth emerged.

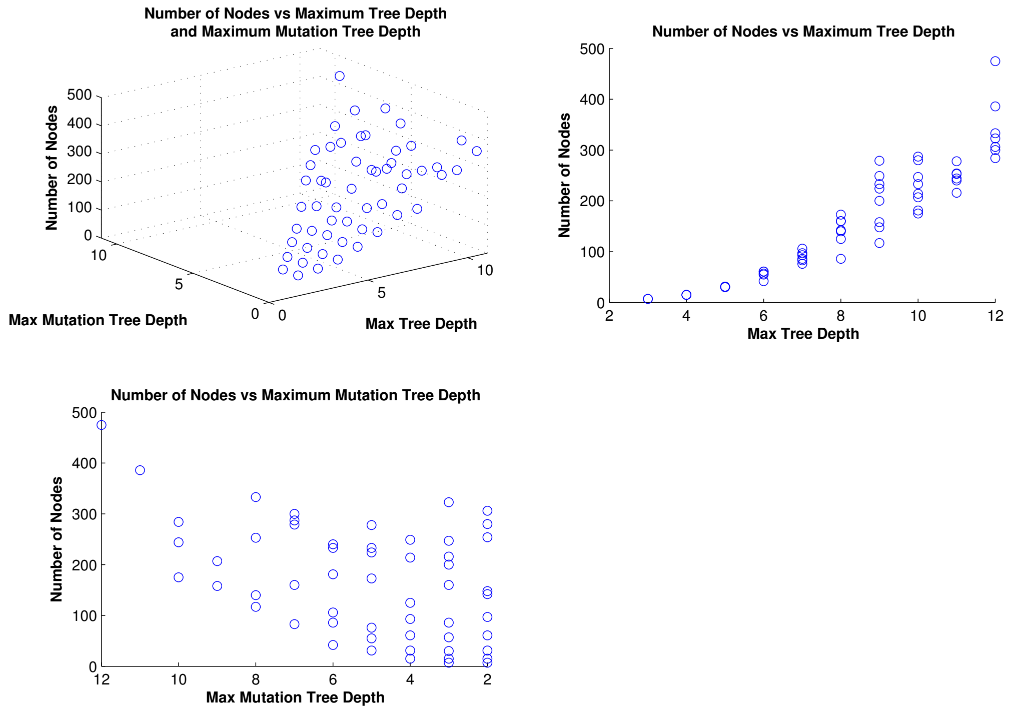


Figure 11: Exploring the effects of maximum tree depth and maximum mutation tree depth on the complexity of the chosen function.

A lot of work has been done in this area, where instead of traditionally fixing the maximum tree depth, a dynamic tree depth limit approach was introduced [9, 10]. This technique have achieved promising results especially in solving complex problems.

6.2 Population Size and Model Complexity

It is of interest to determine whether the population size contributes to model complexity. Thus, fixing the maximum mutation tree depth at 5, the algorithm was run at all combinations of maximum tree depth in $\{5, 7, 9, 11\}$ and populaton size in $\{100, 120, 140, \ldots, 300\}$. The results are plotted in Figure 12. Note that complexity rises with the maximum tree depth. It was thought that when the population is small, the algorithm may have to construct more complex combinations of the population functions, given the smaller pool of genetic material. However, this does not appear to be the case. The bottom scatterplot, which shows a profile of the number of nodes against the population size, exhibits no correlation. In the plot to the right, it is seen that not only does complexity increase, but the variability in the complexity across all the population sizes increases as the maximum tree depth increases.

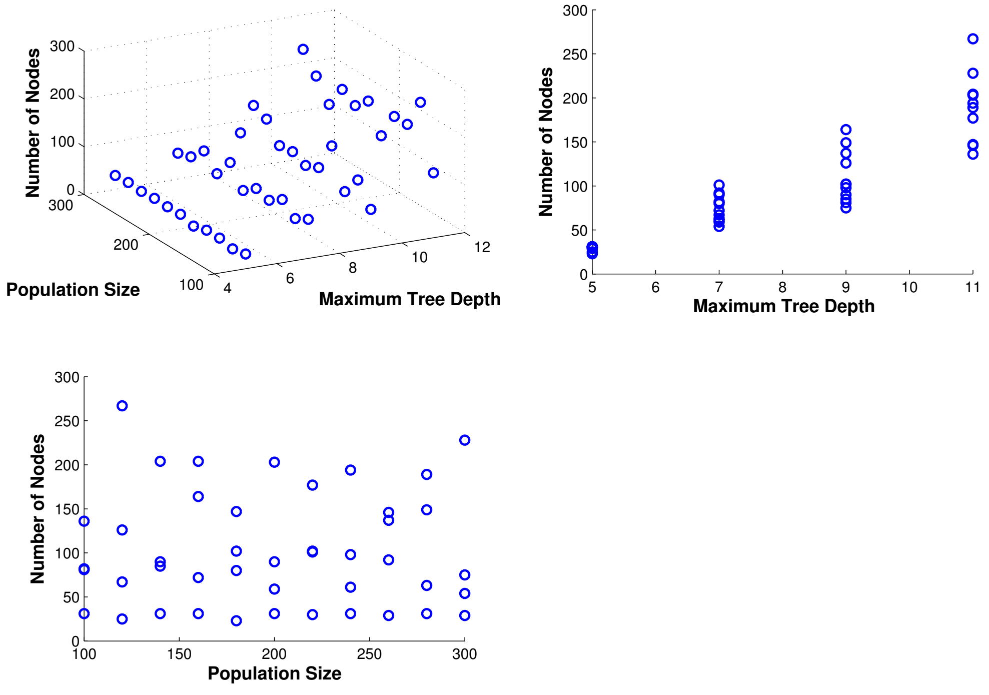


Figure 12: Exploring the effects of population size and maximum tree depth on the complexity of the chosen function.

Figure 13 displays the influence of maximum tree depth and population size on the accuracy of the chosen model. Accuracy appears to be maximized around a population size of 200 or 250. Additionally, in the bottom plot of accuracy against maximum mutation tree depth, there is an indication that too-small or too-large values of maximum tree depth reduce the accuracy of the chosen model; the cluster of points with the highest mean accuracy is located at a maximum tree depth of 7. This is perhaps due to overfitting; if the algorithm is able to perfectly model the training dataset by having a very large population and a high maximum tree depth, then it will not perform as well on the test dataset. If the population is too small and the tree depth too shallow, the algorithm will not be able to find a good fit for the training or the test data.

In [8], studying Genetic Programming with variable population size demonstrates that varying the population size during the algorithm according to a set of rules improves if not matches the results found by standard Genetic Programming while reducing the amount of computational effort.

There is an inherent trade-off between sensitivity and specificity. Consider a test for which it is much more important to detect a 1 than to detect a 0. One could then set all the predicted values equal to 1 and acheive a sensitivity of 1. Conversely, if it were much more important to detect a 0 than to detect a 1, one could predict a 0 every time and acheive a specificity of 1. Between these two extremes lies a curve called the receiver operating characteristic (ROC) curve, which is the curve formed by plotting the true positive rate against the false positive rate—that is sensitivity against 1-specifivity.

To generate a ROC curve for models obtained with Genetic Programming, the fitness function was modified such that the relative fitness cost of false positives and false negatives was swept along a range, at one end of which the fitness function rewarded only true positives and at the other end of which it rewarded only true negatives. The population size, number of generations, maximum tree depth, and maximum mutation tree depth were fixed at 250, 300, 8 and 5, respectively. The left panel of Figure 14 plots sensitivity against 1-specificity across the range of different fitness functions. larger and darker-colored circles represent more complex models in terms of node count. The right panel plots the ROC curve in three dimensions, with

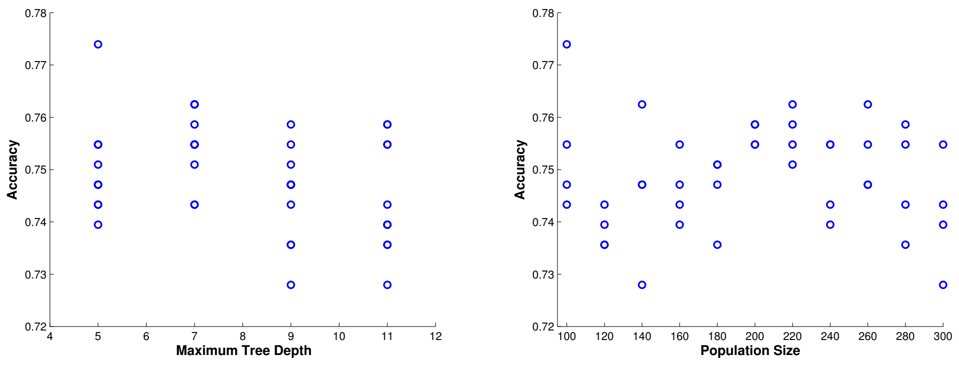


Figure 13: The relationship between accuracy, population and maximum tree depth.

accuracy on the vertical axis. Note that as sensitivity or 1-specificity approach 1, accuracy diminishes.

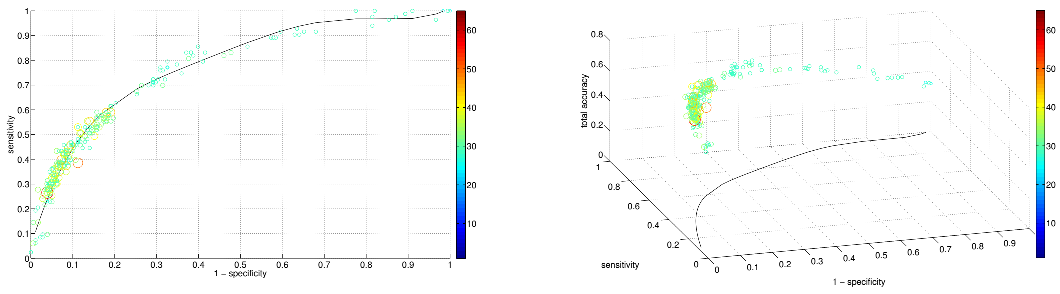


Figure 14: ROC curve for GPTIPS models.

This demonstrates that Genetic Programming can be easily guided by the choice of fitness function. The fitness function describes which traits are desireable, and eventually, the algorithm produces a function having those traits to the greatest degree the data allow.

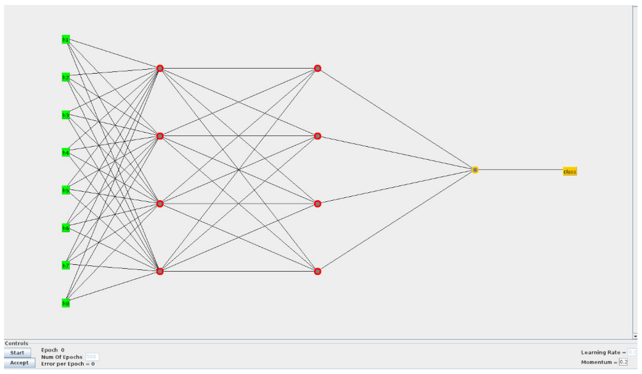
7 Artificial Neural Networks

Multiple layer perceptrons, also known as Artificial Neural Networks, mimic biological neural networks [7]. A multiple-layer perceptron network is comprised of fully interconnected layers including an input layer,hidden layers, and an output layer. The interconnected links have weights. Each hidden layer consists of a number of neurons that have activation functions. The training process is to calculate the weights using the back-propagation algorithm. There are two parameters set during the training process: learning rate and momentum. The learning rate affects the amount by which the weights are updated, and the momentum is applied to the weights during updating.

We employ the multilayer perceptron function in Weka on the classification problem of the diabetes data [6] . Figure 15 shows a screenshot of Weka running with a multilayer perceptron. Some preliminary results are shown in Table 4. The best solution in terms of total accuracy among these results is highlighted in bold.

8 Principal Component Analysis

Principal Component Analysis is one of the ways to analyze and visualize data that has many variables. It generates a new set of variables named principal components by creating uncorrelated linear combinations of



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

Figure 15: Multilayer perceptron in Weka

multilayer perceptrons (hidden-layers,learnRate,momentum,epochs)	**total accuracy**	**sensitivity**	**specificity**
([4],0.3,0.2,500)	0.7126	0.4819	0.8202
([4-4],0.3,0.2,500)	**0.7739**	**0.4940**	**0.9045**
([4-4-4],0.3,0.2,500)	0.7663	0.4940	0.8933
([6-6],0.3,0.2,500)	0.7625	0.5060	0.8820
([8-6],0.03,0.02,2000)	0.7356	0.4940	0.8483
([8-6],0.3,0.2,500)	0.7548	0.4699	0.8876
([4-4],0.4,0.2,500)	0.7625	0.5060	0.8820
([4-4],0.1,0.2,500)	0.7548	0.4819	0.8820
([4-4],0.03,0.02,2000)	0.7395	0.5181	0.8427
([4-4],0.03,0.02,500)	0.7471	0.5060	0.8596

Table 4: preliminary results given by ANNs

the original variables; these linear combinations are called the principle components. For instance, the first principal component is a single axis in the space such that when the original data points are projected along the axis, the variance of the resulting values is maximal. By visualizing the original variables of the data in the new space, we can have a better understanding of their contributions to the viability of the original data.

Figure 17 shows the projections of the original data in the first 2 and first 3 principal component spaces. There are no clear boundaries that separate the two classes completely in the low-dimensional spaces; however, there could be a good classifier existing in a higher-dimensional space.

Figure 18 shows that the first 2 principal components only account for 50% of the total variation, and the rest of the variation is fairly evenly distributed among the other 4 principal components.

Figure 19 shows the contributions of the attributes indicated by vectors to the first two and three principal components, respectively. Thus, all the attributes have positive coefficients for the first principal component, represented by the horizontal axis, whereas only four attributes contribute positively to the second principal component, represented by the vertical axis. All the vectors shown in the right panel of Figure 19 contribute more diversely to the first principal components.

9 Conclusion

In this project we studied the technique of symbolic regression using GP as tool. We used parallel computing to sweep through a wide range of parameters that affected the performance of GP. We found that the number of generations did not have much effect on the accuracy. Based on the importance we gave to sensitivity and specificity we could guide the evolution of the functions toward a model which acheives a desirable rate of true positives and true negatives. It was found that the higher maximum tree depths lead to greater

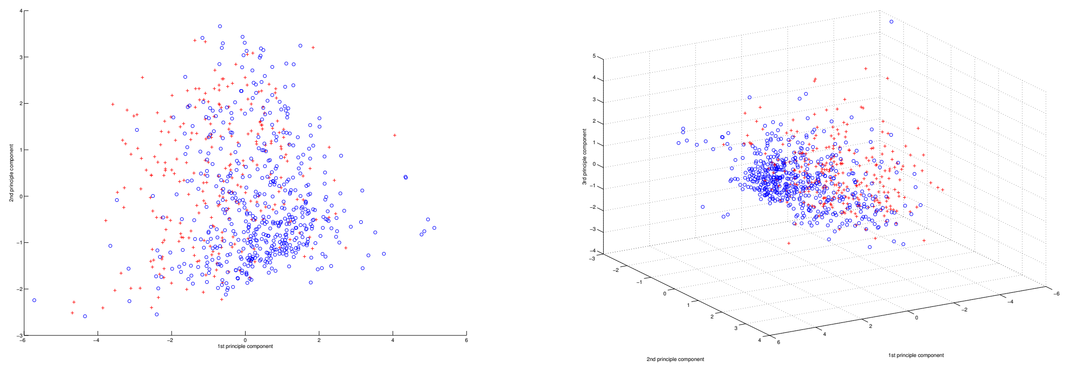


Figure 16: Left: Data projected onto the space formed by the first two principal components. Right: Data projected in the space formed by the first principal components

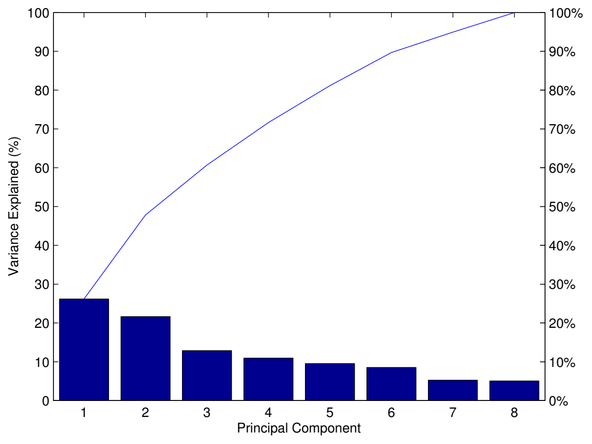


Figure 17: Variance contribution of the principal components

model complexity. Nothing was gained by increasing the population size beyond 200 or 250 in the case of the diabetes data. It is possible that the optimal algorithmic settings will change depending on the dataset. Genetic Programming performed as well as the more conventional methods of Neural Networks and Principal Component Analysis.

References

[1] Poli, Riccardo, W. B. Langdon, Nicholas F. McPhee, and John R. Koza. *A Field Guide to Genetic Programming*. [s. L.]: Lulu, 2008. Print.

[2] Schmidt M., Lipson H. (2009) "Distilling Free-Form Natural Laws from Experimental Data," Science, Vol. 324, no. 5923, pp. 81 - 85.

[3] www.vanillamodeling.com

[4] Lin Edwards. Eureqa, The Robot Scientist (PhysOrg), December 2009.

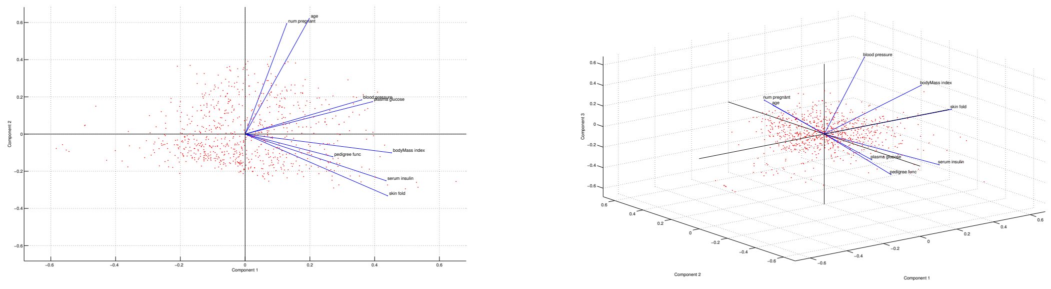


Figure 18: Left: Attribute vector plots in two principal components. Right: Attribute vector plots in three principal components

[5] Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261–265). IEEE Computer Society Press.

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Berhard Pfahringer, Peter Reutemann, Weka data mining software: An update. *SIGHDD Explor. Newsl.*, 11(1):10-18, 2009

[7] Simon Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 1998.

[8] Leonardo Vanneschi and Giuseppe Cuccu. A study of Genetic Programming variable population size for dynamic optimization problems, 2009

[9] Sara Silva, Pedro J.N. Silva, Ernesto Costa. Resource-Limited Genetic Programming: Replacing tree depth limits, 2005

[10] Sara Silva and Ernesto Costa, Comparing tree depth limits and resource-limited GP, 2005

PROBLEM 3: Convective Flow in Porous Media

Trenna Basu, University of South Carolina
Ming-Chun Chen, California State University, East Bay
Sean McNichols, Auburn University
Ben Silva, Florida State University
Wenli Zhou, University of Kentucky

Problem Presenter
Dr. Christopher Kees, U.S. Army Engineer Research and Development Center

Faculty Mentor
Dr. Lea Jenkins[1]

Abstract

The objective of this project is to model fluid flow through submerged vegetation and membranes to understand the effect of porous structures on natural and engineering processes. We use computational tools to generate the structures and to simulate the effect of the structures on fluid movement modeled by a finite element discretization of the two-dimensional Navier-Stokes equations. We then study the effects of the microscopic characteristics of the porous media on the macroscopic drag using the method of volume averaging and statistical tools.

This report contains a description of our modeling process, a discussion of the software tools used to obtain our results, and an analysis of the data obtained through our study.

1 Introduction

This project is motivated by a need to develop extended models for porous media flow through vegetated regions. In the 1850's, Henri Darcy derived a proportionality relationship between the velocity and the pressure for a variety of soil types. Darcy's Law incorporates the viscosity of the fluid, the permeability of the medium, and pressure gradient associated with the fluid. Darcy's Law represents a macroscale-averaged momentum equation, which means that it cannot be used to predict fluid flow on the microscale. However, for many media, including the vegetated media and membranes of interest in this work, understanding the microscale behavior of the fluid flow is vital to developing accurate simulations.

The macroscale behavior of porous media flow has been studied extensively in the groundwater and petroleum engineering fields. The relatively recent increase in computational power, the improved understanding of the physics, and the development of new mathematical and computational algorithms has improved our ability to model at the microscale. However, the

[1]Clemson University

microscale structures are still difficult to understand and are expensive to evaluate at the fine scale. Development of macroscale models for these structures allows for simulation on a larger scale and for simulation of a wider range of flow mechanisms.

Recent natural disasters, such as Hurricane Katrina, the explosion of the Blue Horizon platform, the tsunamis in Indonesia and Japan, and the Mississippi floods, have highlighted the need for accurate models of flow in near-shore regions. These disasters have been devastating in terms of their damage to structures, to human life, and to wildlife regions. Natural flood mechanisms include vegetated regions that exists in bays and estuaries. These regions change the dynamics of fluid movement by removing momentum from the fluid via fluid interactions with the solids regions of the structures. Thus, one idea for improving flood control might consider incorporating new vegetative areas or re-vitalizing existing vegetative areas in near-shore regions. However, especially as in the case of the Blue Horizon disaster, these wildlife regions become contaminated with non-organic substances transported by the fluid. Thus, in order to fully evaluate the effects of any natural disaster, or to measure the full environmental impact of introducing vegetative structures into near-shore regions, the fluid flow models must be accurate enough to generate accurate representations of contaminant transport.

Currently, the most accurate representation of flow through a porous media is realized if one uses the Navier-Stokes equations to model the flow through the network. This requires a microscale understanding of the pore structure, including information on the pore sizes and the pore connectivity network. This information is random, at best, meaning that flow models for a given structure would need to be uniquely determined. Our work seeks to develop a macroscopic flow model for a variety of fluid structures by considering a periodic cell of a given structure as a microscopic representation of the structure. Our objectives in this study are to construct a microscopic representation of a periodic cell; to solve the Navier-Stokes equations over this cell; to use the calculated velocity to develop and upscaled drag law for the macroscopic structure; and to model the drag coefficient as a function of the medium.

2 Mathematical Overview

We construct microscopic models of the flow through the porous medium using the Navier-Stokes equations. These equations are derived from conservation laws associated with mass and momentum. The classical form of the Navier-Stokes equations is given by

$$\nabla \cdot \mathbf{v} = 0 in \Omega \tag{1}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot \left[\mathbf{v} \otimes \mathbf{v} - \nu \left(\nabla \mathbf{v} + \nabla \mathbf{v}^t \right) + \frac{pI}{\rho} \right] - \mathbf{g} = 0 \in \Omega \tag{2}$$

where $\mathbf{v}$ is the fluid velocity, p is the fluid pressure, ρ is the fluid density, and $\mathbf{g}$ is gravity. The spatial domain $\Omega = \Omega_0 \backslash \Omega_s$ is the region within the unit square Ω_0 that is available for flow. Ω_s represents the solid portion of the domain, and we impose no-slip boundary conditions ($\mathbf{v} = 0$) along the solid boundaries that interact with the fluid. Periodic boundary conditions are imposed on the flow boundaries.

Note that $\mathbf{g}$ in the momentum equation above acts as an external force on the system. We altered the momentum balance in our periodic structures by changing the magnitude and direction of this term. Altering this momentum balances changes the velocity of the fluid, thus changing the Reynolds' number and the effects of drag along the solid boundaries.

Darcy's Law can be derived by upscaling the momentum equation above. For many pore structures, the momentum loss to the solid walls is significant, resulting in relatively slow flows through the domain of interest for which Darcy's Law is valid. However, in vegetated structures and many membrane structures, the flow through the non-solid region can be fast enough and/or the void spaces large enough so that Darcy's Law is violated.

In order to understand the connections between pore structures and fluid flow, we constructed several realizations of periodic cells. Our development environment was Python-based [1], and we generated our pore cells using Python scripts implemented in the Sage [2] notebook environment. We evaluated the Navier-Stokes equations using Proteus [**?**], a software tool developed by Dr. Kees and his colleagues Dr. Matthew Farthing and Dr. M.T. Fong at the US Army Engineer Research and Development Center Coastal and Hydraulics Laboratory. Proteus contains a variety of modules for solving flow problems using the finite element method. The Navier-Stokes implementation in Proteus is both mass- and momentum-conserving, which is important for this study. We also chose to consider steady-state solutions only in this work, so in the governing equations above we have $\frac{\partial \mathbf{v}}{\partial t} = 0$.

We will derive the net momentum of the fluid through the cell by considering the momentum loss occurring through the solid boundaries in Ω. Going back to the momentum equation above, and letting

$$\mathbf{f} = -\nu\left(\nabla \mathbf{v} + \mathbf{v}^t\right) + pI,$$

we see that

$$F_D = \int_{\partial\Omega} \mathbf{f} \cdot \mathbf{n}\, ds = \int_{\Omega} \nabla \cdot \mathbf{f}\, dV, \tag{3}$$

where F_D is the momentum loss through the solid boundaries, also known as the drag.

