Detection and Treatment of Almonds with Concealed Damage

BY

TOM CLARK PEARSON
B.S. California State University, Fresno, 1987
M.S. University of California, Davis, 1994

DISSERTATION

Submitted in partial satisfaction of the degree requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Engineering

in the

OFFICE OF THE GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

____

____

____

____

Committee in Charge

1998

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I want like to extend my appreciation to all the committee members who served on this thesis: David Slaughter, Thomas Schatzki, Jim Thompson and Tom Rumsey. I first started working with all four of these people when I came to Davis in 1992 to start my Masters degree research on pistachio nuts. Their many hours of consultation and advice throughout this project has been invaluable. Dr. Slaughter was my advisor at UC Davis for both my Masters and Doctorate degrees. I have gained immensely from his incredible patience, teaching abilities, and broad technical knowledge. Dr. Schatzki, who is a lead scientist at the USDA-ARS, was responsible for funding of my Master's degree work at Davis, then he brought me into his group at the USDA to continue research on pistachio nut sorting. It was Dr. Schatzki's support and encouragement that enabled me to pursue a PhD and to conduct this study on almonds.

Many thanks goes to the Almond Board of California for partial funding of this project.

I'd like to extend my appreciation to my parents and grandparents. These are the people who introduced me to agriculture and engineering. The way that they lived, and pride that they put into their work, is something that I will always admire.

Finally, my acknowledgment would not be complete without expressing my deepest appreciation to my wife, Dr. Hülya Akdoğan. She has constantly been a source of strength throughout this project. Her continuous display of confidence in me and support cannot be overstated.

ABSTRACT

Chemical and physical properties of almonds were studied before drying, after drying and after cooking. During moisture exposure, such as a simulated rainfall, an increase of glucose, decrease of sucrose, and increased oil oxidation was observed. Indicators of concealed damage after drying but before cooking are increased oil oxidation from moisture exposure, slight color changes, and decreased water soluble solids. After cooking, a drop in reactive arginine, histidine, lysine was observed. These characteristics indicate that browning in almonds with concealed damage follows the Maillard reaction pathway.

Use of a lower temperature drying treatment can reduce the incidence of concealed damage. The incidence of concealed damage for nuts receiving the same moisture treatment was 44.4% and 1.2% when dried at 110°C and 55°C, respectively.

A combination of features in the transmission spectra obtained before cooking were used to predict if concealed damage would develop in nuts after cooking. It was observed that nuts with concealed damage have less absorbance in the oil absorption band at 930 nm and increased absorbance in the region between 700 and 750 nm. Physical properties such as mass, moisture changes during cooking, density and thickness, had very low correlations with severity of concealed damage. From near infrared transmission spectra obtained before cooking, discriminant analysis and principle component analysis was used to classify nuts as concealed damaged or normal; validation classification error rates as low as 12.4% were obtained.

A visible and near infrared LED based prototype was built to automatically detect almonds with concealed damage at commercial speeds, about 40 nuts per second. The prototype detects transmitted light through whole almonds from six different LEDs. A validation classification error rate of 20.0% was obtained with regression analysis. Most of the incorrectly classified nuts were on the border between having concealed damage or being normal. Only 2.0% of the almond samples showing little or no browning were classified as concealed damaged and 11.8% of the samples with severe concealed damage were classified as normal. Comparable results were obtained with discriminant analysis.

1. INTRODUCTION

The almond, *Prunus dulcis*, is a member of the rose family and is closely related to the peaches, plums, and apricots. In fact, the seed enclosed within the hard shell of a peach pit closely resembles an almond kernel. The almond is believed to be native to temperate desert zones of western Asia (Rosengarten, 1984). Evidence, dating back to the bronze age, of almond cultivation has been found in Greece and Turkey. Almonds were first planted in North America at Spanish missions along the southern California coast. In the mid 1800s, almonds started to be cultivated in California's San Joaquin and Sacramento Valleys. Today, 99% of the almonds grown in the United States are found in California's San Joaquin and Sacramento Valleys between Bakersfield and Red Bluff (Rosengarten, 1984). As of 1997, approximately 420,000 acres of almonds are cultivated in California and will produce an estimated 680 million pounds of nuts (Anonymous, 1997). Approximately 60% of the world almond production is grown in California (Anonymous, 1994a). There are several varieties or cultivars of almonds grown in California. The top three varieties grown in California are Nonpareil, comprising 49% of California's production, followed by Carmel, comprising 17%, and Mission, comprising 8%. All three of these varieties are preferred as cooking ingredients.

Concealed damage in almonds is defined by the industry as a browning of the kernel interior after moderate to high heat processing, such as cooking or roasting. Figure 1.1 shows an example of a normal almond and concealed damaged nut after cooking. There are no visible defects or indications of concealed damage on the exterior of the kernel before or after cooking or roasting. Bitter flavors are developed in extreme cases of concealed damage after roasting or cooking. Almonds with concealed damage are a problem for the almond industry for a variety of reasons: lower consumer acceptance of their reddish-brown internal appearance, bitter flavor, and possibly lower nutritional value due to degraded amino acids. Concealed damage does not fall into any of the traditional USDA almond defect categories such as discoloration, insect damage, etc. since the disorder cannot be identified from the exterior of the kernel. Normally, a food processor that purchases almonds will set their own specific limits on the allowable quantity of concealed damaged nuts. An almond processor sometimes has a difficult time meeting their customers' defined specifications in seasons when the incidence of concealed damage is high.

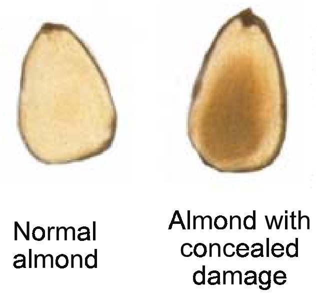


Figure 1.1. Normal almond and almond with concealed damage. Both nuts were cooked at 140°C for 60 minutes in a gravity convection oven.

The concealed damage disorder is apparently initiated when nut kernels are exposed to a warm and moist environment (Kader and Thompson, 1992 and Reil et al., 1996). Nuts are most likely to encounter conditions favorable for development of concealed damage when rain occurs during harvest. However, concealed damage may occur anytime after harvest if the kernels are exposed to a warm and moist environment. Almonds are harvested using a three step process: first, the trees are shaken to knock the nuts to the ground; second, the nuts are swept into windrows; and third, the nuts are picked up by harvesters. Usually a few days, but sometimes a few weeks, will lapse between tree shaking and nut removal from the orchard. It is usually desirable to allow the nuts to dry on the ground before sweeping into windrows (Reil et al., 1996). The disadvantage of this procedure is that the time period when the nuts are lying on the orchard floor creates an opportunity for the nuts to be exposed to rain and for concealed damage to occur. Concealed damage may become more severe if moist nuts are swept into windrows or are collected and stockpiled while wet. On the other hand, concealed damage can be minimized if nuts are removed from the orchard shortly after a rain and promptly dried. Unfortunately, it is often not feasible to get harvesting equipment into the orchard under wet conditions.

The incidence of concealed damage is very sporadic from year to year. It becomes a significant problem every three to five years, when a heavy rain occurs during harvest. Almonds containing more than 6% kernel moisture (w.b.) are dried upon delivery to the processor. During a wet harvest season, the quantity of almonds requiring drying upon delivery to the processor may exceed 20% of the total harvest and exceed the drying capacity of most almond producers. This creates an increased time when almonds are held at a high moisture content which further increases the incidence of concealed damage. Properly dried nut shipments are essentially free of concealed damage while the wet shipments that are dried immediately upon delivery to the processor typically contain 1% to 10% or more concealed damaged nuts (Stoddard, 1995).

A device to sort nuts with concealed damage would ideally be placed after the drying process, when nuts are removed from storage for final processing. Most almonds are used as ingredients in foods and are impossible to inspect after cooking. Approximately 98% of all domestically consumed almonds are shelled. Of the shelled almonds, 60% are sold un-roasted for use as ingredients in candy and other foods, 15% are roasted and salted and packaged for sale as snack food (Woodroof, 1979; Bushnell and King, 1986).

Mission almonds are more prone to concealed damage than other almond varieties. The Mission cultivar is harvested in late September or early October, after all other almond varieties are harvested. It is not known if the harvest time and increased chance of rain at harvest causes the higher incidence of concealed damage or if the reason is physiological. Nonpareil almonds are typically harvested in late August or early September, before all other almond varieties. The Carmel cultivar is harvested in the middle of the almond harvest season and may have significant concealed damage if they are exposed to rain during their harvest.

2. OBJECTIVE

The objective of this research was three-fold. First, the chemical and physical properties of almonds with concealed damage were studied as well as the environmental conditions required to induce concealed damage. Sorting for almonds with concealed damage is preferably done as the nuts come out of storage but before cooking or roasting. Nuts are commonly mixed with other foods before cooking or roasting so there is no opportunity to inspect them. For this reason, emphasis was on properties after drying as this stage provides the best opportunity for sorting, but properties before drying and after cooking were explored as well. The second objective of this study was to explore the treatment and detection of almonds with concealed damage. The effectiveness of a near infrared device to detect early signs of a browning reaction was explored as a possible method to rapidly detect almonds with concealed damage. Thirdly, different drying regimes were evaluated to determine if the concealed damage browning reaction can be inhibited though different drying operations.

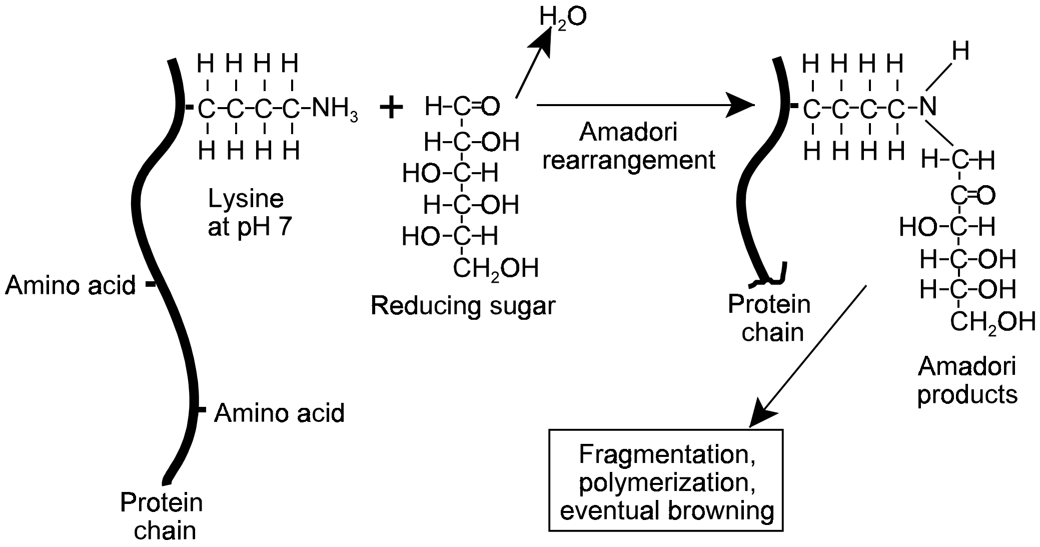
3. REVIEW OF LITERATURE

3.1 Browning Reaction Chemistry

To date, there has been no formal report on the physical and chemical properties of concealed damage in almonds in the literature. Much has been reported about causes of browning reactions in other foods exposed to warm and humid conditions resulting with similar characteristics to almonds with concealed damage. There are three basic types of browning reactions: enzymatic, non-enzymatic, and thermal decomposition. Enzymatic, or oxidative, browning is common in many fruits and vegetables such as apples and bananas. Enzymatic browning occurs if these fruits are cut open and exposed to air for a short period of time (Richardson and Hyslop, 1985). Enzymatic browning requires the presence of oxygen, copper, and some sort of damage to fruit tissue to expose the browning enzyme, phenolase, to the oxidizable tissue. Sufficient tissue damage often occurs with bruising or cutting. Thermal decomposition occurs when sugars are heated in near anhydrous conditions. High temperatures, approximately 200°C, are required to initiate the reaction with sucrose (Lee, 1983). It is not likely that enzymatic browning or thermal decomposition are responsible for concealed damage in almonds. Concealed damage is no more common in sliced almonds where the slicing would likely liberate phenolase from cells than in whole kernel product. Thermal decomposition is not likely either since browning due to concealed damage can occur at temperatures well below those required for thermal decomposition. It is hypothesized among almond industry scientists that the chemical reaction which causes concealed damage is a non-enzymatic browning reaction called the Maillard reaction (Stoddard, 1995). Non-enzymatic browning is common in many processed foods; for example, it is the reaction which causes bread crust to turn brown.

