Undergraduate Research in Mathematics as a Strategy to Increase Academic Achievement and Promote Students to Higher Education in Mathematics[*]

Student Publications and Technical Reports

Summer 2007 – Spring 2009[†]

1. Yesenia Cruz, A Mathematical Model for Detecting Diabetes, Proceedings of Proceedings of the National Conferences on Undergraduate Research, University of Wisconsin La-Crosse, La-Crosse, Wisconsin, April 2009.
2. Abner Ortiz and Greichaly Cabrera, The Non-Linear Cantilever: A Graphical User Interface, Proceedings of Proceedings of the National Conferences on Undergraduate Research, University of Wisconsin La-Crosse, La-Crosse, Wisconsin, April 2009.
3. Desire E. Velzquez Ros, John E. Morales Garca, and Axel Y. Rivera Rodrguez, A GUI For The Analysis of Electrostatic Interactions in Molecular Dynamics Simulations, Proceedings of Proceedings of the National Conferences on Undergraduate Research, University of Wisconsin La-Crosse, La-Crosse, Wisconsin, April 2009.
4. Yesenia Cruz, On the construction of state transfer matrices of close, Proceedings of the National Conferences on Undergraduate Research, Salisbury, Maryland, April 2008.
5. Joyce Fernández, Characterization of textures in microphotographies of leaves epidermis, Proceedings of the National Conferences on Undergraduate Research, Salisbury, Maryland, April 2008.

[*]National Security Agency (NSA) Grant Number H98230-07-1-0114

[†]We included in this report some student papers from Spring 2009 that acknowledge support from this grant.

6. Jessica Flores, The numerical computation of the critical boundary displacement for radial cavitation for composite materials, Proceedings of the National Conferences on Undergraduate Research, Salisbury, Maryland, April 2008.

7. Myrna Merced, Study of stable conformations of cnt-dna hybrids by means of principal component analysis, Proceedings of the National Conferences on Undergraduate Research, Salisbury, Maryland, April 2008.

8. John Morales, Usage of OpenGL for the Representation of a Atom Trajectory, Technical Report, May 2008.

9. Axel Rivera, A methodology to determine the number of clusters in unsupervised hyperspectral image classification, Proceedings of the National Conferences on Undergraduate Research, Salisbury, Maryland, April 2008.

10. Desirée E. Velázquez, A Graphical User Interface to Specify Parameters for Molecular Dynamics Simulations, Technical Report, May 2008.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2009
University of Wisconsin La-Crosse
La-Crosse, Wisconsin
April 16 - 18, 2009

Mathematical Model for Detecting Diabetes

Yesenia Cruz Rosado
Department of Mathematics
The University of Puerto Rico at Humacao
Humacao, Puerto Rico

Faculty Advisor: Prof. Pablo Negrón

Abstract

Diabetes is a syndrome of disordered metabolism, usually due to a combination of hereditary and environmental causes, resulting in abnormally high blood sugar levels. Various hormones in our body such as insulin, growth hormone, glucagon control blood glucose levels, epinephrine best know as adrenaline, glucocorticoids and thyroxine. The two most common forms of diabetes are due to either a diminished production of insulin (Type 1 diabetes), or diminished response by the body to insulin (Type 2 and gestational diabetes). Both lead to hyperglycemia, which largely causes the acute signs of diabetes: excessive urine production, resulting compensatory thirst and increased fluid intake, blurred vision, unexplained weight loss, lethargy, and changes in energy metabolism. We will explain how each hormone is activated and how it affects glucose levels in blood. We present a mathematical model that determines diabetes in patients based in the results on the glucose intolerance test of 5 hours. Our model extends the one proposed by E. Ackerman[2] (1969) to include three instead of two hormones concentrations. In particular we include concentrations for glucose, glucagon and a global variable that includes other hormones such as insulin. The model is based on a 3x3 system of non-homogenous ordinary differential equations. A nonlinear least square method is used to determine the coefficient parameters of the system based on actual data from GTT. The simulations also provide an indicator similar to the one proposed by E. Ackerman (1969), to diagnose a diabetic condition. Additionally, we develop a graphical user interface to facilitate the entering of the patient's data and the visualization of the results.
Keywords: differential equations, diabetes, simulations, graphical user interface

1. Introduction

How do you find out that you have Type 2 diabetes? Often, because there may not be noticeable symptoms, the diagnosis is made during an annual physical or checkup. Your doctor may order a Fasting Blood Sugar (FBS), or an Oral Glucose Tolerance Test (OGTT) better know as GTT to help determine whether you have diabetes. What do these tests mean?

The FBS is a fasting test, meaning that you can't eat for 8-10 hours before you have your blood drawn. Most people like to go for the test first thing in the morning after fasting all night. A fasting blood glucose of 70 mg/dl to 100 mg/dl is normal. If your fasting blood glucose level comes back between 100 mg/dl and 125 mg/dl then you are considered to have impaired fasting glucose or pre-diabetes. A fasting glucose over 125 mg/dl indicates that you have type 2 diabetes. Most doctors like to get a fasting blood sugar on two separate occasions to make sure of the diagnosis. Expected measurements can be found in Table 1.

The GTT is a glucose challenge test. A fasting blood glucose is usually taken first to establish a baseline level. Then you are given a 75 grams glucose drink. Two hours later another blood sample is drawn to check your glucose level. If your blood glucose is under 140 mg/dl then your glucose tolerance is considered normal. If it is 140 mg/dl to 200 mg/dl, then you have impaired glucose tolerance or pre-diabetes. If your glucose is over 200 mg/dl then a diagnosis of type 2 diabetes is made. Again, your doctor will usually perform this test on two different occasions before a definite diagnosis is made. A very serious difficulty associated with this method of diagnosis is that no universal accepted criterion exists for interpreting the results of the GTT.

Diabetes mellitus is the most common endocrine disorder. The diagnosis requires a fasting plasma glucose of (>140 mg/dL) on two occasions. Following ingestion of 75 grams of glucose, the finding of a venous plasma glucose of (>120 mg/dL) after two hours and on at least another occasion during the two-hour test is suggestive. In the case of diagnosing hypoglycemia, it requires a plasma glucose (<45-70 mg/dL) no more than two occasions.

2. Preliminaries

Blood glucose levels are controlled by various hormones in our body such as insulin, growth hormone, glucagon, epinephrine best know as adrenaline, glucocorticoids and thyroxine. Our model's goal is to determine if a patient has diabetes taking in consideration the glucose intolerance test (GTT) and the normal glucose levels that the patient should present (Table 1). The model will determine how much the different hormones influence in those levels of sugar in blood. Therefore an explanation on how each hormone is activated and how it affects glucose levels in blood is now given.

Table 1. Goals for blood glucose in the control of diabetes

	Aceptable		***Ideal***	
Goal	***Mmol/L***	***mg/dL***	***mmol/L***	***mg/dL***
Fasting	3.3-7.2	60-130	3.9-5.6	70-100
Prepandial	3.3-7.2	60-130	3.9-5.6	70-100
Postprandial(1 h)	<11.1	<200	<8.9	<160
3 A.M.	>3.6	>65	>3.6	>65

2.1 insulin

The hormone insulin is made in the beta cells of the pancreas and is secreted when the body presents high blood sugar levels. When only 10-20% of beta cells are working properly then the sings of diabetes tend to show. Insulin causes most of the body's cells to take up glucose from the blood (including liver, muscle, and fat tissue cells), storing it as glycogen in the liver and muscle, and stops use of fat as an energy source. When insulin is absent (or low), glucose is not taken up by most body cells and the body begins to use fat as an energy source (ie, transfer of lipids from adipose tissue to the liver for mobilization as an energy source). When sugar levels are high in the body then the insulin hormone is segregated. When control of insulin levels fail, diabetes mellitus results. On the other hand, an excess of insulin results in hypoglycemia.

2.2 glucagon

Glucagon is an important hormone involved in carbohydrate metabolism. Produced by the alpha cells in the pancreas, it is released when the glucose level in the blood is low (hypoglycemia), causing the liver to convert stored glycogen into glucose and release it into the bloodstream. The action of glucagon is thus opposite to that of insulin, which instructs the body's cells to take in glucose from the blood in times of satiation. In this action if there is no sufficient glucose in blood the glucagon takes the reserves of glucose stored in the liver.

2.3 adrenaline

Also known as epinephrine is a hormone is released form the adrenal glands when danger threatens or in an emergency. The hormone boosts the supply of oxygen and glucose to the brain and muscles, while suppressing other non-emergency bodily processes (digestion in particular). It increases heart rate and stroke volume, dilates the pupils, and constricts arterioles in the skin and gastrointestinal tract while dilating arterioles in skeletal muscles. In times of extreme hypoglycemia it elevates the blood sugar level by increasing catabolism (breakdown) of glycogen to glucose in the liver, and at the same time begins the breakdown of lipids in fat cells. It is important to note that adrenaline mobilizes the glucose reserves in the liver and muscles while glucagon only access the liver reserves. It is important to note that adrenaline is not the automatic response of the body in case of hypoglycemia and therefore we will concentrate in study glucagon segregation that will be given by the glucose level in blood.

2.4 thyroxine

Thyroxine is the major hormone secreted by the follicular cells of the thyroid gland. It is important to note that is involved in controlling the rate of metabolic processes in the body and influencing physical development. Diabetic patients have a higher prevalence of thyroid disorders compared with the normal population [3]. The presence of thyroid dysfunction may affect diabetes control. Hyperthyroidism is typically associated with worsening glycemic control and increased insulin requirements. In patients without any thyroid dysfunction it normally segregates the tryroxine hormone which influence in the metabolism of the body ergo it can either increased or decreased blood sugar levels.

2.5 glucocorticoids

Glucocorticoids is the hormone secreted by the adrenal cortex and plays and important role in the metabolism of carbohydrates. The name "glucocorticoid" derives from early observations that these hormones were involved in glucose metabolism. In the fasted state, glucocorticoid stimulates several processes that collectively serve to **increase and maintain normal concentrations of glucose** in blood.

The metabolic effects include the **inhibition of glucose uptake** in muscle and adipose tissue: A mechanism to conserve glucose and stimulation of gluconeogenesis, particularly in the liver. Gluconeogenesis is a metabolic pathway that results in the generation of glucose from non-carbohydrate carbon.

The vast majority of gluconeogenesis takes place in the liver and, to a smaller extent, in the cortex of kidneys. **This process occurs during periods of fasting, starvation, or intense exercise** and is highly energetic. Gluconeogenesis is often associated with ketosis. Gluconeogenesis is also a target of therapy for type II diabetes, such as metformin, which inhibit glucose formation and stimulate glucose uptake by cells.

2.6 growth hormone (somatotropin)

The growth hormone or somatotropin in **segregated by the delta cells** in the pancreas. It intervenes directly on the **regulation of glycemic** and the **segregation depends on the high levels of glucose**, amino acids and glucagon. In addition to increasing height in children and adolescents, growth hormone has many other effects on the body. This is to reduce liver uptake of glucose and promote gluconeogenesis in the liver, therefore it increases the glucose levels in blood.

It is important to note when the growth hormone is segregated it increases blood sugar levels. It is believed that the growth hormone decreases the sensitivity of muscle and adipose membrane to insulin, thereby reducing the effectiveness of insulin in promoting glucose uptake[1].

3. Mathematical Model

The model proposed serves to interpret the results of the Glucose Tolerance Test (GTT) on either normal or diabetes patients. We know that glucose plays an important role on our performance which depends on the metabolism system. Glucose provides energy to tissue and organisms but the levels provided depend on various hormones such as: insulin, growth hormone, glucagon, epinephrine best know as adrenaline, glucocorticoids and thyroxine. A standard criterion does not exist to analyze the (GTT) results which can be a problem. The model proposed in this paper will separate in three groups the hormones that influence glucose levels in blood. In this way we can group the hormones that elevate glucose levels in blood separated from those that lower them.

3.1 variables

In our model we center our attention on 3 concentrations: G(*t*) denotes blood glucose concentrations; E(*t*) denotes blood glucagon concentrations; H(*t*) denotes the rest of the hormones concentrations. The equations of the model are given by:

$$\frac{dG(t)}{dt} = F_1(G(t), E(t), H(t)) + J(t) \tag{1}$$

$$\frac{dE(t)}{dt} = F_2(G(t), E(t), H(t)) \tag{2}$$

$$\frac{dH(t)}{dt} = F_3(G(t), E(t), H(t)) + K(t) \tag{3}$$

where $J(t), K(t)$ denote external rates of supplied glucose and hormones (like insulin). As it is, the model is quite general. We will consider only small perturbations or variations from a steady state or equilibrium point of the system. If (G_0, E_0, H_0) represents such a state, then it is characterized by the equations:

$$F_1(G_0, E_0, H_0) = 0\,,\ F_2(G_0, E_0, H_0) = 0\,,\ F_3(G_0, E_0, H_0) = 0\,. \tag{4}$$

We assume that *G*, *E* and *H* have achieved the optimal values G_0, E_0 and H_0 by the time the fasting patient has arrived at the hospital. Let

$$g(t) = G(t) - G_0, \quad e(t) = E(t) - E_0, \quad h(t) = H(t) - H_0\,, \tag{5}$$

represent small variations from the corresponding optimal values. Thus we have the linearized version of equations (1), (2) and (3):

$$\frac{\mathrm{d}g(t)}{\mathrm{d}t} = a_1\, g(t) + b_1\, e(t) + c_1\, h(t) + J(t), \tag{6}$$

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} = a_2\, g(t) + b_2\, e(t) + c_2\, h(t), \tag{7}$$

$$\frac{\mathrm{d}h(t)}{\mathrm{d}t} = a_3\, g(t) + b_3\, e(t) + c_3\, h(t) + K(t), \tag{8}$$

Note that glucagon, adrenaline and the growth hormone are hormones that increase blood glucose levels, and that insulin, thyroxin, have the opposite effect. Considerations like these allow us to determine the signs of the coefficients in this system. For example: if glucose levels are high $(g > 0)$ and the glucagon level and other hormones low $(e = 0, h = 0)$, then the glucose level should decrease $(a_1 < 0)$ due to tissue absorption. If the glucagon levels are high $(e > 0)$ and the glucose and other hormones levels are low $(g = 0, h = 0)$, then the glucose level should increase $(b_1 > 0)$ due to glycogen conversion to glucose. Similar considerations lead us to the following inequalities:

$$a_1 < 0,\ b_1 > 0,\ c_1 < 0,\ a_2 < 0,\ b_2 < 0,\ c_2 > 0,\ a_3 > 0,\ b_3 > 0,\ c_3 < 0 \tag{9}$$

We can incorporate this signs explicitly into the system of equations (6), (7) and (8) above by re-writing it as:

$$g'(t) = -\alpha_1\, g(t) + \beta_1\, e(t) - \gamma_1\, h(t) + J(t), \tag{10}$$

$$e'(t) = -\alpha_2\, g(t) - \beta_2\, e(t) + \gamma_2\, h(t), \tag{11}$$

$$h'(t) = \alpha_3\, g(t) + \beta_3\, e(t) - \gamma_3\, h(t) + K(t), \tag{12}$$

where all the constants $\alpha_i, \beta_i, \gamma_i$ are positive numbers.

The basic task of the blood glucose regulatory system is to bring perturbations from the steady state (G_0, E_0, H_0) back to it in time. With this in mind we look for conditions on the coefficient matrix in equations (10), (11) and (12) that guarantee that the equilibrium point (G_0, E_0, H_0) has this *stability* property. This will be so if all the eigenvalues of the coefficient matrix in (10), (11) and (12) have negative real parts. A necessary condition for this is that the determinant of the coefficient matrix be negative. This determinant is given by:

$$-(\alpha_1\beta_2\gamma_3 + \alpha_2\beta_1\gamma_3 + \alpha_3\beta_2\gamma_1) + \alpha_1\beta_3\gamma_2 + \alpha_3\beta_1\gamma_2 + \alpha_2\beta_3\gamma_1 \tag{13}$$

Observe that if β_3, γ_2 are small, with the rest of the coefficients fixed, then the determinant is negative. Henceforth we consider the limiting case of $\beta_3, \gamma_2 = 0$, the coefficient in equations (10), (11) and (12) reduces to:

$$\begin{pmatrix} -\alpha_1 & \beta_1 & -\gamma_1 \\ -\alpha_2 & -\beta_2 & 0 \\ \alpha_3 & 0 & -\gamma_3 \end{pmatrix}.$$

The characteristic polynomial of this matrix is given by:

$$p(\lambda) = \lambda^3 + d_1\lambda^2 + d_2\lambda + d_3, \tag{14}$$

where

$$\begin{aligned} &d_1 = \alpha_1 + \beta_2 + \gamma_3, \quad d_2 = \alpha_2\beta_1 + \alpha_1\beta_2 + \alpha_1\gamma_3 + \alpha_3\gamma_1 + \beta_2\gamma_3, \\ &d_3 = \alpha_2\beta_1\gamma_3 + \alpha_1\beta_2\gamma_3 + \alpha_3\beta_2\gamma_1\,. \end{aligned} \tag{15}$$

Note that the d_i's are all positive. By the Routh-Hurwitz stability criterion[5] we get that all of the roots of $p(\lambda)$ have negative real parts if and only if $d_1 d_2 > d_3$. To recapitulate, our problem reduces to find solutions of the system (10)-(12), where all the constants $\alpha_i, \beta_i, \gamma_i$ are positive numbers such that

$$d_1 d_2 > d_3. \tag{16}$$

Using the elimination method[4], we find that the system of equations (10)-(12) decouples into three equations:

$$L(g) = \widehat{J}(t), \quad L(e) = \widehat{M}(t), \quad L(h) = \widehat{K}(t), \tag{17}$$

where

$$L(u) = u'''(t) + d_1 u''(t) + d_2 u'(t) + d_3 u(t), \tag{18}$$

with d_1, d_2, d_3 as above, and

$$\begin{aligned} \widehat{J}(t) &= J''(t) + (\beta_2 + \gamma_3) J'(t) + \beta_2 \gamma_3 J(t) - \gamma_1 K'(t) - \beta_2 \gamma_1 J(t), \\ \widehat{M}(t) &= -\alpha_2 J'(t) - \alpha_2 \gamma_3 J(t) - \alpha_2 \gamma_1 J(t), \\ \widehat{K}(t) &= \alpha_3 J'(t) + \alpha_3 \beta_2 J(t) + K''(t) + (\alpha_1 + \beta_2) K'(t) + (\alpha_2 \beta_1 + \alpha_1 \beta_2) K(t). \end{aligned} \tag{19}$$

Note that the characteristic polynomial for the operator (21) is given by (14) and (15). By our assumptions on the coefficients of (14) and (15) and condition (19), we have that (14) has either three negative real roots; or one negative root and two complex conjugates roots with negative real part. Assuming the case of complex roots and taking for the moment $J = K = 0$, we have that the solution of the first equation in (10) is given by:

$$g(t) = Ae^{-at} + Be^{-bt}\cos(\omega t - \zeta), \quad a, b > 0, \tag{20}$$

and the blood glucose level is given by the model equation:

$$G(t) = G_0 + Ae^{-at} + Be^{-bt}\cos(\omega t - \zeta), \quad a, b > 0. \tag{21}$$

Note that there are 7 parameters to be the determined in this equation. (One can actually reduce it to 6 by requiring that $g(0) = 0$.) We compute the values of these parameters by a nonlinear least square method. If $G_1, G_2, G_3, \ldots, G_n$ are measurements of the patient's blood glucose concentration at times $t_1, t_2, t_3, \ldots, t_n$, then we find the values of $G_0, A, B, a, b, \omega, \zeta$ that minimize the mean square error function:

$$E = \sum_{k=1}^{n} [G_k - G_0 - Ae^{-at_k} - Be^{-bt_k}\cos(\omega t_k - \zeta)]^2. \tag{22}$$

Note that the requirement $a,b > 0$ implies that condition (19) is satisfied.

We implemented the nonlinear least square method using the functions provided by optimization toolbox in MATLAB™. We created a graphical user interface (GUI) with MATLAB™ as well, that facilitates the entering of the patients GTT data, process it with the least square method, plots a graph of the model (24), and computes a pseudo period for this function that will be used for the diagnosis of a diabetic condition.

4. Results

Since there is not a universal criterion to diagnose diabetes two doctors evaluating a certain patient's GTT results can result in two different diagnoses. Therefore, we want to determine an objective approach to analyze Glucose Intolerance Test (GTT) that can help improve a criterion to diagnose a Diabetes condition. In the model proposed by E. Ackerman (1969), the pseudo period $T = 2\pi / \omega$ was used as an indicator for a diabetic condition. Using data from a variety of sources they found that a value of less than four hours for the indicator indicated normalcy, while appreciably more than four hours implied mild diabetes. In this paper we use the same indicator to test for diabetes. We used data (provided by the Oriental Clinic Laboratory in Humacao) from patients of the Humacao area. The data includes the fasting glucose levels at t=0 and all others glucose levels every hour. We evaluated 15 patients whose name, sex, or age we did not know and compare each of the results to a single diagnosis of a certain doctor. Our results are given in (Table 2).

Table 2 glucose tolerance tests of 15 different patients

Patient Number	Hypothesis			Indicator	Conclusions	
	Normal	Diabetic	Pre-Diabetic		Diabetic	Normal
1	x			3.075		x
2			X	5.564	x	
3			X	1.370		x
4			X	3.1006		x
5	x		X	3.887		x
6			X	1.37		x
7			X	2.0351		x
8			X	5.679	x	
9			X	14.0503	x	
10			X	7.3986	x	
11			X	24.64	x	
12			X	3.352		x
13			X	4.477	x	
C 14	x			7.66	x	
15			X	1.847		x

5. Conclusions

The Oriental Clinic Laboratory provided our data. The results include the fasting glucose levels at t=0 and all others glucose levels every hour. We evaluated 15 patients whose name, sex, or age we did not know with our GUI and compare each of those results to a single diagnosis of a certain doctor. Our results are given in (Table 2).

We encounter that our model was correct in 8 of 15 tests. That gives us a 53% of success. Since our model only considers evaluation between normal or diabetic patients it fails to evaluate pre-diabetic patients. From the 7 patients that were misdiagnosing 5 were pre-diabetic patients and our model diagnosed them as either normal or diabetic. Also the four hour indicator threshold might need to be reevaluated for the Puerto Rican population due to differences in eating habits and preferences.

It is important to note that large deviations of G from G_0 usually indicate severe diabetes or pre-diabetes. Our model precludes by our assumption of small perturbations and the use of the linearization about an equilibrium point.

6. Future Work

We want to validate our method using data that represents better the Puerto Rican population. Also, we want to find a better indicator that considers conditions such as: diabetic, pre-diabetic and normal patient. Finally we will add more features to the GUI, e.g., to provide information on other hormones.

7. Acknowledgments

This work has been supported in part by the National Security Agency, Grant Number H98230-04-C-0486 and the National Science Foundation MARC Program at the UPRH.

8. References

1. M. Braun, “A mathematical model for the detection of diabetes”, in Differential Equations Models, Vol. 1, Edited by M. Braun, C. Coleman, and D. Drew, Spreinger-Verlag, pp.101-108, 1983.
2. E. Ackerman, I. Gatewood, J. Rosevear, and G. Molnar, “Blood glucose regulation and diabetes”,Concepts and Models of Biomathematics, F. Heinmets, Ed., Marcel Decker, pp. 131-156, 1969.
3. P. Wu, “Thyroid disease and diabetes”, Journal: http://journal.diabetes.org/clinicaldiabetes/v18n12000/Pg38.html
4. L.S. Pontryagin, Ordinary Differential Equations, Addison-Wesle 1962.
5. Gradshteyn, I. S. and Ryzhik, I. M. "Routh-Hurwitz Theorem." §15.715 in Tables of Integrals, Series, and Products, 6th ed. San Diego, CA: Academic Press, p. 1076, 2000.
6. Séroul, R. "Stable Polynomials." §10.13 in Programming for Mathematicians. Berlin: Springer-Verlag, pp. 280-286, 2000.
7. C. Landersdorfer and W. Jusko, “Pharmacokinetic/ Pharmacodynamic Modelling in Diabetes Mellitus”. Clin Pharmacokinet 2008; 47(7), pp. 417-448.

Proceeding of the National Conference
On Undergraduate Research (NCUR) 2009
University of Wisconsin-La Crosse
La Crosse, Wisconsin
April 16-18, 2009

The Non-Linear Cantilever: A Graphical User Interface

Abner Ortiz and Greichaly Cabrera
Department of Mathematics
University of Puerto Rico at Humacao
Humacao, PR 00791-4300

Faculty Advisor: Pablo V. Negrón

Abstract

The cantilever problem consists of studying the deformations of a bar or rod that is attached to a wall on one end and is subjected to a torque or applied force on the other end. In the classical cantilever problem, the constitutive functions (the functions characterizing the material that composes the bar) are linear, the material of the bar is homogeneous, inextensible and unshearable; there is no applied torque, and the applied force is vertical. The classical problem was studied by Jas. Bernoulli (1694) and L. Euler (1727). The cantilever problem still has many applications in engineering, and more recently in nano technology. This project is considering a nonlinear model of the cantilever in which the material of the bar is non-homogeneous, extensible and shearable. The nonlinear model is introduced to describe a finite difference numerical scheme for computing approximate solutions of the problem. The resulting nonlinear system of equations is solved with Newton's method, by taking advantage of the structure of the Jacobian matrix (almost tridiagonal) to solve the intermediate linear systems efficiently. Moreover it has been developed a graphical user interface which allowed us to experiment with the model and to control more effectively the different constitutive and force parameters. These tools are used to study the dependence of the bar deformations on thickness variations and the different constitutive parameters and applied forces. Also, this helps in the study of the severity or magnitude of the shear strain as the parameters and forces are changed.
Keywords: Nonlinear Cantilever, GUI

1. Introduction

We consider the problem of finding the shape that assumes a bar composed of a certain material, when it is attached to a wall, and we apply some force or torque on the other end. This problem is known as the *cantilever*. In the classical problem of the cantilever (Bernoulli, 1694; Euler, 1727), the functions that describe the material composing the bar (*constitutive functions*) are assumed to be linear. In this paper we consider nonlinear constitutive functions for the material behavior that include effects for shear, bending, and torsion.[1] For simplicity it is assumed that the cross sections of the bar are circular (see Figure 1).

The problem of the cantilever has many applications in engineering in particular for the construction of bridges. It has also become important in the field of nano science or technology as some deformations of nano fibers can be described very well with a model of a cantilever.[3, 4] An interesting problem would be to compare the results of using the models in this paper based on macro mechanical behaviors, with models of deformations of these nano fibers based on molecular dynamics.

The computer simulations performed nowadays in many areas of modern science use what is known as a *graphical user interface* or GUI. The GUI helps the data entry to the computational module which performs the required computations with the values entered by the user. The computational package MATLAB^{TM} provides several tools to create the graphical user interfaces. Using these tools we constructed a GUI for the non-linear cantilever problem that allowed us to systematically change mechanical and constitutive parameters, and then generate different types of deformations.