2.1 Drag

In fluid dynamics the drag equation is a formula that is used to calculate the force of drag experienced by an object due to movement through a fully-enclosing fluid. In our project we calculated the drag for the different porous media structures that we created. The force on a moving object due to a fluid is

$$F_D = \frac{1}{2}\rho \cdot v^2 \cdot C_D \cdot A \tag{4}$$

where F_D is the force of drag, ρ is the mass density of the fluid, v is the velocity of the object relative to the fluid, A is the reference area, and C_D is the drag coefficient-a dimensionless constant. The reference area A is typically defined as the area of the orthographic projection of the object on a plane perpendicular to the direction of motion. For non-hollow objects with simple shape, such as a sphere, this is exactly the same as a cross sectional area. One of the primary objectives of this project was to find a relationship between the drag and the Reynold's number. Later we will see that an increase in Reynold's number has significant effect on the drag. Furthermore, shape of the porous structure will also impact the drag's relationship to the velocity.

3 Approach

As mentioned earlier, we used the free software Python to build the periodic cell structures, Proteus to resolve the computational flow models, and Sage to integrate Python and Proteus and to generate visualizations of our results. We spent our first several days understanding models of the flow dynamics and developing periodic cell structures that would demonstrate interesting flow dynamics. The periodic cell represents a snapshot of the more complex media that is of interest, and replications of the periodic cell should, in some sense, reconstruct the more complicated structure. We focused on two-dimensional representations in this work, but the general concept is applicable to more general three-dimensional studies. Note that one shortcoming of this approach is the inability to understand the flow dynamics behind non-symmetric and non-periodic structures.

Examples of our periodic structures are shown in Figures 1.1 - 1.7. The pink edges in the figures represent the solid boundary walls through which momentum loss occurs. The regions in the figures outside any circular geometries are open to flow.

Our periodic cells were developed by first establishing a domain and describing the domain in terms of a set of vertices and segments. These vertices and segments are pieced together to generate a planar straight line graph. We often used parametric representations of curves to generate the vertices, but any ordered set of points could be used. The vertices and segments were flagged so that they could be associated with different boundary conditions. The segment flags were associated with different colors so that we could easily visualize the physics associated with the cells we drew. A closer examination of the periodic cell figures shows that solid objects in this porous media receive a flag color of pink, and each side that is considered free flowing will be a different color. Each wall receives a different color for free flowing pieces so that we can assign the direction of the flow through the media. In our cases a green color flag along a free flowing regional wall denotes an incoming flow and magenta denotes outbound flow.

In our first example we present a structure that is representative of square pores or square particles. The square portions of the figure that appear as cut outs in each othe the four corners represent the solid pore structures. We should expect to get free flowing fluid through the cross shaped structure. Below is a diagram that represents one period in a larger period square porous structure.

Figure 1.1

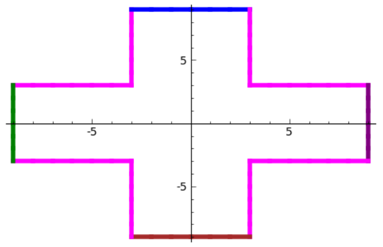
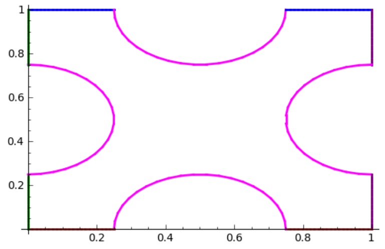


Figure 1.2



In our next example we present a structure that is representative of circular pores or disk like particles. The circular portions of the figure represent sections of the solid circular object. Each circle is obtained through parametric equations that can vary the distance between particles and also the size of each nodule. This depicts several types of structures that have similar molds but varying dimensions. This situation can arise in various soil samples with grains that are large in nature and smaller such as sand.

Figure 1.3

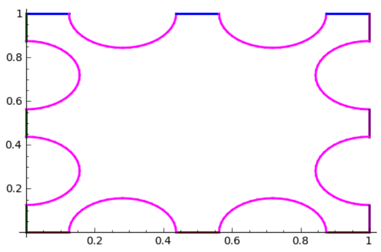


Figure 1.4

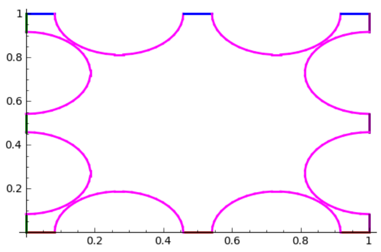
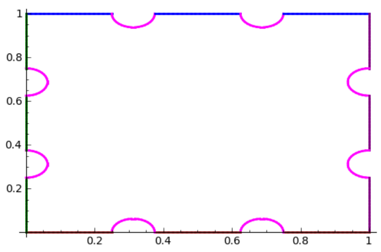


Figure 1.5



In our next structure it is a complex variation of the previous example using circular pores. In this example we present an addition feature that plots a center pore that is stationary. This stationary particle will most likely interrupt fluid flow across the porous media. The last structure consists of packing elliptic particles into a smaller space below a free flowing top. The elliptic particles will populate a small region located at the bottom of what appears to be a cross sectional square. The top of this dedicated region will consist of no pores and therefore be regarded as free flowing. The following figure illustrates precisely what we are trying to accomplish. This diagram will address issues involving boundary conditions at the transitional line or the division between the ellipses and void space.

Figure 1.6

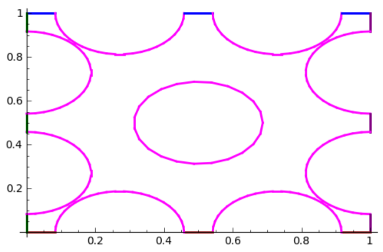
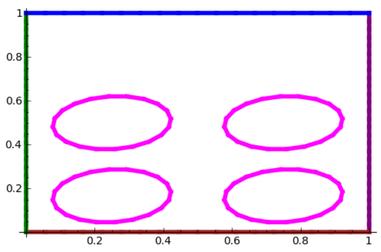


Figure 1.7



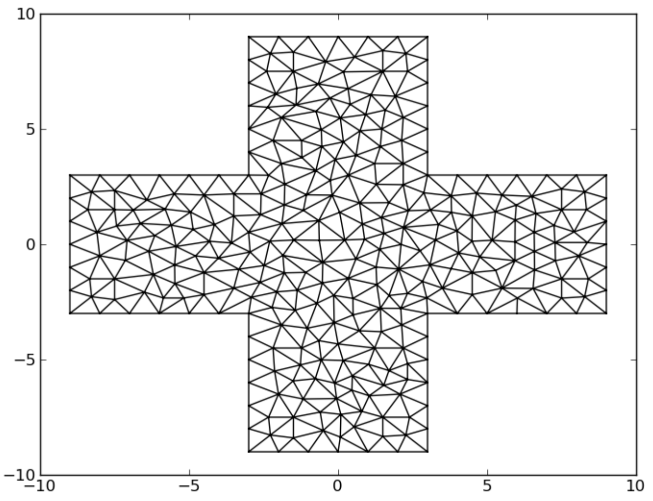
3.1 Mesh Generation

Since coming up with an analytical solution to the 2-dimensional Navier-Stokes equation is difficult in most cases, we need to come with up with a numerical solution using finite element method. This method consists of covering our domain with a triangular mesh and estimating the solution at each point of this lattice. Thus our next goal is to come up with a mesh to cover each of our domains. Further, we can achieve a better approximation by successively refining the mesh that is constructed. Our next step is to generate diagrams illustrating the velocity and pressure gradient of the fluid flow across this mesh or set of points determined by the numerous triangles. To provide a visual representation of this procedure we provide the following figure. This is the mesh for the square pore structure.

3.2 Pressure and Velocity Diagrams

In this section we present two slightly different interpretations of the flow through the media. The left hand graph in these examples will be representative of the pressure gradient across

Figure 0.2



the given structure. Notice that areas colored red have a high pressure since the flow across this media is hitting these solid boundary edges. The graphs that will be on the right hand side show the velocity of the fluid through the figure where the red regions will be indicative of high velocities and blue represents lower velocities. In the first representation fluid will flow left to right horizontally.To Illustrate the fluid flow as it relates to the velocity we have added a third graph on the far right which denotes velocity by arrow measures. Notice that slower blue arrows are forming vortices represented by circular pattern fluid movement.

Figure 0.3

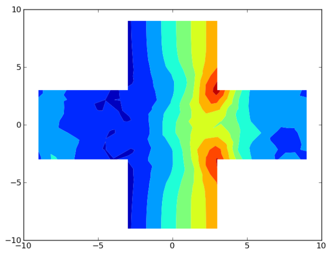


Figure 0.4

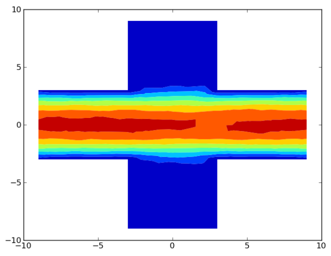
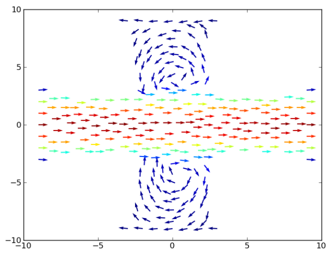


Figure 0.5



In this example of the same structure the fluid will be entering at an angle of 45 degrees or $\frac{\pi}{4}$ and pushed through the porous media. You will notice that the pressure is now highest only on the top right corner of the cross section and the velocity is highest in two directions through the center of the figure.

Figure 0.6

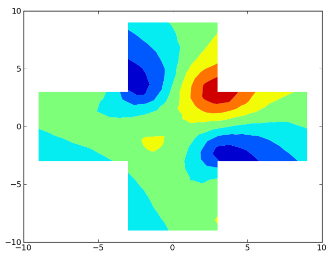
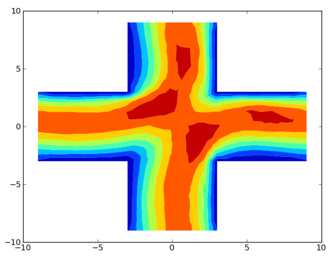


Figure 0.7



These following examples are of the same nature as the previous with regard to the pressure and the velocity across the structure. In these cases we are dealing with varying structures, circular particles and then also submerged ellipsoid particles. There are some nuances about each structure that should be noted. In the first with circular particles the structure can vary by the size of the particles and the distance between them. In the case of the circular particles, these changes in structure mapping would also have some affect on the drag, which will be discussed through statistical analysis. In the case of the submerged ellipsoid structure we have a region above the particles that is a free flowing region and you will notice higher velocities in this region. Also this structure is the best representation of the main objective of this project, which is to study fluid flow through submerged vegetation. The varying qualities in mapping this structure give way to changes in physics and numerics and will also be addressed through statistical analysis.

Figure 1.6

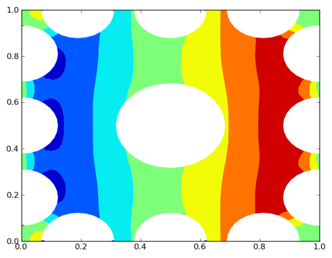


Figure 1.7

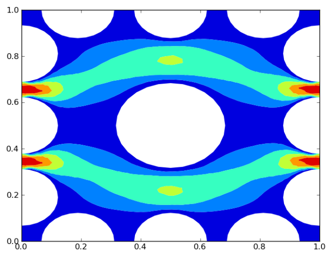


Figure 1.8

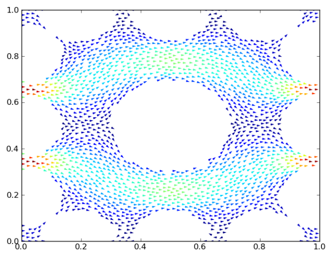


Figure 2.2

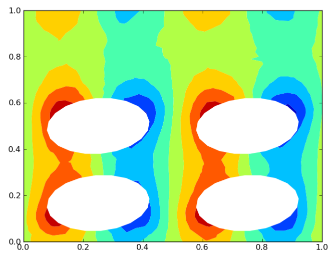
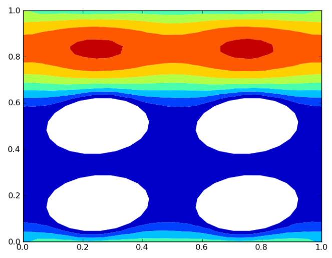


Figure 2.3



4 Submerged Vegetation

Thorough out the course of this study of various porous media we have gathered several structures and have tried to find their correlation to examples in nature. The best approximation is found by using the submerged ellipoid structure and changing several parameters to establish a diagram of what appears to be most like submerged vegetation, such as seaweed. Different from the previous diagrams involving ellipsoid structures you can see that the angle has been changed to allow the ellipsis to appear as almost vertical with small gaps in between solid objects. The reasoning for the gap between solid objects is to account for a porous structure that will move with the flow of fluid, such as the example of seaweed, a non-stationary solid object. The movement of the fluid through the gap is the best simulation of moving structures that can be achieved in this PYTHON environment. The following is a depiction of the fluid flow through this specific type of porous media in the same format that we have established, where the pressure is on the left side and the velocity on the right.

Figure 2.4

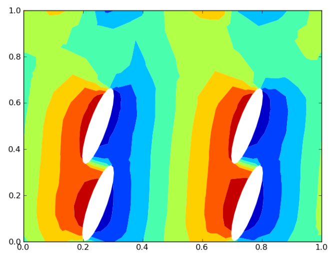


Figure 2.5

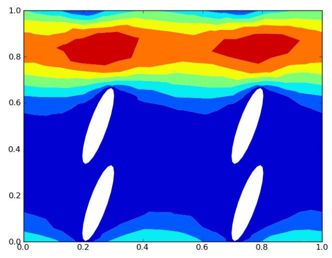
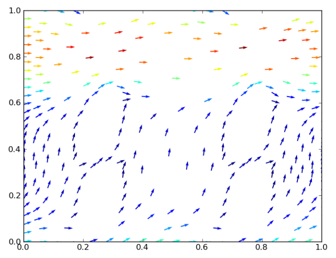


Figure 2.7

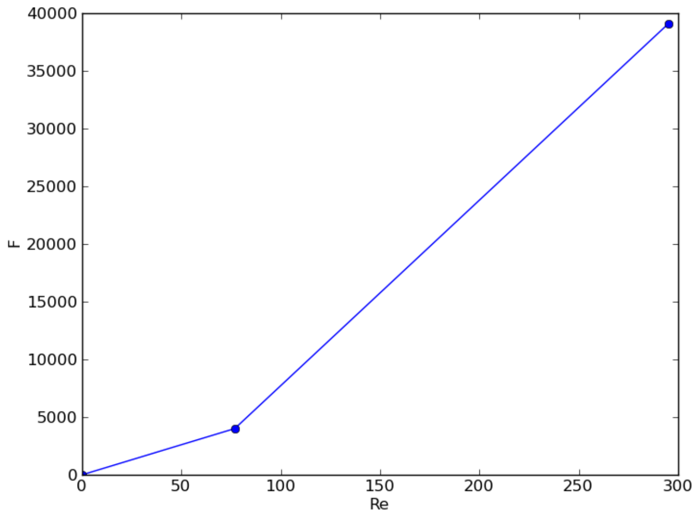


5 Affects on Drag Force

By altering the velocity through the porous media we were able to obtain some data that related the velocity and the drag. From this data we recognized the quadratic nature of the points and were able to develop functions of this nature to best fit the points on a curve. We can find a representation of this quadratic equation through the formulation of the F vs. Re relationship $F = \alpha V + \beta V^2$. We were able to generate a series of equations, each one representing a varying model of the given structure. This was done so that we could make observations about the variations in the structure and its affects on the quadratic equation. The first example that yielded some results involved the structure that contained the circular pores. The first plot represents the data achieved by plotting the Reynold's number, which is directly proportional to the velocity, versus the drag force.

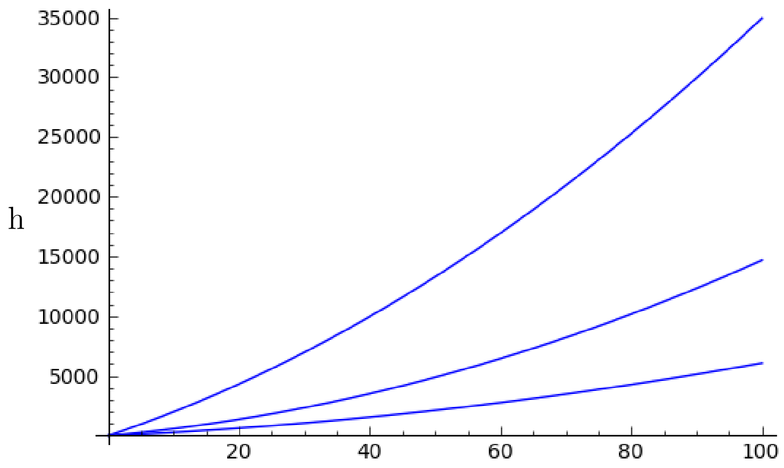
Notice that this representation of the data that we gather clearly would be best represented by a parabolic graph or quadratic equation. From this data we were able to use other statistical software, R, to generate the coefficients of the quadratic equation. Furthermore, this scenario was presented for three different variations of the same structure. The parameter that was slightly adjusted for each representation was the distance between circular pores. This distance had a significant effect on the other parameters, including the radius of the pores, the area of possible fluid flow, and also the Reynold's number. When we analyzed the data further and solved for the quadratic equations for the three varying structures we were able to obtain three curves which are on the same set of axes below also included in the table.

Figure 5.1



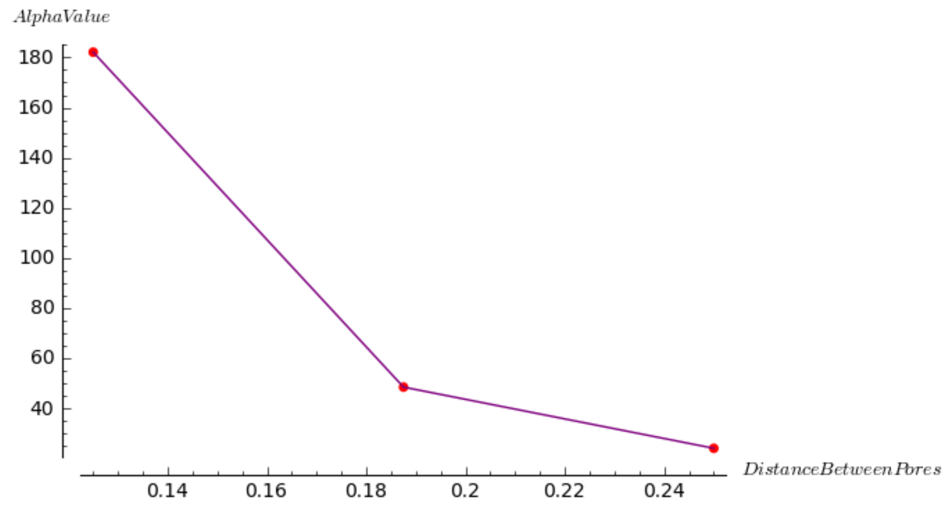
Top	$y = 182.1479x + 1.6702x^2$
Middle	$y = 48.32836x + 0.98412x^2$
Bottom	$y = 23.8780700x + .368035x^2$

Figure 5.2



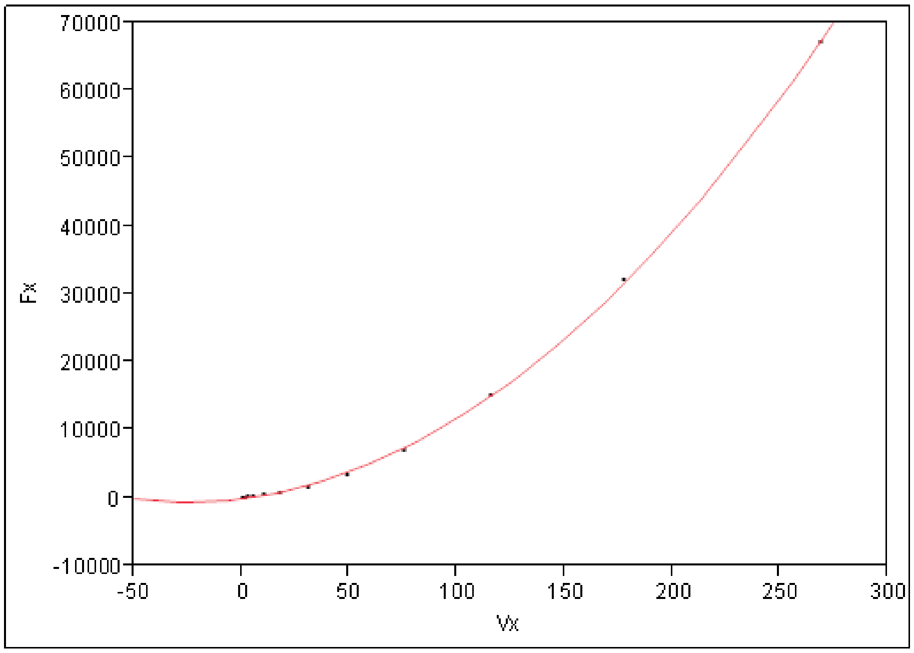
Notice that each curve on the set of axes is slightly quadratic, some more than others. One can also notice that because of the varying distance between circular pores we can see a change in equation that can be analyzed further. The top most curve is representative of the smallest distance between the circular pores, as the distance between the circular pores increases we have the middle and bottom curve in that order. The reasoning behind the separation of curves is suggested by the characteristic length, L, which is seen in the calculation of the Reynold's number. The characteristic length in this specific structure is represented by the area of media that is passable by fluid. Hence the least amount of area within the structure correlates to the graph that has the most drag at a given Reynold's number. When we say drag we can go further in saying that this is representative of momentum lost, since there is less space for the fluid to flow freely. Notice on this last graph we have plotted the α value in the quadratic equation versus the change in the distance between the pores, which is inversely related to the size of the circular pores. You will notice that in this graph as the distance decreases the α value increases dramatically which is indicative of a dramatic increase in the momentum lost or the drag. (Note that the α as mentioned before is the linear component coefficient of the quadratic representation of the Re vs F graph found in Figure 5.1)

Figure 5.3



The second structure that under went some testing and analysis was the pseudo seaweed environment which was constructed by manipulating the domain of the ellipsoid structure. We want to see the effect of changing the environment on momentum loss with velocity corresponding to Reynold's number. The quadratic formula for momentum due to drag force is $F = \alpha v + \beta v^2$. First we find the function of α and β by changing radius of ellipse, horizontal and vertical axis and the depth of the overhead water. For example, we change the value of gravity and keep the other variables such as ellipticity, depth of the water, angle of ellipse constant. We then keep the gravity constant and manipulate other variables one at a time. Changing each variable one at a time creates a different looking structure and in turn a different looking flow. With these variations we eventually end up mapping several different quadratics to best fit the plot of Re vs. F. Once α and β are calculated we can then monitor how that equation will be affected by the change in the specific variable. For example, using the ellipsoid structure we can vary the angle by a specific amount, then draw a best fit line with respect to a quadratic equation. Then reverse the procedure and use the best fit equation as a function of velocity to map that equation against a fluctuation in an angle change. Using this data you can make further correlations between the change in the α value and the change in θ. Below you can see the graph of a quadratic best fit line of Vx vs Fx. This graph can be see through the equation

$$Fx = -1457.011 + 106.05626Vx + 0.7813194(Vx - 41.8634)^2$$



Upon further investigation of the submerged ellipsoid structure we can vary the depth of the ellipsoids while holding all other variables constant. We analyzed the relationship between the β value of the quadratic curve and the variable that we are changing (depth). The resulting output through R is a log transform graph with an adjusted value of .9939, where the value was .8433 before the transformation. This high of an adjusted value for this type of model indicates that there is a strong correlation between the β value and the depth. Also notice that the second graph shows the correlation between α and the depth, represented again by a natural log (ln) model. Choosing the natural log model was due to the fact that the specific data points could not be represented by a quadratic curve. Choosing a linear model and then applying a transformation as we did the α comparison we achieve a R-adjusted value of .973. We also went further to investigate the correlation between the change in the angle that creates the ellipsoid and the quadratic coefficients. The log equation that represents the data involving an angle change shows that as the degree increases so will the α and β values. Notice also that if the angle is 0°, α and β are small, and when the angle is 90° we expect α and β to be highest, meaning that we have the most drag. You will find below the equations of the relationship that best relates the quantities in question.