Non-enzymatic browning has been reviewed and summarized by several researchers. Hodge and Osman (1976) describe non-enzymatic browning having three stages. The initial stage is well understood but the later two stages are not. The first stage produces no recognizable physical characteristics but produces products called Amadori products. Amadori products arise from the product of an initial reaction between amine groups and carbonyl groups. The intermediate stage involves various reactions of the Amadori products which may liberate the bound amine group and form more reactive dicarbonyl compounds. The completion of the intermediate stage is indicated by a slight yellow appearance and strong UV absorption from 277 nm to 285 nm. The final stage involves amino acid condensation with intermediate stage products and aldehyde-amine polymerization to form brown pigments called melanoidins. Before brown pigments are formed, fluorescence may be observed at 425 nm when excited at 375 nm. However, fluorescence disappears upon formation of the brown pigments, making the appearance of fluorescence fleeting and difficult to observe.

The first stage of the Maillard reaction, shown in figure 3.1.1, begins with the condensation of an amine and carbonyl group of an aldehyde or ketone to form what is called a carbonylamine. Usually, the aldehyde or ketone is in the form of a reducing sugar but it can also arise from the oxidation of lipids. Normally this initial reaction is the slowest because the aldehyde or ketone must be in their acyclic forms. The concentration of a reducing sugar in its acyclic form is usually less than 1% of the total monosaccharide concentration. Labuza and Baisier (1992) report that the rate of condensation of the protein, hemoglobin, with glucose is 0.6×10^{-3} $(\mathrm{mM \cdot hr})^{-1}$. The amine group of all free (non-protein) amino acids are highly susceptible to react with an aldehyde or ketone. Protein bound amino acids need to have an amine group on their side chain to be involved in a Maillard reaction. For this reason, lysine is most likely to be involved in the Maillard reaction (Rhee et al., 1981). Arginine and histidine are also susceptible to degradation from the Maillard reaction although at a lesser degree than lysine (Whistler and Daniel, 1985). In almonds, 99% of the amino acids are stored in protein (Soler et al., 1989). Arginine, histidine and lysine together comprise 16% of all total amino acids in almonds (Soler et al., 1989).



3.1.1. The initial stage of the Maillard reaction.

The initial carbonylamine, formed by the condensation of an amine and carbonyl group, undergoes what is called an Amadori rearrangement to finally form Amadori products. Amadori products are more stable than any of the earlier reaction products and their formation is rapid compared to carbonylamine formation (Labuza and Baisier, 1992). Before Amadori product formation, the reaction might be partially reversed before Amadori products are formed, but only under extremely acidic conditions (Cheftel et al., 1985). Amadori products are fairly stable in moist and acidic environments. For the most part, their formation is irreversible. Amadori products are less stable at high temperatures and browning will occur upon heating a food with significant accumulated Amadori product.

The intermediate stage of the Maillard reaction has a variety of pathways. When Amadori products are heated, they can lose one or more water molecules, fragment, or combine with another amino acid. The progression through any of these pathways depends on the pH of the system (Eskin, 1990). Most of the color is produced in the final stage of the Maillard reaction. Here, intermediate stage products may bind with more amino acids and polymerize to form large hetrocyclic nitrogen compounds which have a brown appearance.

The Maillard reaction is highly dependent on time, temperature, and moisture. In low moisture systems, added water increases the mobility of reactants and accelerates browning. However, the browning rate reaches a maximum, and eventually decreases, with increasing moisture because the reactants become diluted. Also, since water is a product of the reaction, high water concentrations will tend to drive the reaction backwards. The browning rate is maximized when the water activity is between 0.4 and 0.8 (Leung, 1987). Lea and Hannan (1949) monitored amino group loss in casein-glucose model systems stored at various relative humilities. They showed that the maximum loss of free amino groups occurred between 0.65 and 0.7 water activity.

Generally, increases in temperature will increase the browning rate. A 10°C increase in temperature will increase the browning rate three to five times, depending on the food involved (Shallenberger, 1974). Eichner et al. (1985) investigated non-enzymatic browning in carrots during drying at temperatures of 110°C, 90°C, and 60°C. All carrots were dried to 7% moisture content and most brown pigments formed after the moisture content dropped below 20%. Browning, as measured by light absorption at 420 nm, was most severe when the drying temperature was 110°C, the highest temperature studied. Accumulation of Amadori product, measured by HPLC, was highest at 110°C also. No browning and little Amadori product accumulated when the carrots were dried at 60°C. They suggested that most browning could be avoided if the drying temperature was simply reduced during the later stages of drying, after the moisture content dropped below 20%.

There have been many studies of a Maillard type reaction in soybeans during prolonged storage. Like almonds, mature soybeans do not contain reducing monosaccharides but do contain small amounts of sucrose and oligosaccharides. When mature soybeans are soaked in water at room temperature for 15 hours, 8% of the bean's oligosaccharides and sucrose disappeared (Rackis, 1978), indicating the onset of germination. Wettlaufer and

Leopold (1991) monitored Amadori product and fluorescent Maillard products in soybeans when stored in warm, humid environments. Fluorescent Maillard products, typical of the final stage, first appeared in soybeans held at 40°C, 100% relative humidity after ten days. Amadori product appeared in soybeans held at 30°C and 75% relative humidity after 7 days. However, fluorescent Maillard products never appeared in soybeans held at these cooler conditions for 28 days.

Similar symptoms of concealed damage have been observed and studied in many other seeds that were subjected to warm and moist environments (Priestly, 1986). In wheat germ, McDonald and Milner (1954) observed a browning reaction accompanied by decreases in protein, indicating non-enzymatic browning. As mentioned earlier, Wettlaufer and Leopold (1991) observed marked increases in Amadori product and fluorescent Maillard reaction products in soybeans after exposure to warm and moist conditions. In the field of seed technology, products of non-enzymatic browning are used to indicate seed deterioration after long periods of storage or after improper storage. These parameters can then be used to predict seed germination rates and vitality.

Interior browning of macadamia nut kernels associated with moisture exposure at harvest time was studied by Prichavudhi and Yamamoto (1965). Increased reducing sugars in macadamia nuts were observed after exposure to moisture. Furthermore, reducing sugar concentration was found to be higher in the center of the kernel, where the darker browning occurs, than the lighter colored outer layer. It was also found that browning increased with higher drying temperatures. When macadamia nuts were dried with ambient air or air heated to 52°C it was found that 15% and 17% of the nuts respectively had brown centers. In contrast, when nuts were dried at 60°C or 71°C, 100% of the nuts had brown centers. The concealed damage like symptoms in macadamia nuts were also found to be due, at least in part, to enzymatic activity. No browning appeared in freshly harvested blanched samples of nuts later exposed to moisture, drying, and roasting treatments while significant browning appeared in control nuts that were not blanched.

In almonds, Reil et al. (1996) noted significant increases in reducing sugars and concealed damage after exposure to simulated rainfall. Like most other seeds during a normal harvest, there are no reducing sugars in almond kernels. In the case of cereal grains, glucose can be formed by enzymatic hydrolysis of starch. In the case of pulses, enzymatic catalyzed oxidation of lipids can form aldehydes that react with amino acids just as reducing sugars do (Eskin, 1990). Lipid conversion generally occurs faster than starch conversion in seeds (Bushuk and Lee, 1978) and enzymatic lipid oxidation increases linearly with water activity. Since approximately 50% of the almond weight is due to lipids, this may be the most significant source of reactants resulting in concealed damage. Almonds do contain approximately 5.8% sucrose (Soler et al., 1989) and some of the sucrose may hydrolyze when an almond takes in moisture. However, Maillard browning from sucrose systems will be limited because sucrose hydrolysis decreases in basic conditions. The initial carbon - amine reaction requires the amine group to be protonated. This is more likely in basic conditions. Also, reducing sugar concentration in their open chain form is increased in basic environments. For these reasons, sucrose is often used in food systems where it is desired to avoid Maillard browning.

Almonds in the field probably do not encounter enough heat to carry the Maillard reaction through the final stage. From literature describing the Maillard reaction in other seeds, it seems feasible that after wet nuts from the field are dried, the first stage of the Maillard reaction progresses through with Amadori product production. Possibly the intermediate stage begins in the dryer with fragmentation of Amadori products into dicarbonyl compounds. After a rain, almonds in windrows may be exposed to continuous moist conditions at temperatures between 20°C and 35°C for several days. Under these conditions, the increased moisture provides mobility for the Maillard reaction to begin. If nuts undergo the beginnings of the first stage of the reaction in the orchard, then drying can carry the reaction through to the beginning of the intermediate stage. The final stages will be completed during roasting or cooking because the high temperatures will drive the reaction at low moisture. At harvest time, almonds on the tree typically will have a kernel moisture content of 5% to 7% dry basis (Soler et al., 1989). Almonds received by the processor containing higher moisture contents are dried in either a batch dryer at about 54°C or a continuous flow dryer with a temperature range between 93°C and 121°C (Thompson et al., 1996). Almonds at various processors are subjected to different drying regimes. Thus, the progression of the Maillard reaction in almonds during drying may vary widely from processor to processor and from dryer to dryer. It is unknown, but possible, that different drying processes have an effect on eventual extent of concealed damage.

Maillard browning can be controlled by chemical and physical treatments, none of which the almond industry utilizes. Maillard browning is controlled in egg and potato products by adding glucose oxidase, an enzyme that causes destruction of glucose. Sulfites may be introduced to foods as a gas or in solution to bind to the carbonyl group and prevent it from reacting with an amino acid. Browning may be prevented if foods are dried slowly but completely before exposure to high temperatures associated with cooking or frying. Almonds are typically dried down to 6% moisture, then stored, then cooked or roasted. Normally, brown pigments in almonds will not form during the drying process. Browning after cooking or roasting might be prevented if wet almonds are initially dried to a lower moisture than the current standard of 6% w.b. with a low temperature operation as was discussed by Eichner et al. (1985).

As mentioned earlier, there has been no formal investigation of the physical and chemical properties of almonds with concealed damage. However, there has been extensive research on almond nutrition after several processing stages as shown in table 3.1.1. As can be seen from table 3.1.1, there are no significant differences in sugar content, basic amino acid content, protein content or oil content due to heat processing. These tests were presumably performed on normal almonds. If concealed damaged nuts were included in the samples, the amount of concealed damaged nuts would be likely be relatively low, so chemical changes in the concealed damaged nuts might not show up in these results.