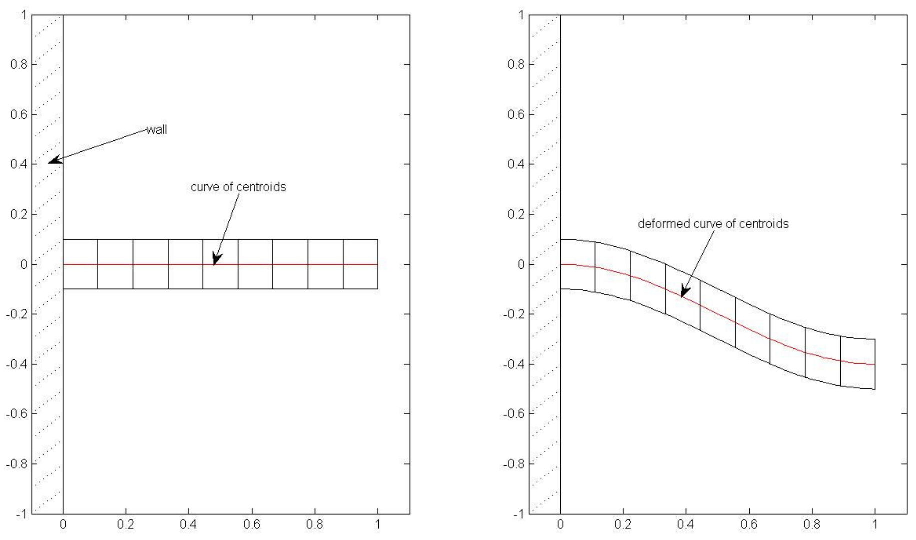


Figure 1: Reference and deformed configurations of the bar.

2. The Mathematical Model

In the Cosserat's special theory, [1] a planar configuration of a column can be described with two functions **r, b**: $[0,1] \to$ span{**i, j**}. The unit vector **b**(s) is called the *directrix* at *s*. If we define the unit vector **a = - k × b**, then **a, b** belong to span{**i,j**}. Hence there exists a function θ(s) such that:

$$\mathbf{a}(s) = \cos\theta(s)\mathbf{i} + \sin\theta(s)\mathbf{j}, \qquad \mathbf{b}(s) = -\sin\theta(s)\mathbf{i} + \cos\theta(s)\mathbf{j}.$$

Since {**a, b**} is a base for span{**i, j**}, we can write:

$$\mathbf{r}'(s) = v(s)\mathbf{a}(s) + \eta(s)\mathbf{b}(s), \tag{1}$$

for some functions v(s), η(s). These functions together with $\mu(s) = \theta'(s)$ are called the *strains* and they completely characterize the deformation of the column. To ensure that the deformation of the bar is not so severe as to make **r** and **b** parallel, we require that $v(s) > 0$, $s \in [0,1]$.

2.1 mechanical behavior

The contact force and torque exerted by the segment [s,1] of the bar on the segment [0,s] are given by **n**(s) and **m**(s), respectively, while the external (body) force and torque per unit length at the point s are given by **f**(s) and **l**(s), respectively. The equations of equilibrium for the deformed bar are now given by[1]:

$$\mathbf{n}'(s) + \mathbf{f}(s) = \mathbf{0}, \qquad \mathbf{m}'(s) + \mathbf{r}'(s) \times \mathbf{n}(s) + \mathbf{l}(s) = \mathbf{0}. \tag{2}$$

Since we are assuming the deformation of the bar is planar, then there exists functions *N*(s), *H*(s), *M*(s) such that

$$\mathbf{n}(s) = N(s)\mathbf{a}(s) + H(s)\mathbf{b}(s), \qquad \mathbf{m}(s) = M(s)\mathbf{k}. \tag{3}$$

2.2 boundary conditions

The boundary conditions at s = 0, 1 can be specified in several ways. We discuss the conditions at the end s = 1 of the bar, the other case being similar. Given the vectors $\mathbf{r}_1$, $\mathbf{n}_1$, we can specify that

$$\mathbf{r}(1) = \mathbf{r}_1, \quad \text{or} \quad \mathbf{n}(1) = \mathbf{n}_1 .. \tag{4}$$

In addition, given $\theta_1, M_1,$ we could have that

$$\theta(1) = \theta_1 \quad \text{or} \quad M(1) = M_1. \tag{5}$$

The *boundary value problem* for the deformations of the bar is given by (2) together with one of the boundary conditions in (4) and another from (5), and the same for s = 0.

2.3 the equations for the nonlinear cantilever

Suppose that the initial figuration of the bar is like in Figure 1 and that **f**(s) = 0, **l**(s) = 0, for all s in (2). From the first equation in (2) we have that **n**(s) = constant. If at s=1 we have an applied force given by the vector $\mathbf{n}_1 = -\lambda\left(\cos\alpha\,\mathbf{i} + \sin\alpha\,\mathbf{j}\right),$ then, we have that

$$\mathbf{n}(s) = -\lambda\left(\cos\alpha\,\mathbf{i} + \sin\alpha\,\mathbf{j}\right),$$

for all s. It follows now from (3) that

$$N(s) = -\lambda\cos(\theta(s) - \alpha), \quad H(s) = \lambda\sin(\theta(s) - \alpha). \tag{6}$$

One can show now that the second equation in (2) is equivalent to

$$M'(s) + \lambda[\nu(s)\sin(\theta(s) - \alpha) + \eta(s)\cos(\theta(s) - \alpha)] = 0\,. \tag{7}$$

So far the functions *N*(s), *H*(s), *M*(s) have not been specified. We assume that

$$N(s) = \widehat{N}(\nu(s)), \quad H(s) = D\eta(s), \quad M(s) = (EI)(s)\mu(s), \tag{8}$$

where $D > 0$,

$$\widehat{N}(\nu) = A\nu^{a} - B\nu^{-a} - A + B, \quad A, B \geq 0, \quad a > 0, \tag{9}$$

and $(EI)(s) > 0$ for all s. The function $(EI)(s)$ contains the information of the geometrical properties of the cross sections of the bar. Using these expressions together with (6) and (7) we can get the functions $\nu(s), \eta(s)$ in terms of $\theta(s)$ and a differential equation for $\theta(s)$:

$$\nu^{a}(s) = \frac{P(s) + \sqrt{P^{2}(s) + 4AB}}{2A}, \quad P(s) = -\lambda\cos(\theta(s) - \alpha) + A - B, \quad \eta(s) = \frac{\lambda}{D}\sin(\theta(s) - \alpha), \tag{10}$$

$$\frac{d}{ds}\left[(EI)(s)\frac{d\theta}{ds}(s)\right] + \lambda[\nu(s)\sin(\theta(s) - \alpha) + \eta(s)\cos(\theta(s) - \alpha)] = 0. \tag{11}$$

The boundary conditions that the bar is attached to a wall on the left side and that a torque is applied on the right side are equivalent to

$$\theta(0) = 0, \qquad \theta'(1) = \gamma, \tag{12}$$

where γ is proportional to the applied torque.

The equations (10), (11) and (12) constitute the boundary value problem for the cantilever. (The case $\alpha = \pi/2, \nu = 1, \eta = 0$ corresponds to the classical cantilever problem.) After solving these equations for the function $\theta(s)$, we have from (1), (10) and $\mathbf{r}(0) = \mathbf{0}$ that the deformed curve of centroids is given by:

$$\mathbf{r}(s) = \left(\int_0^s [\nu(t)\cos\theta(t) - \eta(t)\sin\theta(t)]dt \right)\mathbf{i} + \left(\int_0^s [\nu(t)\sin\theta(t) + \eta(t)\cos\theta(t)]dt \right)\mathbf{j}. \tag{13}$$

3. The Numerical Method

The equations (10), (11) and (12), in general, can not be solved in exact or closed form. It is therefore necessary to turn to numerical methods to approximate its solutions. In this section we will describe a *finite difference method* to approximate these solutions.

First we construct a uniform partition of the interval [0, 1] into *n* sub-intervals. Taking $h = 1/n$ we have that the i-th interval in such a partition is given by $[s_{i-1}, s_i]$, $1 \le i \le n$, where $s_i = ih$, $0 \le i \le n$. We write $s_{i-1/2}$ to represent the mid point of the interval $[s_{i-1}, s_i]$, that is: $s_{i-1/2} = (s_{i-1} + s_i)/2$, $1 \le i \le n$.

Let θ_i represent an approximation of $\theta(s_i)$, $0 \le i \le n$, and ν_i, η_i be given by (10) replacing $\theta(s_i)$ with θ_i. We have now by using twice the mid-point rule for approximating derivatives[2], that equation (11) can be approximated by

$$\begin{aligned} F_i \equiv [(EI)(s_{i+1/2})\theta_{i+1} - ((EI)(s_{i+1/2}) + (EI)(s_{i-1/2}))\theta_i + (EI)(s_{i-1/2})\theta_{i-1}] \\ + \lambda h^2[\nu_i \sin(\theta_i - \alpha) + \eta_i \cos(\theta_i - \alpha)] = 0, \end{aligned} \tag{14}$$

where $1 \le i \le n-1$. Using an end-point formula for approximating derivatives, we get that the boundary conditions (12) can be approximated with

$$\theta_0 = 0, \quad F_n \equiv 3\theta_n - 4\theta_{n-1} + \theta_{n-2} - 2h\gamma = 0. \tag{15}$$

The equations (14) and (15) now form a system of equations whose solutions represent the values of $\theta_1, \theta_2, \ldots, \theta_n$. These equations can be solved using Newton's method.[2] Already calculated the values of $\theta_1, \theta_2, \ldots, \theta_n$, we can obtain the deformed curve of centroids from (13) after approximating the corresponding integrals using for instance the trapezoidal rule.[2]

The linear system to be solved on each iteration of Newton's method when applied to the system (14), (15), can be solved very efficiently if one takes into account the sparsity of the corresponding matrix. If $\mathbf{F}(\theta) = (F_1, F_2, \ldots, F_n)^t$, where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^t$ represents the system (14), (15), then performing the corresponding differentiations one gets:

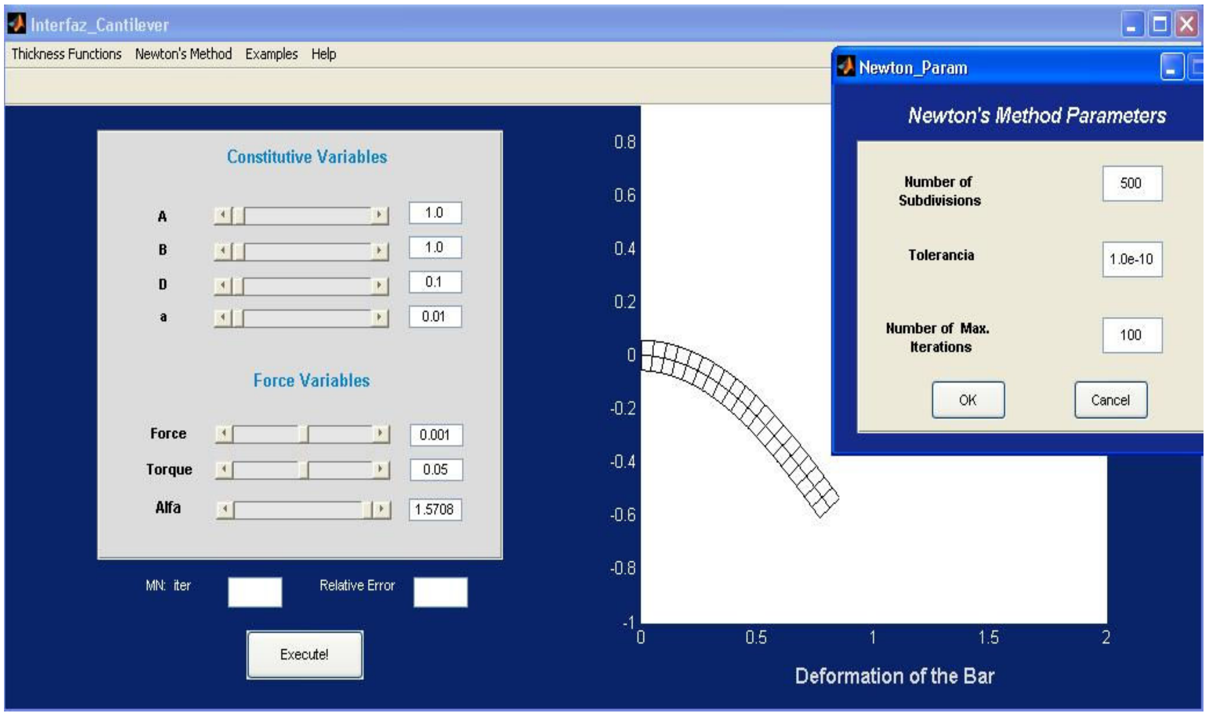
$$\mathbf{F}'(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^t & \beta \end{bmatrix},$$

where $\mathbf{A}$ is an $(n-1)\times(n-1)$ tridiagonal matrix, $\mathbf{b}, \mathbf{c} \in \Re^{n-1}$, and $\beta \in \Re$. One can show now that the corresponding linear system $\mathbf{F}'(\boldsymbol{\theta})\mathbf{w} = \mathbf{h}$ can be solved by solving two tridiagonal $(n-1)\times(n-1)$ systems, plus two additional inner products, that is in $O(n)$ operations.

4. The Graphical User Interface

Using the tools provided by the package MATLAB™, we developed a graphical user interface to experiment with the cantilever problem. The numerical scheme discussed in Section 3 essentially comprises the computational module which is controlled by the GUI. In Figure 2 we show a snapshot of the cantilever GUI. The interface has several sliders and editable text boxes for the user to enter the different mechanical parameters and applied force and torque. There are several menus in the interface to set the bar cross section thickness function, numerical parameters for Newton's method, and some examples for the user to begin familiarizing with the GUI.

After specifying all of these parameters, by pressing the button labeled "Run" (in gray), the GUI executes the computational module with all the specified parameters. After execution the GUI presents a graph of the resulting deformation. The two boxes on the GUI called "MN Iter" and "Relative Error" show the maximum number of iterations performed by Newton's Method and the approximate relative error on the computed solution. These two numbers can be used to assess the convergence or failure of it of the Newton iteration.



A 1.0
B 1.0
D 0.1
a 0.01
Force 0.001
Torque 0.05
Alfa 1.5708
MN: iter
Relative Error

Number of Subdivisions 500
Tolerancia 1.0e-10
Number of Max. Iterations 100

Figure 2: Snapshot of the graphical user interface for the cantilever problem.

5. Numerical Results

In this section we discuss some simulations corresponding to deformations of the nonlinear cantilever, using the GUI and method of Sections 3 and 4, respectively. The function $(EI)(s)$ in (8) was taken to correspond to a bar with circular cross sections and constant mass density. We used four different thickness functions: (1) constant, (2) linear decreasing (a bar thicker at the left end and thinner on the right end), (3) linear increasing (a bar thinner at the left end and thicker on the right end), and (4) quadratic (a bar thicker on both ends and thinner in the middle). The thickness functions were chosen in such a way that the individual total masses of the bars are equal.

In Figure 3 we show the corresponding line of centroids for the deformed bars corresponding to the different thickness functions that we described above. The values we used for the parameters in (8) and (9) are given by

$$A = 1, \quad B = 2, \quad a = 2, \quad D = 0.1, \quad \lambda = 0.0001, \quad \gamma = 0.01, \quad \alpha = \pi / 2. \tag{16}$$

One can see that the bar corresponding to the quadratic thickness function suffers the largest deflection: the thinner part in the middle makes it easier to bend this bar than the others. However, most of the deflection is concentrated on the right end of the bar. The linear decreasing suffers the least deflection. (This might explain why fishing rods are thicker on the handle and thinner on the other end.) Note that both the quadratic and linear decreasing thickness functions have very similar deflection close to the left end. The other two cases: constant and linear increasing are somewhat intermediate with the constant thickness suffering the least deflection. In Figure 4 we show the corresponding shear functions in each case. In all cases the largest shear is close to the left end. However, the linear decreasing thickness function has the least shearing close to the right side while the quadratic has the lowest shear close to the center of the bar where it is thinner.

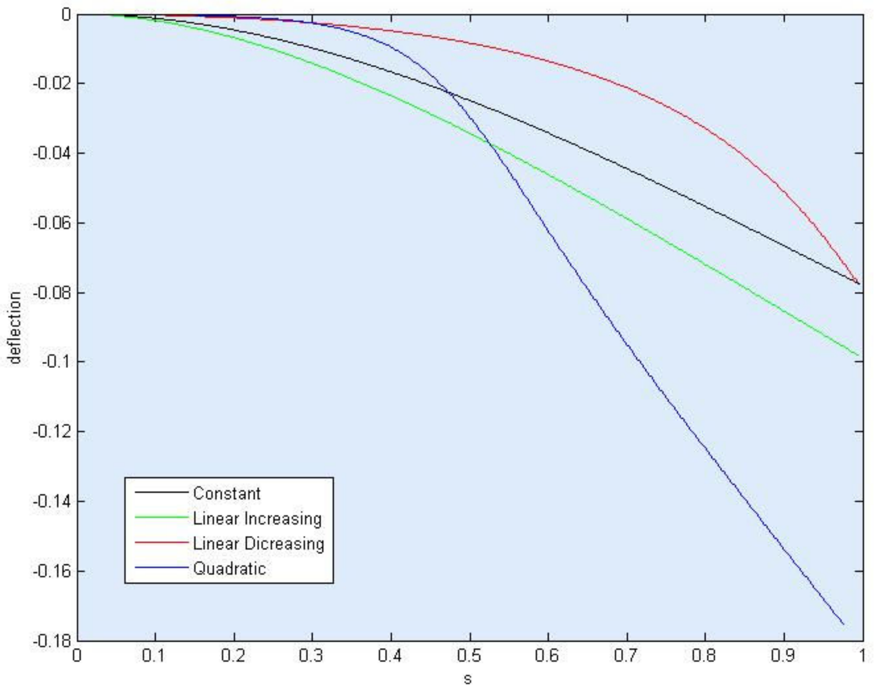


Figure 3: Curves of centroids for different thickness functions corresponding to the data (16).

In our next simulation, we fixed the thickness function to the linear decreasing and changed the shear parameter D in (8). The parameter values are like in (16) except for $\lambda = 0.001$, $\gamma = 0$. We see (Figure 5) that the smaller the value of D, the more shearable is the bar, with most of the shear towards the left end of the bar.

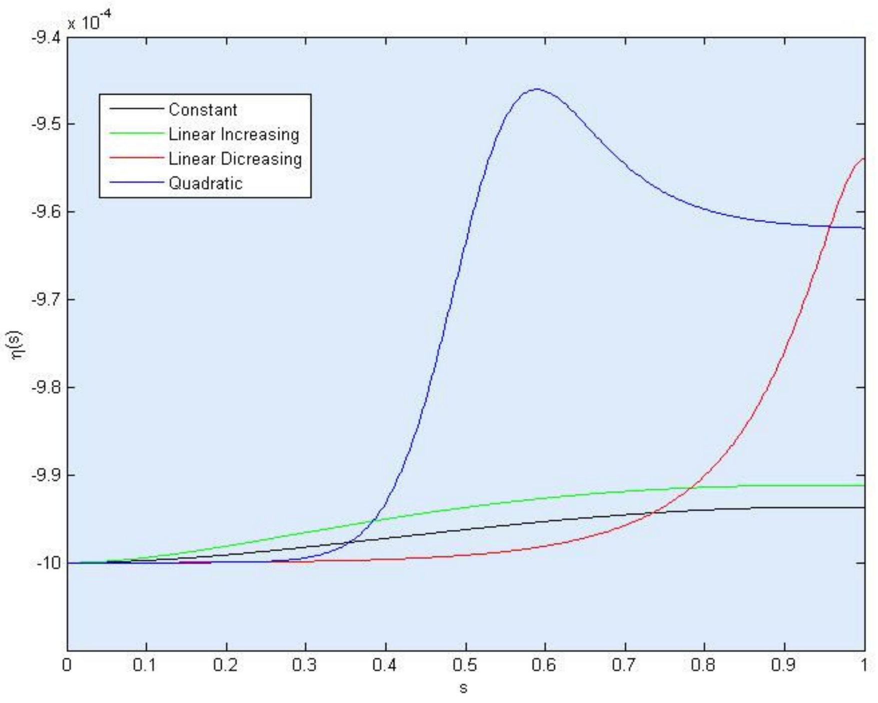


Figure 4: Shear functions for different thickness functions corresponding to the data (16).

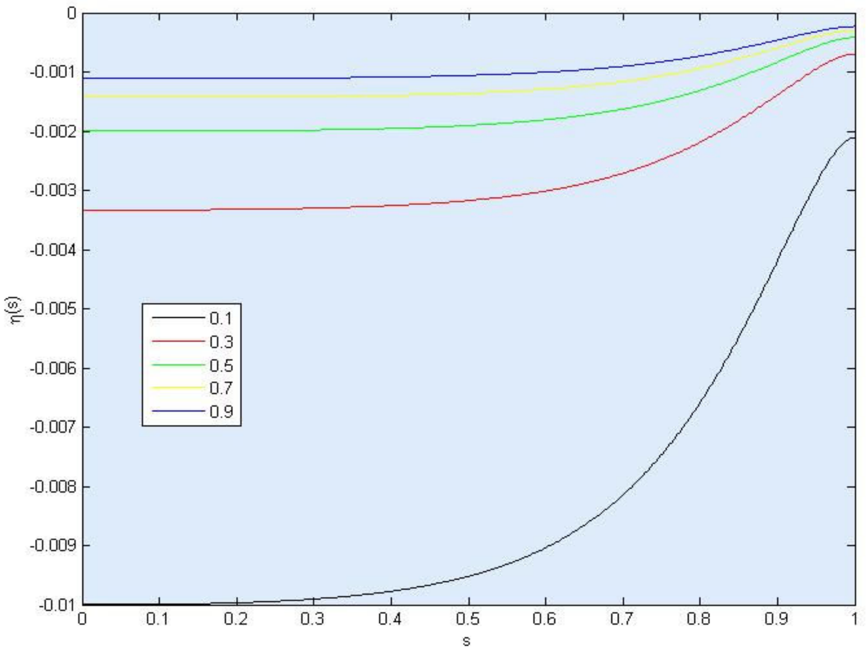


Figure 5: The function η(s) corresponding to different values of the shear parameter D.

6. Conclusions

The proposed numerical scheme and model for the nonlinear cantilever can be used to study the interactions of the various parameters describing the different constitutive and mechanical aspects of this problem. Even for the simple constitutive functions (8), the interrelation among the strains $\nu(s), \eta(s), \mu(s)$ is nonlinear. In a future work we will consider more general constitutive functions depending each of them on all the strains and to correlate the values of the different parameters in the model to actual laboratory data.

Nonlinear models of cantilevers are very common in the literature. Our model differs from many of these other problems either by the type of constitutive equations used or by the corresponding boundary conditions. For example, if we set $\nu = 1, \eta = 0$ in (1), $\mathbf{f}(s) = -q\mathbf{i}, \mathbf{l}(\mathbf{s}) = \mathbf{0}$ in (2) where q is a constant, $(EI)(s)$ equal to a constant in (8), and the boundary condition $\mathbf{n}(1) = \mathbf{0}$, then our equations reduce to the one considered in (6). The emphasis in this paper is on finding analytical asymptotic approximations to the solutions of the corresponding equations. On the other hand in (7) and (8) the deformations of the cantilever are modeled by describing the displacement of the free end of the bar using a forced mass-spring system with friction. The corresponding problems are time dependent (ours is static). The source of the nonlinearities comes from the form of the external or applied force, which is given by an electrostatic force with variable voltage, or by nonlinear (cubic) spring responses[8]. In (7) the dynamics of a single cantilever is studied, with results on the stability of solutions and Hopf bifurcations. In (8) the authors studied a lattice of cantilevers with up to six neighbor interactions and nonlinear spring responses.

7. Acknowledgments

This research has been funded in part by the National Security Agency, Grant Num. H98230-07-1-0114 and the NIH-RISE Program at the University of Puerto Rico at Humacao.

8. References

1. Stuart S. Antman, Nonlinear Problems of Elasticity, *Applied Mathematical Sciences 108*, xviii, Springer-Verlag, New York, 1995.
2. Kendall E. Atkinson, *An Introduction to Numerical Analysis*, 2nd edition, John Wiley & Sons, Inc, 1989.
3. Philip A. Yuya, Yongkui Wen, Joseph A. Turner, Yuris A. Dzenis, and Zheng Li, "Determination of Young's modulus of individual electrospun nanofibers by microcantilever vibration method", *Appl. Phys. Lett.* 90, 111909 (2007), DOI:10.1063/1.2713128.
4. H. Cui,', S. V. Kalinin,, X. Yang, and, D. H. Lowndes, "Growth of Carbon Nanofibers on Tipless Cantilevers for High Resolution Topography and Magnetic Force Imaging", Nano Letters **2004** 4 (11), 2157-2161.
5. Introduction to Elasticity-Wikiversity, http://en.wikiversity.org/wiki/Introduction_to_Elasticity.
6. P. N. Andriotaki, I. H. Stampouloglou, and E. E. Theotokoglou, "Nonlinear asymptotic analysis in elastica of straight bars – analytical parametric solutions, *Arch. Appl. Mech.*, (2006) 76: 525-536, DOI 10.1007/s00419-006-0054-4.
7. S. Liu, A. Davidson, and Q. Lin, Simulation studies on nonlinear dynamics and chaos in a MEMS cantilever control system, Journal of Micromechanics and Microengineering, 14 (2004) 1064-1073.
8. M. Sato, B. E. Hubbard, A. J. Sievers, B. Ilic, and H. G. Craighead, Europhysics Letters, 66 (3), pp. 318-323 (2004), DOI 10.1209/epl/i2003-10224-x.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2009
University of Wisconsin La-Crosse
La-Crosse, Wisconsin
April 16 - 18, 2009

A GUI For The Analysis of Electrostatic Interactions in Molecular Dynamics Simulations

Desirée E. Velázquez Ríos, John E. Morales García, Axel Y. Rivera Rodríguez
Department of Mathematics
University of Puerto Rico at Humacao
Humacao, Puerto Rico, 00792

Faculty Adviser: José O. Sotero Esteva

Abstract

We present a plug-in application with a graphical user interface (GUI) for the Visual Molecular Dynamics (VMD) software for the analysis of electrostatic potentials. It consists of a window for control and visualization of the distribution of charges. It also modifies the representation of the molecules showed on the main VMD window to show the effect of the interaction on selected molecules. We developed an algorithm that approximates computationally intensive calculations of the electrostatic forces with less pseudo-atoms using a N x M x L matrix of boxes, reducing the amount of computation factor of N x M x L. The GUI shows a false-color representation of the pseudo-atoms. A menu is used to configure the results shown by the VMD window. A set of buttons allow the user to change the view points of the canvas and to control the computations. Even with the enhanced algorithms the user may have to wait seconds or minutes depending on the size of boxes being used. Different threads are created in order to show the progression of the computations while they are being performed. Communication between threads is made possible by using shared memory.
Keyword: Molecular dynamics

1. Introduction

Molecular Dynamics (**MD**) simulations often give important insight on the properties of molecules and their interactions. They are based on computing prescribed forces between particles. In the case of classical MD, those particles represent atoms and the forces describe the fundamental forces related to bonds, Van Der Waals and electrostatic forces. Classical MD simulations are well-suited to provide insights into the fundamental properties of CNT-DNA hybrids because they enable calculation of structural properties with atomic resolution. For example, Carbon nanotubes (**CNT**) and single stranded DNA (**ss-DNA**), interesting and important systems in nanoscience, have been used to construct nanoscale chemical sensors. A detailed understanding of electrical properties of these systems is relevant for the design of such sensors.