Figure 5.5	$ln(\alpha)$ vs Depth	$ln(\alpha) = 2.92319 - .20398(\text{Depth})$
Figure 5.6	$ln(\beta)$ vs Depth	$ln(\beta) = -1.515387 - 0.228310(\text{Depth})$
Figure 5.7	$log(\alpha)$ vs sin(Angle)	$ln(\alpha) = -.22275 + 3.2107\sin(\text{Angle})$
Figure 5.8	$log(\beta)$ vs sin(Angle)	$ln(\beta) = -4.8003 - 3.1191\sin(\text{Angle})$

Figure 5.5

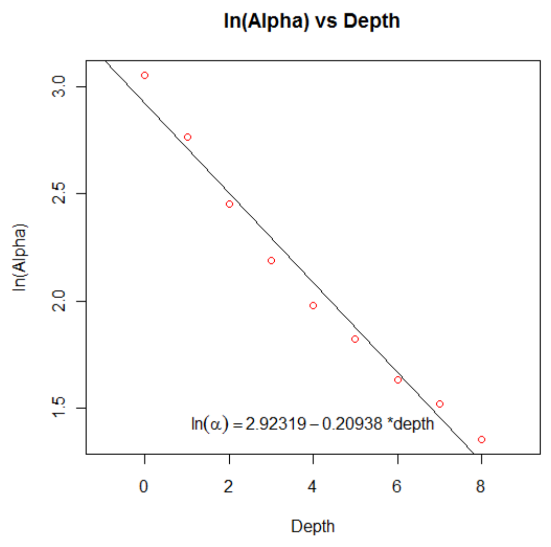


Figure 5.6

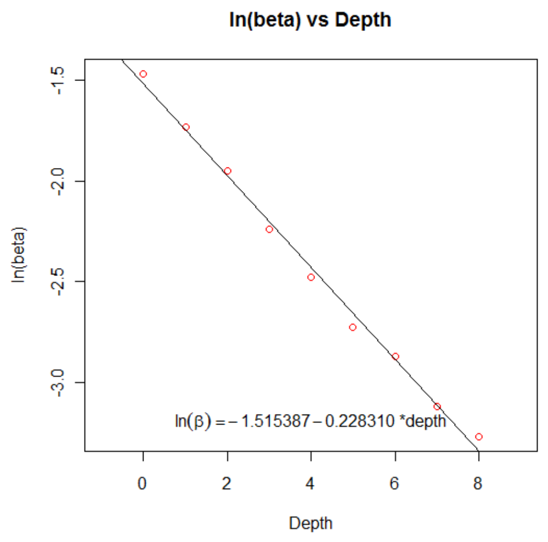


Figure 5.7

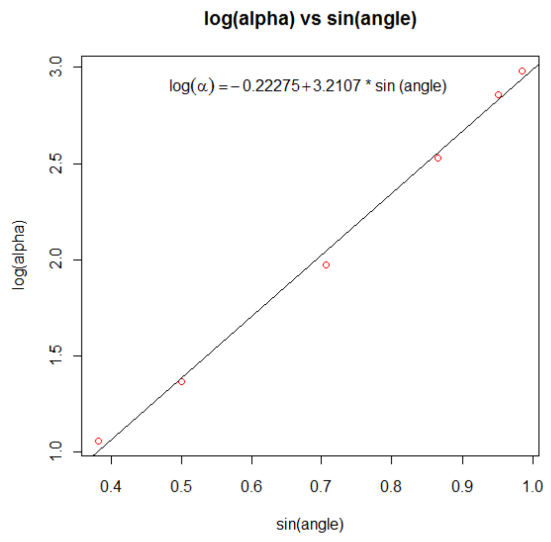
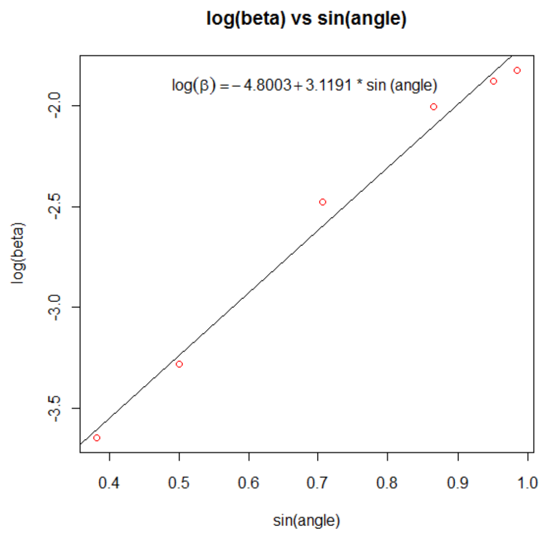


Figure 5.8



There is a very nice tool found in the mathematical community called PYTHON. It's ability to plot and to use various physics equations along with mathematical calculations enabled us to run various tests given specific inputs. To assist us in our computational efforts we implemented a section of the python script called Proteus. Proteus enabled us to use our mapped porous regions as inputs and would generate numerical solutions to the physics equations that govern the fluid flow through a given domain. Without the script Proteus the computations would have been tedious and overbearing. PYTHON is a free tool to all students who are willing to take the time to fully understand its potential in mathematical modeling. These structures were drawn to best represent real world application problems, but much of our work was preliminary and we needed a base line model with an easy representation. Having such a model allowed us to understand the way PYTHON and also the physics within the software worked so that we can achieve further models with more complicated structures. Our dealings with PYTHON and Proteus were run through a front end program, also free, called SAGE. Sage allowed us to bypass the user interface difference between operating systems and allow each team member the ability to access pieces of code regardless of machine.

6 Future Work

This project or idea can be extended to study the case of fluid flow through various vegetation and possibly other examples of porous media in our world today. Also this procedure can be used in a more rigorous modeling situation to fully understand weather disasters such as events that cause storm surge along a coastal front. Further, we have just touched the surface of the potential that this research has for mathematical modelers and also for those working in industry.

For the future research involving incoming coastal waves, one question that stemmed from our thinking was the recession of the fluid through such events as undertow and how fluid will flow when the direction is reversed. Also, possibly looking at media that have movable parts where the fluid carries smaller particles through the media and the effects that collision of material has on the fluid flow.

References

[1] http://www.python.org/.

[2] http://www.sagemath.org/.

[3] A. Bensoussan, J. Lions, and G. Papanicolaou. *Asymptotic analysis for periodic structures.* Stud. Math. Appl. 5. North-Holland, Amsterdam, 1978.

[4] A. Bourgeat, E. Marusic-Paloka, and A. Mikelic. Convective flow in porous media. *Mathematical Models and Methods in Applied Sciences*, 8(6):1–13, 1996.

[5] H. Darcy. *Les Fontaines Publique de La Ville de Dijon.* Librairie des Corps Imperiaux des Ponts et Chaussees et des Mines, Paris, 1856.

[6] L.C. Evans. *Partial Differential Equations*, chapter 4.5.4. AMS, 2010.

[7] P. Forchheimer. Wasserbewegung durch boden. *Z. Ver. Deutsch. Ing.*, 45:1782–1788, 1901.

[8] C. Garibotti and M. Peszynska. Upscaling non-Darcy flow. *Transport in Porous Media*, 80(3):401–430, 2009.

[9] Marco Ghisalberti and Heidi Nepf. Shallw flows over a permeable medium: The hydrodynamics of submerged aquatic canopies. *Transport in Porous Media*, 78:309–326, 2009.

[10] U. Hornung. *Homogenization and Porous Media.* Springer, 1997.

[11] Oleg Iliev, Andro Mikelic, and Peter Popov. On upscaling certain flows in deformable porous media. *Simulation*, 7(1):93–123, 2008.

[12] S.A. Mattis, C.N. Dawson, C.E. Kees, and M.W. Farthing. Upscaling of flow through porous structures and vegetated domains. in review, 2011.

[13] Donald A. Nield and Adrian Bejan. *Convection in Porous Media*. Springer, 2006.

[14] M. Peszynska, A. Trykozko, and K. Augustson. Computational upscaling of inertia effects from porescale to mesoscale. In G. Allen, J. Nabrzyski, E. Seidel, D. van Albada, J. Dongarra, and P. Sloot, editors, *ICCS 2009 Proceedings*, LNCS 5544, Part I, pages 695–704, Berlin-Heidelberg, 2009. Springer-Verlag.

[15] M. Peszynska, A. Trykozko, and K. Kennedy. Sensitivity to anisotropy in non-Darcy flow model from porescale through mesoscale. In *Proceedings of CMWR XVIII*, June 2010.

[16] M. Peszynska, A. Trykozko, and W. Sobieski. *Forchheimer Law in computational and experimental studies of flow through porous media at porescale and mesoscale*, volume 32 of *GAKUTO International Series, Mathematical Sciences Applications*, pages 463–482. Gakkotosho, Tokyo, 2010.

[17] H. Rouse. *Elementary Mechanics of Fluids*. John Wiley and Sons, 1946.

[18] Luc Tartar. *The General Theory of Homogenization: A Personalized Introduction*. Springer, 2010.

A Codes

A.1 Example of Domain Construction in Python

```
#Minimum Distance of a Segment
ds=1/36
#Number of Segments on one boundary of the Graph
Seg=3
Cir=3
#Distance between Circles.
Distance=4/16


def g(Cir,Seg,Distance,ds):
    Arb=0
    if Seg==Cir:
        Arb=Cir
        Cir=Cir-1
        print(Cir,Seg)
    print(Cir)
    Radius=(1-(Seg*Distance))/(Cir*2)

    Length = Distance
    ns=ceil(Length/ds)
    ns2=ceil(Radius*pi/ds)
    ds = (Length)/float(ns) #actual segment width

    x=var('t')
    vlist=[]
    r=Radius
    d=Distance
    TotalLength = Cir*r*2+Seg*d
    print(Cir)
    Centerx = d + r
    Centery = d + r
    i=0
    L=pi*r
    angle=pi/ns2
    Fstep=d/ns
    a=pi
    j=0
    l=0
    #top
    vflag=[]
    slistF=[]
```

```
while l<Seg:
    j=0
    while j<ns:
        vlist.append((0+Fstep*j+(l*(d+2*r)),TotalLength))
        vflag.append(2)
        slistF.append(2)
        j=j+1
    j=0
    if l<Cir:
        while j<ns2:
            vlist.append((r*cos(a+angle*j)+(Centerx)+((2*r+d)*l) ,
                                         r*sin(a+angle*j)+TotalLength))
            j=j+1
            vflag.append(5)
            slistF.append(5)
    l=l+1
#right
a=pi/2
j=0
l=0
while l<Seg:
    j=0
    while j<ns:
        vlist.append((TotalLength,TotalLength-(Fstep*j)-(l*(d+2*r))))
        vflag.append(3)
        slistF.append(3)
        j=j+1
    j=0
    if l<Cir:
        while j<ns2:
            vlist.append((r*cos(a+angle*j)+TotalLength ,
                               r*sin(a+angle*j)+(Centery)+
                                ((2*r+d)*(Cir-(l+1)))))
            j=j+1
            vflag.append(5)
            slistF.append(5)
    l=l+1
#Bottom
a=0
j=0
l=0
while l<Seg:
    j=0
    while j<ns:
        vlist.append((TotalLength-(Fstep*j)-(l*(d+2*r)),0))
        j=j+1
        vflag.append(4)
        slistF.append(4)
    j=0
    if l<Cir:
        while j<ns2:
            vlist.append( (r*cos(a+angle*j)+Centerx+(2*r+d)*(Cir-(l+1)),
                               r*sin(a+angle*j) ))
            j=j+1
            vflag.append(5)
            slistF.append(5)
    l=l+1
#Left
a=-pi/2
j=0
l=0
while l<Seg:
    j=0
    while j<ns:
        vlist.append((0,0+(Fstep*j)+(l*(d+2*r))))
        j=j+1
        vflag.append(1)
        slistF.append(1)
    j=0
    if l<Cir:
        while j<ns2:
            vlist.append( (r*cos(a+angle*j),
                 r*sin(a+angle*j)+Centery+(2*r+d)*l ))
            j=j+1
            vflag.append(5)
            slistF.append(5)
    l=l+1


r=Radius
Number=len(vlist)

slist=[]
for i in range (Number):
    slist.append([i,(i+1) % (Number)])

F=0
f=0
print(Seg,Arb,Cir)
if Seg==Arb:
        angle=pi/ns2
        i=0
        Middle = 1/2
        while f<2*ns2:
            vlist.append(( (r*cos(angle*f)+Middle) ,
             r*sin(angle*f)+Middle ))
            if f<2*ns2-1:
                slist.append([Number+i,Number+1+i])
            f=f+1
            i=i+1
            vflag.append(5)
            slistF.append(5)
        slist.append([Number+i-1,Number])
    A=1-Cir*2*(pi*Radius^2)-(pi*Radius^2)
    L=r*pi*Cir
    print(Arb,Seg,Cir)
    return(vlist,vflag,slistF,slist,ns,ds,A)

(myvertices,vflag,slistF,slist,ns,ds,A)=g(Cir,Seg,Distance,ds)
```

A.2 R Code

```
set.seed(100)
x.2 = floor(runif(512,0,150))
set.seed(40)
x.1 = rnorm(512); y.1 = bubble.sort(x.1)


bubble.sort <- function(x) {
  Length.x <- length(x)
  repeat {
    Change <- FALSE
    Length.x <- Length.x - 1
      for (i in 1:Length.x) {
        if (x[i] > x[i+1]) {
          y <- x[i]
          x[i] <- x[i+1]
          x[i+1] <- y
          Change <- TRUE
        }
      }
     if (!Change) break;
    }
  x
}

insertion.sort <- function(x)
{
   for (i in 2:(length(x)))
     {
        temp <- x[i]
        j <- i-1
        while (j >= 1 && x[j]>temp)
        {
          x[j+1] <- x[j]
          j <- j-1
        }
        x[j+1] <- temp
     }
x
}

merge.sort <- function(a) #create function a
  {
    merge.sort.temp <- function(in1, in2) #create function with 2 variables
    {
       temp <- c()
       while(length(in1) > 0 && length(in2) >0)
       {
      if(in1[1] <= in2[1]) #left is smaller or equal to right
        {
          temp <- c(temp, in1[1])
          in1 <- in1[-1]
        }
         else
        {
          temp <- c(temp, in2[1])
          in2 <- in2[-1]
        }
     }
       if(length(in1) > 0) temp <- c(temp, in1)
       if(length(in2) > 0) temp <- c(temp, in2)
       temp
     }
   Length.a <- length(a)
   if(Length.a <=1) a else
   {
     mid <- length(a)/2
     in1 <- a[1:floor(mid)]
     in2 <- a[floor(mid+1):Length.a]
     in1 <- merge.sort(in1)
     in2 <- merge.sort(in2)
     if(in1[length(in1)] <= in2[1])
   {
     c(in1,in2)
   } else
   {
      merge.sort.temp(in1,in2)
   }
 }
}



t.0 = proc.time()
p.1 = bubble.sort(x.2[1:128])
p.2 = bubble.sort(x.2[129:256])
p.3 = bubble.sort(x.2[257:384])
p.4 = bubble.sort(x.2[385:512])
p.5 = insertion.sort(x.2[1:128])
```

```
p.6 = insertion.sort(x.2[129:256])
p.7 = insertion.sort(x.2[257:384])
p.8 = insertion.sort(x.2[385:512])
p.12 = merge.sort(c(p.1,p.2))
p.34 = merge.sort(c(p.3,p.4))
p = merge.sort(c(p.12,p.34))
proc.time(0) V t.0
```

```
set.seed(100)
x.2 = floor(runif(512,0,150))
set.seed(40)
x.1 = rnorm(512); y.1 = bubble.sort(x.1)
```

```
bubble.sort <- function(x) {
  Length.x <- length(x)
  repeat {
    Change <- FALSE
    Length.x <- Length.x - 1
      for (i in 1:Length.x) {
        if (x[i] > x[i+1]) {
          y <- x[i]
          x[i] <- x[i+1]
          x[i+1] <- y
          Change <- TRUE
        }
      }
     if (!Change) break;
    }
  x
}

merge.sort <- function(a) #create function a
  {
    merge.sort.temp <- function(in1, in2) #create function with 2 variables
    {
       temp <- c()
       while(length(in1) > 0 && length(in2) >0)
       {
      if(in1[1] <= in2[1]) #left is smaller or equal to right
        {
          temp <- c(temp, in1[1])
          in1 <- in1[-1]
        }
         else
        {
          temp <- c(temp, in2[1])
          in2 <- in2[-1]
        }
     }
       if(length(in1) > 0) temp <- c(temp, in1)
       if(length(in2) > 0) temp <- c(temp, in2)
       temp
     }
   Length.a <- length(a)
   if(Length.a <=1) a else
   {
     mid <- length(a)/2
     in1 <- a[1:floor(mid)]
     in2 <- a[floor(mid+1):Length.a]
     in1 <- merge.sort(in1)
     in2 <- merge.sort(in2)
     if(in1[length(in1)] <= in2[1])
   {
     c(in1,in2)
   } else
   {
      merge.sort.temp(in1,in2)
   }
 }
}

t.0 = proc.time()
p.1 = bubble.sort(x.2[1:128])
p.2 = bubble.sort(x.2[129:256])
p.3 = bubble.sort(x.2[257:384])
p.4 = bubble.sort(x.2[385:512])
p.12 = merge.sort(c(p.1,p.2))
p.34 = merge.sort(c(p.3,p.4))
p = merge.sort(c(p.12,p.34))
proc.time(0) - t.0
```

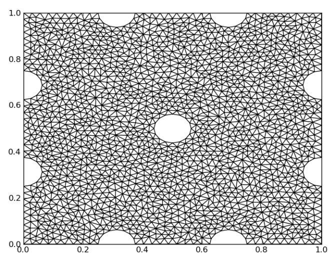
```
insertion.sort <- function(x)
{
   for (i in 2:(length(x)))
     {
        temp <- x[i]
        j <- i-1
        while (j >= 1 && x[j]>temp)
        {
          x[j+1] <- x[j]
          j <- j-1
        }
        x[j+1] <- temp
     }
x
}

p.5 = insertion.sort(x.2[1:128])
p.6 = insertion.sort(x.2[129:256])
p.7 = insertion.sort(x.2[257:384])
p.8 = insertion.sort(x.2[385:512])
```

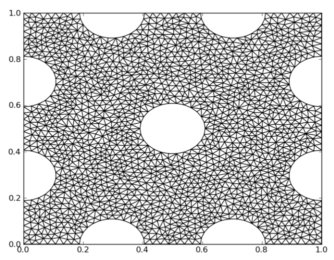
B More Models of Structures

B.1 Meshes of Varying Circular Structures

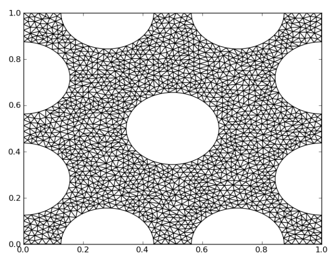
Small



Medium

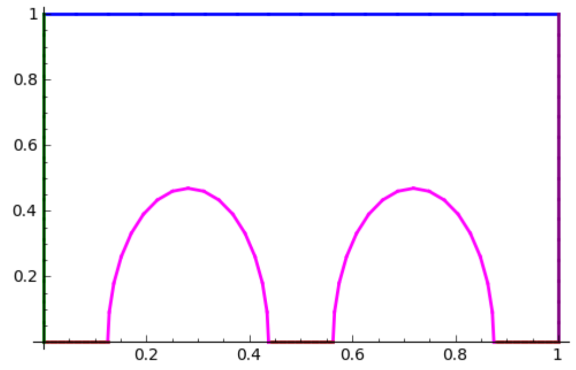


Large

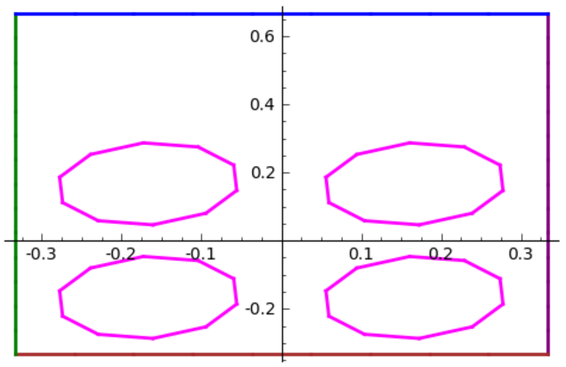


B.2 Various Structures

Mounds

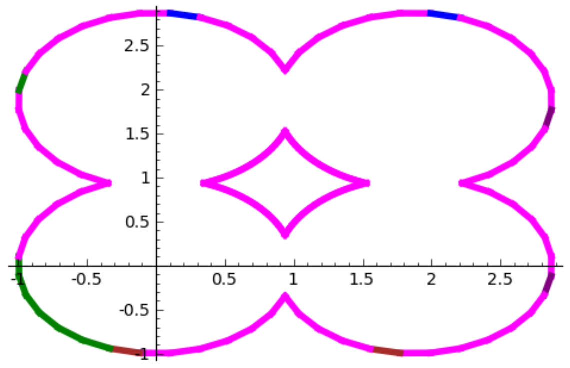


Submerged Ellipsoid

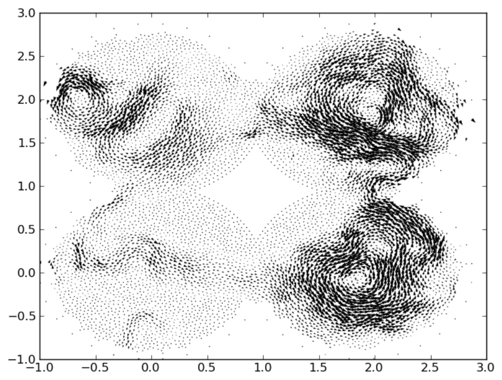


B.3 Honeycomb Structure

Basic Structure



Fluid Flow Diagram



Project 5: Experimental design and inverse problems in plant biological modeling

Matthew Avery[1], Kanadpriya Basu[2], Yansong Cheng[3], Eric Eager[4], Sarah Khasawinah[5], Keri Rehm[6]

Problem Presenter:	Faculty Mentor:
Dr. Laura Potter	Dr. H.T. Banks
Syngenta Biotechnology	North Carolina State University

Abstract

We develop a mathematical and statistical framework to model the actions of underlying metabolites for carbon dioxide assimilation in photosynthesis. This study was motivated by a challenge posed by Syngenta Biotechnology to use modeling to better characterize photosynthesis in plants. We use a dynamical system model proposed by Zhu., et. al. [1], which describes the Calvin Cycle in spinach plants through changes in the concentrations of 38 metabolites using non-linear enzyme kinetic ordinary differential equations and mass-balance laws that contain a total of 165 parameters. In our study of the CO_2 assimilation rate, we pose our research questions with this dynamical system mathematical model and a statistical model to describe the observation process. In particular, we answer three distinct questions:

1. What parameters would be best to estimate?
2. Given simulated data, can we estimate some of the model parameters?
3. Which metabolites are optimal to measure in the first place?

We answer the first problem using sensitivity analysis, which elucidates the dependence of the model's outputs on parameter behavior. We use parameter estimation theory, following the ordinary least squares approach described in [5] to solve the second question. The third question, the problem of choosing the best subset of metabolites to measure, is not well-explored in the existing literature. Here, we propose two *ad hoc* methods. The first, the Simple Linear Regression (SLR) method, models carbon assimilation as a linear function of the metabolites. Using simulated data, we implement a step-wise selection algorithm to determine the best subset of metabolites at each of 10 fixed time points. The second method, the Sum of Squares method, fixes the number of metabolites to be observed and searches for the best combination by sampling from all possible combinations of metabolites. Both methods show promising results.

Our conclusions represent a paradigm for estimating parameters and designing experiments more efficiently in plant biological modeling. Ultimately, our results may be used to help engineer seeds that maximize carbon dioxide assimilation in photosynthesis.

1 Introduction and Motivation

This investigation was motivated by a challenge posed by Syngenta Biotechnology to help better characterize photosynthesis in plants. Understanding how to make photosynthesis more efficient is one approach for improving yield, which is critical for helping farmers meet the impending global challenge to "grow more with less."

In today's world, the demand for more food and energy is skyrocketing while resources are rapidly declining. The population is growing, urbanization is continuing, diets are changing, and our need for more energy as fuel is rising; however, we do not have the land or water resources to sustain this inevitable modernization. Agriculture already uses 40% of the world's land surface and 70% of all available freshwater. Caught in this paradox of producing enough food and fuel for the world while resources are running out, we need to think outside of the box. Mathematics and statistics provide a means towards better understanding these problems and a path towards solving them in a more efficient way.

[1]North Carolina State University
[2]University of South Carolina
[3]Boston University
[4]University of Nebraska-Lincoln
[5]Johns Hopkins University
[6]North Carolina State University

In this paper we introduce analyze the behavior of a mathematical and statistical model of carbon dioxide assimilation in the Calvin Cycle. Our model consists of a system of coupled differential equations describing the relationship between the metabolites involved in the Calvin Cycle [1] and a corresponding inverse problem framework utilizing the statistical model to determine optimal parameter estimates and experimental design.

Our results illustrate that by modifying the ratios of the concentrations of enzymes involved in carbon metabolism or the parameters describing the efficiency of the reactions that use these metabolites, the photosynthetic rate can be changed. In particular, the model output is most sensitive to a few parameters. In addition to aforementioned inverse problem approach, we also develop an innovative algorithm to determine which metabolites are best to measure. These findings in optimal experimental design and parameter estimation theory in plant biological modeling could help to develop more efficient seeds to yield better results despite diminishing resources.

2 Biology Background

Photosynthesis is the process by which plants convert solar energy, using carbon dioxide and water, into chemical energy. The plant stores the chemical energy in the form of glucose and uses this carbohydrate energy source to grow. The chemical equation for photosynthesis in plants is:

$$6CO_2 + H_2O = C_6H_{12}O_6 + 6CO_2 + 6H_2O.$$

Photosynthesis takes place in chloroplasts and involves two stages:

1. Energy Transferring Reactions
2. Carbon Fixation Reactions

In the Energy Transferring stage, pigment molecules capture energy from the sun and store it temporarily in the form of ATP and NADPH. In this process, one molecule of the pigment chlorophyll absorbs one photon and loses one electron. Then, in a process called photolysis, the chlorophyll molecule uses water to regain the lost electron, which releases oxygen.

In the Carbon Fixation stage, ATP and NADPH are used in a sequence called the Calvin Cycle to repackage the energy as sugar and starch. The Calvin Cycle is an intricate series of reactions that involves 38 known enzymes. These details are included in the model to determine which reactions are most important in carbon fixation.

This model is for the Calvin Cycle of a spinach leaf [1], though this prototype could be easily generalized to other plants. We use both statistics and applied mathematics to develop our inverse problem approach. Because the observation process inherently contains errors, which affect resultant parameter estimates, and the system is naturally dynamic, the model will involve a system of coupled ordinary differential equations and a corresponding statistical model to characterize the errors in measurement. We take the ordinary differential equations from this model of the Calvin Cycle and formulate the statistical model using parameter estimation theory in Banks [5].