Table 3.1.1. Almond nutrition data at various processing stages.

	whole, natural (anonymous, 1995a)	whole, natural (Broughton, 1996)	whole, natural (Scherz and Kloos, 1981)	whole, natural (anonymous, 1984)	whole, natural (Soler et al., 1989)	blanched, whole (anonymous, 1995a)	oil roasted, whole (anonymous, 1995a)	dry roasted, whole (anonymous, 1995a)	oil roasted, blanched, whole (anonymous, 1995a)	dry roasted, blanched, whole (anonymous, 1995a)
sugar (g)	2.0	-	-	4.4	5.0	4.0	3.0	4.0	3.0	3.0
lysine (mg)	688	667	-	468		680	510	-	-	-
histidine (mg)	522	558	-	477		510	560	-	-	-
arginine (mg)	2570	2495	-	2270		2500	2560	-	-	-
protein (g)	20.7	19.9	20.0	20.0		21.1	22.3	21.3	20.7	21.1
oleic acid (g)	31.6	-	36.5	-	-	-	33.07	33.0	31.7	
linoleic acid (g)	13.1	-	9.86	-	-	-	13.23	12.9	13.8	
linolenic acid (g)	-	-	0.26	-	-	-	-	0.05	-	-
total fat (g)	50.4	52.2	52.2	-		50.6	58.0	53.2	57.5	56.9
polyunsaturated fat (g)	11.6	10.9	-	-	-	-	13.7	9.3	-	9.6

Note: all quantities are per 100g of almond dry matter.

3.2 Quantification of Maillard reaction products

Quantification of the extent of the early Maillard reaction, well before browning occurs, is important in the nutrition sciences because lysine bound to a sugar likely won't be nutritionally available. Over the past three decades several chemical and microbiological tests have been developed to determine the concentration of nutritionally available lysine in foods. It has been found that most of the protein bound lysine that is bound to an aldehyde is not nutritionally available with normal digestion by humans (Cheftel et al., 1985). In the course of developing rapid chemical methods to determine nutritionally available lysine, the term "reactive lysine" has been introduced (Hurrel et al., 1979). Reactive lysine is defined as lysine with an unbound open side chain. After an aldehyde reacts with lysine, the reaction rapidly progresses to a Schiff's base and Amadori product. Thus, there is usually a good correlation between reactive lysine, available lysine and the extent of the early stage of the Maillard reaction.

The application of several tests to determine reactive lysine after a Maillard reaction were reviewed by Hurrell and Carpenter (1981). Most tests to determine reactive lysine involve dye binding to the open side chain of the lysine molecule. However, all of the dyes used can, to a limited extent, bind to Amadori products as well. This results in underestimating the extent of the early Maillard reaction and overestimating reactive lysine. To account for this problem when the Maillard reaction is involved, Hurrel et al. (1979) developed a two step dye binding procedure that first measures lysine as well as everything else that the dye binds to. In the second step, reactive lysine is bound with propionic anhydride, a compound that very specifically binds to reactive lysine and prevents dye from binding with lysine. The dye binding procedure is applied again to the propionic anhydride treated sample. This measures everything that the dye binds to except the reactive lysine. The difference in the first measurement and second measurement results in total reactive lysine. This procedure has been tested on fish, meat, milk, peanut, and beans and compares favorably with the more traditional but complicated procedure of fluorodinitrobenzene (FDNB) dye binding as described by Booth (1971).

The final products of the Maillard reaction have been monitored and quantified by several methods. The most common method, degree of browning, simply measures light absorption at 420 nm or 490 nm after a heat treatment. Degree of browning has been shown to be highly correlated with the accumulation of early Maillard products or reactive lysine (Labuza and Baisier, 1992 and Pokorny, 1981). Thus, degree of browning can be used to determine the effectiveness of a treatment to prevent browning. For example, if a treatment is devised to inhibit the initial stage of the Maillard reaction, then the treated and control samples can be cooked under the same conditions and degree of browning compared. This can be less time consuming and more indicative of overall effectiveness than measuring early Maillard products by chemical methods.

Another method to quantify the extent of the Maillard reaction is fluorescence measurement at 440 nm when excited by 370 nm light. This was used to quantify Maillard products in soybeans by Wettlaufer and Leopold (1991). Adhikari and Tappel (1973) observed a 500% increase in fluorescence in glucose-glycine model systems heated 10 minutes at 100°C. They also observed increases in fluorescence in coffee after roasting and white bread after toasting. The fluorescent substances appear before the brown pigments, are not identical to the brown pigments, and diminish as brown pigments increase. For these reasons, fluorescence is not commonly used to quantify the extent of the Maillard reaction in food systems (Labuza and Basier, 1992).

As can be seen on figure 3.1.1, there is an increased number of N-C bonds during the initial stage of Maillard browning. This increase is due to formation of Amadori products and Schiff's base. These two browning compounds are likely to be found in concealed damaged almonds after drying. There are several absorption bands of N-C bonds in the near infrared spectrum (Murray and Williams, 1987). The absorption bands are listed below in table 3.2.2 This seems to be a promising method to detect early Maillard reaction products. However, there is nothing reported in the literature indicating the feasibility of this method.

Table 3.2.2. Absorption bands of N-C bonds in the near infrared region

wavelength (nm)	overtone	note
1160	III	isolated
1500	I	N-H interference
1520	III	isolated
1555	II	N-H interference
1785	III	isolated
1855	III	isolated
1925	III	isolated
2035	II	isolated
2330	I	isolated

3.3 Fatty acid quality tests

Unsaturated fatty acids, or those containing double carbon bonds, are much more likely to oxidize than saturated fatty acids. During the oxidation process, a portion of the fatty acid can break off, forming a smaller fatty acid, with one less double carbon bond, and a smaller hydrocarbon molecule. For a specific type of food, the ratio of saturated fatty acids to unsaturated fatty acids is fairly constant before oxidation begins. As can be seen from table 3.1.1, most of the fatty acids in almond oil are oleic acid, which has one double bond, and linoleic acid, which has two double bonds. There are several methods to quantify the number of double bonds, or changes in double bonds, in a sample of oil from foods. Some of the common methods are: Fourier transform infrared spectroscopy, gas chromatography, iodine value, and refractive index (Pike, 1994). Of these methods, gas chromatography and the iodine method can measure the actual number of double bonds in a sample while refractive index is correlated to the number of double bonds in a sample (Hunt et al., 1951). Refractive index can not be used to compare the oxidation level of oil samples extracted from different types of foods, but only from the same type of food (Pomeranz and Meloan, 1994). For example, refractive index can be used to compare oxidation in oils from different almond samples but not between almonds and pistachio nuts because the fatty acid contents of these nuts are different. However, refractive index is much easier and faster than gas chromatography or iodine value methods.

3.4 Real-time detection of food quality

The almond industry currently utilizes a variety of high speed opto-electronic devices to perform automated quality control inspections on individual nut kernels. There are devices which use X-rays to separate rocks, glass, steel or other high density foreign matter from almond kernels. However, most devices use reflected light to inspect the surface condition of the nut. Visible, UV, and near infrared detectors are used to detect surface defects such as skin discolorations, broken skin, insect damage, or embedded shell fragments. However, none of these reflectance or x-ray based devices can detect nuts with concealed damage.

Development of rapid techniques for measuring food constituents is a growing field. Much has been reported in the literature about measurement of proteins, amino acids, lipids and sugars with near infrared reflectance and transmission spectroscopy. For example, Rubenthaler and Bruinsma (1978) found that lysine could be accurately detected in ground samples of wheat using near infrared reflectance. Letellier and Cuq (1991) developed a near infrared reflectance system to determine reactive lysine in wheat flour after heating. Using only four wavelengths, a coefficient of correlation of 0.98 was obtained with reactive lysine dye-binding results. The wavelengths used to obtain this correlation coefficient were 2348 nm, 1940 nm, 1759 nm, and 1680 nm. Delwiche (1995) recently developed a near infrared (850 nm to 1050 nm) transmittance method to detect protein content in individual wheat kernels. The predicted protein content had a correlation of 0.85 to 0.93 to the actual protein content. Random orientation of the wheat kernel created the largest source of prediction error. Orman and Schumann (1992) developed a method of determining oil content in single kernels of maize by near infrared (850 nm to 1050 nm) transmission spectroscopy. The standard error of prediction was only 1.2% and the error was not correlated with seed thickness. Lastly, Lamb and Hurburgh (1991) determined moisture content in single soybeans with near infrared light transmittance.

To date, there have been few developments reported in the literature using a whole spectrum method of data calibration in a real-time, high throughput, situation. This is likely to change with the relatively recent developments in low cost grating type spectrometers using diode array or CCD sensor elements. These devices enable moderately high wavelength resolution (less than 1 nm) and rapid data collection. Depending on the amount of light incident upon the detector, a whole spectrum comprising 1024 points can be collected in 5ms with a sampling frequency of 200kHz. Higher sampling frequencies can be used if the light is available to maintain a sufficient signal to noise ratio. There are two considerations to be made when choosing a detector for rapid spectroscopic measurements of transmitted light. One is response time and the other is signal to noise ratio in low light situations. For low light level applications, charged coupled devices (CCD) are preferred in the visible and near infrared region up to about 1050 nm. At longer wavelengths their sensitivity diminishes. In the region between 1100 nm and about 1800 nm, InGaAs diode arrays are preferred. The digitization, data filtering, deconvolution, and prediction of chemical concentration could be performed concurrently with the data collection from a sensor array. Thus, a predicted chemical quantity can be output nearly at the instant the spectral data is collected. Pearson (1996) used such a concurrent processing scheme in a digital image processing application with a line scan CCD camera being sampled at 200kHz.

3.5 Interaction of light with matter

Molecules in matter constantly vibrate at frequencies corresponding to light frequencies in the infrared region of the electromagnetic spectrum. When a molecule absorbs light, its vibrational, rotational, or electronic energy will increase by the amount of energy within the absorbed photon. The electronic absorptions occur in the UV and visible portion of the spectrum while absorptions due to vibrational and rotational energy occur in the infrared region. Vibrational motion has two different modes. The first, stretching, is a movement of atoms away and towards each other along a line such as an oscillating spring. The second mode, bending, is a movement of an atom or group of atoms that causes a change in bond angle. This can be in the form of twisting, wagging, scissoring, or rocking. Rotational energy refers to molecular spinning about its axis. There are many modes which light can be absorbed by matter. For a nonlinear molecule, which comprises most food molecules, there are *3n-6* modes in which light can be absorbed, where *n* is the number of atoms in the molecule.

Molecules will only absorb light at frequencies that correspond to their own vibrations or rotations. When this occurs, the energy level of the molecule is raised to a higher level. If the absorbed light is of a sufficiently high energy, the molecule's excited energy state will be raised two or three levels. If light of wavelength "X" in the infrared region raises a molecule's energy level from the ground state to the next highest, then light of wavelength 1/2X will raise the energy state two levels and light of wavelength 1/3X will raise the energy state three levels and so on. The light at wavelength X is referred to as the fundamental absorption band. The light at wavelengths of approximately 1/2X and 1/3X are called the first and second overtones, respectively. The actual wavelengths of the overtones deviate slightly from the integer multiples of the fundamental absorption band because the vibration of covalent bonds does not ideally adhere to Hooke's law (Murray and Williams, 1987). The vibrations are not confined to simple harmonic motion. Absorption at fundamental frequencies is often very high and light at these frequencies will not transmit through thick media, such as a whole almond or most other whole foods (Murray and Williams, 1987). Food matter is mainly comprised of carbon, oxygen, hydrogen, nitrogen, phosphorus, sulfur and other trace elements. Most of the overtones of bonds between the main food elements occur in the near infrared region, between 1000 nm and 2500 nm.