MD simulations are limited by the available computational power. State of the art simulations deal with systems composed of hundreds of thousands of atoms. They use time steps in the order of femptoseconds (10^{-12} seconds). Using massively parallel systems and sophisticated algorithms with execution times of several days one may achieve simulations with total simulated time of the order of several nanoseconds (10^{-9} seconds). Then, the large data sets of simulated data, called trajectories, are analyzed with software that is bound by similar limitations.

Visual Molecular Dynamics (**VMD**) is a computer program to visualize and model molecules[7]. This tool was developed for viewing and analyzing the results of molecular dynamics simulations, but it also includes applications for visualizing volumetric data, sequence data, and arbitrary graphics objects. When used for viewing and analyzing MD trajectories, although it may be used in-line as the simulation progresses, most often it is used off-line after the whole trajectory has been produced. Users can implement Tool Command Language (**Tcl**) and Python[1] scripts within VMD to add functionality for the analysis of MD trajectories because it includes embedded Tcl and Python interpreters.

This paper presents a plug-in application with a graphical user interface (**GUI**) for VMD for the analysis of electrostatic potentials. It is meant to provide a first glance of the distribution of charges and electrostatic interactions to the researchers that may be recomputed with more detail and precision off-line. False-color representations of the distribution of charges, projection of the atoms onto 2D planes, and 3D representations of the effect of electrostatic interactions on selected molecules are made available almost instantly. Visual clues provide feedback to the researcher about computation progresses and scales.

The GUI implements an algorithm that approximates computationally intensive calculations of the electrostatic forces by dividing the sample space into N x M x L boxes. Pseudo-atoms that represent averages over each of the boxes are then used to reduce the amount of computation. Multi-threading and the use of C++[2] for the most CPU intensive parts of the code also help to achieve the response times expected from an interactive application.

2. Background

The VMD program is compatible with main file formats produced by MD simulators. This relieves the VMD plug-in programmer of the direct interpretation and manipulation of the data. Access to the positions, types and charges of the atoms is made through the Python modules provided by VMD. Tkinter[3,4], a GUI package for Python, is used for building the window that let the researcher control points of view, atom types being viewed, and computation of interactions.

2.1. Python and CPU intensive code

Python is an interpreted programming language. It is designed to be minimalist in the sense of syntactic complexity. As a consequence the code written in that language is relatively easy to understand and modify even by non-experts. At the same time, it supports programming paradigms such as object oriented programming and structured programming. Functionality pertinent to the construction of computational tools for MD simulations such as graphical user interfaces, threading, interprocess communications and interfacing with compiled languages is provided by a large collection of modules. On the other hand, being an interpreted language with a characteristic of placing syntax clarity over efficiency, it presents further limitations to computationally intensive applications. A study made about performance between different programming languages reports that it took 192 seconds per iterations to solve the Flavius Josephus problem in Python using a code consisting of 41 lines[6].

2.2. VMD modules

VMD provides three modules for accessing and manipulating VMD state with objects that represent important entities. They are referred in the VMD User's Manual as *proxy classes* that "*are written in pure Python and use the lower level built-in interfaces to communicate with VMD*". They provide the classes:

Molecule: a proxy for molecules loaded into VMD;
MoleculeRep: to keep track of the representations in a molecule;
AtomSel: whose instances are independent of the molecules and representations in VMD.

Other non-object oriented modules are provided for interacting with VMD including:

color: used to change the color definitions, color maps, or edit the color scale;
display: controls the VMD camera as well as screen updates;
graphics: used to create custom 3-D objects from graphics primitives such as triangle, line, sphere, text, material, etc.

2.3. electrostatic model

When electrostatic charges are present, the Coulomb potentials between two atoms a_i, a_j is given by

$$v^c(r_{i,j}) = \frac{Q_i Q_j}{4\pi\varepsilon_0 r_{i,j}} \qquad (1)$$

where Q_i and Q_j are the charges of the atoms, $r_{i,j}$ their distance, and ε_0 is a constant. For each atom, the potential between it and all other atoms is computed and added to obtain the net potential on that atom:

$$\vec{v}_i^c = \sum_i v^c(r_{i,j})\vec{r}_{i,j} \quad (2)$$

where $r_{i,j}$ is the unit vector pointing from a_i to a_j.

2.4. test case: CNT-DNA hybrids

CNTs are cylindrical sheets of carbon with diameters of ~1nm and lengths up to a few centimeters. CNTs have electronic and structural properties that vary depending on the diameter, chirality and length. They have many interesting properties such as high mechanical strength and electronic stability. These features make them candidates for practical applications.

Single strand deoxyribonucleic acid (**ss-DNA**) is a variant of the widely known biomolecule that consists only one chain of alternating sugars and phosphates. They are often represented by sequences of the letters C, A, T, and G that correspond to the different base units. It is understood that ss-DNA attaches to the CNT by the $\pi - \pi$ stacking interaction. MD simulations of ss-DNA adsorbing to a CNT used in this project have been done both at the University of Pennsylvania and at the University of Puerto Rico at Humacao[8,9].

3. Methods

3.1. computation of charges and electrostatic potentials

This software serves to visualize two different properties related to charged particles: the distribution of charges throughout the space, and the Coulomb potential at each of the atoms as described in section 2.3. Sometimes it is desirable in a simulation to view the distribution of the charges and the interactions between them. The problem is that many algorithms used for this task are time consuming. A new approach is needed to speed up these calculations. The technique used here helps to improve these calculations and show an average of the distribution of charges and the electrostatic interaction among atoms. This approach takes less time to calculate an average of the interaction among charges. The algorithm first takes a frame in the simulation. It then divides the frame in N x M x L boxes. The mount of subdivisions is entered by the user.

3.1.1. charge's grid

This matrix (grid3D) represents a subdivision of the space in N x M x L cubes (Figure 1). Each matrix entry stores the the sum of the charges of the atoms in the corresponding region. Later, in the computation of electrostatic interactions between all atom pairs, a simulated atom at the center of the box with this sum of charges as its charge will be used by substituting the computations corresponding to all the atoms in that box.

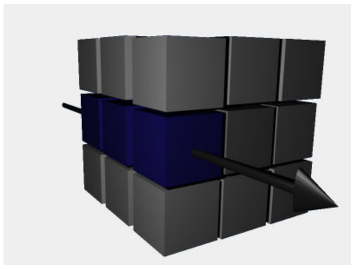


Figure 1.The space occupied by the atoms is divided by 3 x 3 x 3 boxes. The appropriate row or column of boxes is added to compute false color in the 2D projections.

3.1.2. distribution of charges

Isosurfaces are often used to represent the different charge levels throughout a 3D space of continuous models. In our instance we have discrete particles. Therefore a 2D discrete representation was chosen for this purpose. Atoms are projected to a plane. The user may choose between three planes: the X-Y, X-Z, and Y-Z planes. Computing the projections onto these planes is simple: take the corresponding coordinates from the three coordinates. For example, in the X-Y projection,

$$(a, b, c) \longrightarrow (a, b)\,. \tag{3}$$

The similar technique is applied to the grid3D described in section 3.1.1 to produce a 2D projection (grid2D). For instance, in an XY projection, all the magnitudes in the corresponding column are added, that is,

$$grid2D_{i,j} = \sum_{k=j}^{L} grid3D_{i,j,k}. \tag{4}$$

3.1.3. electrostatic potentials

Electrostatic potentials are computed as in section 2.3, but for each atom, instead of computing the sum over all other atoms, it is done over the simulated atoms at the center of the grid boxes. Even with the reduction in computation accomplished by the technique used here, obtaining good approximations requires a grid with enough elements. Assuming a grid with M x N x L cells, computing charges of all atoms takes $O(N * M^3)$. The performance of Python results in a quest limitation.

As explained before, Python is an interpreted language, which makes it slow for big calculations. In order to improve the time spent on calculations, a merge between programming languages was implemented. This merge consists of the C++ and Python programming languages. Since C++ compiles into computer language directly, the running time is faster than Python. To make this merge possible, the Simplified Wrapper and Interface Generator (**SWIG**) library was used. SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG is used with different types of languages including common scripting languages such as Perl, PHP, Python, Tcl and Ruby[5]. Using SWIG libraries helped to make the main calculations using C++ and the GUI in Python. For compiling the C++ files, GNU C++ version 4.3.2 compiler was used.

SWIG libraries' array passing capabilities are limited. These libraries do not accept templates and reference points, which makes it difficult passing the molecule as a parameter for the functions. To solve this just the necessary atom information became the parameters, instead of the whole molecule. This helped the calculations because only the essential atoms and information where used, which results in faster calculations.

Finally, an updated edition of Python is needed for the use of newer and better libraries. The Python plug-in in VMD 1.8.6 uses Python 2.2 libraries. These libraries are not as sophisticated as more recent versions of Python. The source code of VMD was edited, making the principal libraries of the Python plug-in be the Python 2.5 installed in the computer instead of being an extra library that must be added to VMD. This improves the VMD, making use of the most recent Python functions.

3.2. graphical user interface

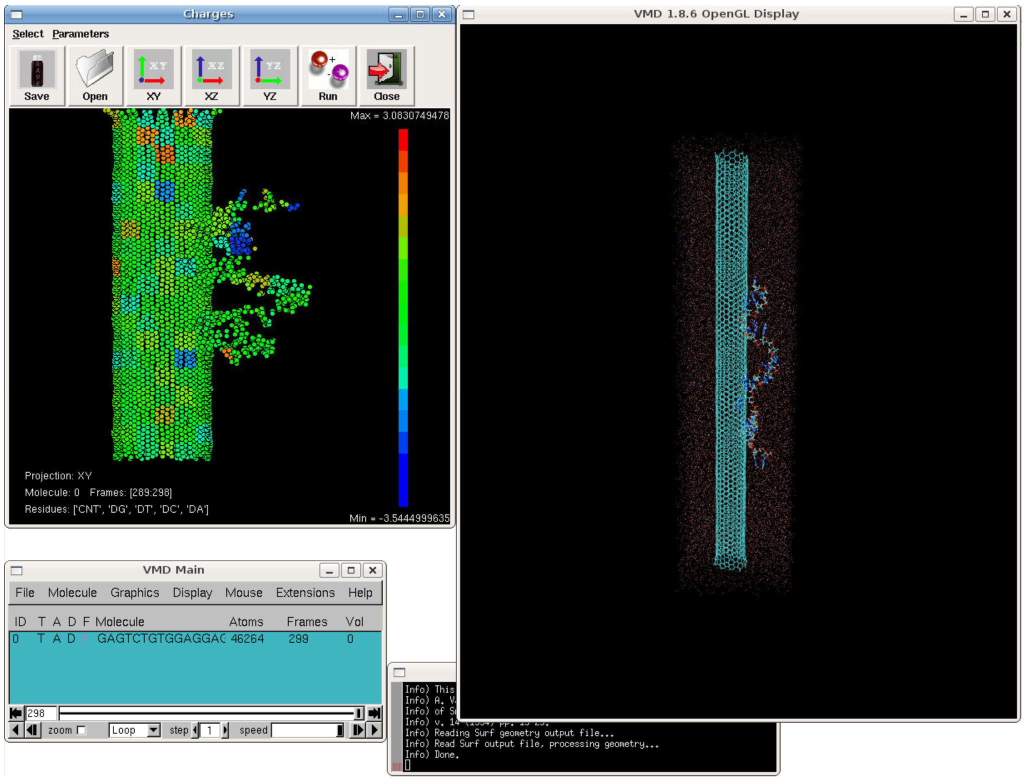
In order to analyze and view the representation of the charges within the system shown in the VMD screen we developed a GUI (Figure 2) that can be called with the VMD. The GUI is divided into two parts that work alongside each other and are written in the same source code; the menu and the canvas. The menu handles the interaction with the user in order to allow them to view the representation of the charges in the system. The canvas handles the calls by the menu and dynamically changes in order to show the representation that the user desires.

3.2.1. menu and buttons

The menu and buttons bar were designed to be as simple as possible while allowing the user to fully understand how each function on screen is intended to work. The buttons bar consists of various buttons where most are visually represented by what each does. A selector menu allows the user to choose between the individual components (residues) of the molecule in order to calculate the whole molecule or each component. This also decides what the canvas will show for the user. The implementation for this selector was made by catching residual names of the molecule loaded into VMD and displaying the names of the components on the menu and sending the selection as a parameter to one of the canvas classes to display only the molecules with the sent name. The parameters menu allows the user to change various parameters during the simulation, such as the subdivisions of the representations of the charges in the canvas, the quantity of frames being analyzed, and the minimum and maximum of the color representation of charges in the canvas and the VMD OpenGL display.

The *save* button stores the state of the simulation. When opened through the GUI it returns the user to the moment of the simulation when it was last saved. Following the *open* button are three buttons called the *XY view* button, the *XZ view* button, and the *YZ view* button, respectively. Each button changes the view of the canvas between each of the main projections for the molecule selected as it is represented in the VMD. This does not change in any way the view of the VMD, it is only relative to its view. Changes to the GUI screen occur after the program has calculated the charges for that view. These calculations are done using the threaded code making the changes that occur within the canvas dynamically.

The following button in the menu is the *run* button. When pressed, the program begins to make the calculations necessary in order to change the representation of the molecule in the main VMD OpenGL display to show the magnitude and direction of the electrostatic potential on each of the atoms that belong to the residues selected by the researcher. The last button, *close*, invokes a small callback function that asks the user whether the application should be closed.



Max = 3.0830749478

Min = -3.5444999635

Projection: XY
Molecule: 0 Frames: [289:298]
Residues: ['CNT', 'DG', 'DT', 'DC', 'DA']

```
ID  T A D F Molecule          Atoms   Frames  Vol
0   T A D F GAGTCTGTGGAGGAC  46264   299     0
```

298 · zoom · Loop · step 1 · speed

```
Info) This
Info) A. V.
Info) of S
Info) v. 14
Info) Reading Surf geometry output file...
Info) Read Surf output file, processing geometry...
Info) Done.
```

Figure 2.Screen shot of all the components of the plug-in. The window in the upper left corner is used to control the computations, projections and residue selection. Progress bars provide feedback about the computations. After completion, electrostatic potentials are shown in the main VMD window (upper right).

3.3.3. representation of distribution of charges

Atoms are particles without colors or any determined shape. In order to represent the distribution of charges, a spherical shape has been assigned to represent an atom (Figure 3). To represent the charge we assigned a false color palette. Each charge will have a color depending on the value of the charge. The colors go from red (the maximum value) to blue (the minimum value) and the other resulting colors are the spectrum between those colors.

3.3.1. pixel representation

With the creation of a grid and the algorithm to transform it into a 2D grid, the transformation of the coordinates into pixel is simple. The representation of the coordinates into pixel is allowed by the following algorithm:

$$\frac{\alpha - \alpha_{max}}{\alpha_{max} - \alpha_{min}} \times D \tag{5}$$

where *α* is the desired coordinate and *D* is the height, if a pixel of the Y coordinate is desired, or the width, if the pixel from the X coordinate is desired. This algorithm allows the conversion of 2D projections by doing the calculation already described by the formula.

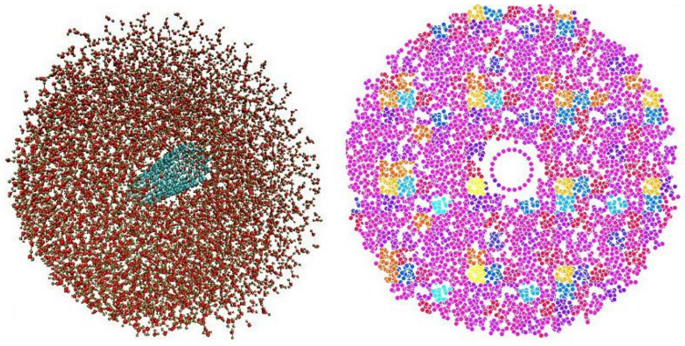


Figure 3. A 3D representation of a carbon nanotube surrounded by water and its XY projection.

3.3.2. color representation

For a better analysis of the charges in the space study, we add color to the propagation of the charges in the studied space. False colors where used to scale the charges. Red is the color of the more charged atom and blue is the color for the less charged atom in this scale. To help the user analyze the charges, a color bar was added in which the colors are sorted from maximum to minimum value. The maximum value of the charge found in the system is at the top of this bar and the minimum value that can be found is at the bottom. This bar is shown in the study space. In order to assign the colors we created a hexadecimal RGB palette and an algorithm to change from decimal base to hexadecimal string base in order to use the color library of VMD. First the algorithm takes the decimal value of the color and then converts it into a string equal to its hexadecimal value.

4. Results

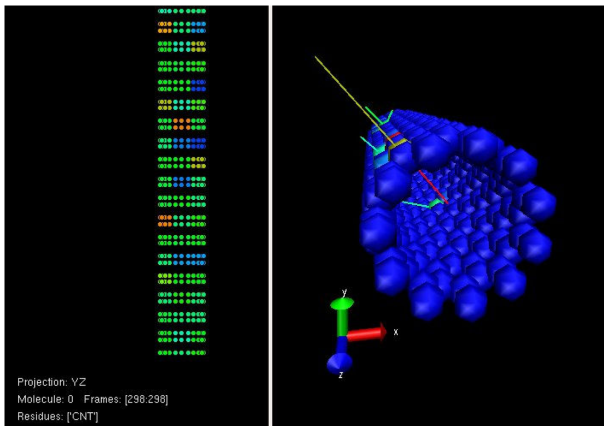

Projection: YZ
Molecule: 0 Frames: [298:298]
Residues: ['CNT']

Figure 4. Charge distribution on the nanotube (left) and electrostatic forces acting on it (right). Lines direct the direction of the force.

4.1. agreement with expected results

In order to analyze our result we first compare our findings of the representation of the charges with previously obtained information. When we represent a small CNT within water, no significant charges can be seen (Figure 4).

With the projections and the color representations of the charges a model or an idea in how the charges are propagated in the polymer and the fibers can be seen. The charges and the trajectories for the projection and the colors were previous calculated data. The result was a success since the projection and the colors met the expectation.

4.2. performance

Over all, the GUI is satisfactory due to the threads used but can still be improved by adding other functions such as an open button.

The edited version of VMD works successfully. The Python plug-in in VMD now uses Python 2.5 version. Also the VMD libraries that can be imported in the original Python plug-in are still usable in this version. A few tests where made using the pickel, sockets, threads and TkInter libraries (which were used for the GUI). The results show that it is stable and the calculations are correct.

The improvement of calculating the charges using C++ was successful. The script was tested using a computer with an Intel 2.66 GHz QuadCore processor and 3 GBs. The tests where done using the original VMD, the edited VMD, with the code using pure Python and the embedded code (Figure 5). Also, the tests were made using two subjects: a short CNT surrounded by water and a longer CNT-DNA hybrid in a water box. The results show an improvement in time for calculating the charges.

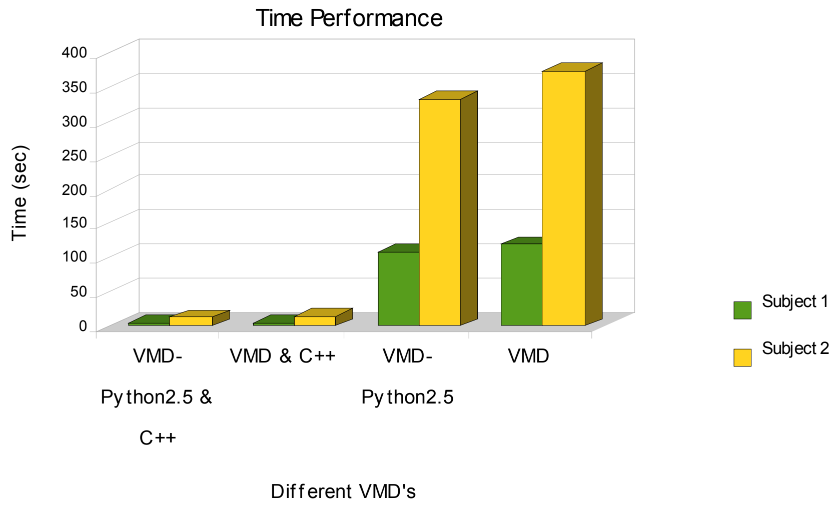


Figure 5. Comparison in time between different versions of VMD and script.

5. Discussion

For future work, the GUI will implement a better selector tool. In the representation area a feature will be added to rotate the 3D view of the VMD display when selecting a 2D representation. Also, the color bar of the electrostatic potential representation will be added to the VMD display.

6. Acknowledgements

The authors wish to express their appreciation to the *Penn-UPR Partnership for Research and Education in Materials* project (NSF-DMR-353730) and the *Humacao Undergraduate Research in Mathematics to Promote Academic Achievement* program (NSA-H98230-04-C-0486). This project was proposed by Robert Johnson of the Department of Physics and Astronomy, University of Pennsylvania in a recent visit to the UPR-Humacao. He also gave advice on the physics of the project and provided simulated data for testing the software.

7. References

1.David M. Beazley, *Python Essential Reference* (Indiana: Sams Publishing, 2006).

2. Joel Adams and others, *C++ An Introduction to Computing*, 3rd ed. (Michigan: Calvin College, 2003).

3. Fredrik Lundh, *An Introduction to Tkinter*, Fredrik Lundh, (1999).

4. John W. Shipman, *Tkinter Reference: a GUI for Python* (New Mexico: New Mexico Tech Computer Center, 2008).

4. David Beazley, "SWIG: an easy to use tool for integrating scripting languages with C and C++", (*Proceeding Of the 4th conference on USENIX Tcl/Tk Workshop*), 2(1996):15.

5. Dhananjay Nene, "Performance Comparison", http://blog.dhananjaynene.com/2008/07/performance-comparison-c-java-python-ruby-jython-jruby-groovy/.

6. Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics", *J. Molec. Graphics*, 14(1996):33.

7. Myrna I. Merced Serrano, "Metrics for the Study of DNA-CNT Hybrids and a Prototype of MoSDAS Graphical User Interface", (*PREM Technical Report*), (2007).

8. Myrna I. Merced Serrano, "Graphical User Interface to Run Molecular Dynamics Simulations of CNT-Polymer Hybrids in VMD", (*PREM Technical Report*), (2007).

Proceeding of the National Conference
On Undergraduate Research (NCUR) 2007
Salisbury University
Salisbury, Maryland
April 10-12, 2008

On the Construction of Column States Matrices of Close Simple Paths for Modeling Linear Polymers

Yesenia Cruz Rosado
Department of Mathematics
The University of Puerto Rico at Humacao
Humacao, Puerto Rico

Faculty Advisor: Prof. José Sotero-Esteva

Abstract

Self-avoiding walks defined over regular grids have been popular because they are used to study and model properties of polymers. The grids used are often restricted to rectangular strips, circles and cylinders among other shapes. Diagrams called column states are used to describe changes in the paths between columns of points. A path can be described also by a sequence of those column states. Only some column states can be successors of another column states in those sequences. Then, a digraph may be associated to a given grid by taking the set of column states as its vertex set and directed edges defined according to allowable column states successors. This work restricts its attention to closed paths over rectangular strips and cylinders and pays a closer look at properties of the column states digraph than what is found in literature. The construction based on the identification of sub-graphs corresponding to smaller grids within the larger graph leads to recurrence equations for counting vertexes and edges, which we present in the case of regular grids. With the algorithm and the recurrence equation the possibility is opened for obtaining better estimates of the size and sparsity of the of the adjacency matrix of the digraph, which in turn is closely related to a transfer matrix that is important for the applications mentioned above.
Keywords:

1. Introduction

Self-avoiding walks (SAW) and cycles (SAC), also called self avoiding polygons, defined over regular grids have been used to study and model polymers since early last century. G. Pólya[1] published a seminal work in this area with applications to the study of hydrocarbons. But this is still an active area of research. In 1980 K. Douglas[2] used (SAW) and (SAC) constrained to 2 dimensional surfaces such as strips, cylinders and cubes to compute polymer distributions. Kloczkowski and Jernigan[3] extended this work for 3 dimensional structures in 2002. A.J. Guttmann and A.R. Conway[5] give an extended review of the progress on the solution of specific problems in this area.

The present work restricts its attention to the modeling of linear polymer conformations with SAC as used by Douglas. In the following sections a notation for the column states that represent transitions on the rectangular grids is introduced. The column states are used as the vertex set of a digraph whose adjacency matrix is related to the transfer matrix used by Douglas. The aim is to find a set of rules that can be used to construct the digraphs algorithmically. A recurrence equation based on the algorithm that gives the number of vertices is posed and an explicit solution is presented. For the number of adjacencies, some facts are proved that allow estimating the minimum and maximum values, thus giving estimates of the density of the transfer matrix.

2. Preliminaries

Modeling and studying properties of polymers with the study of (SAW) and (SAC) is not a new tool, a basic idea of the construction of the vertices can be found in K. Douglas[2]. The approach and notation used thought this paper to study (SWA) and (SAC) is explain in the next section.

2.1 self avoiding walks and cycles

A self-avoiding walk (SAW) is a sequence of moves on a lattice, which does not visit the same point more than once. A self-avoiding cycle (SAC) is a closed self-avoiding walk on a lattice. Self-avoiding walks defined over lattices of width *n* and arbitrary length (Figure 1) are studied here.

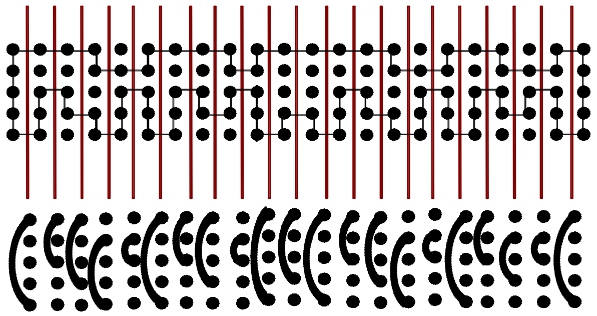


Figure 1. Top: A Self avoiding cycle on a lattice of width n=5. Bottom: The corresponding column states

2.2 column states

Given a rectangular grid one obtains diagrams called **column states** by placing vertical lines between the vertices or pair of points in the lattice (Figure 1). They are used to describe changes in the paths between columns of points. A cycle can be described also by a sequence of those column states. Figure 2 shows all the possible column states for *n*=5.