3 Modeling Techniques

In general, we will follow the notation in [5], while using the differential equations in [1]. Solving this problem requires both a mathematical and a statistical model, which we define below. The mathematical model is given by

$$\frac{d\vec{x}}{dt}(t) = \vec{g}(t, \vec{x}(t), \vec{\theta}) \tag{1}$$

with the associated observation process

$$\vec{f}(t, \vec{\theta}) = \mathcal{C}(\vec{x}(t; \vec{\theta})). \tag{2}$$

The vector $\vec{x}(t;\vec{\theta})$ describes the $m = 38$ state variables, which in this paper represents the concentrations of the metabolites involved in this process and $\vec{\theta}$ is a vector of p parameters. In general, the observation operator $\mathcal{C}$ is some transformation of these state variables, but for our model, we consider only such operators whose output is a subset of $\vec{x}(t;\vec{\theta})$, i.e. $\mathcal{C}$ is a linear transformation (a matrix) that maps $\mathbb{R}^{38}$ into $\mathbb{R}^{p^*}$, where $p^* < 38$ is the number of observed variables.

In practice, a continuous set of data $\vec{f}$ is not observed. Rather, we often observe the state variables at discrete time points $\{t_j\}$, $j = 1, 2, \ldots, n$ giving us a realization of the observation vector

$$\vec{f}(t_j, \vec{\theta}) = \mathcal{C}(\vec{x}(t_j; \vec{\theta})), \quad j = 1, \ldots, n. \tag{3}$$

Because the system has observation error, we formulate the statistical model as

$$\vec{Y}_j = \vec{f}(t_j, \vec{\theta}_0) + \vec{\mathcal{E}}_j, \quad j = 1, \ldots, n. \tag{4}$$

Here, $\vec{Y}_j$ is the vector of state variables observed at the jth time point, $\vec{f}(t_j, \vec{\theta}_0)$ is as in (2), and $\vec{\theta}_0$ is the vector of true parameter values. The $\vec{\mathcal{E}}_j$ is a vector of random variables with constant finite variance over the time and independent across the state variables. No further distributional assumptions are made on the errors. A realization of (4) is

$$\vec{y}_j = \vec{f}(t_j, \vec{\theta}_0) + \vec{\epsilon}_j, \quad j = 1, \ldots, n. \tag{5}$$

The goal of the inverse problem is to determine the optimal $\vec{\theta}$ given the data $\vec{y}_j$. To do this we use ordinary least squares to estimate $\vec{\theta}_0$. That is, we aim to find an estimate $\hat{\theta}_{\text{OLS}}$ of $\vec{\theta}_0$ such that

$$\hat{\theta}_{\text{OLS}} = \arg\min_{\vec{\theta} \in \Theta} \sum_{j=1}^{n} [\vec{y}_j - \vec{f}(t_j, \vec{\theta})]^T V_0^{-1} [\vec{y}_j - \vec{f}(t_j, \vec{\theta})], \tag{6}$$

where Θ is the class of all admissible sets of parameters and

$$V_0 = \text{var}(\vec{\mathcal{E}}_j) = \text{diag}(\sigma_{0,1}^2, \ldots, \sigma_{0,m}^2). \tag{7}$$

We estimate V_0 by

$$V_0 \approx \hat{V} = \text{diag}\left(\frac{1}{n-p} \sum_{j=1}^{n} [\vec{y}_j - \vec{f}(t_j, \hat{\theta})][\vec{y}_j - \vec{f}(t_j, \hat{\theta})]^T \right). \tag{8}$$

Using these estimated variances along with sensitivity variables, we can estimate the covariance matrix for our OLS parameter estimates. For state variable i and parameter k, define the sensitivity variable as

$$\frac{\partial f_i(t, \vec{\theta})}{\partial \theta_k}, \quad i = 1, \ldots, m, \quad k = 1, \ldots, p, \tag{9}$$

where $f_i(t, \vec{\theta})$ is the ith component of $\vec{f}(t, \vec{\theta})$. Then we may calculate

$$D_j(\vec{\theta}) = \begin{pmatrix} \frac{\partial f_1(t_j, \vec{\theta})}{\partial \theta_1} & \frac{\partial f_1(t_j, \vec{\theta})}{\partial \theta_2} & \cdots & \frac{\partial f_1(t_j, \vec{\theta})}{\partial \theta_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m(t_j, \vec{\theta})}{\partial \theta_1} & \frac{\partial f_m(t_j, \vec{\theta})}{\partial \theta_2} & \cdots & \frac{\partial f_m(t_j, \vec{\theta})}{\partial \theta_p} \end{pmatrix}, \tag{10}$$

the $m \times p$ sensitivity matrix at time j. Then the covariance matrix of θ_{OLS} is approximated by

$$\hat{\Sigma}_n = \left(\sum_{j=1}^{n} D_j^T(\hat{\theta}) \hat{V}^{-1} D_j(\hat{\theta}) \right)^{-1}, \tag{11}$$

and the standard error for the kth parameter in θ_{OLS}, labeled $\theta_{\text{OLS},k}$ can be estimated as

$$SE(\theta_{\text{OLS},k}) \approx \sqrt{\hat{\Sigma}_{kk}}. \tag{12}$$

Further discussion of sensitivity matrices, sensitivity variables, and their use in sensitivity analysis is given in the next section.

3.1 Sensitivity Analysis

Broadly defined, sensitivity analysis is a set of techniques that can be used to elucidate the dependence of a model's outputs on parameter behavior. In biological problems, the number of parameters are often sufficiently high to where it is computationally costly or even impossible to attempt to estimate all of them in the inverse problem. In such situations, it is necessary to choose some parameters to estimate and fix the remaining parameters at reasonable estimates. Sensitivity analysis can be employed to help make such decisions. The $m \times p$ sensitivity matrix, $D_j(\vec{\theta})$, contains the values of the sensitivity functions evaluated at time t_j and is defined as

$$D_j(\vec{\theta}) = \begin{pmatrix} \frac{\partial f_1(t_j,\vec{\theta})}{\partial \theta_1} & \frac{\partial f_1(t_j,\vec{\theta})}{\partial \theta_2} & \cdots & \frac{\partial f_1(t_j,\vec{\theta})}{\partial \theta_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m(t_j,\vec{\theta})}{\partial \theta_1} & \frac{\partial f_m(t_j,\vec{\theta})}{\partial \theta_2} & \cdots & \frac{\partial f_m(t_j,\vec{\theta})}{\partial \theta_p} \end{pmatrix} \tag{13}$$

The sensitivity matrix describes how much each component of the vector $\vec{f}$ is dependent upon a particular component of the parameter vector $\vec{\theta}$ at a particular time t_j. Sensitivity matrices for large systems are typically calculated using finite difference techniques or automatic differentiation.

Plotting the values of these matrices using a heat map can give us substantial information with minimal additional effort. When an element $\frac{\partial f_i(t_j,\vec{\theta})}{\partial \theta_k}$ in the sensitivity matrix is far away from 0, it indicates that the value of parameter k has a substantial impact on the output of function f_i at time t_j. Thus, we can identify which parameters we should estimate as well as which time intervals the parameters have the largest impact on the dynamical system. Several additional algorithms for choosing the most important parameters are discussed in [8]. These methods choose a subset of parameters that minimize a selection score related to the covariance matrix of the system while maintaining a full rank sensitivity matrix corresponding to the subset of included parameters. Their proposed algorithm relies on searching each potential set of parameters, however, and is therefore computationally costly, requiring up to

$$\sum_{p=1}^{p_0} \binom{p_0}{p}$$

comparisons, where p_0 is the number of parameters in the full model, if a smaller range of acceptable values of p is not determined prior to the calculations. Additionally, [8] provides theory for only scalar data, not data with multiple observed variables.

We chose to implement the simpler graphing method to determine which parameters were most important in the model. We evaluate the sensitivity functions at a discrete set of time points using the parameter values given in [1] to create the sensitivity matrices $D_j(\vec{\theta})$, calculate the sum $\sum_{j=1}^{n} |D_j(\vec{\theta})|$, and identify parameters that are associated with the largest elements in $\sum_{j=1}^{n} |D_j(\vec{\theta})|$. This set of parameters most strongly influences the behavior of the dynamical system over time.

We believe that one could also use currently undeveloped techniques related to sensitivity analysis to determine which state variables are most strongly associated with the parameters. In the next section, we employ statistical techniques to determine which state variables are most important to measure. The first method, which utilizes linear regression and the Akaike information criterion [2], identifies which state variables are most strongly associated with the outcome of interest, carbon dioxide assimilation. The second, which utilizes least squares minimization and sampling techniques, identifies which state variables provide sufficient information to accurately estimate parameters.

3.2 Choosing Optimal State Variables to Measure

In many applications, the number of state variables theoretically available to measure may exceed the number of state variables that an experimenter actually has the resources to sample. For example, our dynamical system model for the Calvin Cycle includes 38 distinct metabolites that an investigator could record. However, recording this number of metabolites could be prohibitively expensive. It is therefore desirable to find a subset of the available state variables that is optimal in some sense. We propose two methods for choosing this optimal subset of state variables and explore both of them using simulated data from the photosynthesis problem.

Simple Linear Regression The first method transforms the perspective of the problem such that we could appeal to classical variable selection techniques. In our formulation of the mathematical model (1) and statistical mode (3), we view the 38 metabolites as the response variables and time and the parameter $\vec{\theta}$ as the covariates (or inputs). If we had data, we would then find an estimate $\vec{\theta}_{OLS}$ of $\vec{\theta}_0$ using ordinary least squares and find a model solution that describes the metabolite concentrations over time. There are no known statistical or mathematical methods to select optimal outcome variables, and so in this framework we cannot identify which state variables would be best to measure.

In the Calvin cycle, however, the production of metabolites is not the primary goal of the plant: it is carbon assimilation, which leads to plant growth. Carbon dioxide uptake, which is related to carbon assimilation, may be estimated using a linear combination of the velocity rates of a few of the enzyme reactions. In this conceptual framework, we may now identify which state variables to measure. We define the 38 metabolites to be the covariates and we use the estimated CO_2 uptake rate as the response variable. With this change in perspective, we have transformed a complicated problem into a simple problem that may be addressed using standard statistical techniques.

Treating the state variables, metabolite concentrations, as independent variables and CO_2 uptake as the response, we perform simple linear regression at n selected time points t_j, $j = 1, \ldots, n$. The regression model is:

$$Y_j = \beta_0 + \beta_1 M_{1,j} + \beta_2 M_{2,j} + \cdots + \beta_p M_{k,j} + \epsilon_j \tag{14}$$

where Y_j is the CO_2 uptake rate at time t_j, $M_{i,j}$ is the concentration of metabolite i at time t_j, and $p = 38$, the number of metabolites.

To determine which metabolites are most strongly associated with the CO_2 uptake at a particular time t_j, variable selection can then be performed using forward, backwards, or stepwise selection using some selection criterion such as the Akaike information criterion (AIC), described in [2], or the Bayesian information criterion (BIC). The forward selection starts with no predictor variables in the model and adds one at a time if it is statistically significant. The backward selection starts with all predictor variables in the model and eliminates one at a time if it is not significant. Stepwise procedure combines the previous two procedures and considers whether to include or exclude the new variable in the model at each step. In this project, we use the stepwise selection based on the AIC statistic.

AIC [2] is the statistic $AIC = 2(k+1) - 2ln(L)$ where k is the number of predictor variables in the model and L is the maximum likelihood. Smaller AIC indicates the model fits the data better. For each t_j select the model T_j with the smallest AIC and conclude that the terms in T_j provide the best fit of CO_2 uptake and subsequently represent the best set of metabolites to measure at time t_j.

A key step required to conduct this optimization procedure is to generate a plausible dataset if multiple sets of experimental data are not available. To obtain multiple records, we choose n time points and for each discrete time point, we generate a dataset of 100 replicates of metabolite concentrations and CO_2 uptake rates. This dataset was generated by adding 5% noise from the standard normal distribution to each metabolite concentration given by the numerically solved model and CO_2 uptake rate. We perform this procedure at 10 different times, and then run the variable selection linear regression procedure for each time point. Ultimately, we select the metabolites that appear in the union of each of the 10 models more than three times.

Sum of Squares A second method to select optimal metabolites to measure finds the subset of state variables of cardinality p^* that minimizes the following sum of squares:

$$P_\alpha = \sum_{j=1}^{n} ||\vec{Y}_j - \vec{f}(t_j, \vec{\theta})||_2, \tag{15}$$

where $||\cdot||_2$ is the L_2 norm, $\vec{Y}_j$ is the vector of observed metabolite values for one replicate at the jth time point, and $\vec{f}(t_j, \vec{\theta})$ is the corresponding vector of calculated state variables using parameter vector $\vec{\theta}$. To find the combination of p^* state variables that minimizes this expression, one could compute P_α for every $\alpha \in \mathcal{A}_{p^*}$, where $\mathcal{A}_{p^*}$ is the set of all admissible combinations of p^* state variables. However, if there are many state variables and p^* is sufficiently large, this problem becomes computationally intractable. For instance, if we wanted to determine the 10 best metabolites to measure, we would have $\binom{38}{10}$, nearly 473 million combinations to compute! To find each P_α, we need to solve the inverse problem, and the computational cost of doing so is nontrivial for even a small number of parameters estimated. For only 10 parameters, it takes the inverse problem about 5 minutes to run; so it would take days to perform this computation thousands of times. To make this problem computationally feasible, one could first employ sensitivity analysis to reduce the number of parameters to be estimated such that $\mathcal{A}_{p^*}$ is sufficiently small to where a complete census is possible.

Nevertheless, it would be desirable to obtain the optimal set of metabolites without making preemptive assumptions that reduce the state space. In this vein, we propose a solution- below, we develop a statistical technique that takes into account the entire state space and can be conducted on an ordinary computer.

Rather than look at every α in $\mathcal{A}_{p^*}$, we instead look at a randomly selected sample of size N_{samp}. For every α_l, $l = 1, \ldots, N_{samp}$, we compute P_{α_l}. We than rank the P_{α_l} from lowest to highest. For every state variable, we compute a score, denoted S_i where

$$S_i = n_i^{-1} \sum_{l \in L_i} P_{\alpha_l}, \tag{16}$$

where L_i is the collection of integers from $1, \ldots, N_{samp}$ such that state variable i is in α_l and $n_i = card(L_i)$. For each state variable, we get a "score", that is an average of the ranks of the errors for the combinations including that state variable. All that remains is to choose the p^* state variables with the lowest scores, compute the error for this combination, and compare it to all of the sampled combinations. If this combination is now the best, we chose that combination of state variables. If it is not, we can choose the best one from our sampled combinations.

This method is *ad hoc*, and at this time, has no known distributional properties. The genesis came from permutation testing in classical nonparametric statistics theory. Permutation testing estimates p-values by permuting the data into all possible combinations and computing a test statistic for each permutation. If the number of permutations is too large, a sample of possible permutations can be taken. It was this idea of taking a sample over possible combinations or permutations that we borrowed for our method. A more dynamic search algorithm could be implemented, where instead of sampling from the state at random, we assign weights according to the past performance of metabolites. That is, the relative scores of the combinations are used to assign weights to each state variable, making the state variable more likely to be chosen in the next iteration if prior combinations including this state variable had relatively low errors. For now, however, we content ourselves with this simple algorithm to demonstrate the feasibility of this general method for choosing optimal state variables.

4 Results and Analysis

4.1 Sensitivity Analysis Results

We first generated sensitivity matrices D_j for five different sets of ten discrete time points t_j. For each set of time points, we then summed the absolute value of these ten matrices to select the parameters for which the system was most sensitive (see Figure 1). We discovered that KM11, Km521, KI523, KC, Km1221 and Km1241 elicited the largest sensitivity values for all five sets of ten time points. KM12, KM13, KE4, Km131,KE22, Km521 and V58 appeared to generate nontrivial sensitives in some (but not all) of the sets of ten time points. We use this analysis in the next section of parameter analysis.

4.2 Parameter estimation

Before we may conclusively identify the optimal times at which to sample or the optimal metabolites to sample, we must first test the feasibility of predicting a small number of parameters in the Zhu Calvin Cycle model [1]. Since we do not have a set of data, we use the simulated solution to this model, with parameter vector θ_0 with values as described in [1], to calculate eleven data points at times $t_j = \frac{j-1}{10}5000$, $j = 1, 2, \ldots, 11$, so that $T_f = 5000$ seconds. To perform the inverse problem, we add or subtract 25% of the value of the parameters we will estimate and then use the Matlab command `fminsearch` to minimize the unweighted least squares error between the predicted solution and the true solution that uses θ_0.

We perform the inverse problem for the parameter subsets

$$\begin{aligned}
\theta_{0a} &= [KM11, Km521, KI523, KC, Km1221, Km1241], \\
\theta_{0b} &= [KM11, KM12, KM13, KE4, KM131, KE22, Km511, Km521, \\
&\quad KI523, KC, Km1221, Km1241, V58], \\
\theta_{0c} &= [RuBP, SBP, KM11, KM12, KM13, KI13, KE4, KM9, KM131, KI135, KE22, \\
&\quad Km511, Km514, Km521, KI523, KC, Km1221, Km1241, V9, V58, V111],
\end{aligned}$$

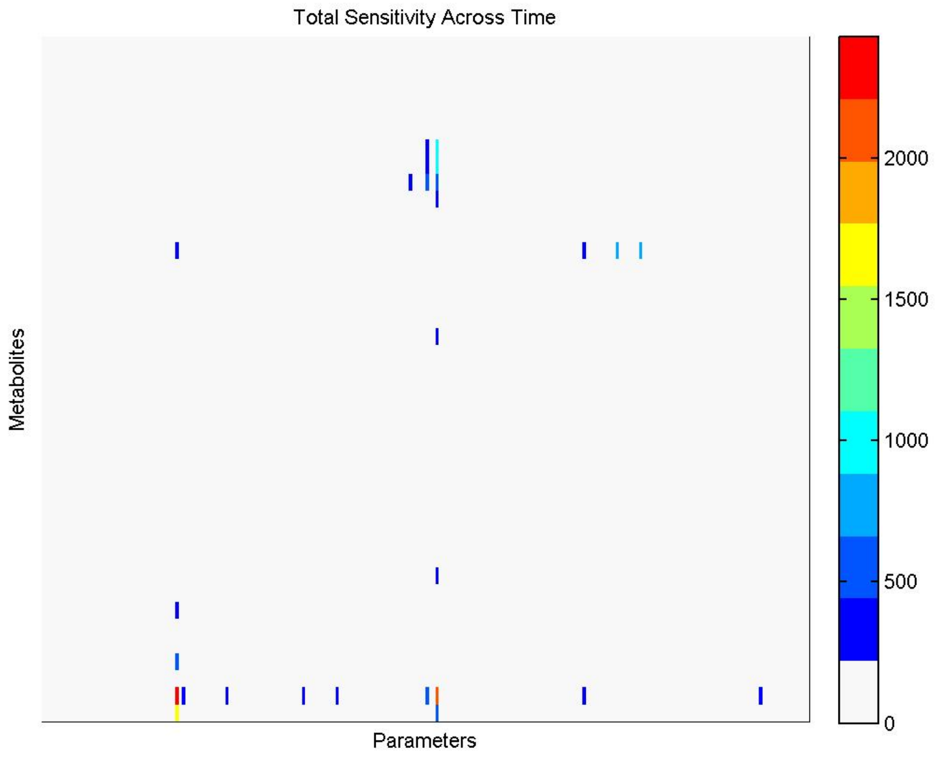


Figure 1: This figure represents a heat map for the sum of the absolute value of the sensitivity matrices for $t = 0.5, 1, 5, 25, 75, 150, 350, 650, 1200, 1850$ of the system in [1]. The horizontal axis indicates the position in the parameter vector $\vec{\theta}$ and the vertical axis indicates the position in the metabolite vector $\vec{x}(t)$.

Table 1: θ_0 and parameter estimates for top 13 parameters

Name	θ_0 value	$\hat{\theta}_a^-$	$\hat{\theta}_a^+$	$\hat{\theta}_b^-$	$\hat{\theta}_b^+$	$\hat{\theta}_c^-$	$\hat{\theta}_c^+$
KM11	0.0115	0.0115	0.0108	0.009	0.0139	0.0081	0.0140
KM12	0.222	-	-	0.1447	0.4354	0.1275	0.377
KM13	0.02	-	-	0.198	0.0298	0.0170	0.0276
KE4	0.05	-	-	0.0502	0.05	0.049	0.0502
KM131	0.05	-	-	0.0492	0.0499	0.0354	0.0561
KE22	0.058	-	-	0.0593	0.0577	0.0547	0.0581
KM511	0.02	-	-	0.0194	0.02	0.0203	0.0201
KM521	0.0025	0.0201	0.0097	5.5e-4	0.0022	4e-5	0.0031
KI523	7e-5	7.1e-5	2.2e-5	2.7e-5	6.2e-5	1e-6	8.5e-5
KC	0.0115	0.0114	0.0425	0.0114	0.0116	0.0062	0.0021
KM1221	0.15	0.0813	0.0023	0.1247	0.2096	0.1007	0.1894
KM1241	0.15	0.0894	0.0236	0.1276	0.2027	0.1064	0.1848
V58	0.0168	-	-	0.0098	0.0168	0.0192	0.0174
Runtime(sec)		282.09	362.80	854.20	640.53	1940.63	2115.75
LSQ Error		0.1429	88.98	0.9895	0.02649	0.1700	0.04131

record the parameter estimates $\hat{\theta}_a^\pm$, $\hat{\theta}_b^\pm$, and $\hat{\theta}_c^\pm$, respectively, for the initial parameter guesses at $1 \pm .25$ times the parameter subset true values, and plot the CO_2 uptake rates for the true solution and the solution generated by the parameter estimate. We also record the least squares error and the runtime for the inverse problem as performed on a laptop with a 2.67 GHz 4-core Intel i7 processor and 4 GB RAM.

Estimating six parameters to which the model is most sensitive provided mixed results when only 25% away from the true value. Starting at 25% below θ_0, the minimization routine was able to estimate KM11, KI523, and KC very well; however, starting at 25% above θ_0 lead to an estimate $\hat{\theta}_a^+$ that is not near θ_0 (Table 1). The variability in results may be due to the capabilities of `fminsearch` or undetected interactions between parameters. For example, when estimating Km521, perhaps KI521 should also be estimated. When estimating the top 13, all parameters could be estimated reasonably well with the exception of KM12, which was off in both $\hat{\theta}_b^-$ and $\hat{\theta}_b^+$, KM521, KC, and KM1221. The performance of the models generated by $\hat{\theta}_b^-$ and $\hat{\theta}_b^+$, however, appear to be very close to that of the model generated by θ_0 (Figure 3). Adding parameter constraints would aid in producing biologically correct parameter estimates. Estimating 21 parameters (the top 13 plus 8 more parameters used in describing reactions that involve RuBP and SBP, which are seen to be related to Rubisco in [1]) was surprisingly effective, with only 6 values in $\hat{\theta}_c^-$ and 2 in $\hat{\theta}_c^+$ being far from their true values (Table 1). The performance of the model generated by $\hat{\theta}_c^+$ is also impressive (Figure 4). Since some of these poorly estimated parameters are also problematic in the 13-parameter case, we may want to refocus our efforts on other parameters or perform more refined sensitivity analysis.

It appears that if the runtime of the parameter estimation problem is a concern, it would be best to estimate no more than 10 parameters; however, the results are sometimes far from the true parameters. Having correct parameter estimates in this context is especially important, as different reaction rates would paint a different picture of how the plant is performing. It is possible to estimate a large number of parameters (over 20) if reasonable initial guesses can be provided. Additionally, the number of parameters that could be estimated - and the quality of those estimates - could be improved if constraints were imposed.

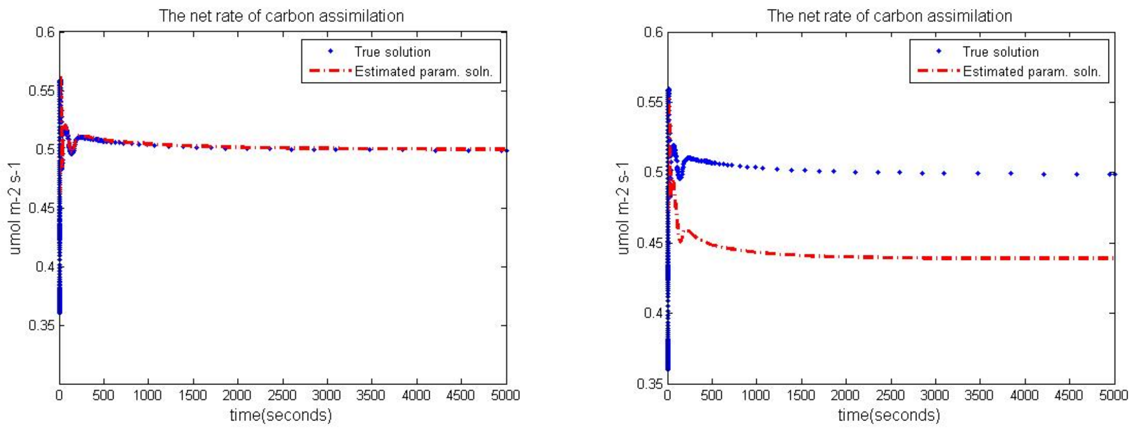


Figure 2: CO_2 uptake rates for θ_0(blue dots) and estimates $\hat{\theta}_a^-$ (left, red) and $\hat{\theta}_a^+$ (right, red).