Light absorption is commonly modeled with the Beer-Lambert law (Murray and Williams, 1987). Referring to figure 3.5.1 the light intensity *I* decreases due to absorption and scattering as it travels through a medium as shown in equation 3.5.1

$$dI = -KIdx - SIdx + SJdx \qquad (3.5.1).$$

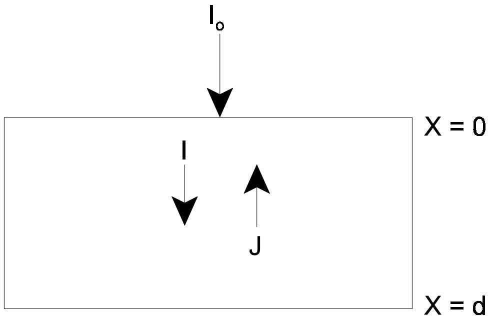


Figure 3.5.1. Schematic for light absorption and scattering (Birth and Hecht, 1987).

where *K* is the absorption coefficient of the media at location *x*, *I* is the light intensity at location *x* where *x* is the location inside the medium parallel to the incident ray I_o, *S* is the scattering coefficient, and *J* is the scattered light intensity. The first term of equation 3.5.1, *KIdx*, is due to absorption, the second term, *SIdx*, is due to scattering, and the third term, *SJdx*, is due to scattered light returning back to the same direction as I_o. Similar to equation 3.5.1, the differential equation for the scattered light is given by equation 3.5.2

$$dJ = -KJdx - SJdx + SIdx \qquad (3.5.2).$$

Referring back to figure 3.5.1, the boundary conditions for equations 3.5.1 and 3.5.2 are $I = I_o$ at $x=0$, and $J=0$ at $x=d$ (Birth and Hecht, 1987). The solution to the two differential equations to get the light intensity, I_d, emerging in the same direction as I_o from the media is given by equation 3.5.3

$$\frac{I_d}{I_o} = \frac{4\beta}{(1+\beta)^2 e^{\int_0^d \gamma dx} - (1-\beta)^2 e^{\int_0^d \gamma dx}} \qquad (3.5.3)$$

where

$$\gamma = \sqrt{K(K+2S)} \qquad (3.5.4)$$

$$\beta = \sqrt{K/(K+2S)} \qquad (3.5.5).$$

For non-scattering media $S=0$, $\gamma=K$ and $\beta=1$. It is usually assumed that γ is independent of x. The absorption coefficient, *K*, is the product the molecular extinction coefficient, *k*, and the chemical concentration, c, in the transmitting media. The molecular extinction coefficient is a function dependent on the type of molecules in the transmitting media and the frequency of the light, I_o. These assumptions lead to a simplified and commonly used form of the Beer-Lambert Law shown in equation 3.5.6 and 3.5.7 (Murray and Williams, 1987)

$$\frac{I_d}{I_o} = e^{-kcd} \qquad (3.5.6)$$

$$\text{absorbance} = \log\left(\frac{I_o}{I_d}\right) = kcd \qquad (3.5.7).$$

The value I_d/I_o is commonly called transmittance and the value $\log(I_o/I_d)$ is commonly referred to as absorbance. In non-scattering samples, absorbance is more commonly used than transmittance because of the linear relation with concentration, c. If a sample is known to be pure or without interfering compounds, the calibration of absorbance to concentration is relatively straight forward with simple linear regression. However, when light is transmitted through a whole food, such as an almond, there are many problems that must be overcome. When light is incident upon a whole food, some is reflected at the surface (specular reflectance), some light penetrates the surface but is reflected back towards the light source at an oblique angle (diffuse reflectance), while other light is absorbed, and other light is transmitted. The proportion of light that is reflected varies depending on the surface conditions. Light that is transmitted is subject to varying degrees of scattering due to different water contents, oil contents, pore sizes, refractive index, crystalline forms and sample shape (Murray and Williams, 1987). This makes it difficult to ascertain the path length the light traveled when transmitting through the sample. In the near infrared region of the electromagnetic spectrum, overlapping absorption overtones from other molecules occurring in varying concentrations make it nearly impossible to determine a chemical concentration without statistical treatment. Nevertheless, the near infrared region is desirable for quantitative analysis in foods because the absorption bands in this region are primarily due to second and third overtones and there are low cost sensors readily available for this region. Light absorption in the near infrared region is low enough that light can be transmitted through a sample, such as a whole almond, while the absorptions are specific enough to predict chemical concentration using appropriate mathematical data treatments. Another advantage of near infrared spectroscopy is that the photodetectors, currently available for this region of the spectrum, have high signal to ratios compared to detectors for other regions of the spectrum.

3.6 Multivariate calibration techniques

A calibration using absorptions at multiple wavelengths is usually required to quantify the chemical concentration in food. If too few absorption factors are used, some interferences will remain uncorrected and the correlation may be poor. However, if too many factors are used, the prediction will start modeling noise in the calibration data set (Martens and Naes, 1987). Deconvolution of the spectra is often computed to overcome the problems of overlapping absorbance peaks and shifting baselines. The second derivative is a common and rapid method of deconvoluting a spectrum. Derivatives can also reduce the effect of spectral shifts due to path length or scattering. However, derivatives are very sensitive to noise, becoming more sensitive as the derivative order is increased. Often the spectral data must be low pass filtered before deconvoluting. Nevertheless, modern spectrometers have sufficient signal to noise ratio that the second derivative can be reliably used.

There are many procedures available to determine the absorption factors needed to develop a prediction equation. The most common methods are stepwise multiple linear regression (SMR), partial least squares regression (PLS), and principle component regression (PCR). All of these methods utilize different methods to select variables from the spectra or transform the spectra into factors to develop a prediction equation for sample chemical concentrations. Spectra usually contain large numbers of data points, 1000 to 2000 points are not uncommon, and high degrees of multicollinearity are often present in spectral data. These two characteristics of spectral data make it difficult to select the best features within the spectra to use in a prediction equation.

Partial least squares regression has gained recent popularity because it is suitable for use with smaller sample sizes and is more robust in the presence of experimental noise in the spectra and chemical data (Martens and Naes, 1987). Like PCR, the PLS method transforms the data to orthogonal linear combinations of the entire spectrum. In PCR, the transformation is independent of the chemical concentration in such a way that the first factor describes the greatest amount of variance in the spectra. The transformation in PLS links the chemical data and spectra together during the transformation. The PLS transformation describes the variance in absorbance data that are relevant to the chemical data variations. Because of light scatter or other chemicals of variable concentrations, the absorbance spectra may contain a high level of variance that is irrelevant to the prediction of chemical concentrations. The PLS procedure takes measures to ignore these irrelevant sources of variance, even though they may be the dominate sources in variance of the absorbance spectra (Haaland and Thomas, 1988). Several PLS algorithms have been developed. PLS1 was developed to relate one chemical variable to the absorbance data. PLS2 was developed to relate more than one chemical variable to the absorbance data. An advantage of SMR over PLS and PCR is that the whole spectrum does not have to be acquired to perform a prediction of chemical concentration after the calibration is performed.

In addition to these methods which in one way or another involve regression, discriminant analysis techniques can be used to classify samples into discrete groups (Mark and Tunnell, 1985). A discrete classification procedure may be more appropriate than regression procedures when it is desired to classify a sample into one of a few different categories. Discriminant analysis has been widely used for this purpose in the field of image processing. However, as with the regression procedures, the problem of selecting which variables to use is complicated by multicollinearity in the data set.

Equation 3.6.1 shows a general model using stepwise multiple regression to determine the concentration of a chemical from a spectrum

$$\boldsymbol{c} = \boldsymbol{Ap} + \boldsymbol{e}_c \qquad (3.6.1).$$

Letting *m* be the number of calibration samples and *n* being the number of spectral variables (wavelengths), ***c*** is a *m x 1* vector of chemical concentrations, ***A*** is a *m x n* matrix of calibration spectra including transformations such as second derivative or ratios of absorbance values, ***p*** is a *n x 1* vector of unknown calibration coefficients, and $\boldsymbol{e}_c$ is the *m x 1* error term. The number of spectral variables, *n*, is reduced as much as possible by stepwise selection procedures. The most popular of these procedures performs a general F test to accept or delete a variable in the model. Once the number of variables are settled upon, the solution for ***p*** that minimizes the error term is given by equation 3.6.2

$$\boldsymbol{p} = (\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t\boldsymbol{c} \qquad (3.6.2).$$

The computation of these matrix equations are straightforward and most statistical packages perform these computations almost automatically. The difficult aspect of SMR is choosing the variables in the ***A*** matrix that will give the best prediction equation. The procedure for stepwise variable selection is as follows. The model starts with the single variable that has the highest correlation with the chemical data. Next, the F statistic is computed for all variables not in the model for their individual contribution to the model. The variable with the highest F statistic is added to the model as long as the F statistic exceeds a preset threshold. Next, individual contribution of the variables already in the model are checked. If a variable already in the model does not produce a significant F statistic, it will be removed from the model. The procedure keeps checking F statistics to add or delete variables from the model and quits when no variables in the data set can be added or deleted. Multicollinearity can prevent the stepwise procedure from picking the best set of variables for the prediction equation. The purpose of the PCR or PLS routines is to eliminate multicollinearity so that variable selection will be more straightforward.

The model for PLS and PCR procedures, given by equation 3.6.3, is similar to the SMR model given by equation 3.6.1

$$\boldsymbol{c} = \boldsymbol{Tp} + \boldsymbol{e}_c \qquad (3.6.3)$$

where $\boldsymbol{c}$, $\boldsymbol{p}$ and $\boldsymbol{e}_c$ are the same as with the SMR procedure. Letting h be the number of scores, or transformed spectrum factors, $\boldsymbol{T}$ is a m x h matrix of the transformed spectral data. Again, the number, h, of variables used in the model is minimized with a stepwise selection or similar technique. In PCR analysis, the linear transformation of $\boldsymbol{A}$ is given by the eigenvector of the covariance matrix of $\boldsymbol{A}$. Letting $\boldsymbol{v}_j$ be the j'th eigenvector of the covariance matrix of $\boldsymbol{A}$, then the transformed spectrum of the i'th sample with the j'th eigenvector, $\boldsymbol{t}_{i,j}$, can be computed with equation 3.6.4

$$\boldsymbol{t}_{i,j} = \boldsymbol{a}_i \boldsymbol{v}_j \qquad (3.6.4)$$

where $\boldsymbol{a}_i$ is the spectrum of the i'th sample. The matrix $\boldsymbol{T}$ is the transformed spectra from all the samples and from h eigenvectors. Members $\boldsymbol{t}_{i,1}$ explain the greatest percentage of variance in the spectrum for the particular sample set. Members $\boldsymbol{t}_{i,2}$, $\boldsymbol{t}_{i,3}$ and so forth explain less variance. If the matrix, $\boldsymbol{T}$, contains as many columns, n, as there were variables in the original spectrum, then 100% of the variance in the original spectra can be explained. A benefit of PCR is that often less than twenty variables will explain nearly 99% of the variance of the original spectra that might contain 2000 or more absorbance values. Since the members of matrix $\boldsymbol{T}$ are independent, stepwise multiple regression can be preformed on this set of data without problems associated with multicollinearity.