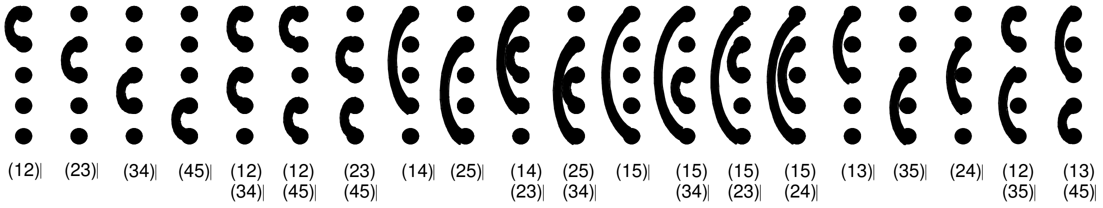


Figure 2. All possible columns states for *n*=5 and their corresponding products of transpositions.

Notation. Column states can be seen to be permutations of the vertices which are transpositions of points such as $(a_1\ b_1)\ldots(a_k\ b_k)$ where $a_i < b_i$ for $i = 1,\ldots,k$ and $a_1 < \cdots < a_k$. Figure 2 shows the corresponding product for all the column states for *n*=5.

Note that only some column states can be successors a given column state. This is because, for some pair of states, any attempt to go from one column state to another will produce collisions in the path. Figure 3 shows that there is an allowable path corresponding to the left pair of states while any attempt to construct such a path with the right pair shows collisions.

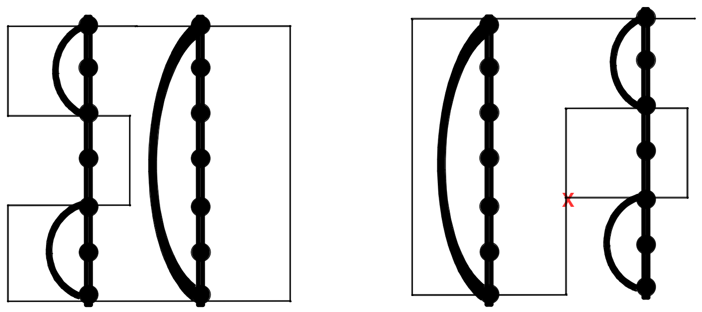


Figure 3. Pairs of column states cannot always be in succession.

2.3 transfer digraph

As stated above, only some column states can be successors of another column state in those sequences. Therefore, there is a natural way of defining a digraph that describes these relations.

Definition. A directed graph or digraph called the **transfer graph** is associated to a given grid by taking the set of all possible column states as its vertex set and directed edges defined according to allowable column states successors that determine adjacencies. The transfer graph of *n* is denoted by $D_n = (V_n, E_n)$, where V_n is the column state set and E_n is the edge set.

Representing the column states as permutations, which are transposition of the vertices, we construct the transfer graph (Figure 4). A single line represents an adjacency between columns states in both directions. An arrow represents an adjacency in one direction. This is from the column state the arrows points to the one in the end. Note that all vertices are adjacent to themselves but for simplicity we do not show those adjacencies in the figures.

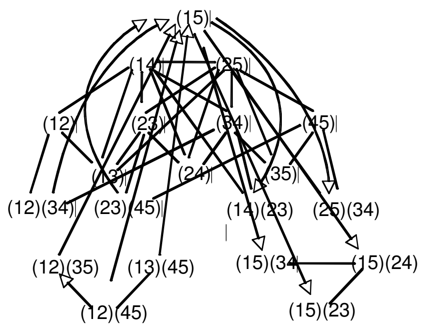


Figure 4. D₅, the transfer graph of *n*=5.

This work restricts its attention to closed paths over rectangular strips and cylinders. It also pays a closer look at properties of the column states of the digraph than what is found in literature. The construction based on the identification of sub-graphs corresponding to smaller grids within the larger graph leads to recurrence equations for counting vertices and edges, which we present in the following section for the case of regular grids. As an example, the transfer graph for D_5 is composed of the transfer graphs of D_3 and D_4 as sub-graphs where we can find copies of them where we only add one.

Notation. Copies of the D_n in which we add a to each element is represented as $D_n^{+a} = (V_n^{+a}, E_n^{+a})$. Where $V_n^{+a} = \{(a_1 + a \quad b_1 + a)\ldots(a_m + a \quad b_m + a) \mid (a_1 \quad b_1)\ldots(a_m \quad b_m) \in V_n\}$ and $V_n^{+a} = \{(v_1^{+i} \quad v_2^{+i}) \mid (v_1 \quad v_2) \in E_n\}$. This can be seen in (Figure 5).

Definition. A **concatenation** of a transposition $(x\ y)$ and a digraph D_k^{+l} is defined and denoted by

$$D_k^{+l} = ((x\ y)V_k^{+l}, (x\ y)E_k^{+l}),$$
$$(x\ y)V_k^{+l} = \{(x\ y)(a_1a_2)\cdots(a_{m-1}a_m) \mid (a_1a_2)\cdots(a_{m-1}a_m) \in V_n\},$$
$$(x\ y)E_k^{+l} = \{(x\ y)(v_1v_2) \mid (v_1v_2) \in E_n\}.$$

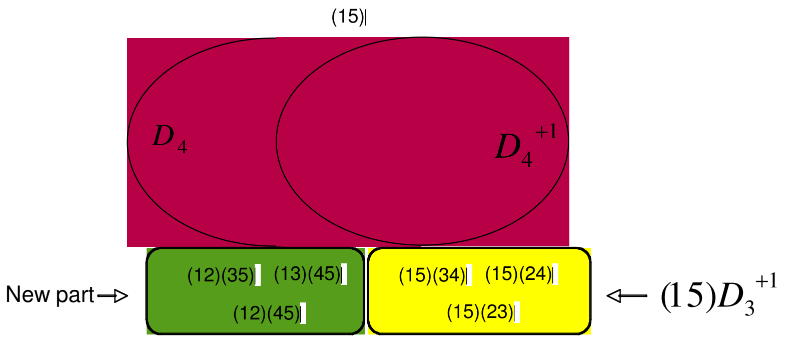


Figure 5. Sub-graphs of transfer graph D$_5$.

2.4 count of size of vertex set

An algorithm that construct the vertex set V_n is based on the identification of subsets that are copies of the sets V_i^{+a}, $i<n$ found within V_n. This algorithm leads to the recurrence equation[6]

$$T_n = 2T_n(n-1) + T(n-2) + 1 + \sum_{i=2}^{n-4} T(i)[T(n-2-i) + T(n-3-i)].$$

The formula for counting column states presented by K. Douglas[2] is a solution of this equation. presents a non-recurrence equation. Note that the number of column states in D_5 is given by the number of vertices of the previous digraphs and other vertices, which are obtained from concatenations. The recurrence equation gives us the number of column states that will be present in the transfer graph D_n.

3. Rules for Determining Adjacencies

We know that we can construct the vertex set of a transfer graph based in the previous vertex set of transfer sub-graphs. Notice that there are certain column state that are before other in the lattice, which tells us that there are certain finite combination of the column states in the lattice, which gives us adjacency. We want to determine the **density of the transfer matrix**, which is the number of nonzero values, where the columns are the column states of D_n. The possibility is opened for obtaining better estimates of the size and sparsity of the of the adjacency matrix of the digraph. We present some results that eventually will allow us to find a lower and upper limit for the density of the transfer matrix.

Theorem: Let $v \in V_n$, $v = \{(a_1\, b_1)...(a_n\, b_n)\}$, $a_i < b_i$ for $i=1,..,n$, and $v \subseteq w$. Then $w \to v$ if and only if for every $(a_i\, b_i) \in v - w$ there is no $(a_i\, b_i) \in v$ with $a_i < a_j < b_j < b_i$.

Proof. ($\Rightarrow$) Let w and v as in the hypothesis. By induction in n, since V_2 has only one element, $(1\ 2)$, the statement is trivially true. Suppose true for $k < n$. Take $v \in D_n$.

Case 1: No transposition in v has 1 or n. Then is true for inductive hipotesis because $v \in D_{n-2}{}^{+1}$.

Case 2: Suppose $v=\{(1\ n)\}$, the theorem holds because $w=v$ and there is a trivial path compatible with the state represented by $w \to v$.

Case 3: $v\ \varepsilon\{(1\ n)\}D_{n-2}{}^{+1}$. Note that w has to have $(1\ n)$. To see this suppose that w does not have $(1\ n)$ that is $(1\, n) \in v - w$. Since $v\text{-}\{(1\ n)\} \neq \phi$, take $(a\, b) \in v - w$. Note that $1<a<b<n$ which is a contradiction, therefore $(1\, n) \in w$.

Now substract $(1n)$ form v and w we can apply case 1 to obtain $(w\text{-}(1\ n)) \to (v\text{-}(1\ n))$. Then it is easy to extend the path compatible with the pair of states $(w\text{-}(1\ n)) \to (v\text{-}(1\ n))$ to one compatible with $w \to v$. That is $w \to v$, see Figure 6.

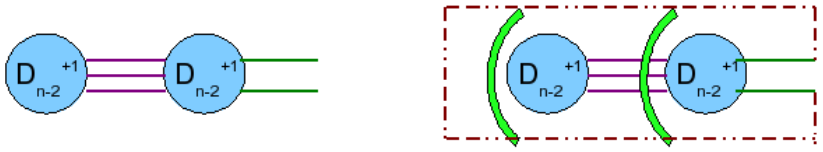


Figure 6. Extension of the path of that show there is a compatible path with w $\to$ v.

Case 4: $(1\, i)$ and $(j\, n) \in v$, $i < j$. Here v is the concatenation of

$v_1 = \{ (a\ b) \in v \mid 1 < a < b < i\}$,and $v_2 = v - v_1$.

w can be divided as well into $w_1 = \{ (a\ b) \in w \mid 1 \leq a < b \leq i\}$, and $w_2 = w - w_1$.

By inductive hipothesis $v_1 \to w_2$ and $v_2 \to w_2$ and the Figure 7 shows that there is a posible path from v to w.

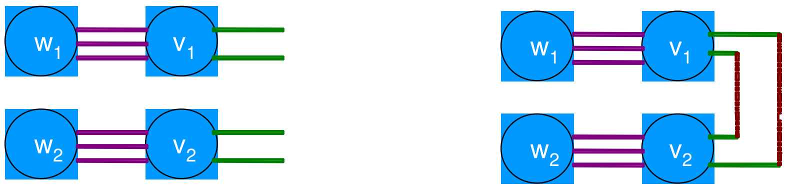


Figure 7. Possible path from v to w.

(®) Suppose w v and v, w are as in the hipothesis. Suppose $\exists(a\, b) \in v$ and $\exists(c\, d) \in w$ with $a < c < d < b$. The Figure 8 shows that one cannot construct a path without collisions.

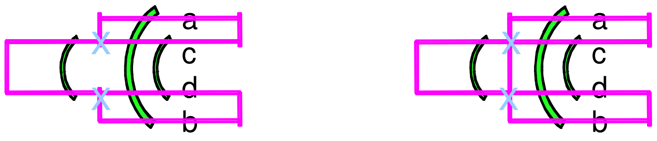


Figure 8. Vertex v is not adjacent to w.

Corollary: Every vertex $v \in D_n$ is adjacent to itself, that is $v \longleftrightarrow v$.

Definition: If $(a_i\, b_i) < (c_i\, d_i)$ this is that $a_i \le c_i < d_i \le b_i$.

Definition: Let v be a vertex. We say that $(a\, b) \in v$ is the shortest transposition in v if $(a\, b) = \min\{n \mid n = \mid a - b \mid\}$.

Theorem: Let $v = \{(a_1\, b_1)\ldots(a_n\, b_n)\}$, $a_i < b_i$ and $w=\{(c_1\, d_1)\ldots(c_n\, d_n)\}$ where $(a_i\, b_i) < (c_i\, d_i)$. Then $w \to v$ and $v \to w$.

Proof: Let v and w be as describe in the theorem. Now construct the path taking the shortest transposition in v and w such that $(a_i\; b_i) < (c_i\; d_i)$, and we draw a possible path we continue this way and when we are done with all of them we unite the paths to construct the final path. This is described in Figure 9.

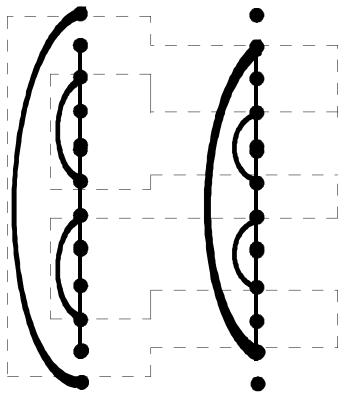


Figure 9. Possible path between column states.

Theorem: Let $v = \{(a_1^1\, a_2^1)\ldots(a_1^k\, a_2^k)\}$, and $w=\{(b_1^k\, b_2^k)\ldots(b_1^k\, b_2^k)\}$, then $w \to v$ and $v \to w$ if and only if a_1^i , $b_1^i > max(\, a_2^{i-1}$, $b_2^{i-1})$ and $b_1^i < a_2^i$ or $a_1^i < b_1^i$ for all $0 < i \le k$ where $0 < k \le [n/2]$.

Proof: ($\Rightarrow$) We will prove this by induction. Let $k=1$, this is that $v = (a_1^1\ a_2^1)$ and $w = (a_1^1\ a_2^1)$. Now since $b_1^1 < a_2^1$ or $a_1^1 < b_1^1$ this is demonstrated in Figure 10.

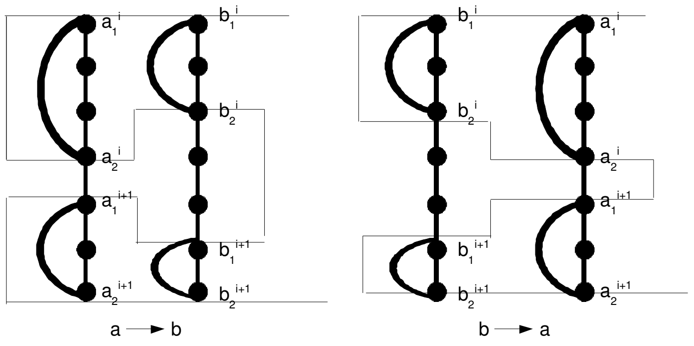


Figure 10. Adjacencies from *v* to *w* and from *w* to *v*.

Suppose this is true for *k*. Now we will prove this for *k+1*. By the construction of the transpositions we have that they can be seen as the union of disjoint transpositions. Therefore we have that there exist a possible path between the first *k* transpositions which we can connect to the last transposition therefore we have that w $\rightarrow$ v and v$\rightarrow$ w.

($\Leftarrow$) Suppose that a_1^i, $b_1^i <= max(a_2^{i-1}, b_2^{i-1})$. By the construction of the vertices we have that *v* and *w* are vertices compose of the disjoint union of transpositions. Therefore without any loss of generalization let v and w be as in Figure 11.

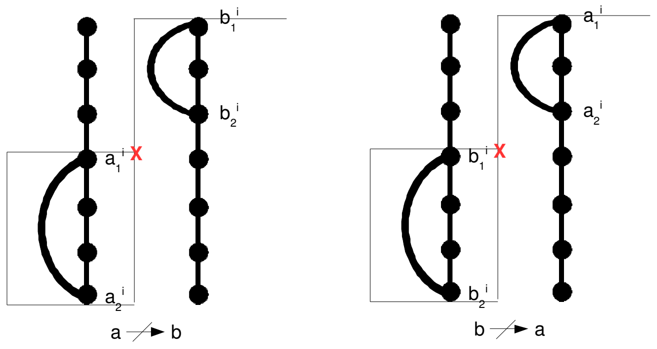


Figure 11. Impossible adjacency from *w* to *v* and from *v* to *w*.

We can see that *w* is not adjacent to *v* and *v* is not adjacent to *w*, which is a contradiction. Therefore a_1^i, $b_1^i > max(a_2^{i-1}, b_2^{i-1})$.

Now suppose that $b_1^i >= a_2^i$ and $a_1^i >= b_1^i$. These cases are represented in Figure 12 and Figure 13.

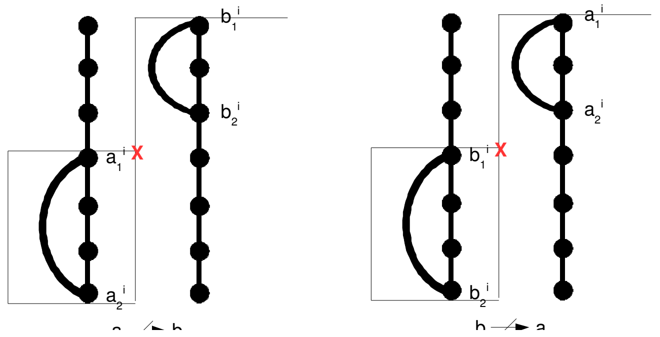


Figure 12. Impossible adjacency from *w* to *v* and from *v* to *w*.

We can see that both adjacency are impossible which is a contradiction. Therefore $b_1^i < a_2^i$ or $a_1^i < b_1^i$ for all $0 < i < k$ where $0 < k < [n/2]$.

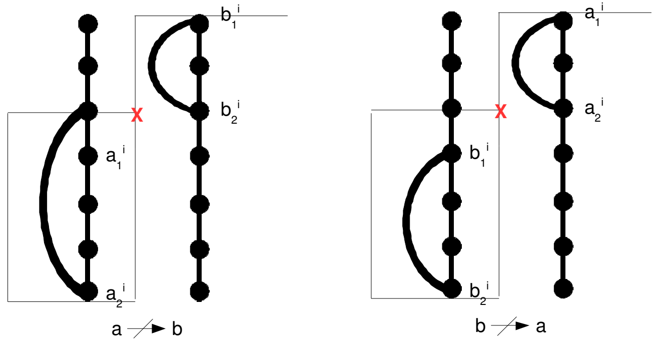


Figure 13. Impossible adjacency from *w* to *v* and from *v* to *w*.

Corollary: Let $v=(1\ n)$ and $w=(i\ j)$ where $1< i,j < n$, then $w \rightarrow v$ and $v \rightarrow w$.

Proof: Since $w=(i\ j)$ and $v=(1\ n)$ we know that for all $0< i,j, \leq n$, $(i\ j) < (1\ n)$, therefore we have that $w \rightarrow v$ and $v \rightarrow w$ by the last theorem.

4. Conclusions and Work in Progress

We establish a recurrence equation based on an algorithm that we found to construct the transfer graph that eventually will help us model some polymer properties. The recurrence equation is the base for the algorithm that we constructed to study and model polymers, because of that we are working on a computer algorithm that will give us the graph which contains the column states and the relations between them that is the edges. Then we are going to study this conformation of polymers in cylinders.

5. Acknowledgements

This work has been supported in part by the National Security Agency, Grant Number **H98230-07-1-0114**; and the National Science Foundation MARC Program at the UPRH and the PREM Program at the UPRH.

6. References

Journals

1. G. Pólya, "Kombitatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen", *Acta Mathematica* 68 (1937), 145-254. Translation to English in G. Pólya, R.C. Read, *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds* (New York: Springer-Verlag, 1987), 1:95.
2. K. Douglas, "Asymptotic Distribution for Self Avoiding Walks Constrained to Strips, Cylinders, and Tubes", *Journal of Statistical Physics.* **3**, (1980), 561-586.
3. A. Kloczkowski, Robert Jernigan, "Efficient Method To Count and Generate Compact Protein Lattice Conformations ", Macromolecules 30 (1997), 6691-6694.
4. R. J. Baxter, Journal of Statistical Physics, Vol 117, Nos. 1/2, 2004
5. A.J. Guttman, A.R. Conway, "Square Lattice Self-Avoiding Walks and Polygons", Annals of Combinatorics 5 (2001), 319-345.
6. J. Sotero-Esteva, Y. Cruz-Rosado, "Construcción de grafos de transferencia de pasos definidos sobre un reticulado", XXIII Seminario Interuniversitario de Investigación en Ciencias Matemáticas (SIDIM 2008), Carolina, Puerto Rico.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2008
Salisbury University
Salisbury, Maryland
April 10–12, 2008

Classification of Textures in Microphotographies of Leaves Epidermis

Joyce M. Fernandez
Department of Mathematics
University of Puerto Rico at Humacao
Humacao, Puerto Rico

Faculty Advisors: Elio Ramos, Denny S. Fernández

Abstract

Microscopic images of leaves, collected from Mona Island dry forest (which is located between Puerto Rico and the Dominican Republic), were analyzed. For each leaf side an image was obtained at two magnifications (200x and 400x). This resulted in four samples of images showing a wide variety of textures and stomata patterns. For each group of images we used the gray-level co-occurrence method to characterize the observed gray level patterns. From the GLCM matrix several texture features were calculated among others: the angular second moment (ASM), the contrast, correlation, inverse difference moment (ISM), and entropy. Visual inspection indicates the formation of three groups of images at 200x magnifications based on the observed patterns. The results of the GLCM analysis indicate consistency between the texture features and the isotropic and anisotropic patterns observed in the leaves
Keywords: microscopic images, texture features, patterns

1. Introduction

The classification and characterization of textures in digital images is of great interest in areas like artificial vision and pattern recognition. The natural world provides a wide variety of examples of textures and patterns that can be observed at different spatial scales. This characterization is very important in many areas of the biological sciences. In taxonomy, for instance, the traditional approach for the discrimination between species is based in the observation of the characteristics. Furthermore, these characteristics may be related to more fundamental issues like the relation between the observed structures and the functions. Through the extensive availability of digital technology and image processing methods, many tasks, like classification, that used to be handled manually, can be performed in an automatic or semi-automatic way using many statistical and artificial intelligence methodologies. This possibility may be attractive but the actual implementation is difficult, mainly due to the correct selection of discriminating features and in many cases the heterogeneous quality of the images.

This paper presents the results of an image analysis for a sample of microphographies of leaves epidermis. The samples show an extensive variety of textures, spatial patterns, cell structures, and stomata configurations. The main objective of this work is to combine several image processing and statistical techniques so that the original group can be divided in sub-samples with similar features. The following section describes the image data sets, and then the data analysis section explains the Gray Level Co-occurrence Matrix (GLCM) method that was used to obtain a texture features matrix. Several multivariate statistical methods were applied to the texture features matrix including principal component and cluster analysis which are described in subsequent sections. Finally, the results and conclusions are presented.

2. Data Set

The data set consisted of four groups of images of leaves epidermis of 1600x1200 pixels at two magnifications (200x and 400x) and sides. The first group (20x_E) consisted of 69 images, the second group (20x_H) consisted of 39 images, the third group (40x_E) consisted of 70 images, and the last group (40x_H) consisted of 60 images. In this paper the results for the 20x_E and 20x_H samples are presented. The leaves were collected from the Mona Island dry forest, which is located between Puerto Rico and the Dominican Republic. The epidermis is the outermost cellular layer that covers the whole plant structure and typically can be observed as a set of closely packed cells without intercellular spaces[4]. Besides the epidermal cells a prominent structure known as the stomata can be observed. The stomata are basically a pore surrounded by two bean shaped cells known as the guard cells (Figure 1). The epidermis has many functions being the most important to allow the sunlight to pass through the chloroplasts which is crucial for the photosynthesis process and to avoid an excessive loss of water from the inner tissues. The stomata allow the gas exchange between the plant and the environment which again is necessary for photosynthesis and respiration. For different plant species a wide variety of patterns of cell epidermis and stomata configurations can be observed. In this sense the observed structures in the images can be used as a discriminator between species or group of species. Traditionally this type of task is performed manually by visual inspection of the images and the corresponding classification. In this paper an automatic procedure is presented that is able to measure some features from the images, followed by a method that allows the construction of groups based on the features.

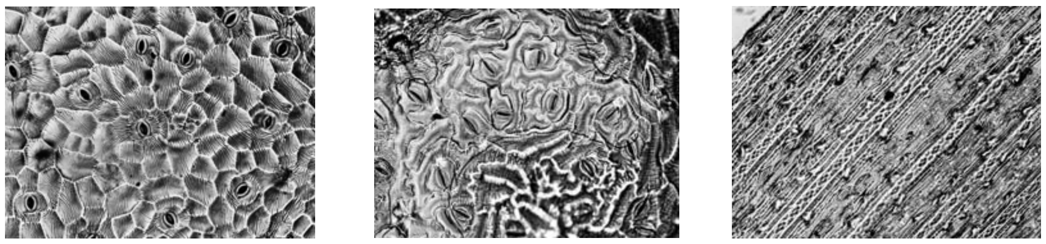

(a) Cissus trifolata (b) Ipomea triloba (c)Tricholaena rosea

Figure 1: Some examples of images prototypes that were found by visual inspection.

3. Data Analysis

Starting with the raw images the data analysis procedure consists of several steps which are described in the following sections.

3.1. pre-processing

Due to the poor contrast in some of the images a normalization procedure (contrast stretching) was carried out for each set of images. In this sense we were able to obtain consistency in the ranges of the pixel values for all the images. Furthermore, the original images were converted to 8 bit gray scale images and the size of the images was reduced in 50% in order to improve the efficiency of the image processing methods. A preliminary visual inspection of the images revealed the formation of three images prototypes which are shown in Figure 1.

3.2 the gray scale co-correspondence matrix

In order to characterize statistically the texture patterns observed in the images the Gray Scale Co-ocurrence Matrix (GLCM) was calculated[6]. The GLCM is a tabulation of how often different combinations of gray levels occurs in a matrix. For an image g we can construct a N x N gray level co-occurrence matrix $M_{d,\theta}$. The elements of $M_{d,\theta}$ represent the probability of the co-occurrence of gray value i,j at points p_1, p_2, separated by distance d and angle θ.

$$M_{d,\theta}(i,j) = Prob\{g(p_1)=i, g(p_2)=j\} \text{ given that } d(p_1,p_2)=\theta \tag{1}$$

where $p_i = (x_i, y_i)$,and i=1,2 and

$$d(p_1,p_2) = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2} \tag{2}$$

$$\angle(p_1,p_2) = \arctan((y_2-y_1)/(x_2-x_1)) \tag{3}$$

Where Equation (1) is the co-ocurrence matrix, Equation (2) is the distance between pixels and Equation (3) is the angle between the selected pixels. From $M_{d,\theta}$ different Haralick texture features can be calculated from the following equations (Table 1). The textural feature were calculated by averaging the co-ocurrence matrix at four angles (0^o, 45^o, 90^o, and 135^o) and at fixed distance of d=1 pixel.