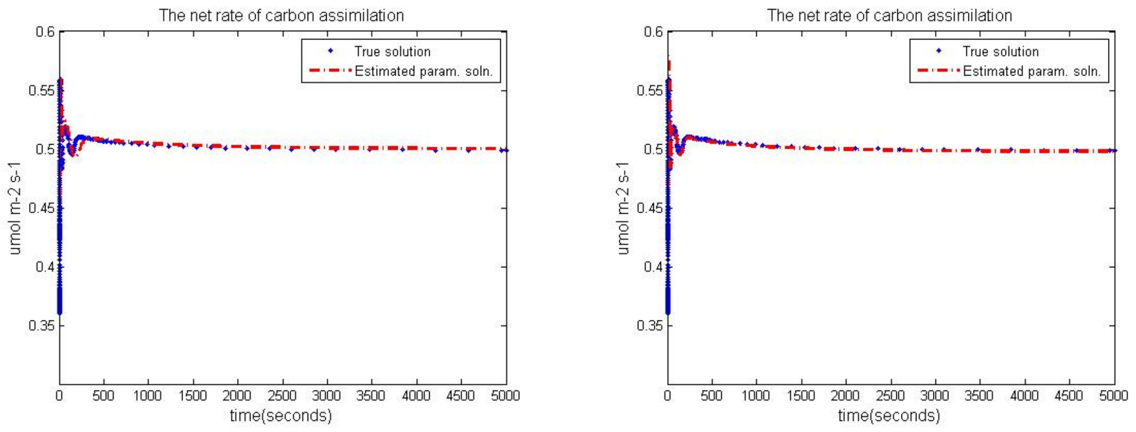


Figure 3: CO_2 uptake rates for θ_0(blue dots) and estimates $\hat{\theta}_b^-$ (left, red) and $\hat{\theta}_b^+$ (right, red).

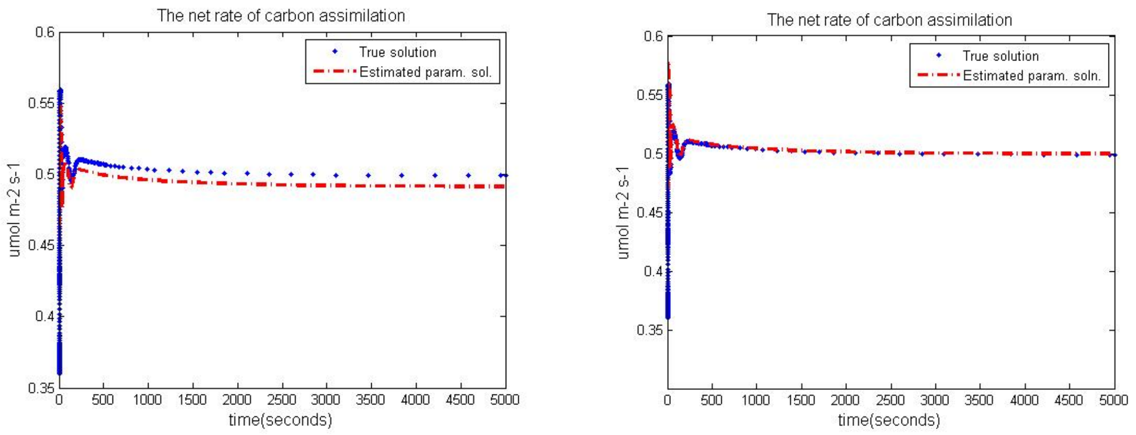


Figure 4: CO_2 uptake rates for θ_0(blue dots) and estimates $\hat{\theta}_c^-$ (left, red) and $\hat{\theta}_c^+$ (right, red).

4.3 Optimal Metabolites to Measure

Using the simple linear regression (SLR) variable selection technique; we obtain the set of metabolites that best predict the CO_2 update rate at each time point. We considered the metabolites that appeared often (more than 3 times), over all of the models for each time point, as the most important features. These metabolites are underlined in Table 2: RuBP, NADPH, GCEA, F26BPc, T3P, GOAc, SUC, FBP, CO2, PGCA, GLUc, NADHc, T3Pc. Since these metabolites most closely associated with CO_2 uptake rate, we recommend measuring these metabolites for an optimal fit.

Model:	**Metabolites:**
T_1	RuBP, E4P, ATP, NADPH, GCEA, GCEAc, HexPc , F26BPc, UTPc
T_2	RuBP, T3P, ADPc, GCEA, GCAc, GOAc, UTPc, SUC
T_3	T3P, FBP, CO2, GCA, PGCA, GOAc, HPRc, F26BPc, SUC
T_4	NADPH, GLUc , GCEA, PGCA, GLYc, HexPc, SUCP, SUC
T_5	T3P, CO2, HexP, NADHc, PGCA, T3Pc
T_6	RuBP, CO2, NADHc, GCAc, SERc, HPRc, T3Pc, F26BPc, SUC
T_7	SBP, NADc, GLUc, GCEA, GCA, SERc, UDPGc
T_8	RuBP, FBP, NADPH, CO2, NADc, ADPc, GLUc , GCEA, GOAc, GLYc, FBPc, UDPGc
T_9	O2, GCAc, HPRc, GCEAc, HexPc
T_10	DPGA, FBP, HexP, NADHc, GCA, PGCA, T3Pc, PGAc

Table 2: This table lists the resultant covariates selected by the step-wise selection algorithm for each model at each of the 10 time points. The underlined metabolites represent those that appear multiple (more than two) times over the 10 models. These highlighted metabolites produce the best fits and therefore represent optimal metabolites to measure.

We also consider the set of metabolites that were most sensitive to the 13 most influential parameters identified by sensitivity analysis. These metabolites are RuBP, PGA, T3P, S7P, SBP, GCEA, SERc, GCEAc, T3Pc, FPBc, and HexPc (Figure 1). This second group overlaps moderately with the set of metabolites that are associated with CO_2 uptake rate, so these groups are combined to form a subset of 20 state variables of interest. The variables in this subset will be used to form the admissible combinations of p^* variables, denoted $\mathcal{A}_{p^*}$.

To exhibit the initial proof of concept and debug the algorithm for the sum of square method to determine the optimal set of variables α_{p^*}, we choose $p^* = 3$, $N_{samp} = 10$, $t_j = \frac{j-1}{10}3000$, $j = 1, 2, \ldots, 11$ and optimize the six parameters in θ_{0a} starting from an initial guess of 75% of θ_{0a}. The code had a runtime of approximately 18 minutes. For one out of the ten selections of three metabolites, we find that the error $P_\alpha = 0$; however, the parameter estimates $\hat{\theta}_a^- \neq \theta_{0a}$. Additionally, two more selections out of the ten yielded $P_\alpha < 5 \times 10^{-3}$, again with $\hat{\theta}_a^-$ being far from θ_{0a}. This indicates that measuring three metabolites does not provide enough information to find acceptable parameter estimates; therefore, we do not calculate the scores S_i for the variables nor rank the variables to find an optimal set of three metabolites.

We also optimize the six parameters in θ_{0a} when $p^* = 5$, $N_{samp} = 450$, and $t_j = \frac{j-1}{10}3000$, $j = 1, 2, \ldots, 11$. This computation requires over 24 hours to complete. While some calculated errors P_α were near zero, none were exactly zero, so sampling only five variables may provide enough information to perform parameter estimation. The carbon assimilation rate calculated using the parameters from the best group of five variables is near that of the true solution, and the rate calculated using the highest individually ranked five variables is even closer (Figure 5). This indicates that when a complete search of all variable combinations is not feasible, using the highest ranked variables based on the selection store S_i may be acceptable.

5 Conclusion

In this paper we used an existing model from [1] to explore some mathematical and statistical modeling questions present in plant photosynthesis. We used sensitivity analysis to determine which sets of parameters were most affecting the metabolites, inverse theory to determine how to best estimate these important parameters, and both simple linear regression and sum of squares to preliminarily select the most important subset of metabolites to measure in an experiment.

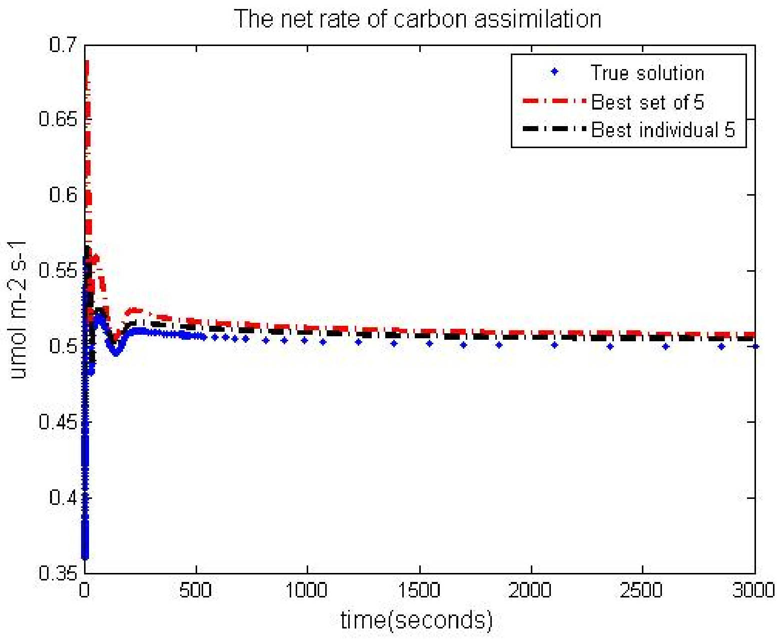


Figure 5: Calculated CO_2 uptake rates using true parameters θ_0 (blue dots), estimated parameters for the best group of 5 parameters (red), and estimated parameters for the best 5 parameters (black).

6 Future Work

Implementation of experimental design techniques to inform plant growth studies has shown great potential; hence, this project has many directions in which it could continue. The long-term goals of this project are to improve the sampling time distribution, select the optimal metabolites to measure when sampling plants, improve how the model accounts for environmental factors such as atmospheric CO_2 and O_2 concentrations, light level, and H_2O availability and how the model includes changes in the plant over time, and investigate cohort- and population-level models for use with destructive sampling techniques.

If our team had one more week to focus on this project, we would further investigate methods to select optimal metabolites. Our proof of concept numerical work indicates that a subset of metabolites may provide enough information to still perform parameter estimation; however, a full check of even all permutations of five metabolites out of 20 would necessitate performing the inverse problem 15504 times, which would take about four weeks. All permutations of 5 metabolites out of the top 15 (as informed by sensitivity analysis and statistical methods) would require solving 3003 inverse problems, which would be within the scope of another week of work if we estimated six parameters. While the code would be running, the team would search for any literature on similar problems and also flesh out the theory behind our method.

If our team had another month to focus on this project, we would provide a more complete solution for selecting the optimal variables. Additionally, such a time frame would also enable us to understand and implement methods to determine the optimal time points. This amount of time would also enable us to better test and improve the algorithms used to solve the ODE system and perform parameter estimation, especially in the areas of defining constraints, implementing algorithms in parallel, and tweaking the performance of the ODE solver. Additionally, we would revisit the sensitivity matrices and apply automatic differentiation instead of finite differences.

With even more time, we would consider the techniques described in [8] and compare selections of different number of metabolites. Both of these problems would require numerous parameter estimations, which would take a great amount of time. However, investigating these problems could potentially save industry partners a great amount of resources by reducing the number of sampling times, reducing the number of metabolite assays required per sample, and informing researchers on what parameters must be accurately estimated.

References

[1] X. G. Zhu, E. de Sturler and S.P. Long: *Optimizing the Distribution of Resources between Enzymes of Carbon Metabolism Can Dramatically Increase Photosynthetic Rate: A Numerical Simulation Using an Evolutionary Algorithm,* Plant Physiology, vol. 145, pp 513-526, 2007.

[2] H. Akaike: *A new look at the statistical model identification,* IEEE Transactions on Automatic Control 19 (6): 716-723, 1974.

[3] H.T. Banks and S. Hu: *Nonlinear stochastic markov processes and modeling uncertainty in populations,* Center for Research in Scientific Computation Technical Report CRSC-TR11-02, NCSU, January 2011.

[4] H.T. Banks, K. Holm and F. Kappel: *Comparison of optimal design methods in inverse problems,* Center for Research in Scientific Computation Technical Report CRSC-TR10-11, NCSU, July 2010.

[5] H.T. Banks, M. Davidian, J.R. Samuels Jr. and K. L. Sutton: *An inverse problem statistical methodology summary,* Center for Research in Scientific Computation Technical Report CRSC-TR08-01, NCSU, January 2008.

[6] H.T. Banks, J.L. Davis, S.L. Ernstberger, S. Hu, E. Artimovich, A.K. Dhar and C.L. Browdy: *A comparison of probabilistic and stochastic formulations in modeling growth uncertainty and variability,* Journal of Biological Dynamics, vol. 3, pp 130-148, 2009.

[7] H.T. Banks, J.L. Davis, S.L. Ernstberger, S. Hu, E. Artimovich and A.K. Dhar: *Experimental design and estimation of growth rate distributions in size-structured shrimp populations,* Inverse Problems, vol. 25, 095003(28pp), September 2009.

[8] H. T. Banks, A. Cinton-Arias, and F. Kappel, *Parameter selection methods in inverse problem formulations,* Center for Research in Scientific Computation Technical Report CRSC-TR10-03, NCSU, November 2010.

[9] H.T. Banks, J.L. Davis and S. Hu: *A computational comparison of alternatives to including uncertainty in structured population models,* Technical Report CRSC-TR09-14, June 2009.

Models for Predicting Indoor Pollution from Questionnaire and Outdoor Pollution Data: The Inner-City Asthma Air Pollution Study

Alexandra Bolotskikh[1], Liang Kong[2], Raymond Mess[3], Bill Shi[4], Yilong Zhang[5]

Problem Presenter:
Agustin Calatroni and Herman Mitchell
Rho, Inc.

Faculty Consultant:
Emily Lei Kang
North Carolina State University

Abstract

Regression analysis plays an important role in asthma studies among inner-city children. It is well known that the outdoor pollution is a risk factor for children's asthma; less, however, is known about the effects of indoor pollution: Direct measurement of indoor exposure is costly, time consuming and difficult to obtain. Fortunately, outdoor particulate matter exposure data is readily available for most major U.S. cities, and we know that outdoor air pollution contributes to indoor concentration levels. These outdoor data, in combination with behavioral questionnaire data (such as smoking, cleaning, cooking, air conditioning use, etc.), may allow us to predict indoor levels of pollution without the cost and difficulty of home visits and direct home pollution measurements. In this report, a generalized additive model (GAM) is developed to predict the indoor $PM_{2.5}$ from the outdoor $PM_{2.5}$ and a questionnaire.

1 Introduction and Motivation

A U.S. Environmental Protection Agency study found a strong association between a number of adverse health effects, including asthma and particulate pollution in outdoor air, [4], [5], [6]. Ultra-fine particles, those with an aerodynamic diameter $< 2.5\mu m$, ($PM_{2.5}$), henceforth to be known as $PM_{2.5}$, may produce most of these harmful effects [12]. Exposure to particulate pollution may cause or exacerbate asthma symptoms [16]. The below figure illustrates how far these ultra-fine particles penetrate into the lungs.

The smallest particles, which are of aerodynamic size four microns or less, can deposit as deep as the lower portion of the lung sacs or the bronchioles. The pervasive infiltration of the lungs by these particles is what makes it so crucial that all the major sources of $\mathrm{PM}_{2.5}$ be well understood.

To understand connections of particulate pollution and asthma the Inner City Air Pollution,(ICAP) was run. The Inner-City Asthma Study was a seven site study sponsored by the National Institute of Allergy and Infectious Diseases (NIAID), and the National Institute of Environmental Health [16].

The Inner-City Asthma Study was designed to evaluate the effectiveness of two types of interventions at reducing asthma morbidity and severity among 937 inner-city children, aged 5 to 11, with moderate to severe asthma. Each participant was enrolled for 2 years: the intervention year and the follow-up year [10].

ICAS did not address indoor particulate pollution. On average Americans spend most of their time indoors; studies give a range from 70 percent [9] up to 90 percent [7]. Thus it is important to be able to determine levels of $PM_{2.5}$ Since little was known about levels of $\mathrm{PM}_{2.5}$ in the home,it became necessary to do a study of indoor levels of $\mathrm{PM}_{2.5}$ [16]. So a new study was attached to ICAS. The new study, the Inner-City Air Pollution,(ICAP), focused on indoor air quality as it related to asthma [16]. The ICAP acquired data in several ways. The first used existing outdoor EPA air pollution monitors to collect data about outdoor $\mathrm{PM}_{2.5}$

[1]University of Florida
[2]Auburn University
[3]University of Wisconsin-Milwaukee
[4]University of North Carolina
[5]George Washington University

and known pollutant gases. The second source of data came from in-home visits made over three 2-week periods at six-month intervals [16]. The third data collection strategy was through in-clinic interviews.

The data collected provided investigators with enough information to obtain important results. Analysis by Wallace et al. confirmed that when present smoking is major contributor to indoor $PM_{2.5}$ levels [16]. The ICAP study also confirmed that cooking is a smaller but significant contributor to indoor $PM_{2.5}$ levels [16]. There was also some indication that dwelling type (highrise apartment,detached home,lowrise apartment etc) may be a factor in indoor $\mathrm{PM}_{2.5}$ levels. Smoking, cooking, and building type data were obtained through in-home visits. The cost of in home visits is very prohibitive. The ICAP study cost approximately 10 million (Mitchell,2011). Since public health officials need to run studies over at least twenty years with larger groups of people, in-home visits become cost prohibitive. For this reason we were asked to design a model which could give investigators a reasonable estimate of indoor $\mathrm{PM}_{2.5}$ levels. We were told the model must be based on information that does not require investigators to enter homes.

The paper is organized in the following manner. Section 1 provides description of the ICAS study and data analyzed. Section 2 gives the initial variable selection and ways to handle missing data. In section 3, we discuss models we built: the intercept model, linear model with interactions, stepwise linear model with interactions and generalized additive model. Also we provide comparison of the prediction behavior of the fitted models. Finally in Section 4, we discuss other possible criteria for variable selection and other models which could be fitted.

2 Data

2.1 Initial Screening of Variables

We recieved the data we used from the Inner-City Asthma Consortium Statistical Coordinating Center at Rho,Inc. in Chapel Hill, NC. The center oversaw both ICAS and ICAP has three levels. The levels are city, household and visit. The data was collected in seven metro areas. Each household was assigned a unique id used throughout the study. Data was collected in during visits to the home, so each data set was assigned a unique three variables name consisting of the city, the household, and the visit number.

Our initial dataset consisted of 277 variables. Because the indoor monitor collected $\mathrm{PM}_{2.5}$ every ten minutes, we had a dataset with over a 1,115,441 observations.

Since we were working with such a really huge data set(1,115,441 observations with 277 different variables), it was inefficient to directly fit the model using the whole data set even for the linear regression. We preferred to first summarize the 10 minutes measurement of indoor $\mathrm{PM}_{2.5}$ to reduce the number of observations. It was reasonable because the $\mathrm{PM}_{2.5}$ will not change too much within 10 minutes. Thus, our challenge became how to suitably summarize the measurement.

We explored three different ways to handle this issue. First, we simply used the daily average of $\mathrm{PM}_{2.5}$ to summarize the data set. The daily average can generally summarize the pattern of $\mathrm{PM}_{2.5}$ and provide the information, and another reason why we use this variable is that we only have the daily average of outdoor $\mathrm{PM}_{2.5}$.

People's indoor activity, however, highly related to the indoor $\mathrm{PM}_{2.5}$. For example, smoking and cooking will dramatically increase the level of $\mathrm{PM}_{2.5}$ which increases the bias if we directly use the daily average $\mathrm{PM}_{2.5}$. Then we try to find another way to summary the 10 minutes measurement. So we decided to use two different average numbers to begin model the dataset.

One is the 3 to 5 AM average of indoor $\mathrm{PM}_{2.5}$. Since most people in the family will still sleep between 3 to 5 AM everyday, it is reasonable to assume that there is strong relationship between indoor and outdoor $\mathrm{PM}_{2.5}$ at this period and achieve good prediction quality.

The last of the averaging variables is peak time average of indoor $\mathrm{PM}_{2.5}$, which calculated by the maximum of the hourly average indoor $\mathrm{PM}_{2.5}$. Since we were also interested in modeling those variables related to the activities for every specific household, we wanted to use the peak time average to find important variables in the questionnaires to build the model. Peak time is the most highly active time for the whole day. Peak time average was therefore more suitable to this feature and helped us understand the prediction performance.

After we finished reducing our number of observations using these averaging techniques we still had a data set with over 200 variables and 7,792 observations. This large number of variables would cause overfitting in a model and we lacked the computational power to run a variable selection process on over 200 variables.

Our next step was to screen out some of the existing variables. First we grouped the variables by type. The primary groups are illustrated in the following table:

The above table illustrates the abundance of variables within some of the groups. We were able to reduce the smoking group variables down to two variables. the first of which is the number of people including visitors who smoked in the home in the last 14 days. The second smoking variable we used was nicotine levels as taken by an in-home nicotine sensor. We could include nicotine in our model since in future studies a mail-in passive sensor will be used and no field researcher needs to enter the home.

We knew smoking was important because analysis in the [16] reported that combustion products such as those related to smoking and to a lesser degree cooking and heating are sources of $PM_{2.5}$ In the same paper, [16], variations in ventilation, such as air filters, open windows or air conditioners of various types, are given as an important group, includeding a group for dwelling types,including such types as highrise apartments, lowrise apartments,detached homes etc. We did this because housing types reflect both economics and possible airflow variations. We included a group that was explicitly of an economic nature based on results from (Wright , 2004) and (Byrant-Stevens , 2008). We also created a group of environmental factors such as weather, chemical gases, molds,etc. Our grouping was aided in part by Sensidyne Inc, a manufacturer of indoor air pollution monitors, and Fig. 1 below is from its website.

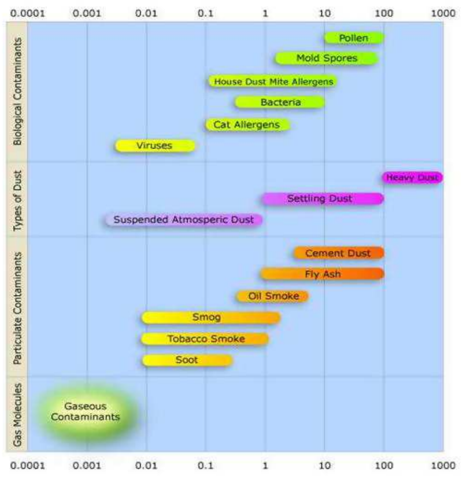


Figure 1: Airborne Particulate

The above graph shows the size of several airborne particulate and illustrates why we had to include some of the groups. After this whole preliminary screening was completed we had 36 variables. Once we finished the preliminary variable selection we needed to use statistical methods to find the truly significant variables.

2.2 Missing Data

For the selected 36 variables(See Appendix table), inevitably, there was some of missing data. We would have to remove the missing values, impute them, or model them. Instead of removing variables or observations with missing data, another approach was to fill in or impute missing values. A variety of imputation approaches could be used that range from extremely simple to rather complex. We describge different methods to impute all the data. These methods keep the full sample size, which can be advantageous for bias and precision; however, they can, however, yield different kinds of bias.

First, the missing value of some factors are below 10 percent, such as the outdoor level of SO_2(2.5%), NO_2(8.5%) and CO(2.1%). For those variables, which they have lower percentage of missing values, so we simply replaced each missing values with the mean value of that household it belonged to, which is called $Mean\ imputation$.

After investigating the background of those variables, we could assume the variable from a specific distribution and then generate the random number from that distribution to impute the missing values, so called

Random imputation. For example, we can assume the log transformation of outdoor level of O_3(10.6%) was from a normal distribution after checking the histogram.

The outdoor daily average of $PM_{2.5}$ and PM_{10} shows a linear relationship, which means we can predict PM_{10} depending on $PM_{2.5}$. Since we have a large percentage (40%) of missing values of PM_{10}, we can predict PM_{10} by the value of $PM_{2.5}$

In sum, we used the most basic way to impute the missing value because of time constraints. We may use some further method to manipulate missing values in the future. Once the problem of missing data had been dealt with, we could move to variable selection.

2.3 Transformation of the Data

To improve the interpretability of the data and make it easier to be visualized, we need to transform the data first. In log transformation, we use natural logs of the values of the variable in our analysis rather than the original raw values. For example, Fig. 2 below describes the log transformation of $PM_{2.5}$. The left panel uses raw data, while in the right panel, both the raw data have been transformed using the logarithm function. After following logarithmic transformations of value of $PM_{2.5}$, the points spread more uniformly as seen from the right panel in Fig. 2.

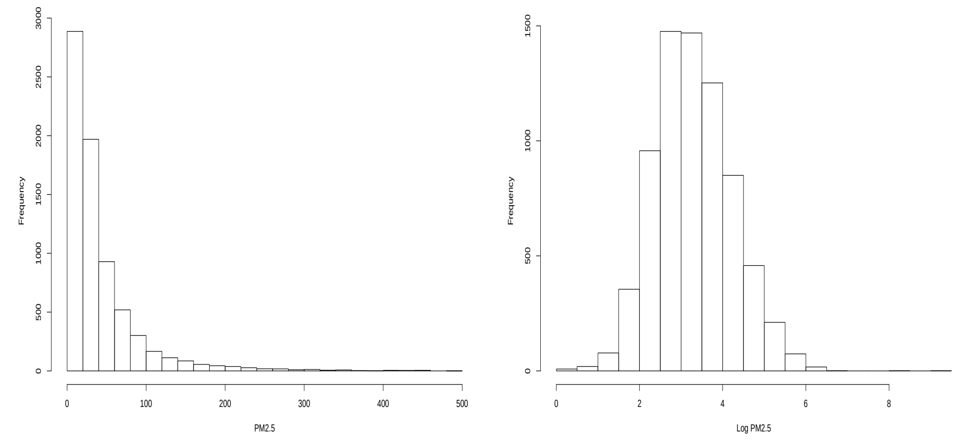


Figure 2: Histograms for raw and log-transformed data.