The computation of the transformed spectrum with PLS is more complex than with PCR. It involves a series of least squares solutions and is performed iteratively in order to compute all the variables desired. The first pass computes a score that maximizes both the variance explained in matrix $\boldsymbol{A}$ and correlation with $\boldsymbol{c}$. The second pass replaces $\boldsymbol{A}$ with the error term in the final regression model, $\boldsymbol{E}_A$. The error term is modeled again and again until the desired number of terms is reached. The scores are independent of each other because, by nature of regression, the error terms are independent. The PLS algorithm is explained in detail by Haaland and Thomas (1988) as well as Martens and

Naes (1987) and only briefly described here. First, $\boldsymbol{A}$ and $\boldsymbol{c}$ are centered by subtracting them from their means to form $\boldsymbol{A_m}$ and $\boldsymbol{c_m}$. A weight loading vector, $\boldsymbol{w}_j^t$, for the *i*'th factor is computed using the least squares solution of equation 3.6.5

$$\boldsymbol{A}_m = \boldsymbol{c}_m\boldsymbol{w}_j^t + \boldsymbol{E}_A \qquad (3.6.5).$$

A score vector, $\boldsymbol{t}_j$, is similarly computed by another least squares solution of equation 3.6.6

$$\boldsymbol{A}_m = \boldsymbol{t}_j\boldsymbol{w}_j^t + \boldsymbol{E}_A \qquad (3.6.6).$$

Another score vector $\boldsymbol{v}_j$ is created to relate $\boldsymbol{t}_j$ to the chemical concentrations with a least squares solution to equation 3.6.7

$$\boldsymbol{c}_m = \boldsymbol{t}_j\boldsymbol{v}_j + \boldsymbol{e}_c \qquad (3.6.7).$$

Vector, $\boldsymbol{b}_j$, is the PLS loading vector for matrix $\boldsymbol{A}$. It is computed with a least squares solution to equation 3.6.8 and the error terms are computed with equations 3.6.9 and 3.6.10

$$\boldsymbol{A}_m = \boldsymbol{t}_j\boldsymbol{b}_j^t + \boldsymbol{E}_A \qquad (3.6.8)$$

$$\boldsymbol{E}_A = \boldsymbol{A}_m - \boldsymbol{t}_j\boldsymbol{b}_j^t \qquad (3.6.9)$$

$$\boldsymbol{e}_c = \boldsymbol{c} - \boldsymbol{v}_j\boldsymbol{t}_j \qquad (3.6.10).$$

At this point, $\boldsymbol{E}_A$ is substituted for $\boldsymbol{A}$, $\boldsymbol{e}_c$ for $\boldsymbol{c}$, *j* is incremented, and the above procedure is repeated until the desired number of loading factors is reached. Letting $\boldsymbol{W}$ and $\boldsymbol{B}$ be matrices composed of *j* rows containing $\boldsymbol{w}_j$ and $\boldsymbol{b}_j$ respectively, a $\boldsymbol{v}$ is formed from the individual $\boldsymbol{v}_j$ terms, the PLS regression coefficients, $\boldsymbol{r}$, are computed with equation 3.6.11

$$\boldsymbol{r}= \boldsymbol{W}^{t}(\boldsymbol{B}\boldsymbol{W}^{t})^{-1}\boldsymbol{v} \qquad (3.6.11)$$

and finally, the predicted chemical concentration is computed with equation 3.6.12

$$\hat{c}= \boldsymbol{a}^{t}\boldsymbol{r} + c_{o} \qquad (3.6.12)$$

where c_o is the average chemical concentration of the calibration samples. Like PCR, the PLS procedure vastly reduces the number of variables needed for calibration as compared to the original spectrum. The advantages of PLS over PCR is that it will usually yield equivalent or superior prediction results with fewer variables than PCR (Martens and Naes, 1987).

Discriminant analysis differs from the previously discussed regression procedures in that it is used to classify samples into one of two or more discrete groups. In discriminant analysis, a sample is classified into a group based on a concept called the generalized distance, or distance between the sample and group centroid. The smaller the generalized distance between a sample and the group centroid, the more likely that the sample belongs to that group. Referring to figure 3.6.1, the Euclidian distance between two points, A and B, in two dimensional space can be computed by the Pythagorean theorem as shown in equation 3.6.12

$$d_{AB}^{2} = \left(A_x - B_x\right)^2 + \left(A_y - B_y\right)^2 \qquad (3.6.12).$$

In vector notation, the distance between two points is given by equation 3.6.13

$$d_{AB}^{2} = \left(\boldsymbol{A} - \boldsymbol{B}\right)^{t}\left(\boldsymbol{A} - \boldsymbol{B}\right) \qquad (3.6.13)$$

where ***A*** and ***B*** are vectors from the origin to points A and B, respectively. In two dimensional space, both ***A*** and ***B*** are 2 × 1 column vectors. More generally, in *p*-variate space, both ***A*** and ***B*** would be *p* × 1 vectors.

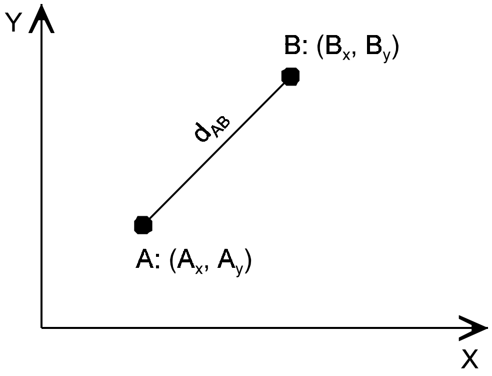


Figure 3.6.1. Distance between two points in a plane.

The generalized distance accounts for the variance of each group. As can be seen from the one dimensional example in figure 3.6.2, the Euclidian distance between the sample, $x=S$ and mean of group A is less the distance from S to the mean of group B. However, the probability of sample S belonging to group B is higher than for group A due to the variance associated with each group.

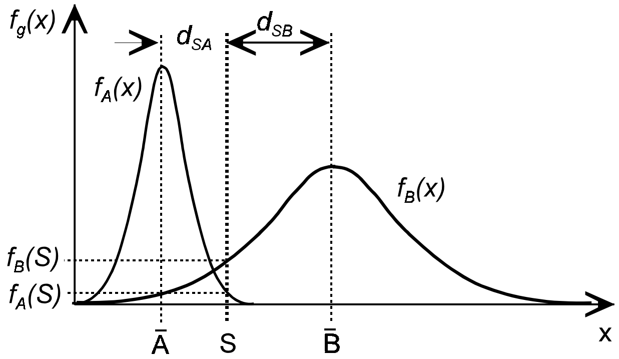


Figure 3.6.2. One dimensional example of generalized distance between two groups.

In contrast to the computation of the Euclidian distance, the covariance matrix, ***S***, is used in the computation of the generalized distance. Equation 3.6.14 is used to compute the generalized distance between a sample, defined by vector $\boldsymbol{X}$, and group centroid, defined by vector $\boldsymbol{X}_A$ (Huberty, 1994)

$$D_{XA}^{2} = \left(\boldsymbol{X} - \boldsymbol{X}_{A}\right)^{t} \boldsymbol{S}^{-1} \left(\boldsymbol{X} - \boldsymbol{X}_{A}\right) \qquad (3.6.14)$$

where

D_A^2 is called the generalized distance from sample X to the centroid of group A,

$\boldsymbol{X}_A$ is a vector describing the location of the centroid of group A,

$\boldsymbol{X}$ is a vector describing the location of the sample,

$\boldsymbol{S}$ is the covariance matrix of group A. The covariance matricies of all groups are pooled if they are equal.

Samples can be classified into groups based on their generalized distance to each group. For example, if the generalized distance from a sample is computed to two different groups, A and B, then the sample would be classified into group A if $D_A < D_B$. However in practice, the probability, *P(A|X)*, of sample X being a member of group A is used for classification. The estimated *p*-variate normal probability density function is given by equation 3.6.15

$$f(X|A) = \frac{1}{\sqrt{(2\pi)^{p}}\sqrt{|S_{A}|}} \exp\left[-\tfrac{1}{2}\left(\boldsymbol{X} - \boldsymbol{X}_{A}\right)^{t} S^{-1} \left(\boldsymbol{X} - \boldsymbol{X}_{A}\right)\right] \qquad (3.6.15)$$

where $\boldsymbol{X}_A$ is a $p \times 1$ vector describing the location of the mean of samples known to belong to group A, and $\boldsymbol{S}_A$ is a $p \times p$ covariance matrix for samples known to belong to group *g*. The values of $\boldsymbol{X}_A$ and $\boldsymbol{S}_A$ are obtained from a training set of samples known to belong to group *A*. When a multivariate normal population can be assumed, the posterior probability, *P(A|X)*, to classify sample X into group *A* is given by equation 3.6.16 when there is a possibility of classification into *k* different groups

$$P(A|X) = \frac{q_A f(X|A)}{\sum_{g=1}^{k} q_g f(X|g)} \qquad (3.6.16)$$

The factor, q_g, is called the a priori probability that a sample is a member of group g without knowing anything about a particular sample. The values of the a priori probabilities are obtained from a training set or other prior knowledge of the population distributions. Classification using a posteriori probabilities is performed by computing the a posteriori probability for membership of all groups and classifying the sample into the group having the largest a posteriori probability.

As with the regression procedures, the difficulty of obtaining an optimal classification equation is in selecting the best, p-variate, set of variables to use. The large size of spectral data sets and multicollinearity make selection of the best set of variables difficult for the classification equation. Stepwise procedures for variable selection similar to those used with stepwise regression are available but not recommended in the presence of multicollinearity (Huberty, 1994). As with regression, one way to overcome the problem of multicollinearity is to transform the data into principle components, then use the stepwise procedure to select the best principle components for the classification.

3.7 Digital sampling of analog data

Digital sampling of data is required for most light measurement applications. Understanding of sampling theory is important, especially when frequency domain filtering is to be used. Several references on sampling theory are available, most of the following discussion is taken from Bringham (1988). There are three main issues that must be addressed when frequency domain information is to be obtained from digitally sampled data: (1) scaling or correspondence of the discrete Fourier transform to the analytical frequency spectrum, (2) aliasing errors, and (3) leakage errors.

A periodic signal, *f(t)*, with period T_o, can be analytically expressed as an infinite sum of periodic sine and cosine functions, called the Fourier series, as shown in equation 3.7.1

$$f(t) = \frac{a_o}{2} + \sum_{n=1}^{\infty}(a_n \cos n\omega_o t + b_n \sin n\omega_o t) \qquad (3.7.1)$$

where:

$$\omega_o = \frac{2\pi}{T_o} \qquad (3.7.2)$$

$$a_n = \frac{2}{T_o}\int_{-T_o/2}^{T_o/2} f(t)\cos(n\omega_o t)dt \qquad (3.7.3)$$

$$b_n = \frac{2}{T_o}\int_{-T_o/2}^{T_o/2} f(t)\sin(n\omega_o t)dt \qquad (3.7.4)$$

$$a_o = \frac{2}{T_o}\int_{-T_o/2}^{T_o/2} f(t)dt \qquad (3.7.5).$$

Defining $C_n = a_n - jb_n$ where $j = \sqrt{-1}$, then

$$C_n = \frac{2}{T_o}\int_{-To/2}^{To/2} f(t)\left[\cos n\omega_o t - j\sin n\omega_o t\right]dt \qquad (3.7.6).$$

The magnitude of C_n is the amplitude at frequency $n\omega_o$, and the set of coefficients, $\{C_n\}$, are called the frequency spectrum of *f(t)*. The Fourier transform for a periodic function with period T_o is defined by equation 3.7.7. Usually the Fourier transform and frequency spectrum of periodic signals differ only by a scaling factor.

$$F(f) = \int_{-\infty}^{\infty} f(t)\left[\cos n\omega_o t\right]dt - j\int_{-\infty}^{\infty} f(t)\left[\sin n\omega_o t\right]dt \qquad (3.7.7)$$