Texture feature	**Formula**	**Texture feature**	**Formula**
Absolute value	$\sum_{i,j=0}^{N-1} \lvert i-j \rvert M_{d,\theta}(i,j)$	Entropy	$\sum_{i,j=0}^{N-1} M_{d,\theta}(i,j)\log_2\left[M_{d,\theta}(i,j)\right]$
Inverse difference	$\sum_{i,j=0}^{N-1} \frac{M_{d,\theta}(i,j)}{1+(i-j)^2}$	Correlation	$\frac{\sum_{i,j=0}^{N-1} ijM_{d,\theta}(i,j) - \mu_x\mu_y}{\sigma_x\sigma_y}$
Homogeneity	$\sum_{i,j=0}^{N-1} \frac{M_{d,\theta}(i,j)}{1+\lvert i-j \rvert^2}$	Energy	$\sum_{i,j=0}^{N-1} M_{d,\theta}(i,j)^2$
Contrast	$\sum_{i,j=0}^{N-1} (i-j)^2 M_{d,\theta}(i,j)$		

Table 1: Texture features utilized in the GLCM analysis of the images with the corresponding formula[2]

The GLCM and the corresponding features from Table 1 were calculated for each of the 4 groups of images. The GLCM features were complemented with four first order statistics namely the average gray intensity, the standard deviations and the corresponding skewness and kurtosis. The GLCM method was implemented as a plugin with the ImageJ[1] public domain Java image processing software. The plugin, `GLCM.java` was developed in Java and based on the texture analysis plugin developed by Julio E. Cabrera from NIH[3]. The correspondence between the image number and the plant species (from visual identification) are given in the next two figures.

1 Achasp1011(20x)E	19 Cistri962(20x)E	37 Jatgos957(20x)E	55 Schfru998(20x)E
2 Agasis981(20x)E	20 Cocdiv1014(20x)E	38 Jatmul1003(20x)E	56 Sespor989(20x)E
3 Alover978(20x)E	21 Cocmic993(20x)E	39 Krufer1000(20x)E	57 Sidobo953(20x)E
4 Amyele1027(20x)E	22 Cocuvi949(20x)E	40 Leuleu1024(20x)E	58 Stajam958(20x)E
5 Antacu965(20x)E	23 Comdod954(20x)E	41 Morcit1006(20x)E	59 Stastr1030(20x)E
6 Barasi990(20x)E	24 Comele1029(20x)E	42 Nepmul979(20x)E	60 Stiema963(20x)E
7 Bousuc1008(20x)E	25 Corglo956(20x)E	43 Pasto(1)984(20x)E	61 Surmar946(20x)E
8 Bursim968(20x)E	26 Cyphum1018(20x)E	44 Phyama980(20x)E	62 Swimah1013(20x)E
9 Caebon985(20x)E	27 Eupcor973(20x)E	45 Pilmar1019(20x)E	63 Tercat952(20x)E
10 Caemon1012(20x)E	28 Euppet1001(20x)E	46 Pisalb1026(20x)E	64 Thepop947(20x)E
11 Caklan1031(20x)E	29 Ficcit977(20x)E	47 Pluobt997(20x)E	65 Tilutr1023(20x)E
12 Canros988(20x)E	30 Goshir955(20x)E	48 Porole1016(20x)E	66 Toumic961(20x)E
13 Canwin1015(20x)E	31 Guadis1017(20x)E	49 Porrub974(20x)E	67 Triros982(20x)E
14 Capbif950(20x)E	32 Guaoff1004(20x)E	50 Preagg975(20x)E	68 Uromax1032(20x)E
15 Capfle1002(20x)E	33 Guasan1005(20x)E	51 Psymon994(20x)E	69 Vercin986(20x)E
16 Cenvir969(20x)E	34 Guekru1020(20x)E	52 Ranacu1028(20x)E	
17 Chanic1009(20x)E	35 Hipman1007(20x)E	53 Raunit945(20x)E	
18 Chialb1022(20x)E	36 Ipotri983(20x)E	54 Reynuc960(20x)E	

Table 2: List of the 69 images at 200x magnification by E side and its corresponding numbers. The name of the files correspond to the abbreviated scientific name of the identified species.

1 Achasp1011(20x)H	11 Cocdiv1014(20x)H	21 Nepmul979(20x)H	31 Stajam958(20x)H
2 Agasis981(20x)H	12 Cocmic993(20x)H	22 Pasto(1)984(20x)H	32 Stastr1030(20x)H
3 Alover978(20x)H	13 Cocuvi949(20x)H	23 Phyama980(20x)H	33 Styham971(20x)H
4 Boeere987(20x)H	14 Comele1029(20x)H	24 Pilmar1019(20x)H	34 Surmar946(20x)H
5 Caklan1031(20x)H	15 Cyphum1018(20x)H	25 Pisalb1026(20x)H	35 Thepop947(20x)H
6 Canros988(20x)H	16 Eupcor973(20x)H	26 Porrub974(20x)H	36 Tilutr1023(20x)H
7 Capbif950(20x)H	17 Goshir955(20x)H	27 Raunit945(20x)H	37Tricis951(20x)H
8 Cenvir969(20x)H	18 Ipotri983(20x)H	28 Schfru998(20x)H	38 Triros982(20x)H
9 Chanic1009(20x)H	19 Jatgos957(20x)H	29 Sespor989(20x)H	39 Uromax1032(20x)H
10 Cistri962(20x)H	20 Leuleu1024(20x)H	30 Sidobo953(20x)H	

Table 3: List of the 39 images at 200x magnification by H side and its corresponding numbers. The name of the files correspond to the abbreviated scientific name of the identified species.

3.3 Principal Component Analysis

The Principal Component Analysis (PCA) is one of the main tools of exploratory multivariate data analysis. A very common situation in multivariate data analysis, like in the features table obtained from the GLCM analysis, is the availability of several variables (features) for a single observation. In this sense each observation is a point in a multidimensional space. However, in general multidimensional data is very difficult to visualize and consequently hard to identify patterns.

The PCA is a method that reduces data dimensionality by performing a covariance analysis between factors. This method involves a mathematical technique to solve for the eigenvalues and eigenvectors of a square symmetric matrix with sums of squares and cross products. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component. The sum of the eigenvalues equals the trace of the square matrix and the maximum number of eigenvectors equals the number of rows or columns of this matrix.

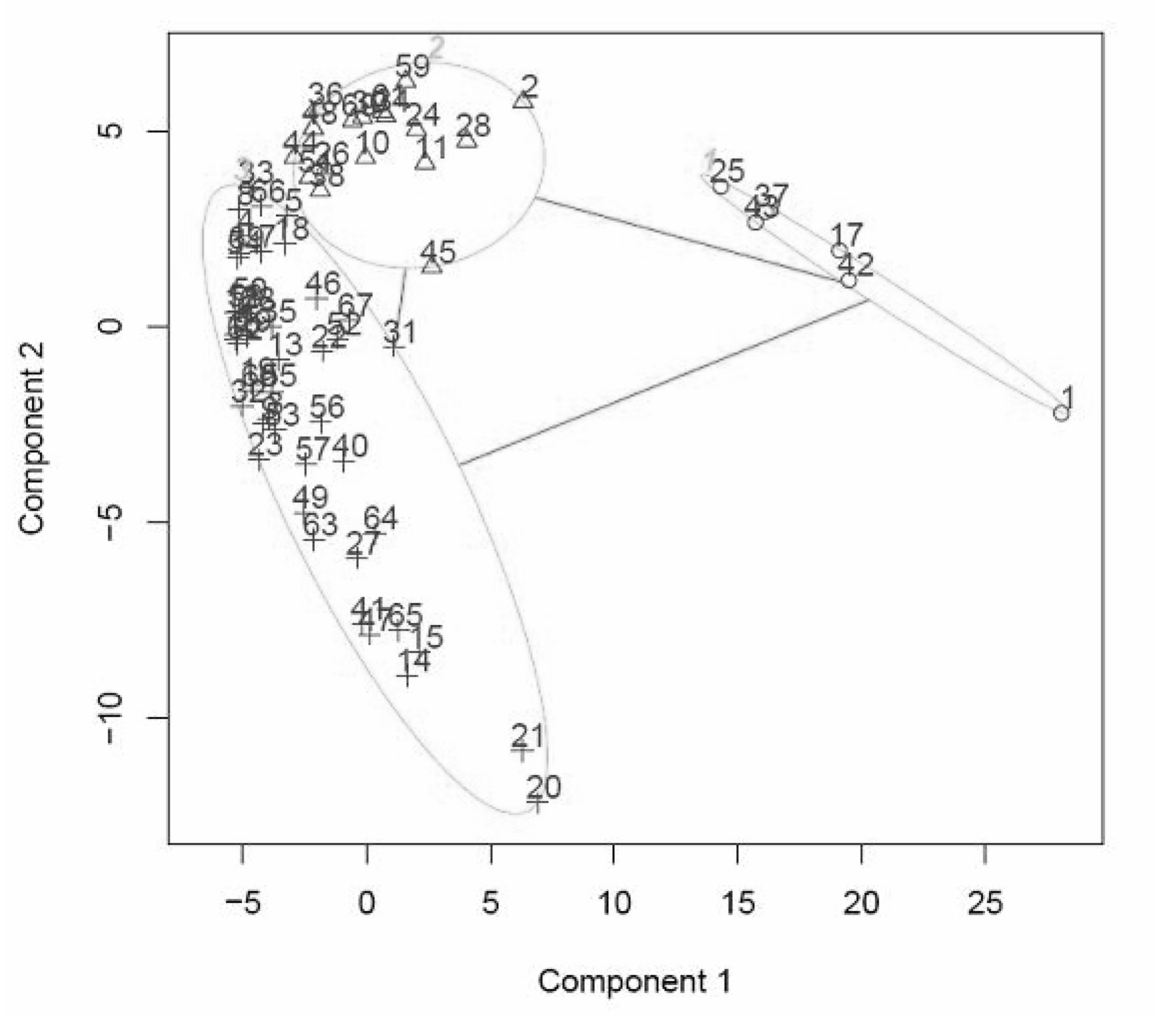


Figure 1: Principal Component Analysis (PCA) Plot at 200x and E side. The two components describe 89% of the variability observed in the features matrix. The groups were obtained from the CLARA optimal partitioning method.

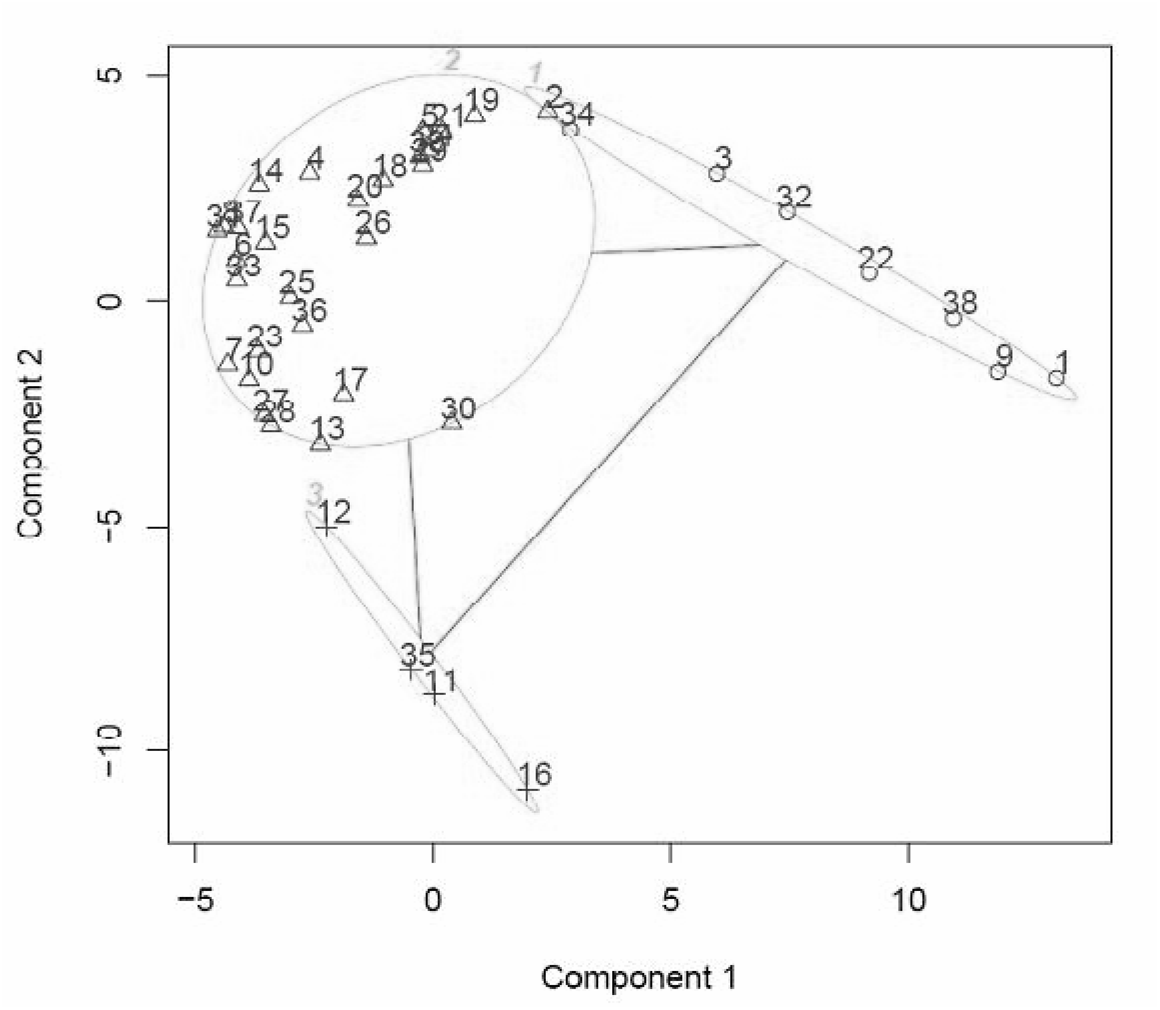


Figure 2: Principal Component Analysis (PCA) Plot at 200x and H side. The two components describe 91% of the variability observed in the features matrix. The groups were obtained from the CLARA optimal partitioning method.

The PCA method was applied to the features table for each of the image samples. A two dimensional plot of the first two components was obtained (see Figure 4 and Figure 5) and subsequently analyzed. Each point in the PCA plot was associated with an image in the sample. The relation between the numbers and the images was obtained from Figure 2 and Figure 3. Once the PCA plots were obtained two optimal partitioning methods[9] (PAM and CLARA) were applied in order to identify groups in the data sets.

4. Results and Conclusions

An image and multivariate data analysis was performed on several samples of images of leaves epidermis in search of patterns that allow an automatic or semi-automatic grouping or characterization of the images. In the first phase of the analysis eleven textural features were estimated for each image using the Gray Scale Co-occurrence Matrix (GLCM) method. A Principal Component Analysis (PCA) was performed on the features data and using an optimal partitioning method three groups of images were identified. Examination of the images in each group revealed consistency between the visual features of the images and the groups obtained from the textural features, PCA and partitioning methods. Is interesting to note that an independent analysis[7] of the images using visual morphological features from the epidermis (type of epidermis, type of stomata, structure of guard cells, etc.) reveals the formation of four groups. A future work may reveal the connection (if any) between the textural classification and this morphological classification.

5. Acknowledgments

This research has been sponsored by the National Security Agency (NSA) Grant NUM. H98230-07-1-0114. We would like to thank Dr. Elvia Meléndez-Ackerman from CREST-Center for Applied Tropical Ecology and Conservation (CATEC) for providing the images utilized in the analysis.

6. References

1. Abramoff, M.D., Magelhaes, P.J., Ram, S.J. "Image Processing with ImageJ", 2004.
2. Bharath Kumar, S.V. et al, Proc. of International Conference on Signal Processing and Communication, 2004 Biophotonics International, volume 11, issue 7, pp. 36-42, 2004.
3. Cabrera J., Texture Analyzer, http//rbs.info.nih.gov/ij/plugins/texture.html 2005
4. Carpenter, K.J., *Am. J. Botany*, 92, 2005
5. Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. Graphical Methods for Data Analysis, Belmont, CA: Wadsworth. 1983.
6. Haralick, R.M., Proceedings of the IEEE, 67, 1979.
7. Melendez-Ackerman, E. Private Communication, 2008.
8. Smith, L.I., A tutorial on Principal Component Analysis, http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf 2002
9. Venaples, W.N., Ripley, B.D., Modern Applied Statistics with S, Springer-Verlag, New York, 2002.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2008
Salisbury University
Salisbury, Maryland
April 10 – 12, 2008

The Numerical Computation of the Critical Boundary Displacement for Radial Cavitation for Composite Materials

Jessica Flores
Department of Mathematics
University of Puerto Rico at Humacao
Humacao, Puerto Rico 00791

Faculty Advisor: Dr. Pablo V. Negrón

Abstract

The phenomenon of void formation in bodies under tension has been observed in laboratory experiments. Ball (1982) showed, in the context of nonlinear elasticity, that void formation or "cavitation" can decrease the (potential) energy of a body in tension when the tension is sufficiently large. An important related problem is that of characterizing or computing the critical boundary displacement at which cavitation occurs. As cavitation can point to the initiation of fracture or rupture in a body, the computation of such critical boundary displacement is important from the point of view of design. In Negrón-Marrero and Sivaloganathan[4] a numerical scheme for computing the critical boundary displacement for cavitation is proposed that applies to a very general class of compressible homogeneous materials. In this paper we study the generalization or extension of this method to composite (non-homogeneous) materials.
Keywords: Cavitation, Critical Boundary Displacement, Numerical Scheme.

1 Introduction

Void formation, also known as "cavitation" can point to the initial fracture of a body in tension (Figure 1). Over the years, this phenomenon has been studied and it was shown by Ball (1982) that void formation decreases the potential energy of the body. This happens when the boundary displacement is sufficiently large. The particular boundary displacement at which cavitation appears is called the *critical boundary displacement for cavitation* and is denoted by λ_{crit}. The computation of such critical boundary displacement is important from the point of view of design. Most of the attempts of computing λ_{crit} have been based on finding exact solutions of the equations describing such deformations. For a nice review on these and other related results on cavitation we refer to Horgan and Polignone[3]. In Negrón-Marrero and Sivaloganathan[4] a numerical scheme for computing the critical boundary displacement for cavitation is proposed that applies to a very general class of compressible homogeneous materials. This method is based on the solution of a sequence of problems with punctured domains. That is, a small hole is put in the center of the body, and the problem is solved for such a domain. Then we proceed to make the hole smaller and repeat the process. It is known, Sivaloganathan[5], that this process converges to a solution of the corresponding problem for the solid body. In the method of Negrón-Marrero and Sivaloganathan[4], which is called the *inverse method*, in addition to the punctured domain, the inner cavity size of the deformed body is specified as well, and a sequence of problems with both the hole in the reference configuration and that of the deformed configuration approaching zero, is solved. In Negrón-Marrero and Sivaloganathan[4] it is shown that this process converges to λ_{crit}. With the specification of the inner cavity, one then is confronted with solving a sequence of initial value problems (c.f. (10), (11)) instead of a sequence of nonlinear boundary value problems (c.f. (6), (7), (8), (9)).

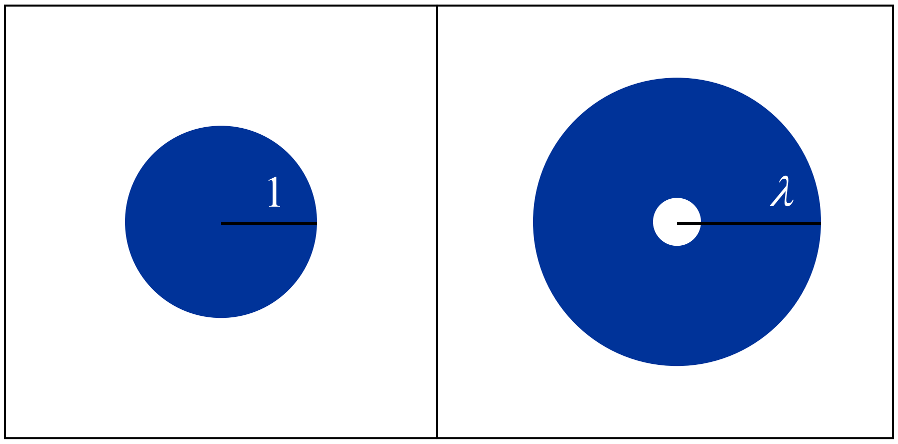


Figure 1. Void formation or cavitation on a spherically symmetric body.

The inverse method proposed by Negrón-Marrero and Sivaloganathan[4] and its convergence properties are for homogeneous materials. In this paper we consider the generalization of this method to bodies composed of two different homogeneous materials. We consider a sphere or ball with a center core and an outer core of different materials each (Figure 2). We study how the critical boundary displacement depends on the properties of the two material and the relative sizes of the cores. For the purpose of this work we consider cores composed of materials that can be described by the stored energy function (1):

$$\Phi(v_1, v_2, v_3) = v_1^{-2} + v_2^{-2} + v_3^{-2} + c v_1 v_2 v_3 . \tag{1}$$

Each core will be described by a function of this type. For this particular example we study how λ_{crit} varies as the material parameter *c* changes for each core, and how it depends on the relative sizes of the cores.

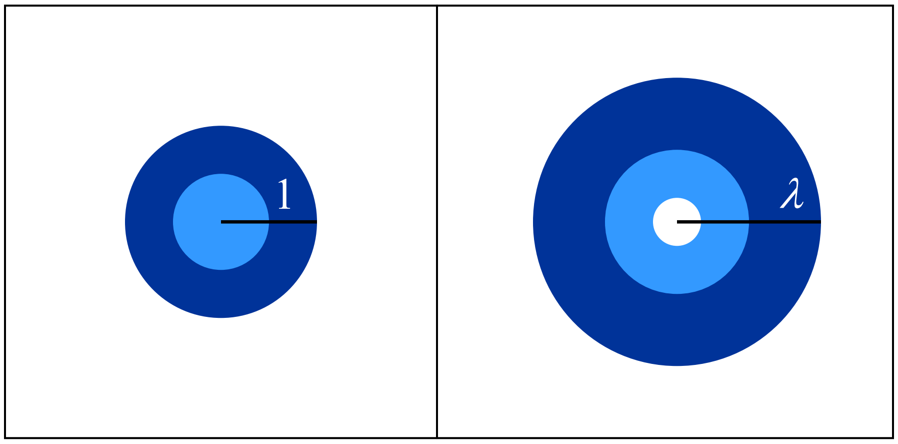


Figure 2. Non-homogeneous material (left). Non-homogeneous material with cavitation (right).

2 Formulation of the Problem

We considered a unit sphere B as the reference configuration of the body. A deformation of B is a function $u : B \to \Re^3$. The derivative $\nabla u(x)$ is called the *deformation gradient.* The requirement that the deformation *u* *preserves orientation* is equivalent to:

$$\det \nabla u(x) > 0, \quad x \in B. \tag{2}$$

If $W : B \times M_+ \to \Re$, where $M_+ = \{F \in \Re^{3\times3} : \det F > 0\}$, represents the stored energy function for the material of the body, then the total stored energy associated with the deformation *u* is given by:

$$E(u) = \int_B W(x, \nabla u(x))\, dx. \tag{3}$$

We look for deformations *u* that minimize this total stored energy functional among an appropriate class of functions and satisfying the boundary condition:

$$u(x) = \lambda x, \quad x \in \partial B.$$

In this paper we look for solutions of this problem that are *radially symmetric*, that is, solutions *u* of the form:

$$u(x) = r(R)\frac{x}{R}, \quad x \in B,$$

where $R = \|x\|$, and $r : [0,1] \to \Re$. It follows that the condition (2) is satisfied provided:

$$r'(R), \frac{r(R)}{R} > 0, \quad R \in [0,1].$$

The stored energy functions *W* that we consider will be of the following form:

$$W(x, F) = \begin{cases} \Phi_1(v_1, v_2, v_3), & 0 < R < a, \\ \Phi_2(v_1, v_2, v_3), & a < R < 1, \end{cases} \tag{4}$$

where v_1, v_2, v_3, are the proper values of $(F'F)^{1/2}$, called the principal stretches, and $a > 0$ is the radial size of the inner core. The total stored energy in the body due to deformation $r(R)$ is now given by:

$$I(r)=\int_0^a R^2\Phi_1\left(r'(R),\frac{r(R)}{R},\frac{r(R)}{R}\right)dR+\int_a^1 R^2\Phi_2\left(r'(R),\frac{r(R)}{R},\frac{r(R)}{R}\right)dR. \tag{5}$$

In the following we use the notation $\Phi_{i,j}$ for the partial derivative of Φ_i with respect to the *j*-th variable, and

$$\Phi_{i,j}(r(R))=\Phi_{i,j}\left(r'(R),\frac{r(R)}{R},\frac{r(R)}{R}\right),\quad \text{etc..}$$

By considering smooth variations v such that $v(1)=0$, one can show that the Euler-Lagrange equations for the functional (4) are given by:

$$\frac{d}{dR}\left[R^2\Phi_{1,1}(r(R))\right]=2R\Phi_{1,2}(r(R)),\quad 0<R<a, \tag{6}$$

$$\frac{d}{dR}\left[R^2\Phi_{2,1}(r(R))\right]=2R\Phi_{2,2}(r(R)),\quad a<R<1, \tag{7}$$

with boundary conditions:

$$r(0)\geq 0,\quad r(1)=\lambda,\quad \Phi_{1,1}\left(r'(a^-),\frac{r(a)}{a},\frac{r(a)}{a}\right)=\Phi_{2,1}\left(r'(a^+),\frac{r(a)}{a},\frac{r(a)}{a}\right). \tag{8}$$

Moreover, if $r(0)>0$, then we must have that[1]:

$$\lim_{R\to 0^+}\left(\frac{R}{r(R)}\right)^2\Phi_{1,1}(r(R))=0. \tag{9}$$

Note that in general $r(R)$ will be a continuous function, differentiable everywhere except at $R=a$. However the radial stress is in general continuous for all R. In fact, the last condition in (8) is just a statement that this radial stress is continuous across $R=a$.

3 Numerical Scheme

In this section we describe the numerical scheme that was used to approximate the critical boundary displacement for cavitation. As mentioned in the introduction, the method is based on the solution of a sequence of problems with punctured domains. The punctured domains are given by:

$$B_\varepsilon = \{x : \varepsilon < \|x\| < 1\},$$

where $\varepsilon > 0$. We denote by $r_\varepsilon(R)$ the solution of (6), (7), (8), and (9) in this new domain. In this case the condition (9) reduces to:

$$\Phi_{1,1}\left(r'_\varepsilon(\varepsilon), \frac{r_\varepsilon(\varepsilon)}{\varepsilon}, \frac{r_\varepsilon(\varepsilon)}{\varepsilon} \right) = 0.$$

Under some physically reasonable assumptions on the stored energy functions Φ_1, Φ_2, we can get that the equations:

$$\Phi_{i,1}(\nu, \tau, \tau) = P, \quad i = 1,2,$$

are equivalent to

$$\nu = \phi_i(\tau, P), \quad i = 1,2,$$

where $\phi_i : (0,\infty) \times \Re \to (0,\infty), \quad i = 1,2$, are smooth functions.