3 Modeling

3.1 Linear Model

The main goal of our project is to predict indoor $PM_{2.5}$ using outdoor $PM_{2.5}$ and other variables, affecting $PM_{2.5}$. Since measuring indoor $PM_{2.5}$ is very expensive, only data which can be obtained without in-home visits is available to build the model and make predictions. To make the best use of our data, we randomly divided data into two parts. The first part contains 90% of the observations and it was used for building and validating the models. The second part contains the remaining 10% of the observations which were used to check the prediction performance of fitted models.

We first modeled the daily average of indoor $PM_{2.5}$. To have a baseline model for comparisons we fitted the intercept model. Then we started constructing models from the simplest linear model with interactions which we fit using the linear model function, 'lm', in R. The fitted linear model shows a reasonable fit with the coefficient of determination equal to 0.4252. But this model included all 36 variables, and it is computationally inefficient. So this model needs to be improved in particular in order to reduce the number of explanatory variables. Besides, there are statistical reasons why such selection is necessary. First, too many variables may introduce the problem of multi-collinearity or increase the variance of estimates. Second, unnecessary variables tend to complicate the subsequent use of the model. Last, but not least, fewer variables indicates less cost on collecting data in practice.

To select variables to be included in the model, we used one of the criterion-based methods to find a linear model which fits the data well. The most straightforward criterion to measure the "fitness" of a given model is the coefficient of multiple determination R^2 [14]. It is, however, easy to see from its definition that R^2 increases as the number of variables increases, and the model with the most variables will be chosen. To solve this issue, penalty terms for models with a large number of variables have been suggested and corresponding criteria are proposed, such as Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC). Since our analysis is carried out in a non-Bayesian framework, the AIC measurement is used in this work. The formal definition of AIC [1] is:

$$AIC_p = n \log(\frac{SSE_p}{n}) + 2p \tag{1}$$

where n is the number of observations, and SSE_p is the sum of squared errors of the model with p variables. It is derived from estimating the total of the mean squared prediction bias across all observations of the sample, and the second term in the right hand side of equation (1) is the penalty function of the number of variables. The model with small AIC is preferred. Note that the choice of AIC does not mean it is a dominant measure over others. None of these statistics should be regarded as definitive in itself [15]. Any result needs to be combined with judgement from the data itself to produce a reasonable model. We used the stepwise regression method to find the model with the smallest AIC. Given the total number of available variables P, the algorithm proceeds as follows:

Step 1 Given the current p regressors, we try to add the other $P - p$ variables one at a time and calculate the corresponding AIC; then the one which results in the largest decrease in AIC is included.

Step 2 For the resulting model obtained from step 1, we checked whether dropping any of the variables in the present model decreases AIC. If yes, the one whose deletion reduces AIC the most is excluded.

Step 3 We repeated Step 1 and Step 2. When neither of them reduces AIC, the procedure stops, and the current model is chosen.

We ran the above algorithm in R using a built-in stepwise linear regression function. This model provides a similar fit to the linear model with interaction terms with coefficient of determination equal to 0.4227. By using stepwise regression, we narrowed the number of explanatory variables from 36 to 24.

After building the models based on 90% of the data, we take the remaining 10% for prediction. So now we make predictions based on our three linear models: the intercept model, the full linear model with interactions and the model from stepwise linear regression. We compared predictions for the model via comparing AIC and Mean Squared Predictive Error (MSPE), where MSPE is obtained from the following formula:

$$MSPE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{2}$$

In theory, the function from observed values to predicted should be the identity function. And we can see from Fig. 3 that the intercept model is really a bad predictor of indoor $PM_{2.5}$. Based on AIC and MSPE both the full linear model with interactions and the model obtained from stepwise procedure shows very similar predictive performance. Also the coefficients of determination for these models are roughly the same. But the model from linear stepwise regression has less predictors, while still showing good prediction results. Therefore stepwise linear regression is preferred.

3.2 General Additive Model:

Since we know from exploratory analysis of the data that some of the variables show non-linear behavior with respect to indoor $PM_{2.5}$. we needed to try a more general model to fit our data set to improve the performance of prediction of indoor $PM_{2.5}$ from the outdoor one. Generalized additive models (Hastie, 1993) will be a good candidate, because they can use smooth spline or other nonlinear regression method to fit the continuous variables.

For our project, we developed Generalized Additive Models (GAM) for the daily average of $PM_{2.5}$. For simplicity we will use the same notation in Hastie's book [Hastie] to describe the model:

$$g(y) = \alpha + \sum_{j=1}^{p} f_j(X_i) + \epsilon$$

where $g()$ is the link function. In our project we considered $g() = log()$, X_i is the predictor, y is the response, α and $f_1, \ldots, f_p$ are unknown and should be evaluated. Typically, $f_1, \ldots, f_p$ will be fitted by smoothing spline with backfitted algorithm (Hastie, 1993). Errors ϵ are independent of the X_j's, $E(\epsilon) = 0$ with $Var(\epsilon) = \sigma^2$.

After we selected the variables from the linear model by the stepwise method, we got our full model for GAM with 24 different variables. We still used AIC criteria to choose the best model with GAM by the stepwise method. Then we get our final model as

$$\begin{aligned} log(pm25_{indoor}) &= f_1(log(pm25_{outdoor})) + f_2(nico) + f_3(no_2) + f_4(co) + f_5(o_3) + f_6(distance) + \\ & smoke.num + density + site + bheb1 + visit + dewp + fipb11 + frysum + \\ & weekend + temp + fipb9 + fenvb2 + weather + f_7(log(pm25)) \times smoke.num \end{aligned}$$

Model	**AIC**	**MSPE**	R^2
Intercept	17588.87	0.65	0
Linear with Interaction	14178.55	0.54	0.43
Stepwise	14163.37	0.56	0.42
GAM	14136.4	0.52	0.42

Table 1: Modeling results for daily average $PM_{2.5}$

Model	**AIC**	**MSPE**	R^2
Intercept	19449.25	1.01	0
Linear with Interaction	17026.45	0.78	0.33
Stepwise	17001.16	0.79	0.32
GAM	16854.11	0.78	0.34

Table 2: Modeling results for 3 am–5 am average $PM_{2.5}$

Model	**AIC**	**MSPE**	R^2
Intercept	20862.97	2.06	0.02
Linear with Interaction	18556.81	2.21	0.32
Stepwise	18535.62	2.19	0.32
GAM	18480.29	2.20	0.31

Table 3: Modeling results for peak time average $PM_{2.5}$

The R^2 for GAM is 0.419, which is quite close to the R^2 for the model chosen by the stepwise method, which is 0.423 in the previous part. We also used the 10% of the data left to check the performance of the GAM model by MSPE. The performance is also quite good. For GAM MSPE is 0.52, and for the linear model it is 0.56. The benefit of GAM is that we used 19 variables to fit the model, compared with 20 for the linear model, without loss of the prediction performance.

After doing analysis for the daily average of $PM_{2.5}$, we thought that it might be interesting to look separately on what characteristics of the household affected $PM_{2.5}$. For that purpose, instead of taking daily

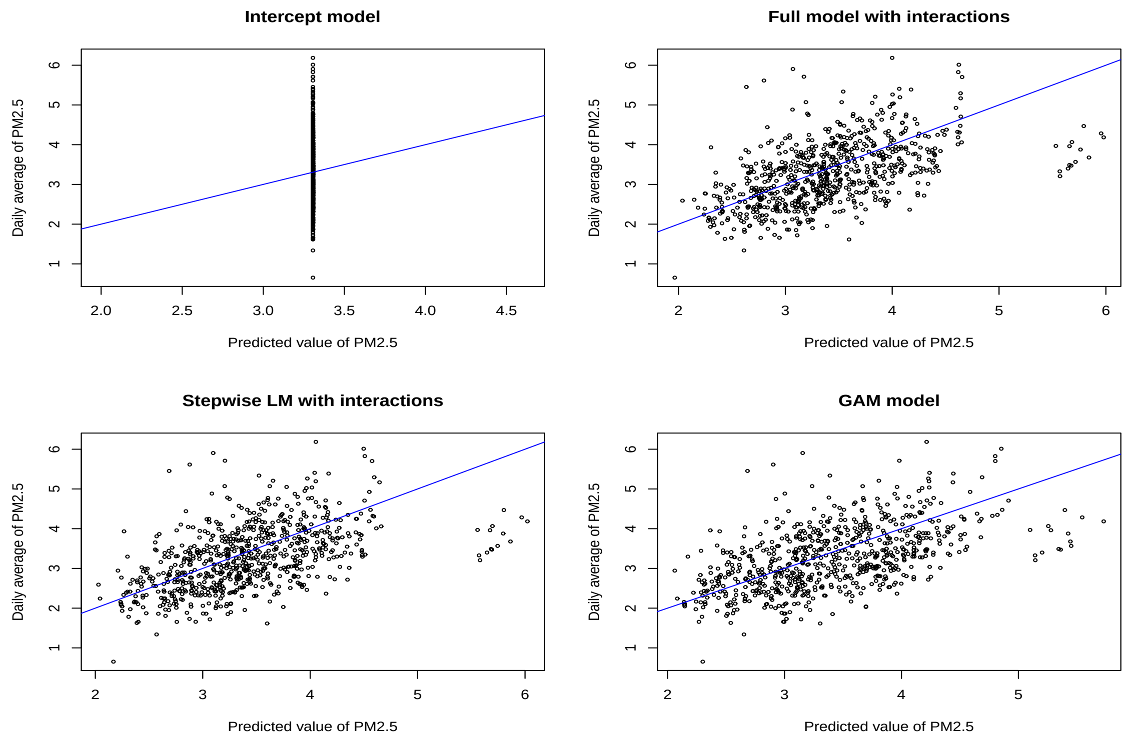


Figure 3: Prediction results for daily average $PM_{2.5}$

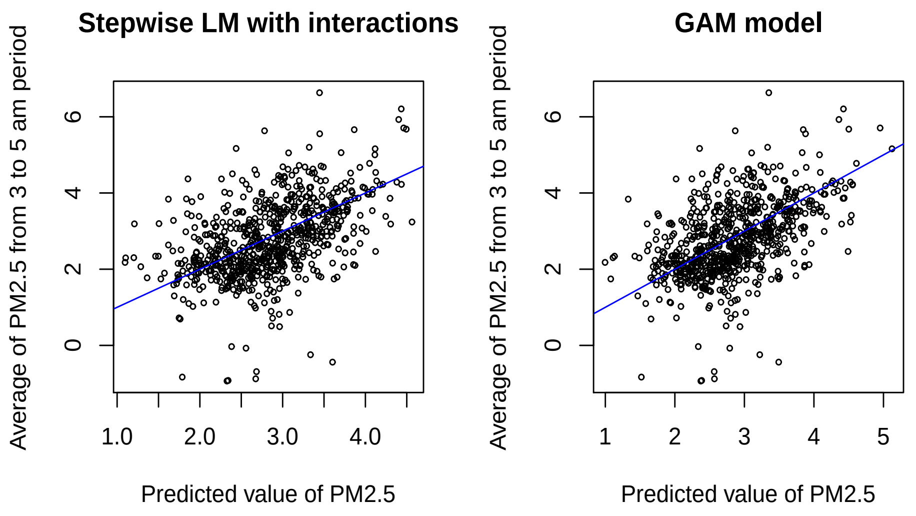


Figure 4: Predicion results for 3 am–5 am average $PM_{2.5}$

averages of $PM_{2.5}$, we took the average of $PM_{2.5}$ during period from 3 to 5 AM. In this case, effects of human behavior such as smoking, frying, etc. are minimized. We did similar analysis for $PM_{2.5}$ from the 3 to 5 AM average using stepwise procedures and GAM.

Again we found that the GAM model fit and predicted data the best.

$$log(pm25_{indoor}) \quad = \quad log(pm25_{outdoor}) + smoke.num + dewp + visit +$$

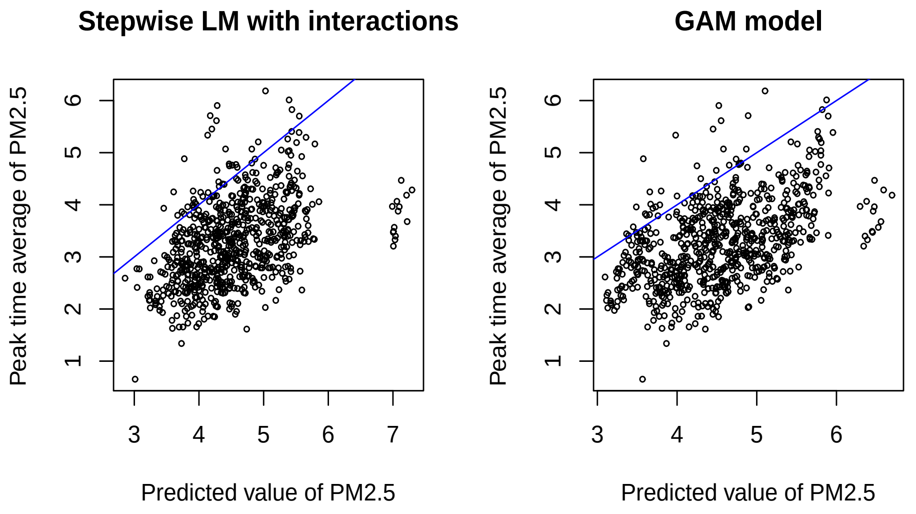


Figure 5: Prediction results for peak time average $PM_{2.5}$

$$f_1(nico) + f_2(no2) + density + bheb1 + frysum + stsum + fipb11 + site + f_3(co) + f_4(dist) + inc.cat + pets + weather + fipb10 + f_5(o3)$$

As we could expect, this model did not include an indicator variable of whether the measurements were taken over the weekend (see Appendix B) since even for the weekend, effects of human behavior are still minimized. The significance of the variables describing the amount of frying and toasting can also be explained: If windows were not open after people fried something, the frying effect would still be present at night. We also looked at what characteristics of human behavior affected $PM_{2.5}$ the most. Here we fitted the model using the peak time average of $PM_{2.5}$ as our response. In this case, that model fit well as seen in the Fig. 3 through Fig. 5, but there is some bias in prediction (Fig. 3 through Fig. 5), which can be explained since every household behaves differently in the peak time, and we cannot predict one of them using the others. The final GAM for peak time average of $PM_{2.5}$ is given by

$$\begin{aligned} log(pm25_{indoor}) &= log(pm25_{outdoor}) + smoke.num + density + f_1(o3) + f_2(nico) + site + bheb1 + \\ &\quad stsum + inc.cat + dist + dewp + fipb11 + frysum + weekend + visit + \\ &\quad prcp + f_3(no2) + weather + in.opnh + hepa \end{aligned}$$

4 Conclusions and Discussion

To confirm that the linear and GAM models contain the most important variables, we ran a regression tree procedure on the original thirty-six variables. Regression tree is a non-parametric method to select from a collection of variables the ones which are most important for determining a numeric response variable. The regression tree analysis and the models generated by the GAM process agree on the most important variables thus further supporting our use of the GAM. (Please see appendix A for technique details about the regression tree method.)

In the analysis above, we have shown that outdoor $PM_{2.5}$is a good predictor of indoor one. The other variables which can be used to predict $PM_{2.5}$ can be summarized into three different groups. First group contains matters that are the elements of $PM_{2.5}$ like O_3, CO, NO_2 and Nicotine . Second group is environmental factors like temperature, distance to the highway, weather. Third group includes the features of the household like the number of people who smoke, population density of that household. They all contribute to the prediction level of indoor $PM_{2.5}$ as it was shown in the data analysis part above.

On the other hand, there are still a lot of things that can be done to improve the modeling procedure. First, we can try to use different criteria like Bayes Information Criteria (BIC) and general cross validation (GCV) with stepwise method to do the variable selection. Moreover, other types of variable selection techniques can be applied, such as principle component analysis and bi-clustering method. Second, residual analysis shows that we can continue to use more difficult modeling technology to improve the modeling performance. For instance, since our project naturally separates into two different levels, city and household it is reasonable to add random effect to model the different residual features in each household and city. Also for prediction part of the project the cross validation method can be used to estimate how well the model can predict the indoor $PM_{2.5}$.

However, we have been given only one week to conduct the project. So we had to focus on simpler models, which we were able to construct and analyze within one week, instead of complicated models for which it takes too much time just to fit the model. We could not cover the modeling in every detail. But even our current relatively simple model have shown well prediction performance.

In the quest to improve performance,given more time to work on the project we would build and test a Generalized Additive Mixed Model(GAMM). We have also discussed with the Karl Gregory from the Robot Scientist group a possible collaboration between the two groups in which we could compare the performance of the robot scientist technology with traditional statistical methods. Robot scientist technology uses genetic programming and symbolic regression to build large trees of equations which model the data and then through testing and hybridization of the equations after many iterations the robot scientist produces a final model which is superior to previous models. Their is a large computational cost to using such technology. Still it would be of value to compare the various model building methods so that a comparison of computational cost and model prediction performance could be made.

References

[1] H. Akaike, A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (6): 716C723

[2] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and Regression Trees, 1st ed. Chapman and Hall/CRC, 1984.

[3] T. Bryant-Stephens, Asthma disparities in urban enviroments, Journal of Allergy and Immunology, June 2009

[4] U.S. EPA,Air Quality Criteria for Particulate Matter, Published by U.S. Environmental Protection Agency, Research Triangle Park, NC,1996

[5] U.S. EPA, Third External Review Air Quality Criteria for Particulate Matter, Published by National Center for Environmental Assessment-RTP Office, Research Triangle Park, NC, April 2002

[6] U.S. EPA,Air Quality Criteria for Particulate Matter (Final Report), Published by U.S. Environmental Protection Agency, Washington, DC 2004

[7] U.S. EPA, Questions about your community: Indoor Air url(http://www.epa.gov/region1/communities/indoorair.html), 2011

[8] T. J. Hastie,R. J. Tibshirani,Generalized additive models, Chapman & Hall/CRC,1990

[9] N. Klepeis, A. Tsung, JV Behar,Analysis of the National Human Activity Pattern Study (Analysis of the National Human Activity Pattern Study (NHAPS) Respondents from a Standpoint of Exposure Assessment. Final Report), Published by the Environmental Protection Agency, Las Vegas, NV 1995

[10] H. Mitchell, INNER-CITY ASTHMA STUDY (ICAS): Taking health care where it is needed most, url(http://www.rhoworld.com/rho/federal-research/projects/asthma-study)

[11] , H. Mitchell,SAMSI Summer Workshop Talk, oral communication, July 7,2011

[12] J. Schwartz, L.M. Neas, Fine particles are more strongly associated than coarse particles with acute respiratory effects in schoolchildren, Epidemiology, Vol11 No. 1, 2000

[13] Sensidyne Inc., Airborne Particulate, url(http://www.sensidyne.com/dust-particulate-monitor-nephelometer/dust-monitoring.php),2011

[14] R.G.D Steel, J.H. Torrie, Principles and Procedures of Statistics, New York: McGraw-Hill, 1960, pp. 187, 287.

[15] R.L. Smith, K.D.S. Yound, Linear Regression, preprint, Spetember 2009

[16] L. A. Wallace, H. Mitchell, et. al.,Particle Concentrations in Inner-City Homes of Children with Asthma: The Effect of Smoking, Cooking, and Outdoor Pollution, Enviromental Health Perspectives, Vol.111, Number 9, July 2003

[17] R. Wright, H. Mitchell, et al., Communtity Violence and Asthma Morbidity: The Inner-City Asthma Study, American Journal of Public Health, Vol 94, Number 4, April 2004

Appendices

A Regression Tree Method

Regression Tree is a non-parametric method to select from a collection of variables the ones which are most important for determining a numeric response variable. It begins with the entire sample, and then divides the entire sample into two subgroups according to a splitting rule and goodness of split. Normally the splitting rule has the form: whether $X_i \leq b$ or not, where X_i is one of the variables, and b is a number called the splitting point. When X_i is a categorical variable, the condition becomes: whether $X_i \in B$ or not, where B is a subset of the range of X_i. Then different splitting rules are tried and a goodness of split criterion is used pick up the best splitting rule. And this rule is used to disaggregate the sample into two subgroups. The process is repeated with the new subgroups until the tree reaches a maximal size, or it cannot be split anymore. Specifically, we use the goodness of split measure in [Breiman], and it proceeds as follows:

1. We start with the root node which contains all the observations.
2. At each node, every variable and its possible splitting points are used to split the node, and finally we pick the one which results in the largest goodness of split.
3. Step 2 is repeated on each of the two child nodes obtained, until none of them can be split.
4. Having generated a large tree, we utilize the technique in [Breiman] to prune the tree and find the optimal one.

Using the above algorithm, we identify Smoke or Nicotine are important ones from 36 variables. The regression trees produced for three response variables (daily average of $PM_{2.5}$, 3-5am average of $PM_{2.5}$, peak time average of $PM_{2.5}$) are presented below.

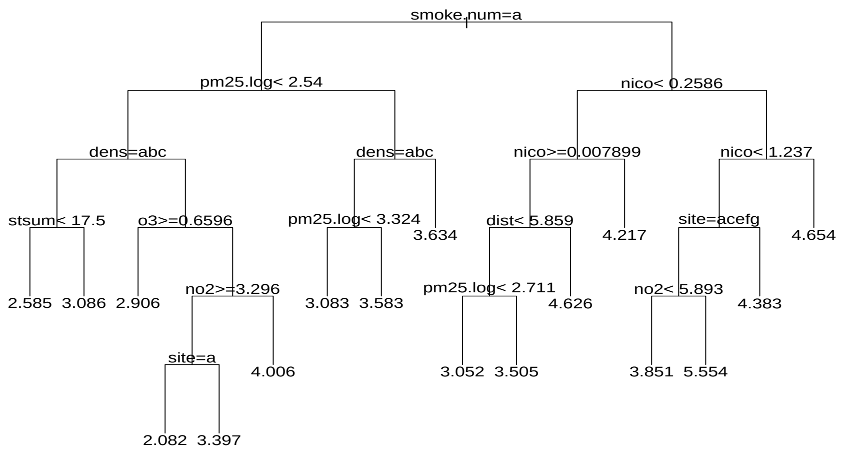


Figure 6: Daily.Avg.tree

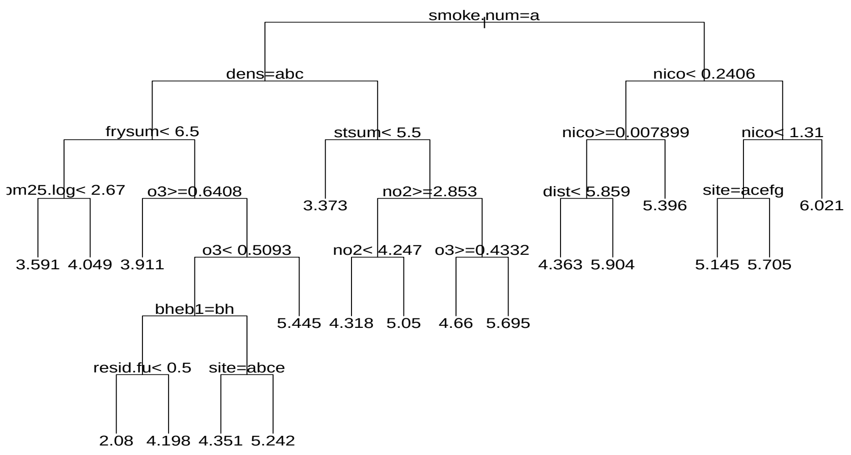


Figure 7: Peak.Avg.tree

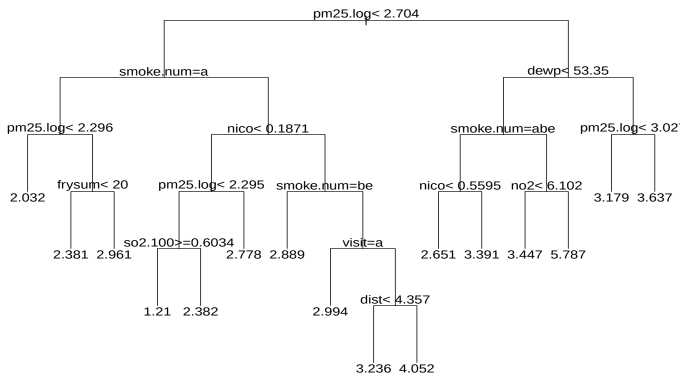


Figure 8: AM.Avg.tree

B Selected Variables

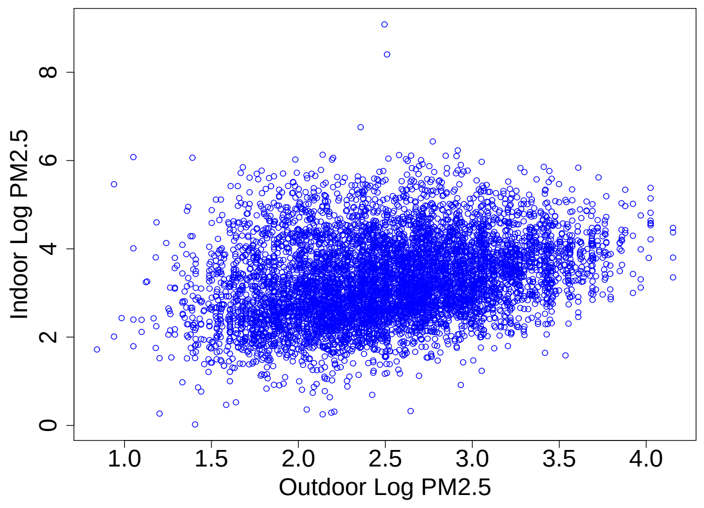


Figure 9: Predictions for indoor $PM_2.5$ from outdoor measurements.