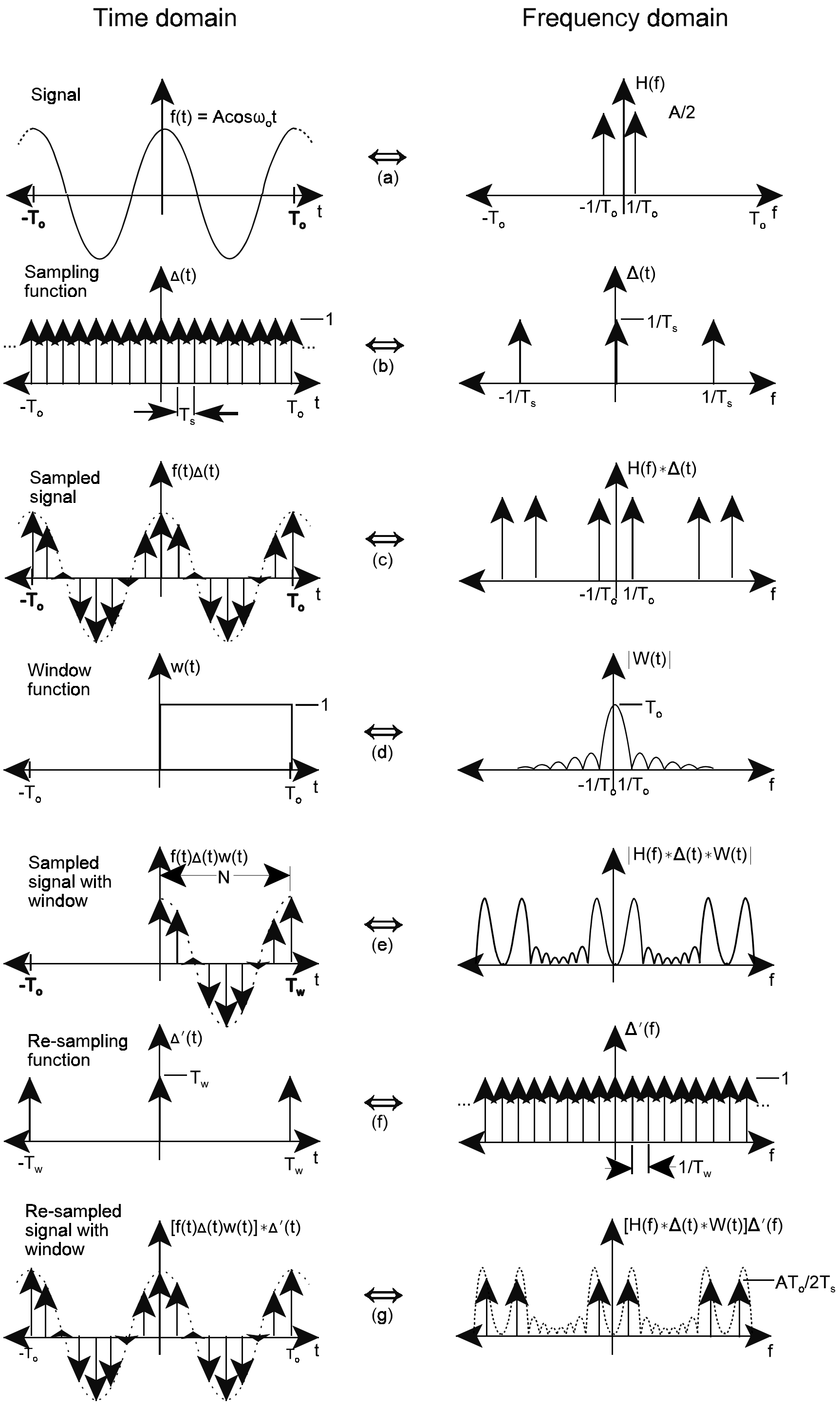


Figure 3.7.1. Graphical representation of a discrete Fourier transform of a band limited periodic waveform with square window function equal to one period. From Bringham (1988).

In figure 3.7.1(a), a band limited periodic waveform, *f(t)*, with period T_o and its corresponding analytical frequency spectrum are displayed. In figure 3.7.1(b), is a sampling function, *Δ(t)*, and its corresponding frequency spectrum. The sampling function is defined as an infinite series of impulse functions with a period T_s. In figure 3.7.1(c), the sampled wave form is shown as the product of the initial wave, *f(t)* and the sampling function, *Δ(t)*. The frequency spectrum of the sampled waveform, *f(t)Δ(t)*, shown in figure 3.7.1(c), is the convolution of the two individual frequency spectrums of each function, $F(f) * \Delta(f)$. Figure 3.7.1(d) displays the window function, *w(t)* and its frequency spectrum, *W(f)*, used during sampling. The shape of the function for this example is rectangular and its duration, T_w, is for one period of the sampled waveform, *f(t)*; however, as will be discussed later, other window functions may be preferentially used. Regardless of the shape, the duration of the window function also dictates the number of samples, *N*, that will be taken. In the time domain, multiplying the window function to the sampled waveform then gives a finite length sampled waveform, *f(t)Δ(t)w(t)*, as shown in figure 3.7.1(e). In the frequency domain, the three individual frequency spectrums are convolved, $F(f) * \Delta(f) * W(f)$, also shown in figure 3.7.1(e). Analytically, this is the frequency spectrum of the finite duration sampled waveform. Note that this frequency spectrum, figure 3.7.1(e), is quite different than the frequency spectrum of the original waveform, *F(f)*, shown in figure 3.7.1(a). This is due to the discontinuity created by the edges of the finite duration sampled waveform, *f(t)Δ(t)w(t)*. Correspondence between the analytical frequency spectrum, *F(f)*, and the finite duration sampled waveform, $F(f) * \Delta(f) * W(f)$, is obtained by multiplying $F(f) * \Delta(f) * W(f)$ by a dummy frequency sampling function, $\Delta'(f)$, with a period of $1/T_w$. The frequency sampling function, $\Delta'(f)$, and its corresponding time domain function, $\Delta'(t)$, are shown in figure 3.7.1(f). The function $\Delta'(t)$ is an infinite series of impulse functions with period equivalent to the duration of the window function. Since the frequency sampling function, $\Delta'(f)$, was mathematically multiplied to $F(f) * \Delta(f) * W(f)$ in the frequency domain, it is convolved with *f(t)Δ(t)w(t)* in the time domain as shown in figure 3.7.1(g). In this example, the duration of the sampling window, *w(t)*, was an integer multiple of the period, T_o, of the original waveform, *f(t)*, the convolution, $[f(t)\Delta(t)w(t)] * \Delta'(t)$, results in a smooth replica of the original sampled waveform, *f(t)Δ(t)*. This leads to a frequency spectrum that is scaled by T_o/T_s from the analytical frequency spectrum. However, if the sampling window duration were not an integer multiple of the input signal's period, or if the input signal was random or noise contaminated, then discontinuities would exist in the result of *[f(t)Δ(t)w(t)] *Δ′(t)*, and these would distort the resulting frequency spectrum. This effect is called leakage error.

Under many conditions, a signal to be sampled is not bandlimited due to noise or by the inherit nature of the signal. Under these conditions, leakage errors are unavoidable but can be greatly reduced by the use of different window functions, such as the Hanning window as shown in figure 3.7.2 and defined in equation 3.7.8

$$w(t)=\begin{cases}\frac{1}{2}\left[1+\cos\left(\frac{2\pi t}{T_o}\right)\right] & \text{for } |t|\le\frac{T_o}{2}\\ 0 \text{ for } |t|>\frac{T_o}{2} & \end{cases} \qquad (3.7.8).$$

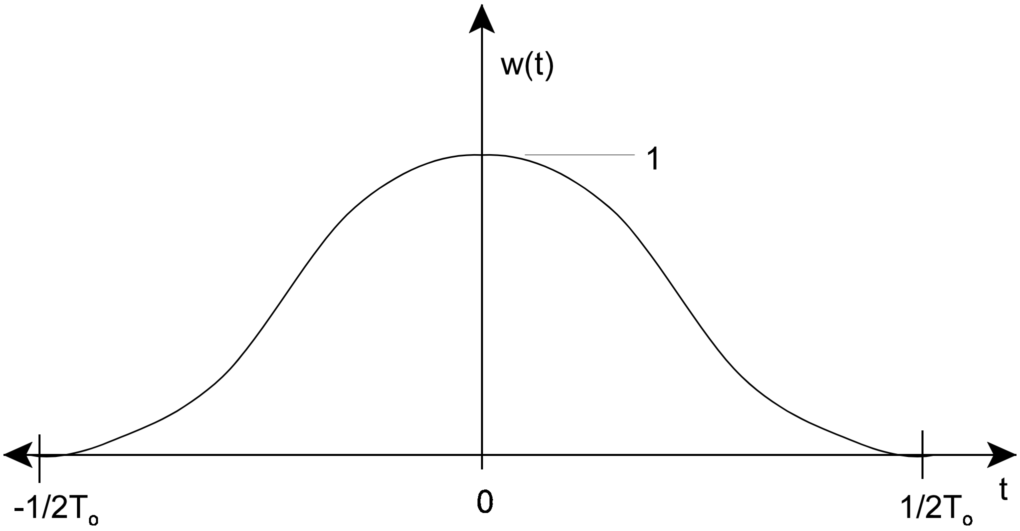


Figure 3.7.2. Hanning window function.

The Hanning window assures that there will be no discontinuities in the frequency sampled time domain result of *[f(t) Δ(t)w(t)] *Δ′(t)*. However, the resolution of the sampled frequency spectrum between frequencies 0 and $T_o/2$ will be diminished. This is due to the broader side lobe in the frequency spectrum of the Hanning window than the rectangular window function. During convolution in the frequency domain of the sampled waveform with the window function, *[F(f) *Δ(f)] *W(f)*, more of the data is "smeared" together due to the broader side lobe on the Hanning window. Obtaining more samples from the original waveform can help overcome this problem.

3.8 Computation of the discrete Fourier transform

The sampling function was defined as an infinite series of impulse functions so the product of the original waveform *f(t)* with the sampling function *Δ(t)* is computed as shown in equation 3.8.1

$$f(t)\Delta(t) = f(t)\sum_{k=-\infty}^{\infty}\delta(t-kT_s) = \sum_{k=-\infty}^{\infty} f(kT_s)\delta(t-kT_s) \qquad (3.8.1).$$

For the sampling window, it is assumed that there are *N* equally spaced samples within the sampling duration, so $N = T_w/T_s$. Equation 3.8.2 shows the result of multiplying a rectangular sampling window with equation 3.8.1

$$f(t)\Delta(t)w(t) = \left[\sum_{k=-\infty}^{\infty} f(kT_s)\delta(t-kT_s)\right]w(t) = \sum_{k=0}^{N-1} f(kT_s)\delta(t-kT_s) \quad (3.8.2).$$

Lastly, the sampling of *F(f) *Δ(f) *W(f)* in the frequency domain is accounted for. In the time domain, this is accomplished with convolving the sampled and truncated wave of equation 3.8.2 with the sampling function, equation 3.8.3, in the time domain. The result is shown in equation 3.8.4 and 3.8.5

$$\Delta'(t) = T_W \sum_{r=-\infty}^{\infty} \delta(t - rT_W) \qquad (3.8.3)$$

$$f(t)\Delta(t)w(t) * \Delta'(t) = \left[\sum_{k=0}^{N-1} f(kT_S)\delta(t-kT_S)\right] * \left[T_W \sum_{r=-\infty}^{\infty}\delta(t-rT_W)\right] \quad (3.8.4)$$

simplifying and letting $\tilde{f}(t) = f(t)\Delta(t)w(t)*\Delta'(t)$:

$$\tilde{f}(t) = f(t)\Delta(t)w(t)*\Delta'(t) = T_w \sum_{r=-\infty}^{\infty}\left[\sum_{k=0}^{N-1} f(kT_s)\delta(t-kT_s-rT_w)\right] \qquad (3.8.5).$$

The discrete Fourier transform of $\tilde{f}(t)$ can now be computed with equation 3.8.6

$$\tilde{F}(\frac{n}{NT_s}) = \sum_{k=0}^{N-1}\tilde{f}(kT)e^{-j2\pi nk/N} \qquad \text{where} \qquad n = 0,1,...,N-1 \qquad (3.8.6).$$

Choice of the sampling frequency is constrained on the low end by the Nyquist sampling theorem which states that data must be sampled at a minimum frequency of twice the bandwidth of interest to avoid aliasing. Sampling frequency and number of samples, *N*, are usually chosen to achieve the desired frequency resolution of the discrete Fourier transform which is $1/NT_s$.

Equation 3.8.6 shows the computation of the discrete Fourier transform if the sampling window is rectangular. However, it is often desirable to use other sampling windows, especially the Hanning window. It can be shown that when a Hanning window is used, equation 3.8.6 should be multiplied by a scale factor of $\sqrt{8/3}$ (Bendat and Piersol, 1986) to obtain the correct magnitude of the Fourier transform.