If instead of λ in (8), we prescribe $r_\varepsilon(\varepsilon) = c$, then the problem of finding $r_\varepsilon(R)$ can be stated now as the following initial value problems:

$$\frac{d}{dR}\left[R^2 \Phi_{1,1}(r_\varepsilon(R))\right] = 2R\Phi_{1,2}(r_\varepsilon(R)), \quad \varepsilon < R < a, \tag{10}$$

$$r_\varepsilon(\varepsilon) = c, \quad r'_\varepsilon(\varepsilon) = \phi_1\left(\frac{c}{\varepsilon}, 0\right),$$

$$\frac{d}{dR}\left[R^2 \Phi_{2,1}(r_\varepsilon(R))\right] = 2R\Phi_{2,2}(r_\varepsilon(R)), \quad a < R < 1, \tag{11}$$

$$r_\varepsilon(a^+) = r_\varepsilon(a^-), \quad r'_\varepsilon(a^+) = \phi_2\left(\frac{r_\varepsilon(a)}{a}, P^-\right), \quad P^- = \Phi_{1,1}\left(r'_\varepsilon(a^-), \frac{r_\varepsilon(a)}{a}, \frac{r_\varepsilon(a)}{a} \right).$$

The idea now is to solve these problems for a sequence of ε, c converging to zero. For the case of the single core it is shown in Negrón-Marrero and Sivaloganathan[4] that the sequence of boundary displacements generated according to $\lambda = r_\varepsilon(1)$, converges to the critical boundary displacement for cavitation λ_{crit}. This procedure can be described by the following pseudo-algorithm:

3.1 Procedure

Let $\{(\varepsilon_k, c_k)\}$ be a sequence converging to (0,0).

1. For k = 0, 1, 2,…,

   a) Compute an approximate solution $v_{k,1}$ of the equation: $\Phi_{1,1}\left(v_{k,1}, \frac{c_k}{\varepsilon_k}, \frac{c_k}{\varepsilon_k}\right) = 0$.

   b) Compute an approximate solution $r_{k,1}(R)$ of the IVP given by the equation (3) on (ε_k, a) subject to

   $r(\varepsilon_k) = c_k$, $r'(\varepsilon_k) = v_{k,1}$.

   c) Compute an approximate solution $v_{k,2}$ of the equation:

   $$\Phi_{1,1}\left(r'_{k,1}(a), \frac{r_{k,1}(a)}{a}, \frac{r_{k,1}(a)}{a}\right) = \Phi_{2,1}\left(v_{k,2}, \frac{r_{k,2}(a)}{a}, \frac{r_{k,2}(a)}{a}\right).$$

   d) Compute an approximate solution $r_{k,2}(R)$ of the IVP given by the equation (5) on $(a,1)$ subject to

   $r(a) = r_{k,1}(a)$, $r'(a) = v_{k,2}$.

   e) Set $\lambda_k = r_{k,2}(1)$.

2. Repeat steps (a) to (e) until $\{\lambda_k\}$ satisfies a certain stopping criteria.

4 Numerical Results

Procedure 3.1 was implemented in MATLAB. This computational environment provides for very efficient routines for solving initial value problems. For the numerical simulations we used the following stored energy functions for the cores:

$$\Phi_i(v_1, v_2, v_3) = v_1^{-2} + v_2^{-2} + v_3^{-2} + c_i v_1 v_2 v_3, \quad i = 1,2,$$

where $i = 1,2$ denote the inner an outer cores respectively.

We ran several simulations in which we used Procedure 3.1 to compute λ_{crit} for different values of a, c_1, c_2. In the first case $a = 0.2, c_1 = 1.5$ and we vary c_2. As c_2 increases from 1.5 to 5, the outer core becomes "harder". We can see (Figure 3) that it becomes "easier" (smaller λ_{crit}) to open a hole in the center as expected due to the "harder" outer core.

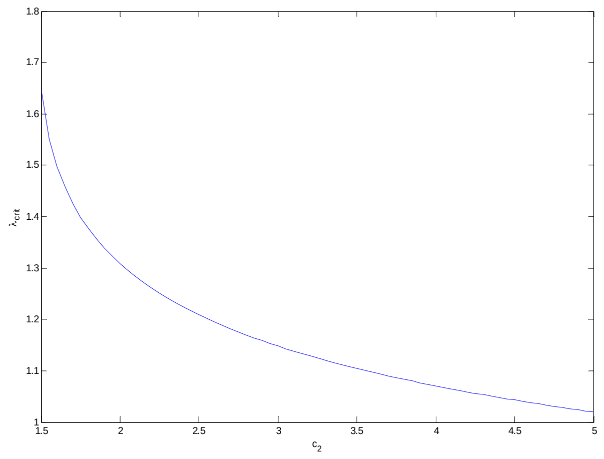


Figure 3. λ_{crit} as a function of c_2 for $a = 0.2, c_1 = 1.5$.

In the next simulation we have $c_1 = 2, c_2 = 5$. Thus the outer core is harder than the inner one. We vary the inner core radius. We can (Figure 4) that as the inner core radius increases, it becomes "harder" (larger λ_{crit}) to open a hole at the center. This is the effect of the increasing inner core. As a gets close to 1, one can see that λ_{crit} approaches the value of 1.3087 which is that corresponding to a single core of $c_1 = 2$.

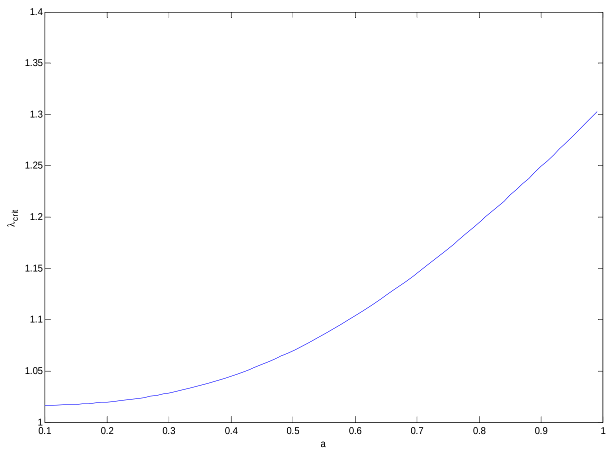


Figure 4. λ_{crit} as a function of a for $c_1 = 2, c_2 = 5$.

In the last simulation we consider variations both in c_2, a while c_1 is fixed at 2. c_2 varies between 1.5 and 5, while a changes between 0.1 up to 0.5. We show (Figure 5) the corresponding surface of λ_{crit} as a function of c_2, a. It is interesting to observe that for values of c_2 between 1.5 and 2, the value of λ_{crit} is a decreasing function of a. For a small, the softer material given by c_2 occupies most of the body. One has to pull "harder" to open a hole in the “harder” center ($c_1 = 2$) because the outer material yields more easily. As a increases, this effect becomes less marked. The opposite behavior holds for values of c_2 between 2 and 5.

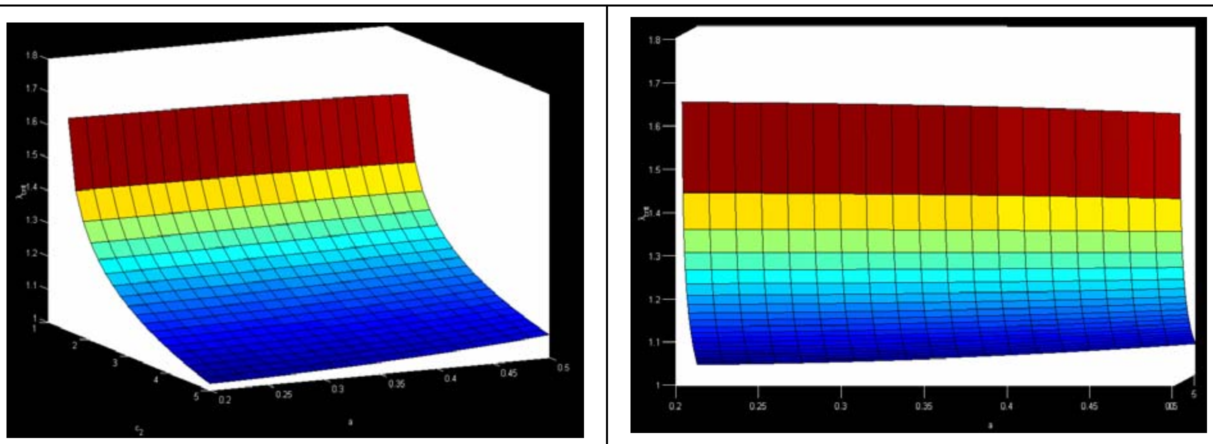


Figure 5. λ_{crit} as a function of both c_2 and a.

5 Conclusions

In the more general case of a composite material, the inverse method proved to be a useful scheme for computing the critical displacement for cavitation. We studied the behavior of λ_{crit} as a function of some of the constitutive parameters of the materials of the cores and as the size of the cores changed. The stored energy function used for the simulations is good only for small deformations because the term corresponding to the determinant, which is $v_1 v_2 v_3$, is finite under extreme compressions. In a future paper we consider more realistic stored energy functions (3) as well as other types of non-homogeneities. Also one needs to study the theoretical convergence properties of the scheme under these more general conditions.

6 Acknowledgements

This work has been funded by the National Security Agency, Grant Num. H98230-07-1-0114.

7 References

1. Ball, J. M., *Discontinuous Equilibrium Solutions and Cavitation in Nonlinear Elasticity*, Phil. Trans. Royal Soc. London, **A 306**, 557-611, 1982.
2. Horgan, C.O., and Abeyaratne, R., *A bifurcation problem for a compressible nonlinearly elastic medium growth of a micro--void*, J. Elasticity, 16, 189-200, 1986.
3. Horgan, C.O. and Poligone, D.A., *Cavitation in Nonlinearly Elastic Solids: A Review*, Appl. Mech. Rev., Vol. 48, No. 8, 1995.
4. Negrón-Marrero, P. V. and Sivaloganathan, J., *The Numerical Computation of the Critical Boundary Displacement for Radial Cavitation,* To appear in the Journal of Mathematics and Mechanics of Solids, 2008.
5. Sivaloganathan, J., Cavitation the incompressible limit, and material inhomogeneity, Quarterly of Applied Mathematics, Vol. XLIX, No. 3, 521-541, 1991.
6. Sivaloganathan, J., *Uniqueness of regular and singular equilibria for spherically symmetric problems of nonlinear elasticity*, Arch. Rat. Mech. Anal., 96, 97--136, 1986.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2008
Salisbury University
Salisbury, Maryland
April 10 - 12, 2008

Study of Stable Conformations of CNT-DNA Hybrids by Means of Principal Component Analysis

Myrna I. Merced Serrano
Department of Mathematics
University of Puerto Rico at Humacao
100 Tejas Avenue
Humacao Puerto Rico 00791

Faculty Adviser: José Sotero Esteva

Abstract

Single walled carbon nanotubes (swCNT) functionalized with single strand DNA (ss-DNA) in a non-covalent interaction have been successfully used as gas sensors. Molecular Dynamics (MD) simulations have been used before to understand the interactions between swCNT and ss-DNA that influence electron transport such as π-π stacking. But studying how gas analytes interact with these hybrids requires detailed observations of the conformation of ss-DNAs. Principal Components Analysis (PCA) is often used to analyze conformations of molecules, mainly proteins, produced by MD simulations. A set of *N* atoms is selected within the molecule. The conformations of the molecule at different moments can be traced by using PCA to map the sequence of points in a 3N-dimensional space into a 2D or 3D space. Clusters of points indicate similar conformations. A PCA analysis was performed on trajectories generated by MD simulations of swCNT-ss-DNA hybrids at room temperature in aqueous solution. Ss-DNAs corresponding to three specific sequences of 21 oglionucleotides used in actual gas sensor experiments as well as four Poly-Cytosine of various lengths were explored. The selection of atoms sets for the PCA analysis was also varied ensuring uniform selections along the DNA. The PCA analysis shows that the paths of the 3D points progress to eventually form clusters indicating that the shapes of the DNAs evolve into stable configurations. No returns in the paths nor jumps between clusters indicate that the DNA reaches a final conformation. These results are consistent with observations made in the laboratory proving that the PCA method is suitable for the study of the conformation of CNT-DNA hybrids.
Keywords: molecule conformations, DNA-CNT hybrids, Principal Components Analysis (PCA)

1. **Introduction**

The combination of single stranded DNA (ss-DNA) with a single walled carbon nanotube (swCNT) is a particularly elegant example of a self assembled nanodevice that has been experimentally proven to be sensitive to a set of gases. The robustness of the functionalization of the swCNT is crucial for the usefulness of the device[1]. Molecular Dynamics (MD) simulations are well-suited to provide insights into the fundamental properties of DNA-CNT hybrids because they enable calculations of structural properties with an atomic resolution. MD simulations have been used by A.T. Johnson *et.al.* to understand the interactions between swCNT and ss-DNA[2] that influence electron transport such as π-π stacking[3].

Since its first use by A. García[4] for the analysis of MD simulations results, Principal Component Analysis (PCA) is often used to trace the conformations of molecules especially proteins (see R.L. Jernigan *et.al.*[5] for a recent example). Often the *x, y, z* coordinates of the positions of atoms of the molecule serve as the input for the PCA method although the use of the dihedral angles in the backbone has been explored recently[6]. The problem of determining an appropriate sample size has been studied from the point of view of the length of the simulation[7]. Results on the influence of the selection criteria of atoms or angles on the PCA are difficult to find.

This paper presents the application of PCA to the novel studies of conformation of ss-DNA onto CNTs. In order to validate the selection of atoms for the PCA in further studies a comparison is made between the results obtained from several selections of atoms.

2. **Background**

2.1 **carbon nanotubes**

Carbon nanotubes (CNT) are cylindrical sheets of carbon. CNT have diameters of ~1nm and lengths up to a few centimeters[8]. They have many interesting properties. They can have tensile strength as high as sixty times larger than steel. They also show electronic stability. Nanotubes can accommodate current densities 1000 times higher than copper and silver.

2.2 **ss-DNA**

Single stranded DNA (ss-DNA) is a DNA molecule consisting of only one chain of alternating sugars and phosphates. It can assume different structures depending on the solvent and ionic environment. For this work some ss-DNA composed of a repeating sequence of cytosine (Poly-Cytosine) and specific sequences of ss-DNA were used. Figure 1 shows a ss-DNA onto a CNT.

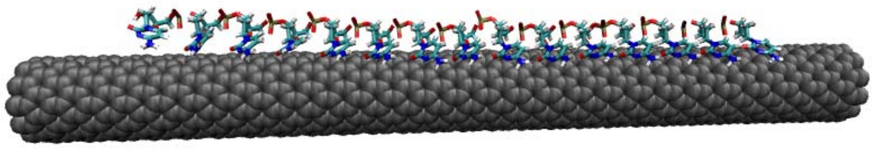


Figure 1. ss-DNA onto a CNT.

2.3 **MD simulations**

MD simulations calculate the trajectories of *N* interacting atoms by numerically solving Newton's equations of motion for each atom. Since atomic forces are conservative, they can be described by a potential function. There are a variety of atom’s interactions some of these exists between atoms that do not share a chemical bond (electrostatic and the Van der Waals forces). They are other forces that act on atoms that share chemical bonds. These forces are bond stretching (bond force), bond bending (angle force) and bond twisting (torsion force).

2.4 **PCA**

Principal Component Analysis (PCA) is a technique used to reduce data that is represented in a high number of dimensions to 2D or 3D. PCA allows to visualize the similarities and/or differences in a set of data. Among the wide range of applications of the PCA, it can be used to study the shapes of molecules studied by means of MD simulations[4]. The technique has been used successfully in the analysis of proteins[9].

3. **Software and methods**

The GROMACS MD package[10] was used to perform MD simulations. A detailed script was written in order to precisely the set up of the system[11]. In the following sections is the outline of the procedure to set up and to run the MD simulations. Also, there is a detailed description of the organization of the data and the PCA performed into the data.

3.1. **MoSDAS**

The Model building, Simulation and Data Analysis Script (MoSDAS[12]) was developed to automate the production of the MD system. The development of MoSDAS simplifies and avoids the most of the errors in the simulation process. All the commands of MoSDAS are in *bash* programming language. The main purpose of MoSDAS is to call and run other programs.

The first step is to generate the ss-DNA with *nucleic* program, which is part of the Tinker molecular modeling package[13], a CNT 30Å longer than the ss-DNA is also generated. The ss-DNA and the CNT were joined with *tleap,* which is a subprogram of the AMBER7 MD package[14]. The system was placed inside of a box and hydrated with water. The water inside of the tube was removed with the *tcl* script *rem-wat-interior.tcl* developed by Robert Johnson[3]. The *index* file is the file that classified the atoms by groups, for example WATER, DNA_20L, etc. This file was made with *make_ndx* program, a program that is part of GROMACS. The *topology* file of the system was generated with an *awk* script. The *master input* file was edited from a template. The editions of the *master input* file were made with a script. Periodic boundary conditions in all directions were used.

3.2 **simulations**

Before each MD simulation a minimization of the system is required. Some systems also need a relaxation after the minimization.

MD simulations of Poly-Cytosines of 5, 15, 25, and 30 monomers as well as the sequences:

Sequence 1: 5’ AAA ACC CCC GGG GTT TTT TTT TTG 3’
Sequence 2: 5’ CTT CTG TCT TGA TGT TTG TCA AAC 3’
Sequence 3: 5’ GAG TCT GTG GAG GAG GTA GTC 3’

These specific sequences are the same used in actual experiments of gas sensors by A.T. Johnson *et.al.*[3]. Each of these simulations was run for 10ns and not all of them have been equilibrated yet.
After the simulation the trajectory data was visualized with Visual Molecular Dynamics (VMD)[15]. The program VMD is a molecular visualization program for displaying, animating and analyzing large biomolecular systems using 3D graphics and built-in scripting[16]. The visualization of the data was made because we wanted to be sure that the system was stable.

3.3 **organization of the trajectory data**

The trajectory file produced by GROMACS that contains all the coordinates, velocities, forces and energies as in the GROMACS *master input* file. This file is in portable binary format and can be read with *gmxdump*, program given by GROMACS.

3.3.1 *atoms sets*

This study is focused on the shape of the ss-DNA only. Different subsets of the ss-DNA atoms were chosen ensuring a uniform distribution across the length of the molecule.

For Poly-Cytosine we have five sets:
1. All ss-DNA atoms except for Hydrogen atoms
2. All atoms in the backbone of the ss-DNA
3. All atoms in all the rings of the ss-DNA
4. The N4 atoms (it is on the end of each ring)
5. The P atoms (they are at backbone)

For the specific sequences we have three sets:
1. All ss-DNA atoms except for Hydrogen atoms
2. All atoms in the backbone of the ss-DNA
3. All atoms in all the rings of the ss-DNA

3.3.2 *script to generate the data matrix*

To perform the PCA of the trajectory data of the different atoms sets it is required to create a data matrix. This matrix needs to have a row for each frame of data, and the columns are the x, y, z coordinates of the atoms. For example, the coordinates of atom one are the first three columns of the matrix and all the rows in these three columns are the trajectory of this atom. To generate this matrix a *bash* script was developed. A matrix for each atoms set was generated.

3.4 **PCA in the trajectory data**

After the matrices for each sequence and each atoms set were built, a PCA for all matrices was produced. To make the PCAs, MatLab was used. MatLab is a high-performance language for technical computing[17]. To obtain the PCA the `princomp(x)` function of MatLab was used. After the PCA, the standard scores were analyzed. The first scores were used to make plots of all atoms sets of the ss-DNA. A qualitative comparison of the data was made.

4. **Results and Discussion**

A visual inspection of Poly-Cytosine ss-DNA reveals that the molecule conforms onto the CNT in a counterclockwise helical way[2] (Figure 2).

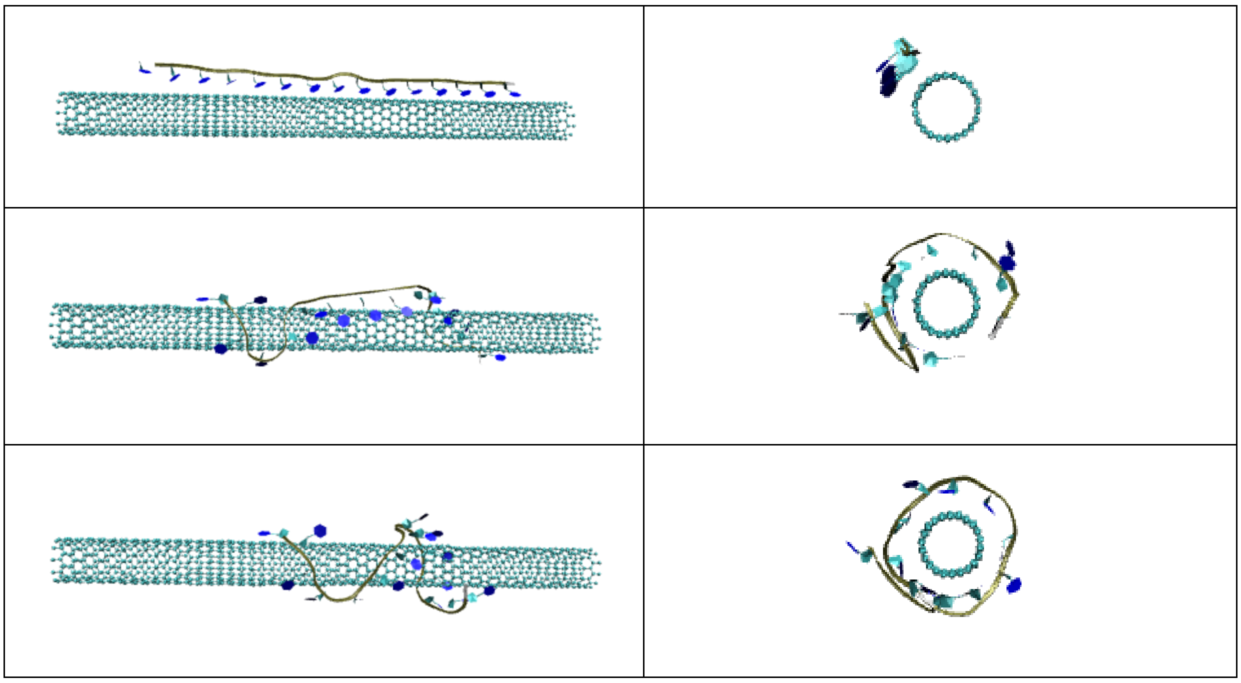
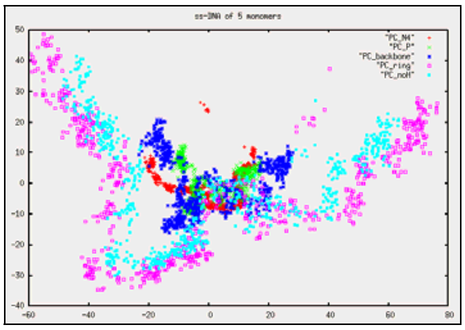
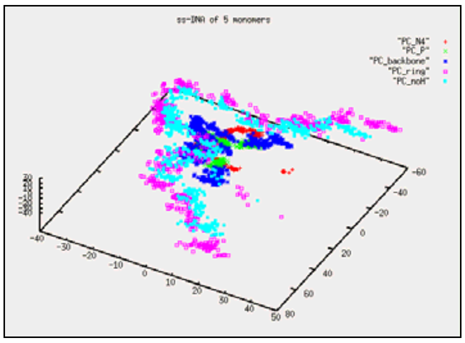
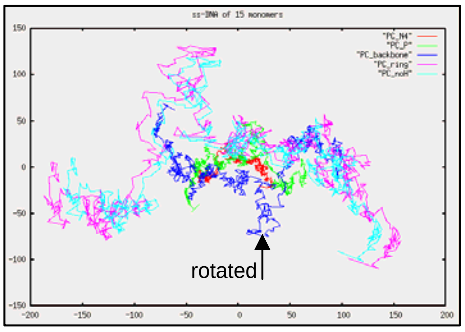
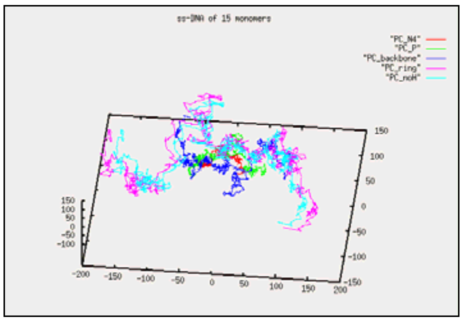


Figure 2. Snap shots of Poly-Cytosine ss-DNA of 15 monomers during the MD simulation.

Figure 3 shows the 2D and 3D PCA reductions of all Poly-Cytosine ss-DNA. In this figure the red line corresponds to N4 atoms set, green corresponds to P atoms set, blue corresponds to backbone atoms set, pink corresponds to ring atoms set, and light blue corresponds to the set of all atoms except for Hydrogen.

Monomers	2D	3D
5		
15		

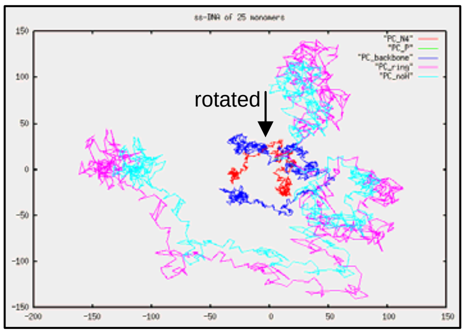
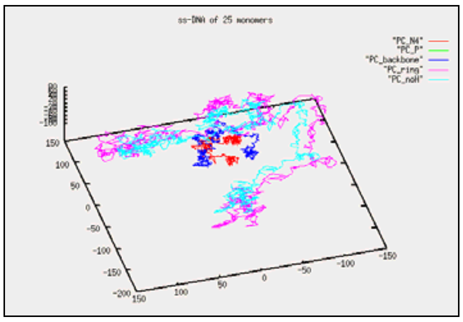
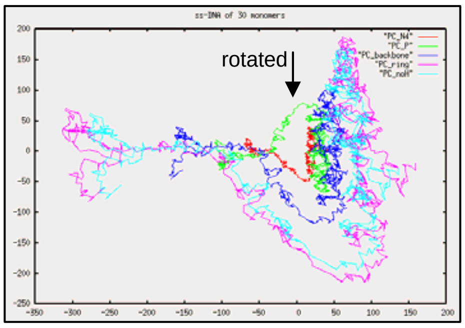
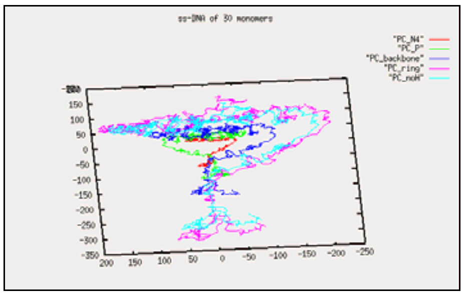
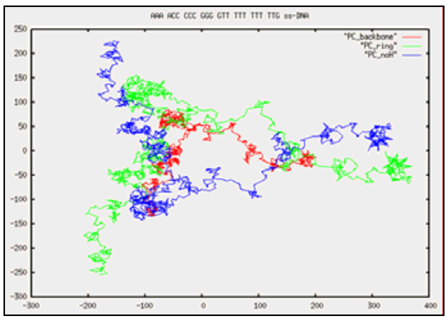
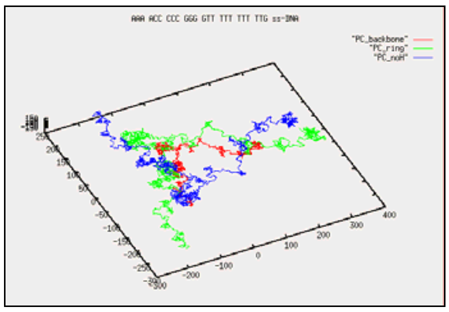
25		
30		

Figure 3. Plots of scores of PCA of all Poly-Cytosines.