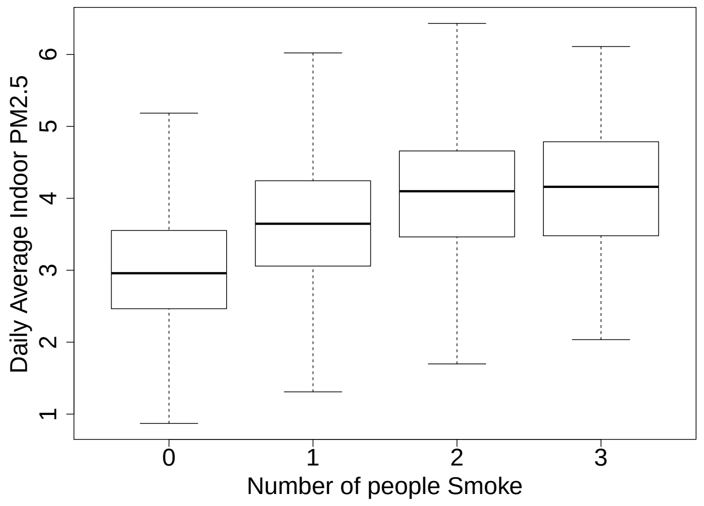


Figure 10: Boxplots for log-transformed daily average of indoor $PM_2.5$ grouped with various number of people smoke

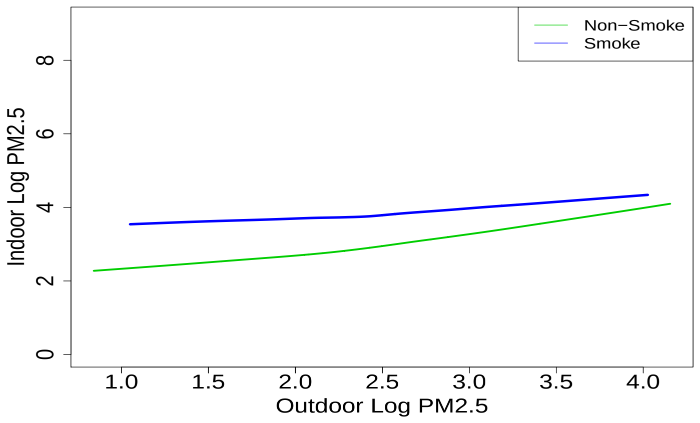


Figure 11: Relationship between log-transformed indoor and outdoor daily average $PM_2.5$ for two groups: Non-smoking and smoking households.

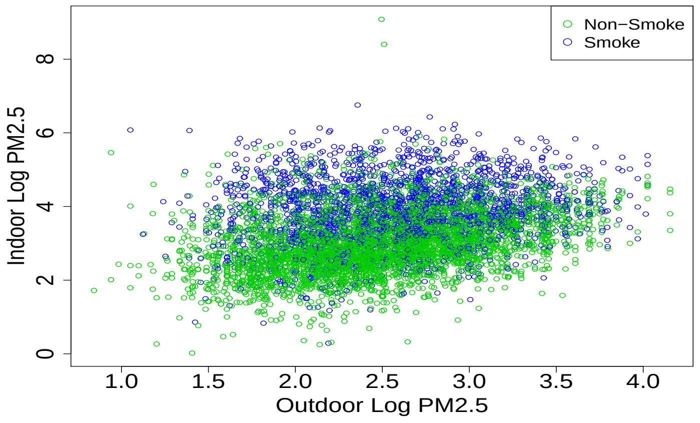


Figure 12: AM.Avg.tree

Labels	**Description**
bheb1	Type of dwelling (e.g. detached house, low rise apartment, etc.)
CO	Parts per billion of outdoor CO
density	Number of people per room
dewp	Outdoor dew point
distance	Distance from major roads (meters)
fenvb2	Indicator variable of type of air condition
fipb9	Number of times any part of the house was vacuumed
fipb10	Number of times any part of the house was dusted
fipb11	Number of times any part of the house was swept
frysum	Sum of number of days when food was fryed
hepa	Indicator variable of whether the HEPA air filter was used
inc.cat	Indicator variable of whether the income is less than 15000
income	Indicator variable of the income(7 categories)
job	Indicator variable of household member has a job
max	Maximum outdoor temperature
min	Minimum outdoor temperature
nico	The amount of nicotine in mgrams collected per visit
no2	Parts per billion of outdoor NO2
no2.ppb	Parts per billion of outdoor NO2 per visit
o3	Parts per billion of outdoor O3
o3.ppb	Parts per billion of outdoor O3 per visit
pets	Indicator variable of present rate of furry pets
pm10	Outdoor particulate matter 10
pm25indoor	Indoor particulate matter 2.5
pm25outdoor	Outdoor particulate matter 2.5
prcp	Total precipitation (rain and/or melted)
resid.fu	Number of years spent in this household
site	Indicates the city the household in
smoke.num	Number of people who smoke in the household
so2	Parts per billion of outdoor SO2
studyid	Id of the household
stusum	Sum of number of days when food was toasted
temp	Outdoor temperature
visit	Number of the visit
weather	Indicator variable of weather
weekend	Indicator variable of weekend measurements
win.opnh	How many hours windows were open during the day

Table 4: The 36 variables selected from the stepwise procedure described in Section 3.1

IMSM 2011: Credit Risk Quantification

Xiao Han Penn State University
Jerome Ouedraogo Georgia State University
Susan VanderPlas Iowa State University
Anran Wang North Carolina State University
Yan Zeng University of Rochester

Jared Bogacki BB&T
Jeff Scroggs North Carolina State University

July 15, 2011

1 Introduction

Analysis of investment portfolio risk involves many interesting modeling challenges: the new potential investment must be classified into a category or multiple categories to assess its similarity to already-held assets, the correlation between the potential investment and other assets must be estimated, and the asset's individual risk must be assessed by consideration of its leverage values. Quantification of these factors in credit risk is an inherently complex process, and automation of that process is extremely desireable as it reduces the potential for error and provides a summary of the complex information that contributes to the decision making process.

2 Project Objectives

The goal of this project is to develop and enhance algorithms for automatically classifying potential assets, assess their potential correlation with other assets, and summarize the risk involved in the acquisition of the debt for the bank. Our specific goals for the week are as follows:

1. Develop questions that will provide useful discriminating information between broad sectors of industry represented by NAICS codes
2. Use text processing to create a list of suggested NAICS codes, of which several codes may be selected (as many companies have business across several relevant sub-sectors)

3. $\boxed{\text{as of Wed. 7/13}}$ Develop a method for post-processing NAICS code suggestions to expand potential selections in the case of an unsatisfactory result.

4. $\boxed{\text{as of Wed. 7/13}}$ Rank NAICS code suggestions in terms of strength of lexical match, remove matches that are noticeably weaker compared to other results.

5. Assess correlation between classes of NAICS codes which can then be utilized to calculate the diversity of the portfolio given potential new assets (using stock prices and lexical processing)

6. Develop a model for the "risk rating" of a potential asset ~~given the contents of the portfolio that accounts for the correlation of default risk between portfolio members.~~ compared to assets in the same industry

3 Classification of Assets into NAICS Categories

3.1 Filtration Parameters for NAICS Classification

3.1.1 Partitioning NAICS Codes by Description

The first two-digit portion of the NAICS code describes the broad industry classification of a particular enterprise. Tabulation of the most commonly occurring words for each broad class suggested the partition shown in Table 1.

3.1.2 Filtration of Potential NAICS Codes To Improve Algorithm Efficiency

As such, it would be reasonable that our first "filtration" question should concern the general industry (and should allow multiple responses), for example:

$$\text{1. Does Company X} \begin{bmatrix} \square & \text{Produce, Manufacture, or Create a Product?} \\ \square & \text{Provide a Service?} \\ \square & \text{Sell Products and/or Services?} \\ \square & \text{Other} \end{bmatrix}$$

One followup question (with multiple potential responses) should be sufficient to narrow the search space in preparation for the LSI algorithm, which utilizes a paragraph of input text to identify potential NAICS code matches by a measure of semantic matching. Any further questions would narrow the field to so few poential matches as to allow a single forced-choice code matching.

Followup questions were written using the broad overview categories formed by considering the first two digits of the NAICS codes. For instance, if the answer to the first question is only "Produce, manufacture, grow, mine, or create a product", the second question is:

Classification Category	Code	Common Descriptors
Production of Goods	11	farming, production, cattle
Production of Goods	21	mining, ore, gas
Production of Goods	22	power, electric, generation
Production of Goods	23	construction, contractors, building
Production of Goods	31	manufacturing, mills, cut
Production of Goods	32	manufacturing, product, plastics
Production of Goods	33	manufacturing, equipment, metal
Retail	42	merchendise, wholesalers, equipment
Retail	44	stores, dealers, clothing
Retail	45	stores, dealers, department
Retail	49	storage, warehousing, delivery
Services Retail	52	insurance, credit, carriers
Services Retail	53	rental, leasing, equipment
Services	48	transportation, freight, air
Services	54	services, consulting, advertising
Services	56	services, waste, collection
Services	61	schools, training, colleges
Services	62	services, offices, health
Services	72	hotels, services, buffet
Services	81	maintenance, repair, organization
Services	92	administration, offices, programs
Entertainment	71	sports, amusement, artists
Production of Goods Services Entertainment	51	publishers, motion, picture

Table 1: Two-Digit NAICS Code partitions

2. Is the Company

- ☐ Farming or Crop Related?
- ☐ Mining and Extraction Related?
- ☐ Power Generation or Distribution Related?
- ☐ Construction or Contractor Related?
- ☐ Food and Clothing Manufacturing Related?
- ☐ Other Manufacturing?
- ☐ Entertainment Product Related?
- ☐ Other

An important feature of this pre-filtering is that it allows multiple choices, so that the field is narrowed as much or as little as desired. This allows forced compatibility with the previous algorithm (simply selecting "Other" at each stage of questions includes all options). Additionally, as users must select multiple choices for their inclusion, so there is a required work input to allowing multiple options, which may serve as somewhat of a deterrent to overly broad searches. Finally, the filtration algorithm alone reduces the potential scope of choices to the point of allowing a more focused forced-choice selection, which is far more likely to be accurate. The use of the LSI algorithm to further filter (and weight) the results provides an even more significant improvement in the suggested NAICS code

3.2 Latent Semantics Indexing

3.2.1 Algorithm Description

Latent Semantic Indexing (LSI) also called latent semantic analysis (LSA) is an indexing and retrieval method that uses a mathematical technique (Singular value decomposition) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings.

To perform a LSI, we first construct a Term-Document Matrix (TDM) to identify the occurrences of the m unique terms within n documents. In other words, a TDM is a mathematical matrix that describes the frequency of terms that occur in a collection of documents.

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ . & \dots & . \\ . & \dots & . \\ . & \dots & . \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

Each column is one NAICS code, Rows are all words appearing in all codes. Each element represents the number of occurrence each word (row) was found in a code (column). We can used TMG (Text-to-Matrix Generator) toolbox to construct TDM in Matlab. TMG is constructed to perform preprocessing and filtering steps that are typically performed on text documents. With TMG, we can:

- Create the TDM corresponding to a set of documents (matrix A);
- Create the query vectors from user input (Q);
- Update existing TDM by incorporation of new documents;
- Broaden existing TDM results by deletion of specified documents.

Step 1

Use TMG to build NAICS codes TDM. We use 19,720 NAICS codes description (6 digits) to build the TDM (matrix A)

Step 2

- Get the company description, format it in a text file
- Use TMG to create a query matrix (step 1) with the company description

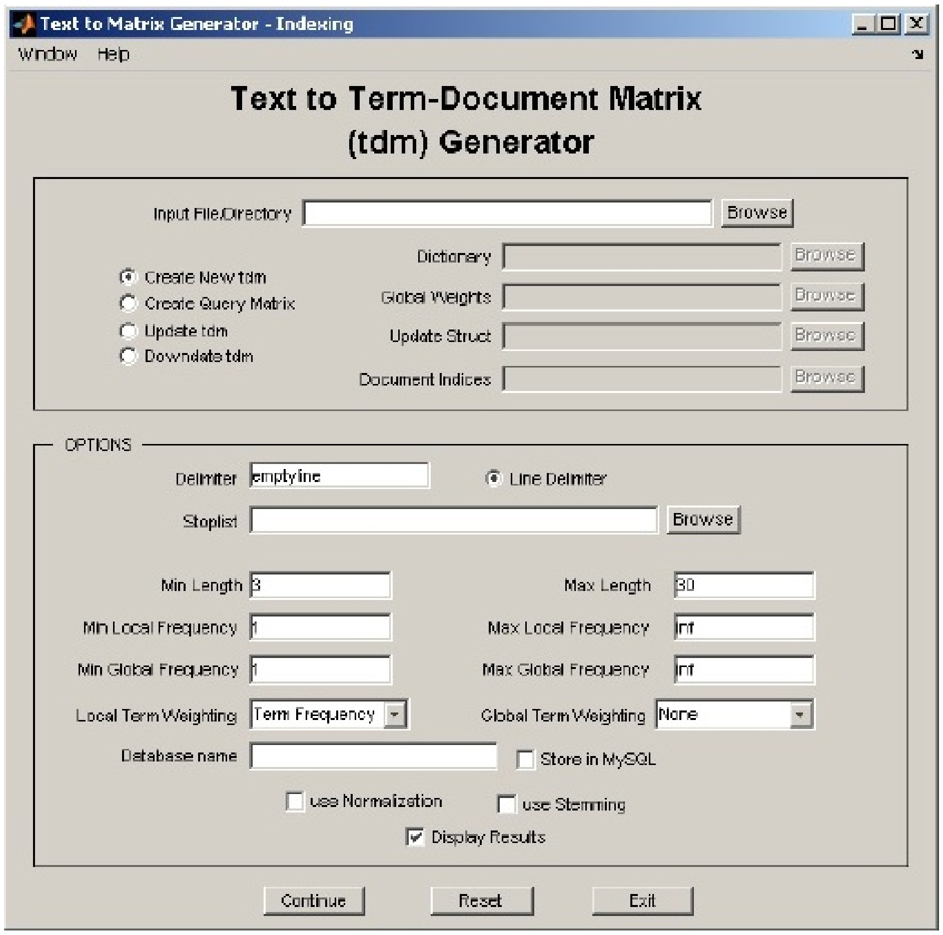
Step 3

Display the best NAICS codes (columns) matching the company description through angle comparison In this step we rank documents in decreasing order of query-document cosine similarities.

$$sim(C_i, q) = cosine(C_i, Q) = \frac{(C_i.Q)}{\|C_i\|\|Q\|}$$

where C_i for $i = 1$ to $n = 19,720$ are the columns of the matrix A.

We modified the procedure for running the algorithm to produce more accurate results. The main difference between the two approaches is the way in which matrices are generated using TMG. The second step, as described above, is slightly different: previous groups used Update TDM whereas in the new, we use Create a new Query, producing different results.



Text to Term-Document Matrix (tdm) Generator

Input File/Directory:
Dictionary:
Global Weights:
Update Struct:
Document Indices:

[✓ Create New tdm] Create Query Matrix Update tdm Downdate tdm

OPTIONS

Delimiter: emptyline
[✓ Line Delimiter]
Stoplist:
Min Length: 3
Max Length: 30
Min Local Frequency: 1
Max Local Frequency: Inf
Min Global Frequency: 1
Max Global Frequency: Inf
Local Term Weighting: Term Frequency
Global Term Weighting: None
Database name:
☐ Store in MySQL
☐ use Normalization
☐ use Stemming
☑ Display Results

Figure 1: TDM Algorithm Options. Using "Create a New Query" produces superior results to "Update TDM"

3.2.2 Results

Our results seem to be significantly more accurate than those produced in the previous work. One major problem with the previous algorithm is best demonstrated by its performance given a description of T-Mobile:

Code	Description
518210	Data entry services
484220	Mobile home towing services, local
484230	Mobile home towing services, long-distance
712110	Mobile museums
722330	Canteens, mobile

Table 2: Results for T-Mobile, using the previous LSI algorithm

By comparison, using the new approach, the algorithm suggests the following NAICS codes:

Cosine ∡	Code	Description
0.4000	517210	wireless data communication carriers except satellite
0.2582	454112	auctions internet retail
0.2582	515111	broadcasting networks radio
0.2582	515120	broadcasting networks television
0.2582	515210	cable broadcasting networks
0.2582	515210	subscription television networks
0.2582	517110	cable tv providers except networks
0.2582	518210	data entry services

Table 3: Results for T-Mobile, using the modified LSI algorithm

Clearly, the new method produces more accurate results. The LSI algorithm is no longer skewed by the abundance of the word "mobile" in the description of T-Mobile, and while the previous algorithm suggested various transportation and entertainment industries and did not suggest cellular phone networks, our modified algorithm performs quite successfully.

Additionally, If we select only the choices that are reasonable given the answers to the initial two questions, the new algorithm produces this output:

Cosine ∡	Code	Description
0.4000	517210	wireless data communication carriers (except satellite)
0.2582	517110	cable tv providers (except networks)
0.2582	515111	broadcasting networks radio
0.2582	515120	broadcasting networks television
0.2582	515210	cable broadcasting networks

Table 4: Results for T-Mobile, using the modified LSI algorithm and preprocessing

As an added advantage, searching only the NAICS codes corresponding to the answers to the initial questions reduces the computational load required by narrowing the search space. As a result, we expect that implementing the two initial questions as a "filter" will provide increased accuracy and decreased computation time. From a human-factors perspective, preprocessing also increases accuracy with minimal human input, so even if the paragraph does not provide useful information, if the two initial questions are answered correctly, the results will be at least somewhat useful.

Additional suggested modifications LSI is a powerful and automatic searching technique in the sense that the only input required is description of the company, with which the algorithm performs its code searching procedure. However, our implementation requires an analyst familiar with MatLab, and a MatLab license.

Additionally, TMG has a module that can compute and retrieve the NAICS code, but the output is in html file (given the rank of column and the angle). TMG can also use (or not) SVD in its analysis. It should be reasonably simple to automate the output of TMG so that it produces the NAICS code directly. Additionally, it would be useful to examine the distribution of angle size to determine the optimal cutoff for "good" NAICS code matches, and from that distribution, it would also be useful to obtain a "match probability" that would allow the end user to more easily determine whether there might be multiple reasonable NAICS matches.

3.3 Conclusion and Recommendations

The advantages of this approach are significant:

- The LSI algorithm produces list of potential NAICS codes, rather than a single NAICS code. This is an advantage, as a firm can have multiple NAICS codes (e.g., Walmart has both a retail division and a wholesale division, Nike produces athletic gear, clothing, and shoes, etc.)
- The filtration questions provide a reasonably narrow list of codes with very minimal effort, increasing accuracy of company information

4 Measuring Distance-to-Default

In this section, we try to evaluate credit risk of grouped companies based on their distance-to-default. The distance-to-default measure is first introduced by Moody's KMV[1], which defined as the number of standard deviations the asset value is away from the default barrier. The concept of distance-to-default is straightforward. Default of a company occurs if its assets fail to meet liability payments. In general a company will default when its asset value reaches the book value of its total debts, that is, when its market net value reaches zero. Thus the higher the value of the firm assets, relative to the default barrier, the company would be farther away from default.

$$DD = \frac{V_A - D}{V_A \sigma_A}$$

where

DD is the distance-to-default,

V_A and σ_A are the value and volatility of a company,

and D is the default barrier or the book value of the debts.

Thus, the main elements that determine the distance-to-default of a company can be summarized by asset value, asset risk and financial leverage. For a publicly traded company, we can use the information from its market price of equity and liability to estimate the distance-to-default. To do so, we

1. Estimate implied asset value and volatility by market value and volatility of the equity and the book value of the debts.

2. Compute the distance-to-default based on obtained asset value and volatility and the book value of liabilities.

The R package `quantmod` [3] was used to obtain this information. More information is available in the Appendix.

4.1 Estimation of Asset Value and Volatility

The options pricing approach can be extended to compute the implied value of asset value here. Since the equity holders have limited liability, as the asset value goes down and reaches the default barrier, they have the right but not the obligation to pay off the debt. Otherwise, as the asset value moves up against the default point, the equity holders gain the premium or the net asset value. As a result, the equity of a company could be modeled as a perpetual call option on the value of its assets with a strike price equal to the default barrier, or the book value of the company's debts in the simplest case. Then the value of the equity can be written as:

$$V_E = \max\{V_A - D, 0\},$$

where

V_E is the market value of the company's equity, as shown in Figure 2

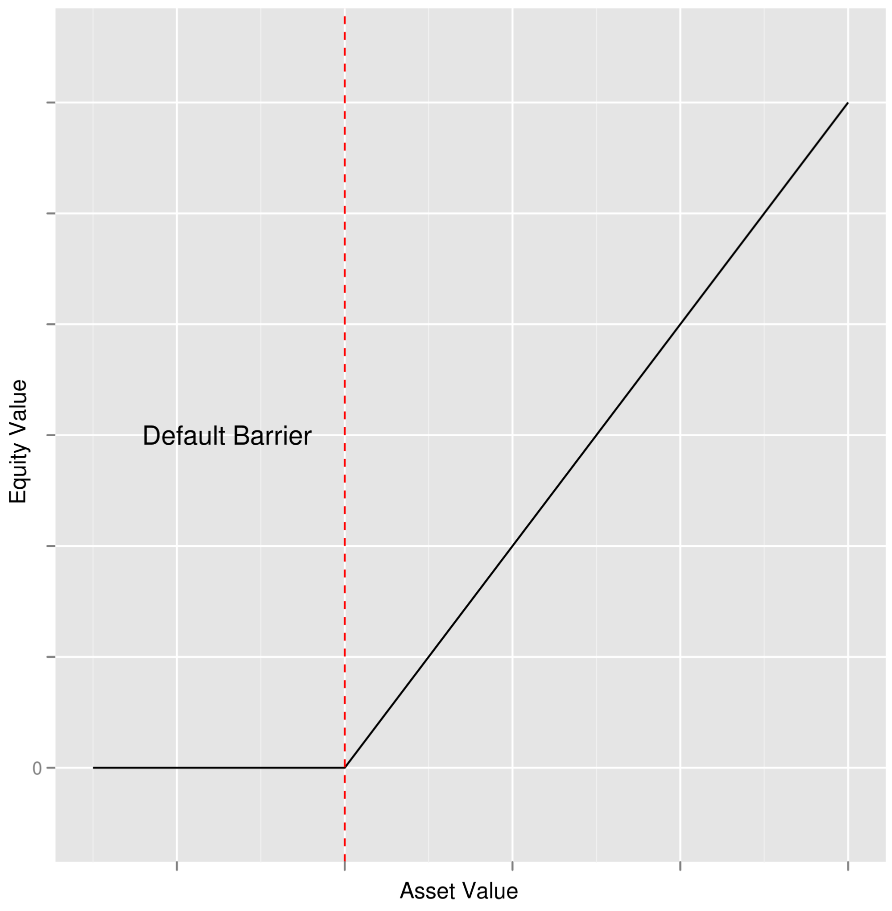


Figure 2: Default occurs when the value of a company's assets is less than or equal to its debts

Based on Black-Scholes-Merton framework, model the relationship between the market value of assets and the market value of equity assuming the asset value of the company follows the Geometric Brownian Motion. The it follows that the asset value is

$$dV_A = \mu V_A dt + \sigma_A V_A dz$$

where

μ is the drift rate of the asset value,

and z is a Wiener process.

The value and volatility of equity can be expressed as:

$$V_E = V_A N(d_1) - De^{-rT} N(d_2)$$

where

$d_1 = \dfrac{\ln(\frac{V_A}{D})(r + \frac{\sigma_A^2}{2})T}{\sigma_A \sqrt{T}}$,

$d_2 = d_1 - \sigma_A\sqrt{T}$,
$N(d)$ is the cumulative normal,
r is the risk free interest rate,
and T is time to maturity.

$$\sigma_E = \frac{V_A}{V_E}\Delta\sigma_A$$

where
σ_E is volatility of equity value,
and $\Delta = \frac{\partial V_E}{\partial V_A}$.

Using the observed market value and volatility of equity of the company, the implied value and volatility of company assets can be estimated by solving the two equations inversely.

4.2 Calculation of the Distance-to-Default

As a result, under the Black-Scholes-Merton framework as stated above, the expression of distance-to-default is:

$$DD = \frac{\ln\frac{V_A}{D} + (\mu - \frac{\sigma_A^2}{2})T}{\sigma_A\sqrt{T}}$$

Based on data from Yahoo Finance including 3600 publicly traded companies with stock price and its standard deviation, number of shares, and total debt, we calculate the distance-to-default for each company. Here we use the interest rate of three-month Treasury Bill as the risk free interest rate, and incorporate a fifth order polynomial to estimate the cumulative normal distribution[2].

4.3 Comparison of Industry-wide Average Distance to Default

Finally, we obtained each company's SIC code and converted it into a NAICS code [1]. After grouping all the companies by their NAICS code into 22 sectors, we found the median of the obtained distance-to-default amongst the companies for each sector with the two digit NAICS code, and used the median as an indicator for each sector, as shown in Figure 3. Therefore, given a company, we can first locate their two digit NAICS code and then estimate its risk by the assigned indicator in the sector. The results from the quantification procedure can provide a picture of risk rating for each industry.