If it is desired to compute the discrete Fourier transform at all possible frequencies, then it is usually more computational efficient to use the fast Fourier transform (FFT) algorithm. The result of the FFT is the same as the discrete Fourier transform but the number of computations are minimized. A constraint of the FFT algorithm is that the number of samples, *N*, must be a power of 2, so that $N=2^p$, where *p* is any positive integer. The number of multiplications required for FFT procedure is equal to *Np* while the number of multiplications with the discrete Fourier transform is equal to N^2. Another constraint of the FFT is that all of the samples must be obtained in order to start the computations. If the sampling rate is sufficiently slow, then the computations of the discrete Fourier transform as shown in equation 3.8.6 can be carried out between samples so that the Fourier transform will be complete upon finishing the sampling. Finally, if the entire Fourier transform is not required, that is, if the magnitude at only a few specific frequencies in the Fourier transform are of interest, then it may also be more efficient to compute the discrete Fourier transform with equation 3.8.6, especially if the computations can be computed between sampling points.

3.9 Light detectors

There are two main mechanisms of photo detection in widespread use today: the external photoelectric effect and the internal photoelectric effect (Palais, 1988). The photomultiplier tube is an example of the external photo electric effect where electrons are liberated from the surface of a metal due to the energy absorbed by incident photons. Semiconductor devices are examples of internal photo electric effect where free electrons and holes are generated by absorption of photons.

Vacuum photomultiplier tubes (PMT) are used in very low light level applications. They have very fast response times and are not effected by thermal noise sources. Disadvantages of PMT's are high cost, large size, they are fragile and can be significantly effected by external magnetic fields (Anonymous. 1994b). A schematic of a PMT is shown in figure 3.9.1.

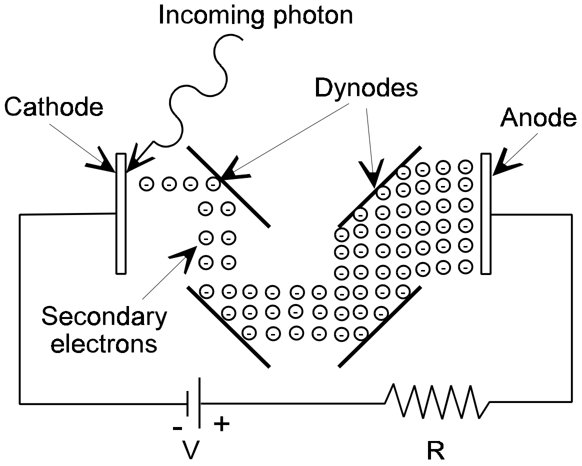


Figure 3.9.1. Schematic of PMT, from Palais (1988).

As can be seen in figure 3.9.1, a bias voltage is applied, making the anode positive and cathode negative. A minimal amount of current will pass through the resistor when no light is incident on the cathode. When light is absorbed by the cathode, energy from the incident photon is transferred to electrons in the cathode material. If sufficient energy is transferred from a photon to electron, the electron will be liberated from the cathode. The free electron will accelerate towards the first dynode because it is placed at a higher voltage than the cathode. This acceleration causes the electron to collide with the dynode with a high kinetic energy. As a result of this high energy collision, two to six electrons will be liberated from the dynode (Palais, 1988). These electrons are then accelerated towards the next dynode and so on until reaching the anode. The influx of electrons to the anode cause current to flow through the external circuit and a voltage across the resistor is proportional to the intensity of incident light on the cathode.

Semiconductor photodetectors are more rugged, economical, and can be contained in a much smaller package than PMTs. However, they are not as sensitive as PMTs and their frequency response is not as fast as a PMT. The output of a semiconductor photodetector usually must be amplified which creates another source for noise. Most semiconductor detectors are made of silicon and are sensitive to light in the visible and near infrared region up to 1100 nm (Anonymous, 1994b). Semiconductor photodiodes can be made out of other materials to achieve different spectral responses. For example InGaAs photodiodes have usable spectral responses between 800 nm and 1700 nm (Anonymous, 1994b). A schematic of a silicon photodiode is shown in figure 3.9.2. The diode is constructed of both n type semiconductor material and p type semiconductor material. The n material contains impurities that form covalent bonds with silicon atoms and make an excess of free electrons. The p material also contains impurities, commonly boron, which form covalent bands with silicon atoms but with insufficient electrons creating voids of electrons. These voids are called “holes” and serve as positive charge carriers.

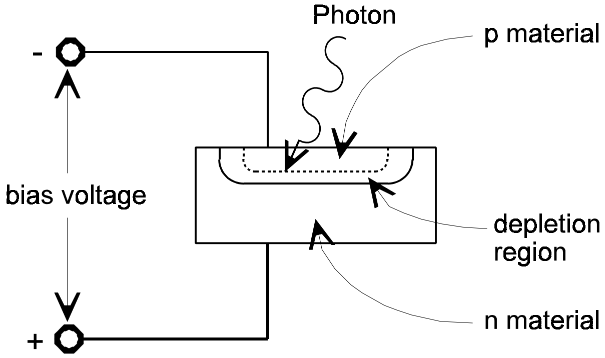


Figure 3.9.2. Schematic of a silicon photodiode (Anonymous, 1994b).

As can be seen in figure 3.9.2, the region between the n material and p material is called the depletion region when no light is present. This region has no free electrons or holes so current is not carried through this region. The absorbed energy of a photon passing through the p material and into the depletion region causes a bound electron in the depletion region to pass from the valence to the conduction band. The electron can now move, creating a hole. Electrons will move towards the n material and the holes moves towards the p material, resulting in a positive change on in the p material and negative charge on the n material. These moving charges cause a current flow through the output leads of the photodiode (Palais, 1988).

There are two modes of operation of a silicon photodiode: photovoltaic or photoconductive. In photovoltaic mode, there is no bias voltage applied to the photodiode and the voltage generated by the migrating electrons and holes is proportional to the incident light intensity. The generated voltage usually needs to be amplified by a high input impedance amplifier. In photoconductive mode, a reverse bias voltage is applied to the diode as shown in figure 3.9.2 and the output current is proportional to the incident light. The current is usually converted to a voltage with a transimpedance amplifier. The response time of a photodiode is faster in the photoconductive mode than in the photovoltaic mode, but the noise in the photoconductive mode is higher.

Avalanche photodiodes are usually silicon devices, but like a PMT, have an internal gain. Avalanche photodiodes utilize a high bias voltage, 100 to 500V, to facilitate the multiplication process. A free electron and hole created by an absorbed photon in the depletion region is excited to a high kinetic energy by the high bias voltage. When these high energy electrons collide with a neutral atom, additional electron hole pairs are liberated. These liberated charges can then free other electron hole pairs. This process is called avalanche multiplication. The internal gain of avalanche photodiodes are not as high as a PMT and their output signals are slightly more noisy than that of a PMT signal. However, avalanche photodiodes are more rugged than PMTs and cost less than PMTs but much more than normal silicon photodiodes.

Photodiode arrays and charge coupled devices (CCD) are comprised of several hundred to several thousand individual photodetector elements integrated into one package. The output of each photodetector is called a pixel. The photodetector array is commonly arranged in a single line or in a rectangular arrangement. Imaging applications widely utilize CCDs. Both photodiode arrays and CCDs are also used with grating type spectrometers for rapid spectrum acquisition. The design and operation of these array sensors are discussed by Tseng et al. (1985). The output of each pixel element flow out of the array package in a serial fashion. When the response of all pixels in the array have been output, the serial output can repeat itself almost immediately. The electrical response of each pixel element to the incident light upon it is usually integrated over the time duration required to output all of the pixel responses of the entire array. That is, while a pixel in not being accessed for output, it’s response is being integrated. The main difference between a CCD and photodiode array is how the integration and data transfer physically take place.

A schematic of a photodiode array sensor is shown in figure 3.9.3. The shift register, controlled by an external clock, opens one MOS switch at a time to allow reading of each photodiode element, one at a time. When a photo diode element is not being read, its output charges the capacitor adjacent to it. So the output of each photodiode element is proportional to the light intensity integrated over the time duration taken while all of the other photodiode elements are being read.

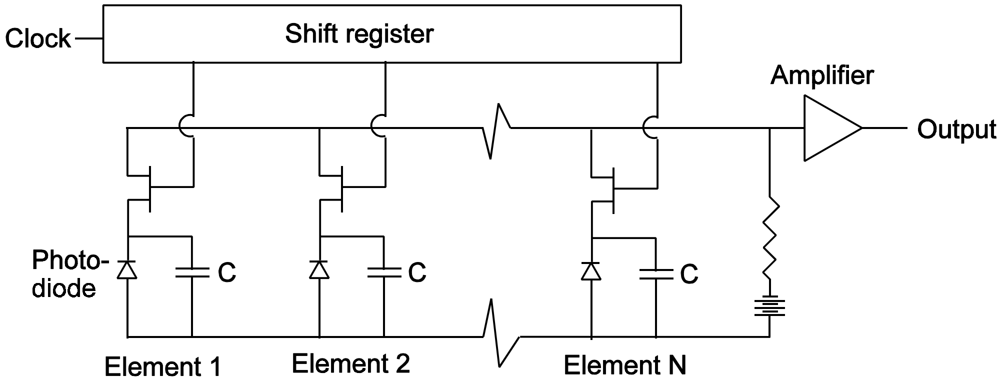


Figure 3.9.3. Simplified schematic of a linear photodiode array, from Tseng et al. (1985).

The operation of CCD sensors are explained in detail by Tseng et al. (1985) and Gibson et al. (1996). The transfer of charges out of a CCD occurs in what is called a charge transfer device. In this device, a charge is temporarily stored and transferred along a string of integrated MOSFET's and capacitors (Cooley and Belina, 1996). A group of charges can be serially transferred as well. A simple schematic of a linear array CCD is shown in figure 3.9.4. Incident light is converted to electrical energy by a tiny photodiode which also stores this energy until it is drained. Thus, the integration in CCD sensors occurs within the photodiode material. The charges accumulated by all the photodiodes are then transferred in parallel to the charge transfer region by activation of the transfer gate. The charge transfer region has individual storage registers which hold the charge transferred from each photodiode. After the photodiode charges are transferred to the charge transfer region, the photodiodes can begin integration again. Referring to figure 3.9.4, one clock pulse to the charge transfer region shifts all charges of each storage register to the right by one register. The register to the far right is output, amplified, and represents the signal from the far right pixel. The next clock pulse shifts the all storage registers to the right once again and outputs another pixel signal. This cycle is repeated until all pixels have been output. The whole process of transferring the charges from the photodetector elements to the charge transfer region can then repeat itself almost immediately. More elaborate architectures for the charge flow are commonly used but these basic concepts are still used.

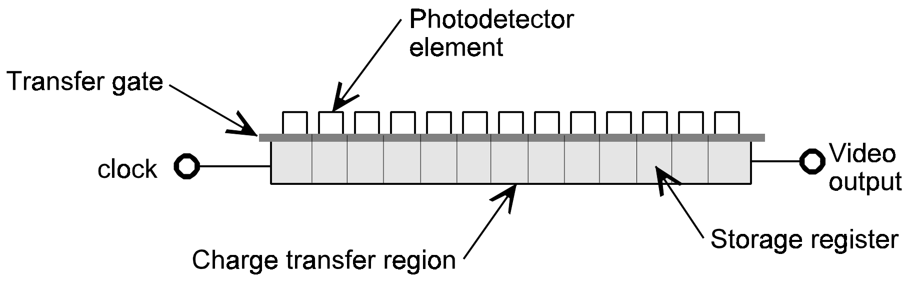


Figure 3.9.4. Simplified schematic of a linear CCD, from Tseng et al. (1985).