In these plots (Figure 3) clusters of points indicate stable configurations of the system. In all cases the clusters around the point that marks the final frame of the simulation are notably densest of all. Moreover, the absence of multiple paths between clusters and of any return path to previous clusters demonstrates that those last conformations are stable and the process is irreversible within the time frame of the simulation. This phenomenon is independent of the length of the Poly-Cytosine.

In Figure 3 is easy to see that the shapes of the lines are similar for all atoms sets. For example, the 2D and 3D plots of Poly-Cytosine of 30 monomers show that the shapes of the lines are almost the same. Also, the set of *P* atoms is rotated compared with the other four sets. The orientation of lines can change occasionally but that does not affect the interpretation of the results. Another important result is that the sets that contain more atoms show that the line in the plot is more extended in comparison with the smallest sets. This is related with the way in which the PCA is computed. In Poly-Cytosine of 30 monomers, the set of all atoms except for Hydrogen and the set of the ring atoms are the biggest ones. This is consistent with all Poly-Cytosine ss-DNA. The smallest set for all ss-DNA is the N4 atoms set. The red line in the plot confirms that because is the most compressed line in each plot. These observations are consistent for all Poly-Cytosine ss-DNA. It is essential to see that although they are some of the lines rotated, compressed or extended that does not affect our results because this study is based on searching for clusters to determinate that the molecule has stable conformations.

Visual inspection of the system of the specific sequences of ss-DNA and CNT shows the ss-DNA conforming onto the CNT in a way similar to Poly-Cytosine. In Figure 4 the red line corresponds to the set of backbone atoms, green corresponds to ring atoms set, and blue corresponds to the set of all atoms except for Hydrogen.

Sequence	2D	3D
1		

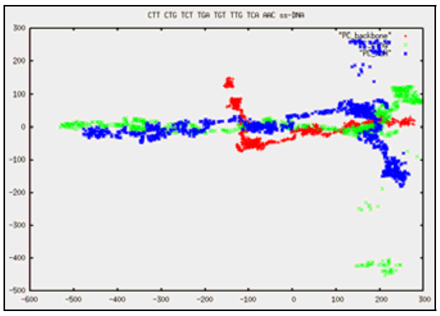
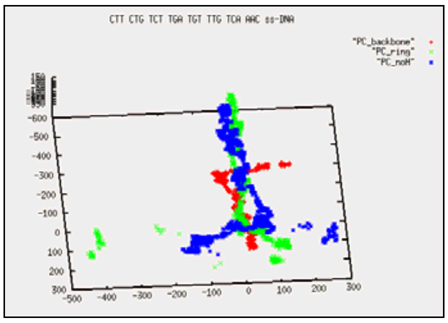
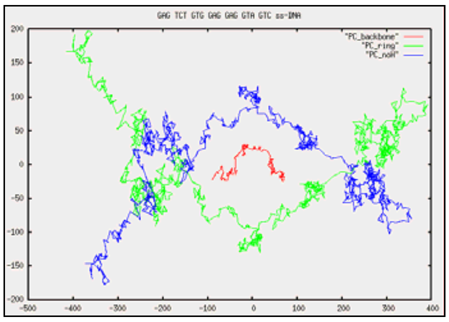
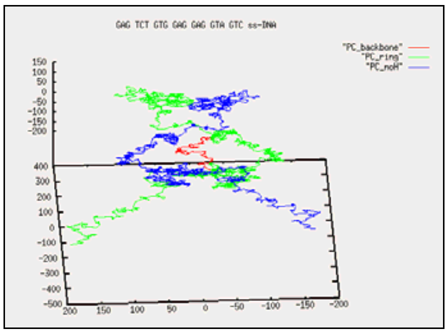
2		
3		

Figure 4. Plots of scores of PCA of the specific sequences.

The PCA depicted in Figure 4 shows patterns that demonstrate that the last conformations of the molecule are stable, and irreversible within the time frame of the simulation for the specific sequences of ss-DNA. The shape in each sequence is conserved no matter the atoms sets. As was in the Poly-Cytosine for a big set of atoms the lines were extended, as well for small sets the lines were compressed. They are lines rotated but this does not affect the results because the shape of the lines is conserved. Some of these plots show the trajectory of the molecule since the simulation was start. Examples for this observation are the sequence 1 and 2, here in the 2D plots the start is at the left side of the plots and the clusters at the right side represent a stable configuration for this ss-DNA. Also, there are no jumps in none of the plots.

5. **Conclusions**

The PCA analysis shows that the paths of the 3D points progress to eventually form clusters indicating that the shapes of the DNAs evolve into stable configurations. This is consistent no matter which atoms set its plotted. No returns in the paths nor jumps between clusters indicate that the DNA reaches a final conformation. These results are consistent with observations made in the laboratory proving that the PCA method is suitable for the study of the conformation of CNT-DNA hybrids.

6. **Future work**

Currently we are working on a Graphical User Interface (GUI) to setup, run and analyze systems of Polymer-CNT hybrids. This study justifies the inclusion of the PCA into this application.

7. **Acknowledgments**

This work was supported by Penn-UPR Partnership for Research and Education in Materials project (NSF-DMR-353730), Humacao Undergraduate Research in Mathematics to Promote Academic Achievement program (NSA-H98230-04-C-0486) and Minority Access to Research Careers program (NIH- 5T34GM008156-18). The author of this paper wants to thank to Robert Johnson and A.T. Johnson from the University of Pennsylvania for the introduction to the problem of MD Simulations, and to Robert Jernigan from Iowa State University for showing some of the analysis techniques used in this paper.

8. **References**

[1]Staii C., Johnson A.T., Chen, M., Gelperin, A., "*DNA-Decorated Carbon Nanotubes for Chemical Sensing*", Nano Letters, 2005, 5(9), 1774-1778.
[2]Johnson, R., Johnson, A.T., Klein, M., “*Probing the Structure of DNA-Carbon Nanotube Hybrids with Molecular Dynamics Simulations*”, American Physical Society, APS March Meeting, March 5-9, 2007.
[3]Johnson, A.T., Staii, C., Chen, M., Khamis, S., Johnson, R., Klein, M., Gelperin A., “*DNA-decorated carbon nanotubes for chemical sensing*”, 2006 Semicond. Sci. Technol. 21 S17-S21.
[4]Garcia, A.E., “*Large-amplitude nonlinear motions in proteins*”, Physical Review Letters, 68 (17). 2696-2699, 1992.
[5]Yang, L., Song, G., Carriquiry, A., Jernigan, R.L., “*Close Correspondence between the Motions from Principal Component Analysis of Multiple HIV-1 Protease Structures and Elastic Network Modes*”, Structure, Volume 16, Issue 2, 321 – 330.
[6]Altis, A., Nguyen, P., Hegger, R., Stock, G., “*Dihedral angle principal component analysis of molecular dynamics simulations*”, The Journal of Chemical Physics, Vol. 126, No. 24. (2007).
[7]Grossfield, A., Feller, S., Pitman, M., “*Convergence of Molecular Dynamics Simulations of Membrane Proteins*”, PROTEINS: Structure, Function, and Bioinformatics 67:31–40 (2007)
[8]Zheng, M., *et.al*, “*Structure-Based Carbon Nanotube Sorting by Sequence-Dependent DNA Assembly*” Science 28 November 2003: Vol. 302. no. 5650, 1545 – 1548.
[9]Teodoro, M., Phillips, G.N. Jr., Kavraki, L.E, “*A dimensionality reduction approach to modeling protein flexibility*”, In Proc. ACM Int. Conf. on Computational Biology, 2002, 299-308.
[10]GROMACS, http://www.gromacs.org/gromacs/features/feature-summary.html.
[11]Merced M., “*Metrics for the study of DNA-CNT hybrids*”, Proceedings of the National Conference on Undergraduate Research (NCUR 2007).
[12]Merced M., “Automatization of a Molecular Dynamics Simulation and the Evaluation of Metrics for the study of DNA-CNT Hybrids”, PREM Technical Report, December 2006.
[13]Tinker Molecular Modeling Package, http://dasher.wustl.edu/tinker/.
[14]The Amber Molecular Dynamics Package, http://amber.scripps.edu/.
[15]Humphrey, W., Dalke, A., Schulten, K., “*VMD- Visual Molecular Dynamics*”, J. Molec. Graphics 1996, 14.1, 33-38.
[16]Visual Molecular Dynamics, http://www.ks.uiuc.edu/Research/vmd/.
[17]The Math World- MATLAB and Simulink for Technical Computing, http://www.mathworks.com/.

Usage of OpenGL for the Representation of a Atom Trajectory

John E. Morales García
University of Puerto Rico at Humacao
100 Teja Avenue
Humacao 00791

Faculty Adviser: José Sotero Esteva

INTRODUCTION

MoSDAS is a easy to use Graphical User Interface (GUI) developed for the study of hybrid system as a plug-in for the VMD to develop Molecular Dynamics (MD) simulations. This work is part of the beginning of the graphic representation of the molecular trajectory of a atom or molecules. The main purpose is to use OpenGL to make the representation of this trajectory real, in 3D using as base PyOpenGL.

SOFTWARE AND METHODS

In this work the softwares utilize were Dr.Python (python 2.5 programming language) and PyOpenGL (GLUT library).

MAIN PURPOSE

To find a way in which a atom trajectory can be represented in a three dimensional view, using OpenGL as a base.

PyOpenGL

What is PyOpenGL and the GLUT library?

PyOpenGL is a standard specification defining a cross-language cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface have over 250 different functions calls which can be used to draw from complex three dimensional scenes from simple primitives. Almost all modern computers have it installed. Glut (pronounced like the glut in gluttony) is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL.

FROM C++ TO PYTHON

Since python is a object oriented computer language, most of the initial approach was based on the translation of C++ language code to Python language. We chose examples that shoold result in code that we expect to use in the final implementation of MASDAS-GUI.

Here is part of the pseudo code from the C++ Glutlib library, created by Andrew William Proksel of the Northwestern University:

This piece of code do all the initialization and setup needed to use the graphic library.

```
void InitGraphics()
{
      {int argc=0; char **argv=(char **)0;
      glutInit(&argc, argv);
      }

      glutInitDisplayMode(GLUT_DOUBLE| GLUT_RGBA);

      glutInitWindowPosition(NU_SCREEN_XPOS, NU_SCREEN_YPOS);
      glutInitWindowSize(NU_SCREENWIDTH,NU_SCREENHEIGHT);
      glutCreateWindow("Northwestern University - EECS-110 ");

      glClearColor(1.0,1.0,1.0,0.0);
      glColor3d(0.0,0.0,0.0);
      glPointSize(3.0);

      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0, (GLdouble)NU_SCREENWIDTH, 0.0, (GLdouble)NU_SCREENHEIGHT);

      glutDisplayFunc(DisplayCallback);
      glutKeyboardFunc(KeyboardCallback);
      glutSpecialFunc(SpecialCallback);
      glutMouseFunc(MouseCallback);
      glutIdleFunc(myIdle);

      glutMainLoop();
```

which its equivalent in Python is:

```
def InitGraphics(DisplayCallback):
      argv= []
      glutInit(argv)


      glutInitDisplayMode(GLUT_DOUBLE| GLUT_RGBA)

      glutInitWindowPosition(NU_SCREEN_XPOS, NU_SCREEN_YPOS)
      glutInitWindowSize(NU_SCREENWIDTH,NU_SCREENHEIGHT)
      glutCreateWindow("UPRH ")

      glClearColor(1.0,1.0,1.0,0.0)
      glColor3d(0.0,0.0,0.0)
      glPointSize(3.0)

      glMatrixMode(GL_PROJECTION)
      glLoadIdentity()
      gluOrtho2D(0.0, NU_SCREENWIDTH, 0.0, NU_SCREENHEIGHT)


      glutDisplayFunc(DisplayCallback)
      glutKeyboardFunc(KeyboardCallback)
      glutSpecialFunc(SpecialCallback)
      glutMouseFunc(MouseCallback)
      glutIdleFunc(myIdle)

      glutMainLoop()


angle = 0.0
NU_ANGLESTEP = pi / 180.0

rom10  = GLUT_BITMAP_TIMES_ROMAN_10
rom24  = GLUT_BITMAP_TIMES_ROMAN_24
helv10 = GLUT_BITMAP_HELVETICA_10
helv12 = GLUT_BITMAP_HELVETICA_12
helv18 = GLUT_BITMAP_HELVETICA_18
```

SIMULATIONS

Heat Transfer.

The heat transfer simulation consist of showing how a metal plate behave when heat its apply into one of the corners. Using of reference some example of Matrix without Linear Algebra of the course MATE 3009 given at the University of Puerto Rico at Humacao in 2006.

This part of the pseudo code defines dibujaPlaca(). To define this function, we most first scan the matrix that have the information and then paint a square with a color that represent the temperature assigned.

implemented in C++

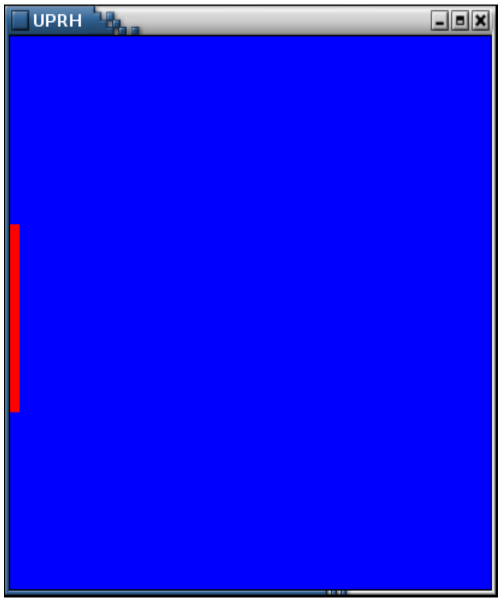
```
void dibujaPlaca(){
   int anchoCuadrito = 400 / N;
   for(int n=0; n < N; n++)
   for(int m=0; m < M; m++){
      double intensidad = (placa[n][m]tempMin)/(tempMaxtempMin);
      SetPenColor(intensidad, 0.0, 1.0intensidad);
      int baseX = n * anchoCuadrito, baseY = m * anchoCuadrito;
      DrawFillBox(baseX, baseY, baseX+anchoCuadrito,
                   baseY+anchoCuadrito);
   }
}
```

In Python since dibujaPlaca is declare void, theres no need for a formal function call dibujaPlaca(), we just can add the definition into the display function and we still getting the same results.

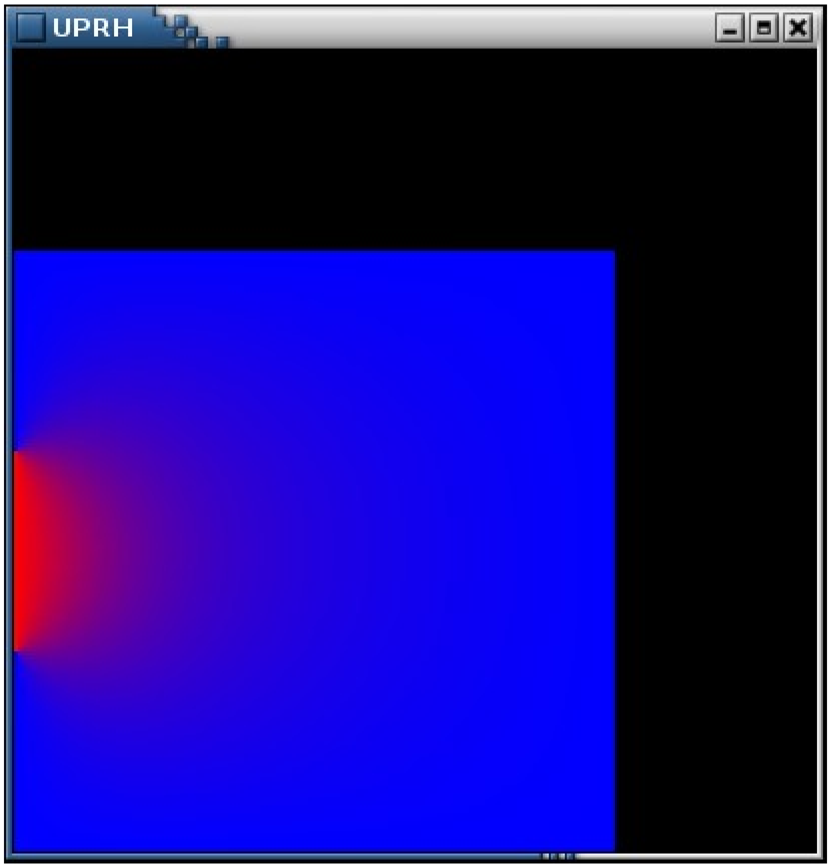
implemented in Python with PyOpenGL

```
def myDisplay():
      global placa, tInterior, NMAX, tempMax, NU_SCREENWIDTH
      calculaTemperaturas(2.2, placa, NMAX)

      # dibuja
      anchoCuadrito = NU_SCREENWIDTH / NMAX
      intensidad = random.randrange(0.0,1.0)

      for m in range(NMAX):
            for n in range(NMAX):
                  intensidad = (placa[n][m] - tInterior) / (tempMax - tInterior)
                  SetPenColor(intensidad, 0.0, 1.0 - intensidad)
                  baseX = n * anchoCuadrito
                  baseY = m * anchoCuadrito
                  DrawFillBox(baseX, baseY, baseX + anchoCuadrito,baseY + \
anchoCuadrito)
```

Here are some pictures of the Heat Transfer Simulation.



In here, heat is apply to the middle left corner of the plate. (red area represent the heat, and the blue area is the plate).



In this other picture we can see the propagation of the heat over the plate.(the red haze area).

Movement of Spherical Bodies.

The main purpose of this simulation was to study the movement of spherical bodies and they're surface. To learn more of spherical objects I used as reference the OpenGL Programming Guide , better know as The Red Book , which is in C++ and not in Python.

Step 1. Building a wired solar system:

Since the Red Book is write in C++, we need to translate it into Python language.

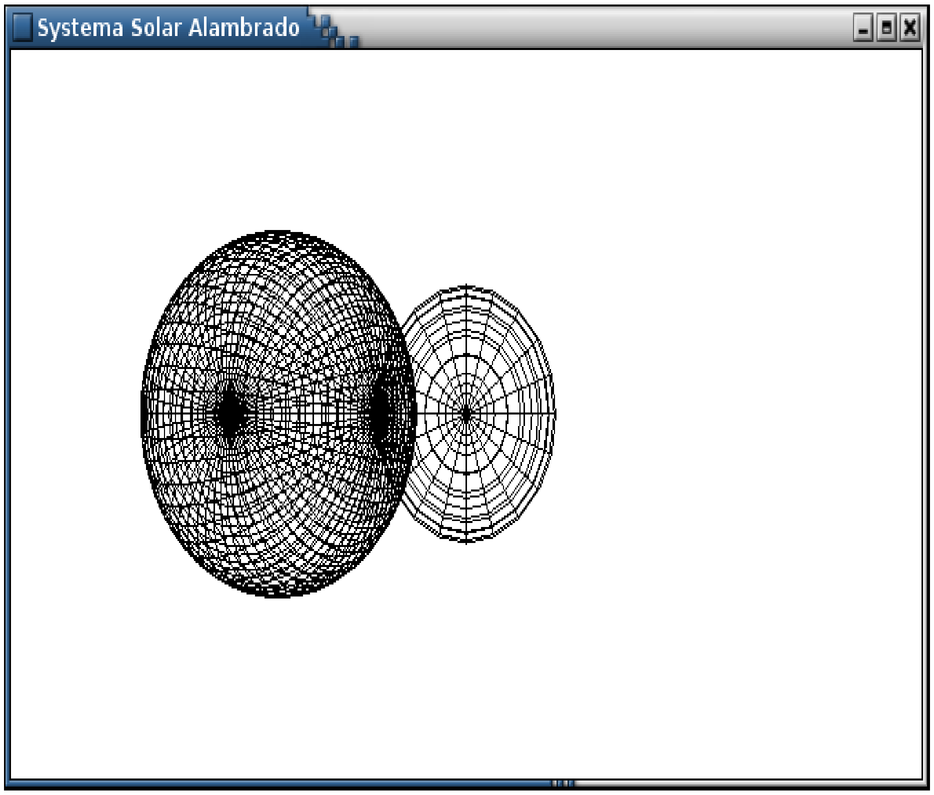
Since we are interested in Spherical Bodies, heres the function that display the spheres, this the display function of the program.

```
def display():
      glClear(GL_COLOR_BUFFER_BIT)
      glColor3f(0.0,0.0,0.0)

      glPushMatrix()
      glutWireSphere(1.0,20,16)
      glRotatef(year,0.0,1.0,0.0)
      glTranslatef(2.0,0.0,0.0)
      glRotatef(day,0.0,1.0,0.0)
      glutWireSphere(1.0,60,40)
      glPopMatrix()
      glutSwapBuffers()
```

This part of code is the responsible for giving movement to the solar system. It have the keyboard command to make the sphere that represent the planet to rotate in its own axis and around the sphere that represents the sun.

```
def keyboard(key, x, y):
      global year, day
      if key == 'd':
            day = (day + 10) % 360
            glutPostRedisplay()

      elif key == 'D':
            day = (day - 10) % 360
            glutPostRedisplay()

      elif key == 'y':
            year = (year + 5) % 360
            glutPostRedisplay()

      elif key == 'Y':
            year = (year - 5) % 360
            glutPostRedisplay()
```

This how the wired solar system looks at the End.

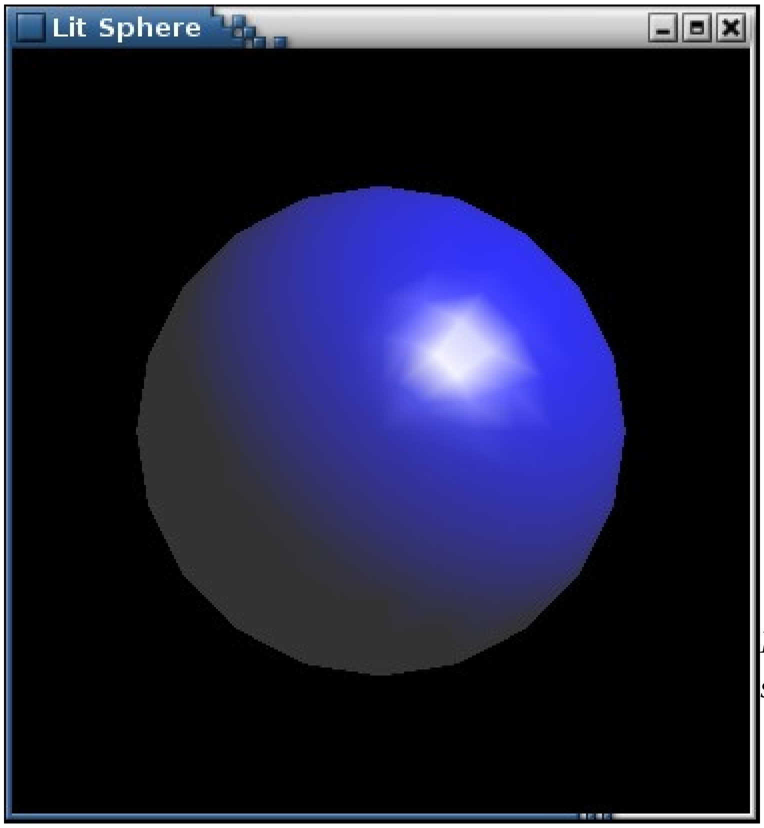


Step 2. Applying texture to a sphere, the Lit Sphere :

Since we now know how to make a wired sphere, we can now try to put some texture, light, color and light. The next function is the initialization for the rendering of a lit sphere.

```
def init():
      mat_specular = [1.0,1.0,1.0,1.0]
      mat_shininess = [50.0]
      light_position = [1.0,1.0,1.0,0.0]
      white_light = [0.0, 0.0,1.0,0.0]
      lmodel_ambient = [1.0,1.0,1.0,0.0]
      glClearColor(0.0,0.0,0.0,0.0)
      glShadeModel(GL_SMOOTH)
      glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular)
      glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess)
      glLightfv(GL_LIGHT0,GL_POSITION, light_position)
      glLightfv(GL_LIGHT0,GL_DIFFUSE, white_light)
      glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lmodel_ambient)

      glEnable(GL_LIGHTING)
      glEnable(GL_LIGHT0)
      glEnable(GL_DEPTH_TEST)
```

With this code we are able to position one or more light source. It also define material properties for the object in the scene and define the level of global ambient light and the effective location of the viewpoint.



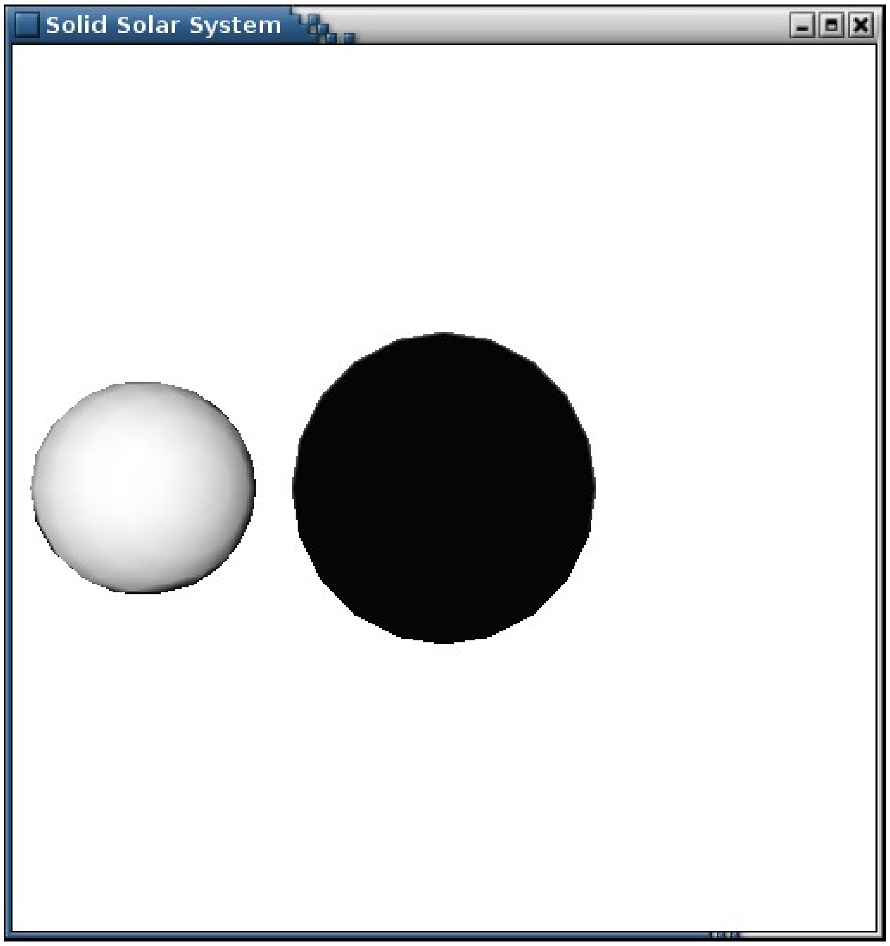
Here are the final results of the Lit sphere program.