[1]Corresponding Codes taken from `http://naics.com/naicsexcel%20files/SIC%20to%20US%20NAICS%20Cross%20Reference%202011%20Version.xls`. Cases for which there were multiple NAICS codes were reduced to the first available NAICS code for simplicity (though as this portion of the project uses only the two-digit NAICS code, this should have no effect).

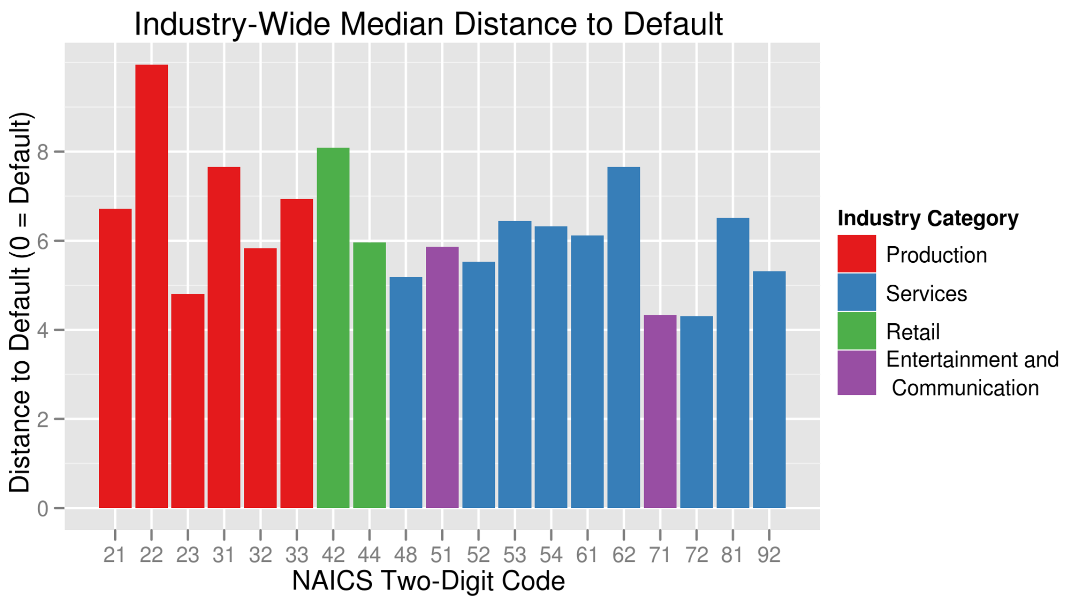


Figure 3: Median Distance to Default, by Industry

To extend this approach, it would be reasonably simple to compare a specific company to its industry norms to determine its relative stability compared to other similar companies. Figure 4 shows the distance to default of all 3600 companies considered in this analysis.

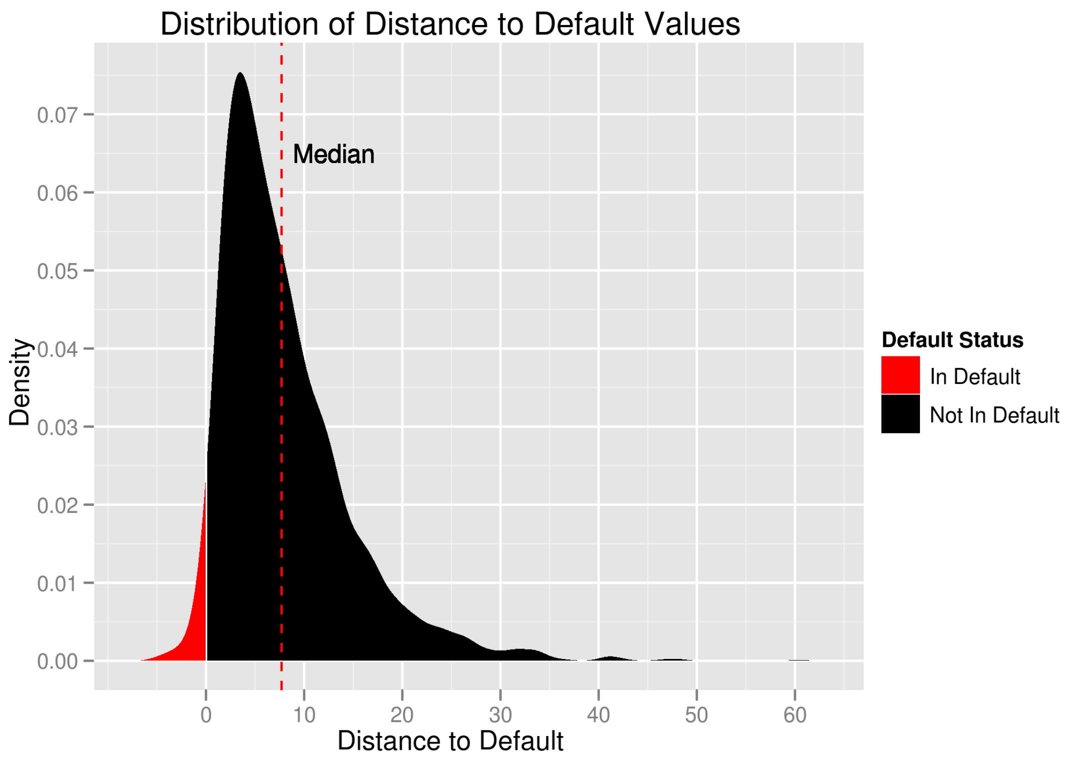


Figure 4: Distribution of Distance to Default Values for 3600 companies considered in this analysis. A distribution of these values could be computed for each sector, and this could be used to determine the relative fiscal health of a given company in that sector.

Separating each 2-digit NAICS code, we see that the distribution is generally similar for each NAICS code, with the notable exception of sector 22, Power and Utility Generation. As these industries are heavily subsidized by the government, it would make sense that the distance to default distribution is shifted right somewhat (that is, default appears to be less probable).

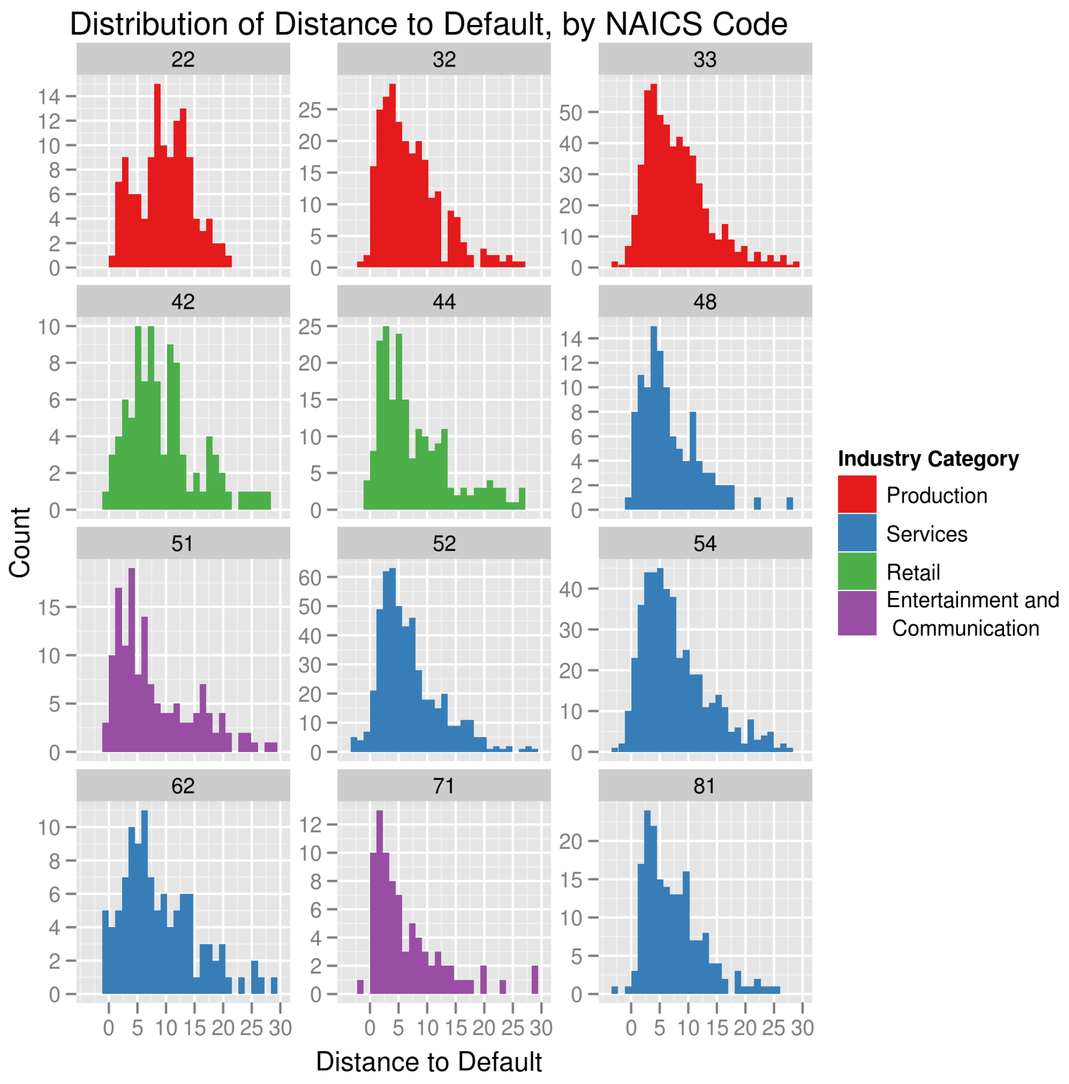


Figure 5: Distribution of Distance to Default Values for a selection of 2-digit NAICS codes.

Similarly, across the broad categories that we used to filter NAICS codes in Section 3, the distributions of distance-to-default are quite similar - skewed, with most of the mass between 0 and 20.

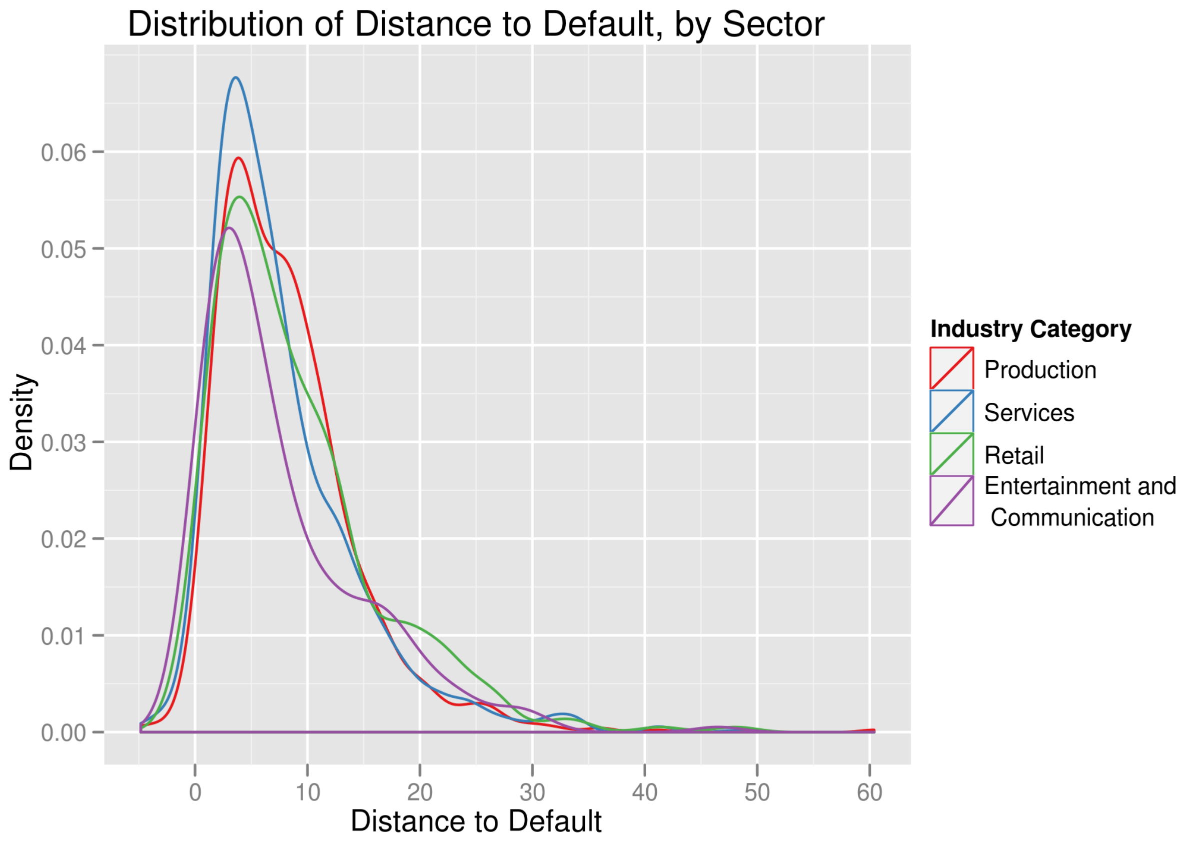


Figure 6: Distribution of Distance to Default Values by Sector.

5 Estimation of Asset Correlation

Correlation between distinct assets results in a portfolio that can quickly become unstable due to market fluctuations. This correlation should be considered as a risk factor during acquisition of a new asset, as a new asset may either increase or decrease the diversity of the portfolio, depending on its correlations with individual assets. These correlations may be assessed with NAICS codes (similar industries should be correlated) or with historical data (though this may lead to incorrect conclusions given a novel scenario, as occurred during the recent housing crash). We will consider both approaches in subsequent sections.

5.1 Linguistic similarity of NAICS code descriptors

NAICS code descriptions provide a useful (though simplistic) proxy for relatedness of industry wide effects. We would expect, for instance, that the success of sawmills would be related to the state of the construction industry, and vice versa. As such, we used the short

NAICS description and created a matrix of the proportion of similar words across the first four digits of NAICS codes.

While this method is crude, it could be expanded quite elegantly to account for the reoccurrence of words (i.e. farming) and relationships between words (i.e. tree, wood, lumber) with some effort. Due to limited time, we chose to limit our analysis to this quick and dirty approach.

Grouping across four-digit NAICS codes produces the following measures of similarity between 9 sample companies:

	BBBY	CVS	DUK	GBX	HD	M	ODFL	RCII	SHEN
BBBY		0.29	0.00	0.00	0.37	0.35	0.00	0.06	0.00
CVS	0.29		0.00	0.00	0.24	0.30	0.00	0.15	0.00
DUK	0.00	0.00		0.10	0.00	0.00	0.00	0.00	0.00
GBX	0.00	0.00	0.10		0.00	0.00	0.05	0.04	0.00
HD	0.37	0.24	0.00	0.00		0.28	0.00	0.06	0.00
M	0.35	0.30	0.00	0.00	0.28		0.00	0.00	0.00
ODFL	0.00	0.00	0.00	0.05	0.00	0.00		0.00	0.00
RCII	0.06	0.15	0.00	0.04	0.06	0.00	0.00		0.00
SHEN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

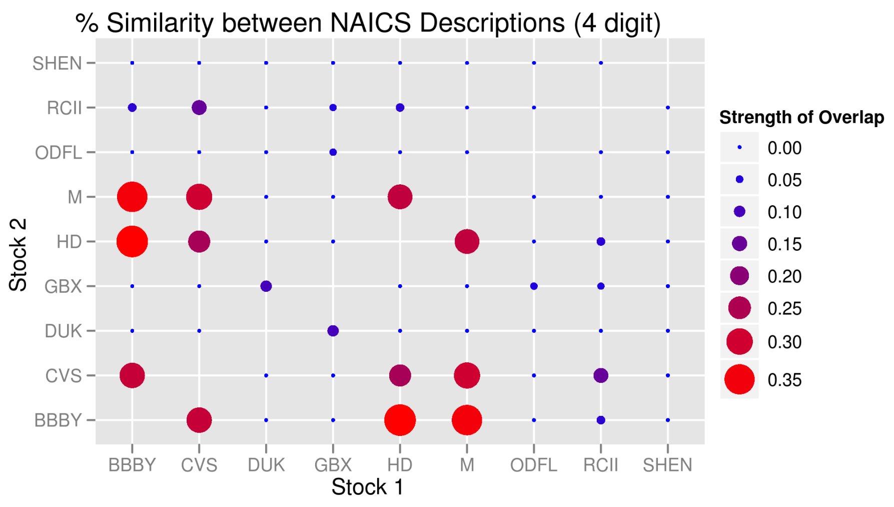


Figure 7: 4 Digit NAICS code Description Linguistic Similarity

Home Depot, Macys, and Bed Bath and Beyond show the most lexical similarity, as they are all retail-based stores.

Using two-digit NAICS codes produces a more coarse-grained measure of similarity that identifies only major industry similarities. In situations such as the current recession, this may be useful for predicting industry-wide correlations, since retail-based sectors as a whole seem to suffer. However, the measure of sector similarity would likely improve given a longer textual description of the NAICS two-digit codes (this would differentiate certain classes of retail businesses, for example).

	BBBY	CVS	DUK	GBX	HD	M	ODFL	RCII	SHEN
BBBY		1.00	0.06	0.15	1.00	0.33	0.06	0.12	0.04
CVS	1.00		0.06	0.15	1.00	0.33	0.06	0.12	0.04
DUK	0.06	0.06		0.04	0.06	0.04	0.11	0.03	0.03
GBX	0.15	0.15	0.04		0.15	0.07	0.10	0.13	0.04
HD	1.00	1.00	0.06	0.15		0.33	0.06	0.12	0.04
M	0.33	0.33	0.04	0.07	0.33		0.08	0.07	0.04
ODFL	0.06	0.06	0.11	0.10	0.06	0.08		0.23	0.01
RCII	0.12	0.12	0.03	0.13	0.12	0.07	0.23		0.02
SHEN	0.04	0.04	0.03	0.04	0.04	0.04	0.01	0.02	

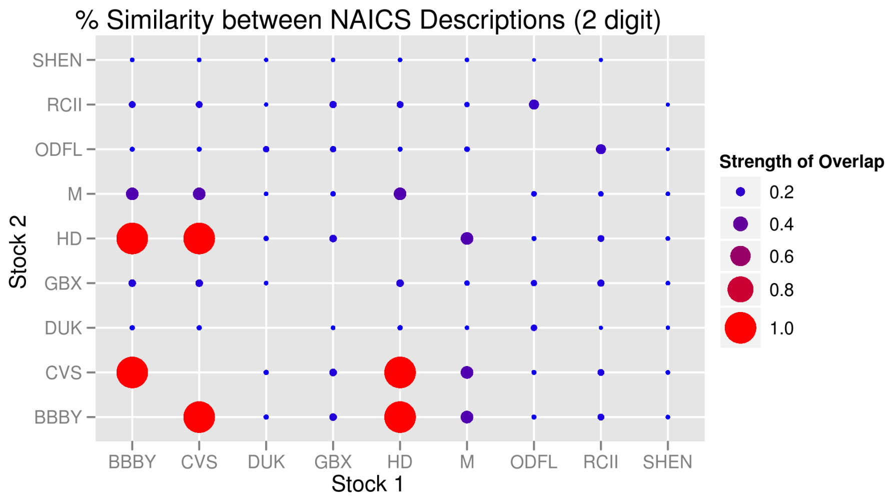


Figure 8: 2 Digit NAICS code Description Linguistic Similarity

We should note that this measure is not a true "correlation", but rather a measure of semantic similarity, as there is no possibility of negative values. However, we believe that given more textual information and/or a more sophisticated algorithm, this could be a very useful measure of the relationship between sectors.

One way to expand the lexical correlations between companies would be to compare the company descriptors (or longer NAICS code descriptors) to get a better measure of lexical similarity. It would also be reasonable to apply the LSI algorithm to measure lexical similarity between NAICS codes (rather than applying it to determine which NAICS code is lexically similar to a paragraph).
An alternate method to calculate the relationship between two NAICS codes or Industries would be to construct a graph, such that the NAICS code is the node and any similar words in the NAICS descriptions are edges connecting the nodes (and are possibly weighted, based on the word similarity). Using this method, the interconnectedness of several different industries could be mapped simultaneously, providing a (likely) more robust method of assessing the interconnectedness of different assets.

5.2 Correlation of Historical Performance as a Measure of Asset Similarity

Another method for assessing similarity of stock performance is to measure the correlation of historical performance. Specifically, to assess the correlation in performance of Company A and Company B (in order to assess the likelihood that A and B will default in close temporal proximity, we can calculate

$$\mathrm{Cor}(A, B) = \frac{E\left[(A - \mu_A)(B - \mu_B)\right]}{\sigma_A \sigma_B}$$

where $E[\cdot]$ is the expectation, μ_A is the overall mean of the price of stock A (or some other assessment of the well-being of A), and σ_A is the standard deviation of the price of stock A.

We have created a matrix of correlations of historical stock data for some 3750+ companies to use as a proof-of-concept for the purposes of this project. A sample of the stock correlations for 9 companies is shown below:

	BBBY	CVS	DUK	GBX	HD	M	ODFL	RCII	SHEN
BBBY		0.88	-0.30	0.94	0.85	0.79	0.22	0.94	-0.21
CVS	0.88		0.01	0.81	0.71	0.58	0.32	0.79	0.08
DUK	-0.30	0.01		-0.35	-0.23	-0.30	0.31	-0.32	0.56
GBX	0.94	0.81	-0.35		0.82	0.81	0.15	0.95	-0.13
HD	0.85	0.71	-0.23	0.82		0.87	0.01	0.87	-0.17
M	0.79	0.58	-0.30	0.81	0.87		-0.12	0.90	-0.13
ODFL	0.22	0.32	0.31	0.15	0.01	-0.12		0.13	0.27
RCII	0.94	0.79	-0.32	0.95	0.87	0.90	0.13		-0.08
SHEN	-0.21	0.08	0.56	-0.13	-0.17	-0.13	0.27	-0.08	

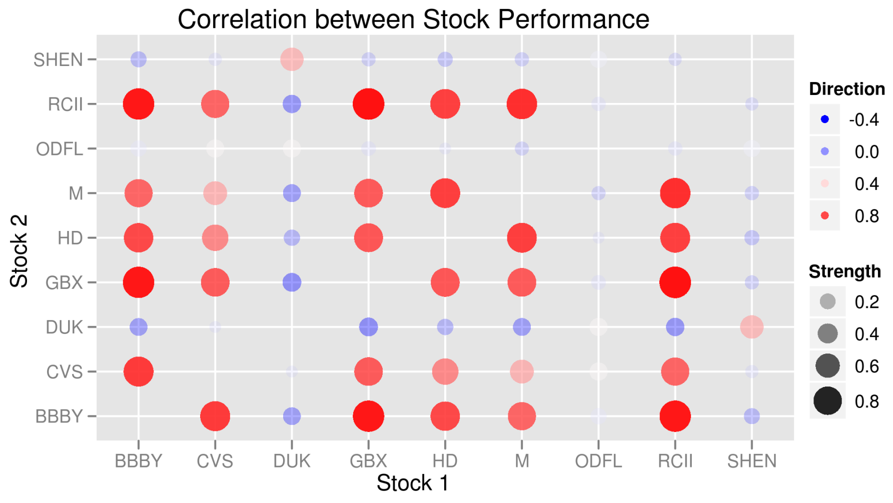


Figure 9: Correlation Between Stock Prices of Sample Companies

This measure of performance correlation is much more computationally intensive to calculate, but it can be stored in a database and updated only when necessary, so this is not a performance concern. It appears to provide a much better measure of correlations in performance of each company, but does not account for industry class. Additionally, privately held companies cannot be compared using this method. It should be possible, however, to use other numerical assessments of a company's health over time to correlate historical performance and determine whether a potential investment is likely to correlate with other assets.

Using either (or both) methods for assessing the correlation between the performance of two companies, it would then be possible to evaluate the relative merit of acquisition of an asset based on its contribution to the diversity of the portfolio. Using this measure and a measure of the asset's distance to default compared to the average for a company in the same market, it should be possible to create a simple numerical descriptor of the asset's relative risk and potential reward compared to the entire portfolio.

References

[1] P. Crosbie, J. Bohn, *Modeling Default Risk,* Moody's KMV Company, 2003.

[2] J.Hull, *Options, Futures, and Other Derivative,* Prentice Hall, New Jersey, 1993.

[3] Jeffrey A. Ryan (2010). quantmod: Quantitative Financial Modelling Framework. R package version 0.3-15. `http://CRAN.R-project.org/package=quantmod`

[4] R Development Core Team (2010). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. `http://www.R-project.org/`

A Data Acquisition and Cleaning

R[4] statistical software was used to acquire and clean much of the data used in the project. Specifically, the package `quantmod`[3] was used to acquire stock data, containing the debt and equity values (reported quarterly) and closing stock prices. As debt and equity reports are made to the SEC on a quarterly basis, we had access to the previous year's data, and then the previous 4 years' annual data. Presumably, this data is available on a more detailed level from financial services, but for the purposes of this project, quarterly and annual data were deemed sufficient. Data was acquired from Yahoo! finance [2] by means of functions in the `quantmod` package, notably `getFinancials` and `getSymbols`, which retrieve the equity/debt reports and daily closing stock prices respectively.

Data were acquired as available for any company listed on the NYSE, any component corporation of NASDAQ, and any component corporation of the S& P 500. It happened that for many companies, the Yahoo! finance page was not parseable (or did not exist), so these companies were summarily excluded.

Additionally, the SIC code was acquired from the SEC website, and matched to the corresponding NAICS code(s). In cases in which this was not a one-to-one matching, the first match was chosen (for simplicity).

[2] `http://finance.yahoo.com/`