Step 3. A solid solar system:

Now that we have the wired solar system, lets try to apply the Lit sphere method to it. If we add the previous code from the Lit sphere program and also change the display function of the solar system (previously shown) we can make a solid solar system.

```
def display():
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
      glColor3f(0.0,0.0,0.0)

      glPushMatrix()
      glutSolidSphere(1.0,20,16)
      glRotatef(year,0.0,1.0,0.0)
      glTranslatef(2.0,0.0,0.0)
      glRotatef(day,0.0,1.0,0.0)
      glutSolidSphere(0.5,20,10)
      glPopMatrix()
      glutSwapBuffers()
      glFlush()
```

The key definition here is glutSolidSphere(), which is the one responsible for solidifying the spheres.



This figure show the results of the solid solar system. (The Grey ball is the planet and the black one the sun).

REVIEW

PyOpenGL is a cross language, cross-platform that let us use the object oriented language Python, work well with the GLUT library. The C++ codes for some programs can be easily translate into Python without trouble. Simulations such like heat transfer and solar systems, can be of help in the development of new approach to MD. The behavior of spherical bodies is a good approach to the representation of a atom behavior.

CONCLUSION

The main purpose of all this simulations in OpenGL, is to apply them to the MoSDAS to create a way to represent the atom trajectory in a three dimensional view. With this methods of movement of spherical bodies we can give the atoms a sphere behavior to make the work of molecule trajectory more easier and since the adaptation of C++ language to Python works very well, we are able to work in a new methods and ideas in the future of Molecular Dynamics.

REFERENCE

Mason Woo, Jackie Neider, Tom Davis, DaveShreiner. OpenGL Programming Guide. Third Edition.Pearson Educatin Corporate Sales Division. 201 W.103rd Street

Tom McReynolds, David Blythe. Advanced Graphics Programming Using OpenGL . 500 Sansome Street, Suite 400, San Francisco, CA 94111.

Graphical User Interface to Run modelecular Dynamics Simulations of CNT-Polymer Hybrids in VMD , Myrna I. Merced, Paper, December 2007.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2008
Salisbury University
Salisbury, Maryland
April 10 - 12, 2008

A Methodology to Determine the Number of Clusters in Unsupervised Hyperspectral Image Classification

Axel Y. Rivera (Luis G. Jaimes)
Department of Mathematics
University of Puerto Rico at Humacao
Humacao, PR 00791

Faculty Advisor: Luis G. Jaimes

Abstract

A hyperspectral image is a collection of images using a large number of channels from the spectrum interval. These images are used to identify different natural phenomenon like variation of vegetation, minerals, etc. It is difficult to identify the components in these images. One of the principal problems in unsupervised classification is to determinate the natural clusters in which the data is distributed and their number. The focus of this paper is to provide a methodology to determine the number of clusters when unsupervised classifications over hyperspectral images were made. The method implemented for this purpose consists in using different clustering algorithms (hierarchies and partitioning) in combination with different validation methods (external and internal). These combinations gave clues about possible best number of clusters. The first experiments were done with some hyperspectral images reduced for algorithm calibration purposes. These experiments were successful using images that contain well defined and separated objects, also with images that contain objects that are overlapped by other objects.
Keywords: Clustering, Data Mining, Hyperspectral Images

1. Introduction

Hyperspectral images, that are taken most of the times from airborne scanners or satellites, are a combination of images taken in different light frequencies. These light frequencies are based in the electromagnetic spectrum chart and are used to identify things that can not be seen by human eyes. Thanks to this configuration, these images provide useful information like variation of vegetation, minerals, etc. This information can be used for moisture studies, forests ignitions and others.

These images are composed from one hundred to three hundred bands. Each band is the same picture in different light frequencies and each pixel, depending from the sensor, can represent approximately thirty by thirty square meters. Because of these properties, these images are composed a huge collection of data. A classification method is needed because the human eye can not identify objects at this level and the quantity of data is massive. The classification methods that are used often are: supervised and unsupervised (clustering) classification.

The supervised classification is bases on classifying *n* objects in different groups. These groups have user-defined training sets, it means, the user defines the properties of the groups where the objects are going to be classified.

Unlike the supervised classification, the unsupervised classification, or clustering, tries to group *n* objects in *k* partitions without any user-defined training sets, in other words, the objects are classified by themselves and not by user defined properties. These properties leave an opening to the problem of determining the natural clusters in which the data is distributed and their number.

2. Preprocessing

The hyperspectral images are composed by many bands. This collection of data is organized as a stack of pictures from lowest to highest wave length. After the data is organized in this form, it forms a three dimensional matrix (cube). This cube is called the hyperspectral cube.

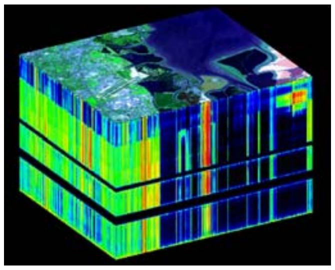


Figure 1. Example of a Hyperspectral Cube

In this cube a pixel can be seen as an array of data. This property helps to transform the cube into a matrix. In this matrix each row represents a pixel and each column represents the value of the pixel in each band. This transformation helps to apply the clustering algorithms that are going to be used.

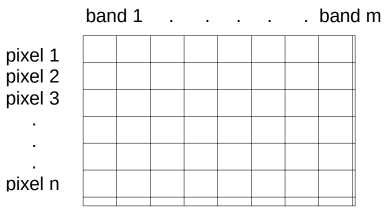


Figure 2. Transformed Matrix

3. **Algorithms**

3.1 **clustering algorithms**

A clustering algorithm is a partitioning method that tries to group a data set into subsets without any user-defined properties, it means, the data arrange by themselves. Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics. The data that is each cluster (partition) share some properties in common, most of the times the distance between the object and the cluster center.

3.1.1 *k-means*

$$K_{mn}(S) = \sum_{i=1}^{K} \sum_{X_n \in S_i} \left((X_n - C_i)^2 \right) \tag{1}$$

$$\text{K} = \text{clusters quantity}, \text{S}_j \subset \text{K}, \text{xn} = \text{data vector}, \text{M}_j = \text{centroid}$$

The K-Means clustering method is an algorithm that groups *n* objects in *k* partitions, where *k* < *n*. The objective that it tries to achieve is to minimize total intra-cluster variance, or, the squared error function. The centers of the clusters in this algorithm are based on centroids. A centroid is an approximation of a subset center and is given by the average of the sum of all elements in the subset. A centroid is defined as:

$$X = \{x_{1,} x_{2,} x_{3,} ..., x_n\}, n > 0$$
$$C_x = \frac{x_1 + x_2 + x_3 + ... + x_n}{n} \qquad (2)$$

3.1.2 *k-medoids*

$$K_{md}(R) = \frac{1}{(P)} \sum_{i=1, p \in P}^{K} \left((p - M_i(P))^2 \right) \qquad (3)$$
P = data, R c P, p ∈ P, r*(p)* = medoid

K-medoids algorithm have the same purpose as K-means, this means, it groups a set of *n* objects into *k* partitions, where *k* < *n* and tries to minimize total intra-cluster variance. The only difference is that K-medoids' centers are based on medoids. A medoid is the element of a *K* cluster that has the minimum distance to all elements in the same cluster. A medoid is represented as:

$$X = \{x_{1,} x_{2,} x_{3,} ..., x_n\}, n > 0$$
$$M_X = min(x_i)_{x_i \in X} \left\{ \frac{1}{|X|} \sum_{j=1, j \neq i}^{|X|} \{ \gamma(x_i, x_j) \} \right\} \qquad (4)$$

3.2 **validation algorithms**

When a data has been classified, it needs to be verified. Validation algorithms are those that tend to verify how good the data classification was. These algorithms are important, especially when clustering is used, because they can tell how accurate a classification was and can provide information about the best partition number.

3.2.1 *average silhouette*

The Silhouette validation technique calculates the silhouette width for each element in the data set, average silhouette width for each subset and overall average silhouette width for a total data set. It is based on the comparison of its tightness and separation between clusters. The average silhouette width could be applied for evaluation of clustering validity and also could be used to decide how good the number of selected clusters is. In this validation method the best number of cluster is given by the greater Silhouette value. It is defined as:

$$AS = \frac{1}{|K|} \sum_{i=1}^{|K|} \left\{ \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \right\} \qquad (5)$$
a(i) = average dissimilarity of i object to all objects in same cluster
b(i) = minimum of average dissimilarity of i object to all other clusters

3.2.2 davies-bouldin

Davies-Boulding validation algorithm tries to identify the clusters that are compact and well separated. The objective is to obtain clusters with minimum intra-cluster distances. Therefore, this index is minimized when looking for the best number of clusters. It is defined as:

$$DB = \frac{1}{|C|}\sum_{j=1}^{|C|} max_{j \neq i}\left\{\frac{\Delta(x_j)+\Delta(x_i)}{\delta(x_i, x_i)}\right\} \tag{6}$$

C = clustered data, X_i = cluster, (X_n) = intra - cluster distance, $\delta(X_i, X_j)$ = inter - cluster distance

3.2.3 *calinski & harabasz*

The Calinski & Harabasz's algorithm tries to find the best partition between *k* clusters. The results in this validation algorithm will be high positive numbers. The greater result is considered as the best number to divide the data. It is defined as:

$$CH = \frac{[SSB(k-1)]}{[SSW(n-k)]} \tag{7}$$

SB = between cluster squares' sums, SSW = within cluster squares' sums, k = clusters quantity, n = data quantity

3.2.3 *dunn's index*

This validation method has the same purpose as the Davies-Bouldin algorithm. It tries to identify the compact and well separated clusters. The main goal of the measure is to maximize the inter-cluster distances and minimize the intra-cluster distances. Therefore, the number of clusters that maximizes *Dn* is taken as the optimal number of the clusters. This algorithm is defined as:

$$Dn = min_{1 \leq i \leq |C|}\left\{ min_{1 \leq j \leq |C|, j \neq i}\left\{\frac{Dist(C_i, C_j)}{max_{1 \leq k \leq |C|}\{Diam(C_k)\}}\right\}\right\} \tag{8}$$

$\mathrm{Dist}(C_i, C_j) = \min\{\delta((x, y)\}, x \in C_i, y \in C_j, \mathrm{Diam}(C_i) = \max\{\gamma((x, y)\}, x, y \in C_i$

3.3 **distances implemented**

The clustering and validation methods are based on the distance between two points (δ(x,y)) function, making them one of the most important function for this work. Different distance functions can be used (*Euclidean, Manhattan, etc.*), but for this work the distance measure used was the *Euclidean* distance. The *Euclidean* distance equation is:

$$p_n \in P, q_n \in Q, n \in N$$
$$\gamma(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{9}$$

3.3.1 *intra-cluster distance*

The intra-cluster distance is the distance between the elements that are in the cluster. For this work the intra-cluster distance implemented is the *centroid diameter* distance. This distance reflects the double average distance between all of the samples and the cluster's center. It is defined as:

$$\Delta(S) = 2\left(\frac{\sum_{x \in S} \gamma(x, v)}{|S|}\right) \tag{10}$$

v = center

3.3.2 *inter-cluster distance*

The inter-cluster distance is the distance between two or more clusters. In this work the inter-cluster distance used is the *centroid linkage*. This function reflects the distance between the centers of two clusters. It is given be the equation:

$$\delta(S,T) = \gamma(vs, vt) \tag{11}$$

vs = center of S, vt = center of T

4. **Methodology**

The methodology used is the following. The first step consists in the preprocessing of the data. Here the hyperspectral cube is transformed into the matrix mentioned before for clustering.

The next step consists in finding the number of the natural clusters where the data is distributed. This process is carried out by the following methodology. The first step is to select one of the two clustering algorithms. The transformed matrix is loaded to it and is clustered using different partition quantities, most of the time from 2 to *k*. This process is performed with both clustering algorithms.

All partition quantities are passed through all four validation methods. The algorithms will give a number. The number that is repeated more between all validation methods is selected as the best number. This process is repeated *n* times. Each time that a process is repeated is called an experiment.

A final table is constructed with the results of the experiments. This table contains a percent rate of the number that appears most of the times in each experiment between *x* clustering method with *y* validation method combination. Results with high percent rate represent that the combination is good for finding the optimal number. The number that appears most of the time in this table is selected as the best number for clustering.

5. **Implementation**

Given that a hyperspectral image is a huge collection of data, it was not possible to use Matlab and R most popular clustering algorithms. MatLab was only used for the transformation of the hyperspectral cube to the matrix. A tool was developed for the rest of the work. This tool is divided in two parts: the clustering methods and the validation methods.

The clustering methods were implemented using the C Clustering Library developed by Michiel de Hoon, Seiya Imoto, Satoru Miyano at the University of Tokyo. This library is a collection of numerical routines that implement the clustering algorithms that are most commonly used, for example: K-Means, K-Medoid, Hierarchical and Self-Organizing Maps. It was developed using C programming language, but the developers implemented these routines available for use with Python programming language. When the library is used in Python, it gives the parameters to C and works with the routines. After C finishes the clustering, it returns the results to Python.

For the integration of the validation methods, it was needed to be developed in C++ programming language. This programming language is regarded as a mid-level language; it is a combination of both high-level and low-level language features. These properties permit massive routines to be calculated in less time.

To combine this whole collection of routines and libraries, a graphical user interface (GUI) was developed. This GUI was made in Python using the wxPython libraries. To make possible the interaction between both programming languages (C++ & Python), the SWIG (Simplified Wrapper and Interface Generator) tool was used. SWIG is a tool used to connect language and makes possible the usage of libraries and programs between them. This tool made possible that the routines written in C++ can be added to the GUI without any problem.

6. **Results**

The developed tool was tested with two training images. These images represent the two extreme cases. One image shows four well separated figures with ten bands. The other image contains three figures, one over the other, with ten bands. The parameters were: clustering from two to ten clusters and the experiments were repeated five times. These images where used for algorithm calibrations and methodology tests. The tables with the percents rates mentioned before were made using the results obtained from the tests (see Table 1, Table 2, Table 3 and Table 4).

6.1 **four figures image**

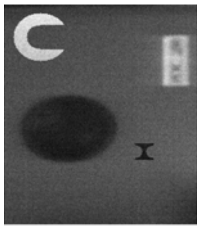


Figure 3. Four well separated objects image

6.1.1 *k-means results*

Test Number		1	2	3	4	5		Best Number	Percent (%) Rate
Silhouette		3	3	3	3	3		3	100%
DB		3	3	3	3	3		3	100%
Calinski		7	7	5	7	7		7	80%
Dunn		3	3	3	3	3		3	100%

Table 1. Four figures image's K-means results

6.1.2 *k-medoids results*

Test Number		1	2	3	4	5		Best Number	Percent (%) Rate
Silhouette		4	4	4	4	4		4	100%
DB		4	4	4	4	4		4	100%
Calinski		4	4	4	4	4		4	100%
Dunn		4	4	4	4	4		4	100%

Table 2. Four figures image's K-medoids results

6.2 **three figures image**

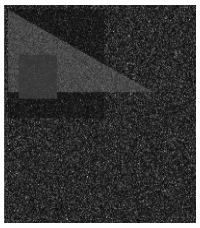


Figure 4. Three overlapped objects image

6.2.1 *k-means results*

Test Number		1	2	3	4	5		Best Number	Percent (%) Rate
Silhouette		3	3	3	3	3		3	100%
DB		3	3	3	3	3		3	100%
Calinski		8	8	8	8	8		8	100%
Dunn		3	3	3	3	3		3	100%

Table 3. Three figures image's K-means results

6.2.1 *k-medoids results*

Test Number		1	2	3	4	5		Best Number	Percent (%) Rate
Silhouette		3	3	3	3	3		3	100%
DB		3	3	3	3	3		3	100%
Calinski		7	7	8	9	7		7	60%
Dunn		3	3	3	3	3		3	100%

Table 4. Three figures image's K-medoids results

7. **Conclusion**

The clustering algorithms were tested in two extreme cases, one with well separated objects image and another with very united components image. For the well separated objects image, the results showed that K-Medoids method responded very precise in combination with all validation methods. It showed that for all cases these combinations gave the correct cluster numbers. The combination between K-Means method with all validations methods did not assert in any case.

The image with the objects that are overlapped showed that all combinations of clustering and validation methods asserted the correct number of clusters with the exception of both clustering algorithms with Calinsky & Harabasz validation method.

These results give a recommendation that the best number for clustering a hyperspectral image can be found with the combination of K-Medoids algorithms with Dunn's Index, Average Silhouette or Davies Bouldin validation methods.

8. **Acknowledgement**

Thanks to Michiel de Hoon, Seiya Imoto and Satoru Miyano at the University of Tokyo for providing the PyCluster libraries. Also, thanks to Santiago Velasco from the University of Puerto Rico at Mayagüez for providing the Hyperspectral Image Generation Tool. This work is sponsored by URMAA (H98230-07-1-0114) and the Puerto Rico Louis Stokes Alliance for Minority Participation.

9. **References**

1. Bolshakova N. & Azuaje F., “Cluster validation techniques for genome expression data”, 2002
2. de Hoon M., et al., “The C Clustering Library”, 2005
3. Jaimes L., “Uso de técnicas de clasificación en conglomerados para describir perfiles en grandes bases de datos educativas”, 2004
4. Rousseeuw P., et al., “Clustering in an Object-Oriented Environment”, 1998
5. Velasco S. & Manian V., “Improving Hyperspectral Image Classification using Spatial Preprocessing”, 2008
6. Bolshakova N., et al., “Cluster Validity Algorithms”, http://machaon.karanagai.com/validation_algorithms.html
7. Fulton W. & Beazley D., “Simplified Wrapper and Interface Generator (SWIG)”, http://www.swig.org

A Graphical User Interface to Specify Parameters for Molecular Dynamics Simulations

Desirée E. Velázquez Ríos
842-06-9756
Department of Mathematics
University of Puerto Rico at Humacao
Faculty advisor: José O. Sotero Esteva

I. Introduction

I.1 Molecular Dynamics

A Molecular Dynamics (MD) simulation requires the specification of many parameters, including the name and localization of several data files; physical parameters such as temperature, and simulation parameters such as the amount of time-steps, periodic boundary conditions and many others. These configuration files are written most of the time using a text editor. This is a difficult task and the source of many errors that can result in the waste of execution time in sophisticated computer systems.

The reason is that the configuration files are composed by files of the extension type *.pdb* and *.psf* that contain the following information: name of the compound, species and tissue from which is was obtained, authorship, revision history, journal citation, references, amino acid sequence, stoichiometry, secondary structure locations, crystal lattice and symmetry group, and finally the ATOM and HETATM records containing the coordinates of the protein and any waters, ions, or other heterogeneous atoms in the crystal. These files are organized in a specific way so that each file can be reused without having to change much the information contained. As mentioned before, these files are then used to create the configuration files, with the extension .conf for NAnoscale Molecular Dynamics (NAMD).

I.2 MOSDAS-GUI

A Graphical User Interface (GUI) to specify parameters for MD simulations was developed in order to simplify the earlier mentioned process and reduce the number of errors the can waste execution time. The interface presents to the user a set of templates to the specification of the configuration parameters. The templates group the parameters by categories according to their nature: physical, data, etc. Special attention has been put into the usability of the GUI. Suggestions of the values of the parameters are made based on previous choices such as commonly used data sources and physical parameters, as well as the choices of parameters that the user is making during the session.

At the start, the GUI is opened through the Visual Molecular Dynamics (VMD) program, which shows the menu bar from where the user will work. It prevents the user from accessing anything while nothing is on screen, or has been created. Once the user creates a molecule or system of molecules, such as fibers that can include a strand of DNA with a carbon nanotube or different types of polymers such as Copolymers or Homopolymers that include Polyaniline, the GUI opens up new options that the user is now able to use. When creating a system one is able to determine the length or components of said system

But before the user is able to access options like moving or centralizing the systems, he must first select which system or systems of molecules he would like to work with. Moving the system allows the user to change the xyz coordinates of the selected system, while centralizing it puts the geometrical center of the system in the 000 coordinate. If the user selects two or more systems, the merging option is available. This option merges the two or more selected systems, creating one whole, new system but the previously selected systems will be unable to be accessed as separate systems again. Other options, such as boxing the systems for simulations, are available as well.

The GUI also guides the user through the parameter specifications process by visually suggesting the order in which they should be entered. This guarantees that the user has finished entering a set of data before moving onto the next. And thus the GUI has proven to be a valuable tool for the specification of parameters for a MD simulation.

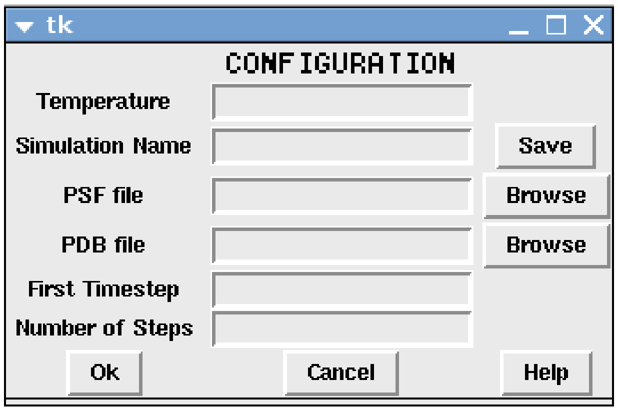
II. Window for MD configuration parameters

At first, I constructed the configuration window, that would help in the simplification of configuring parameters in MD simulations, for the GUI (spoken about in the last report), in order for it to be incorporated into the MOSDAS-GUI. This window was created using the Python programming language, the Tkinter library and the tkFileDialog module. In order to program well, the DrPython Interpreter was used. It has a feeling of the DevBloodshed C++ so it was easy to get accustomed to it.

To begin, we create a window using the Tkinter library. The window is divided by eight rows and three columns. For each phase that describes what is required of the user, a Label is used. A variable, containing the label is appended to the desired position of this 8x3 table window. In the first row, only the label of the title of the window is contained in the second column.

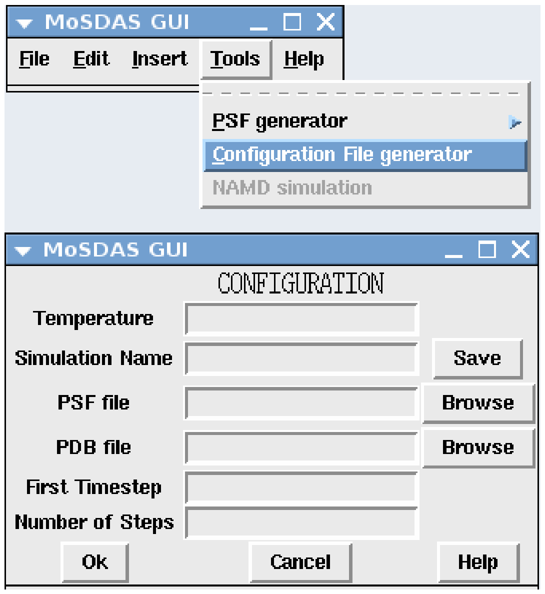
In the second, sixth and seventh rows, labels that require only text written by the user are contained in the first column. The second column for these rows contains an empty space box each where the user will write the required information. For the third, fourth and fifth rows the same process just mentioned is applied. The only difference is that the user need not write the names for all three rows. The third row requires the user to write a line of text, but he is able to browse where he'll save the configuration file, along with the option of just selecting a previously saved file to replace. The fourth and fifth rows share a similarity to the third row, but it only requires the user to select the name of a *pdb* and *psf* file to add to the configuration file. This process is accessed through the use of buttons and thanks to the tkFileDialog module and its asksaveasfilename() and askopenfilename() functions which can be selected through the use of said buttons.

The last row contains three important functions. The first column contains the *ok* button, where once the user has pressed it, the configuration window saves the information written and sends it the default folder or the folder selected by the user through the button on the third row. Once it has saved and sent the configuration file, the window closes. The *cancel* button does as its name implies, it cancels the activity and closes the window without saving anything. Lastly, the *help* button will give suggestions in case the user does not know what he should write. A picture of the finished window running from the computer directly can be seen below.



III. Results and Discussion

Once incorporated and the user has done the earlier mentioned steps, he will be able to save the system(s) through the VMD menu or through the configuration window, shown below again, this time running through the VMD program.



With this, we are able to tell that our window works just fine. But we can improve the configuration file generator by adding suggestions in the empty box spaces.

IV. Future Work

To include suggestions for each option that the user utilizes. By adding suggestions in the space boxes that are empty right now, the user will be able to have a clearer idea of what it is that he needs to put in. Another way of improving the configuration window is by adding a bar that the user can slide for options such as temperature or number of steps, allowing him to know from where to where he should select the information.

V. References

- *Un Interfaz Gráfico de Usuario para Especificar Parámetros en Simulaciones de Dinámica Molecular* (poster), SIDIM 2008, UPR Carolina, March 1, 2008

- *A Graphical User Interface to Specify Parameters for Molecular Dynamics Simulations* (poster), Symposium on Nanotechnology, Palmas del Mar, Humacao, May 2, 2008

- *Python Essential Reference 3rd Edition*, David M. Beazly, Sams Publishing (2006)

- *Tkinter reference: a GUI for Python*, John W. Shipman, New Mexico Tech (2004), http://www.nmt.edu/tcc/help/pubs/tkinter

- *An Introduction to Tkinter*, Fredrik Lundh (1999), http://www.pythonware.com/library/tkinter/introduction/

- http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/namd-tutorial-unix.html (NAMD Tutorial by the Computational Biophysics Workshop of the Beckman Institute from the NIH Resource for Macromolecular Modeling and Bioinformatics in the University of Illinois at Urbana-Champaign)