In general, CCD arrays are preferred over photodiode arrays in low light level applications. The data transfer in the CCD architecture is less prone to noise contamination than in a photodiode array architecture. However, the CCD array does tend to cost more and the photodiode array architecture allow more energy per pixel to be integrated. Thus, the CCD pixel elements are more likely to become saturated in higher light level applications (Tseng et al., 1985).

3.10 Photodetector noise sources

There are four or five different sources of noise associated with photodetectors, depending on the type. In addition to detector noise, there may other sources of noise in the system, such as amplifier noise.

One type of noise, shot noise, is caused by photons arriving at the sensor in a random order. The random nature of the photons hitting the detector cause the detector current output to randomly vary. Shot noise increases with incident radiant power on the photodetector. Also, the frequency spectrum of shot noise is uniform throughout the usable bandwidth of the photodetector (Horowitz and Hill, 1989). The second type of noise is called generation-recombination noise. When photons liberate electron hole pairs in the depletion region, some pairs may reunite before they reach the p material or n material. Also, additional pairs of electrons and holes may be liberated by thermal energy. These two processes occur randomly and are jointly called generation-recombination noise. A third type of noise, Johnson noise, is caused by fluctuations in a detector's internal resistance due to thermal energy creating random motion of electrons and holes. These movements create a voltage with a uniform frequency spectrum (Horowitz and Hill, 1989). Higher thermal energy causes more random motion of the electrons so cooling the detector will reduce this source of noise. Unlike shot noise, Johnson noise is independent of the incident light intensity. Johnson noise usually dominates in low light level applications. The fourth type of noise is called flicker noise. The cause of flicker noise is not well understood but it occurs only in detectors operating in the photoconductive mode (Anonymous, 1994b). The last type of detector noise, readout noise, occurs only in detectors arranged in an array. When charges are transferred between different storage registers, there is an uncertainty involved leading to this type of noise (Anonymous, 1994b).

Of these noise sources, either Johnson noise or shot noise or both will usually dominate (Palais, 1988). Average current generated by Johnson noise, i_j, can be modeled with the relation shown in equation 3.10.1

$$i_J = \sqrt{\frac{4kT\Delta f}{R_L}} \qquad (3.10.1)$$

where k is the Boltzmann constant, T is the absolute temperature, Δf is the photodetector's operating bandwidth, and R_L is the internal resistance of the photodetector. Average current generated by shot noise, i_{sn}, is given by equation 3.10.2

$$i_{sn} = \sqrt{M^n 2eI\Delta f} \qquad (3.10.2)$$

where e is the magnitude of the charge on an electron, I is the average current flowing in the detector, Δf is the photodetector's operating bandwidth and M is the internal gain in a photodetector such as an avalanche photodiode or PMT. The variable n is a correction factor for avalanche photodiodes because shot noise is more prevalent in these detectors. For PMT's, the shot noise increases linearly with gain, so n equals 2. But for avalanche photodiodes, *n* falls between 2 and 3. Both the shot noise and the Johnson noise are proportional to the square root of the photodetector's operating bandwidth, Δf.

3.11 Electronic amplifier noise

Almost all photodetectors will require some sort of amplifier to boost the electrical output of the sensor to a usable level. In general, noise created in the photodetector will be multiplied by the gain of the external amplifier. Also, the amplifier will add additional noise which is proportional to the gain of the amplifier. The nature of amplifier noise is similar to some of the photodetector noise sources, namely Johnson noise, shot noise, and flicker noise. Johnson noise in amplifiers is due to the same phenomena as in photodetectors, random motions of electrons caused by thermal energy (Horowitz and Hill, 1989). Shot noise in amplifiers is due the random, discrete, flow of electric charges within a circuit, similar to the discrete flow of photons arriving at the photodetector. As with photodiodes, the cause of flicker noise in amplifiers is not well understood; however, the magnitude of this noise source is heavily dependent on the quality of connections within the amplifier circuit (Horowitz and Hill, 1989).

3.12 Digital noise suppression techniques

Shot noise and Johnson noise are always going to be present to some degree, no matter how well the light detection system is designed. Usually some noise suppression is required for all spectroscopic light measurements. Two of the most common methods for suppressing noise, or low pass filtering, in acquired spectra are box car averaging and Savitzky-Golay smoothing (Hruschka, 1987). Box car averaging simply replaces a point by the mean of a set number of points surrounding it. Savitzky-Golay smoothing replaces a point by the predicted value of an *n* degree polynomial fit of points surrounding the point to be smoothed. The replaced point is always the center point in the set of points fit by the polynomial. Both box car averaging and Savitzky-Golay smoothing are easy to implement. Both methods work equally well if the true absorbance peaks in the spectrum are wide compared to the number of points used in the box car average or polynomial fit. However, if the absorbance peak is sharp or narrow, then Savitzky-Golay smoothing may be more desirable (Hruschka, 1987).

Both shot noise and Johnson noise are proportional to the square root of the photodetector's operating bandwidth, *Δf*. One way to significantly reduce these two noise sources is to modulate the incident light at a specific frequency, and band pass filter the photodetector output at that frequency (Anonymous 1994b). Letting Δf_{bp} be the band pass width, then the contribution of both shot noise and Johnson noise will be reduced by a factor of $\sqrt{\Delta f / \Delta f_{bp}}$. This method also can allow simultaneous acquisition of several different signals by one discrete detector, a method used in early analog communication networks (Palais, 1988). If several different light signals, modulated at different frequencies, are incident upon a single detector, then the detector will output the summation of all the modulated signals. The detector output signal can then be separated into individual components, due to each of the modulated light signals, by a set of band pass filters or by computing a Fourier transform on the acquired signal and selecting the desired frequency component. A schematic of such an analog modulation system is shown in figure 3.12.1.

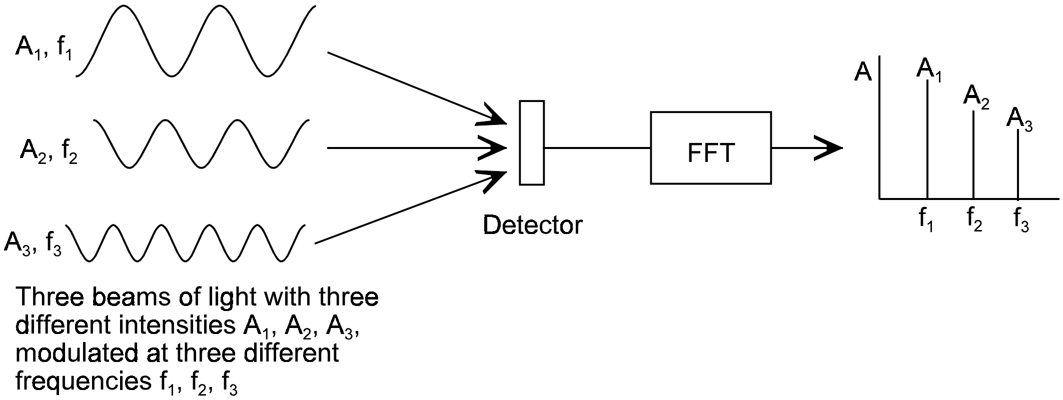

A_1, f_1
A_2, f_2
A_3, f_3
Three beams of light with three different intensities A_1, A_2, A_3, modulated at three different frequencies f_1, f_2, f_3
Detector
FFT
A
A_1
A_2
A_3
f_1
f_2
f_3

Figure 3.12.1. Schematic of a system to detect three beams of light with three different modulation frequencies.

3.13 Light emitting diodes

Light emitting diodes (LED) offer several advantages over incandescent light sources such as: ability to operate at high modulation frequencies, long lifetime, ability to withstand shock and vibration, and low power consumption (Hodapp, 1996). The operation of a LED is similar to that of semiconductor photodetectors in reverse. A LED is constructed of a pn junction that emits light when forward biased (Palais, 1988). The forward bias voltage provide the free electrons, contained in the n region, enough energy to cross into the p region and unite with a hole. This results in an energy loss which is converted to optical energy (Palais, 1988). Most commercial LED devices are constructed of a few elements such as Aluminum (Al), Galium (Ga), Indium (In), Nitrogen (N), Phosphorous (P), Arsenic (As), and Antimony (Sb) (Hodapp, 1996). Emission wavelengths of devices constructed of GaInP are in the range of 640 to 680 nm, AlGaAs devices emit light from 800 to 900 nm, and GaAs devices emit light between 900 and 1000 nm (Palais, 1988). The actual light emission range and peak wavelength depends further on the internal crystal structure of the semiconductor material (Anonymous, 1996).

In an ideal case, the radiant power output of a LED is linearly proportional to the electrical current passing through the LED. Impurities and imperfections in the LED semiconductor material can cause the relation between LED current and radiant output power to be slightly nonlinear. However, most of the nonlinearities occur at operating currents at the very low end or very high end of the LED's operating range (Hodapp, 1996).

The radiant output power of LEDs and in the peak emission wavelength are effected by their operating temperature and length of service. The internal resistance of the LED changes with temperature and this effects the current flow through the device, resulting changes in radiant output power. Also, radiant output power for a given current will deteriorate with time. This is due to dislocations and shifts in the crystal structure of the semiconductor material (Hodapp, 1996). The radiant output power, *P*, decreases exponentially with operating time as shown in equation 3.13.1

$$P = P_o e^{-\beta t} \qquad (3.13.1)$$

where P_o is the initial radiant output power, β is a constant called the deterioration factor, and t is the total operating time. Thus, the radiant output power of an LED may drop rapidly during an initial "burn in" period. Afterwards, the radiant output power will decrease gradually for the duration of the life of the LED.

3.14 Color theory

Young (1802) stated that any color can be reproduced by mixing an appropriate set of three primary colors. Much later, it was shown that the normal human eye has three different color sensors, called cones, with different absorption spectra, $S_1(\lambda)$, $S_2(\lambda)$, $S_3(\lambda)$ (MacAdam, 1970). The spectral sensitivity of all normal cones lie between 380 nm and 780 nm but the peak responses for each cone falls in the yellow-green region of the spectrum for $S_1(\lambda)$, in the green region for $S_2(\lambda)$, and in the blue region for $S_3(\lambda)$. Color sensation due to the response from visible light with a spectral energy distribution described by $C(\lambda)$ can be described by three separate responses, $\alpha_1(C)$, $\alpha_2(C)$ and $\alpha_3(C)$ computed by equation 3.14.1

$$\alpha_i(C)=\int_{380}^{780}S_i(\lambda)C(\lambda)d\lambda \qquad i=1,\ 2,\ 3 \qquad (3.14.1).$$

The mathematics of color matching is described by Jain (1989). Defining three independent primary sources of light, $P_k(\lambda)$, k = 1, 2, 3, where

$$\int_{380}^{780}P_k(\lambda)d\lambda=1 \qquad (3.14.2)$$

and three proportions β_k, where k = 1, 2, 3. Then the perception of light with spectral energy, $C(\lambda)$, could be matched by the proper mixture of $\beta_k P_k(\lambda)$, k = 1, 2, 3. The three cone responses $\alpha_1(C)$, $\alpha_2(C)$ and $\alpha_3(C)$ can be described equation 3.14.3 in terms of β_k and $P_k(\lambda)$

$$\alpha_i(C)=\int_{380}^{780}\left[\sum_{k=1}^{3}\beta_k P_k(\lambda)\right]S_i(\lambda)d\lambda=\sum_{k=1}^{3}\beta_k\int_{380}^{780}S_i(\lambda)P_k(\lambda)d\lambda \quad i=1,\ 2,\ 3 \qquad (3.14.3).$$

Values of for β_1, β_2 and β_3 can be found by solving the three equations for $\alpha_1(C)$, $\alpha_2(C)$ and $\alpha_3(C)$ to describe the cone response to light with a spectral energy distribution, $C(\lambda)$, using three known primary sources, $P_1(\lambda)$, $P_2(\lambda)$, and $P_3(\lambda)$. The primary colors are usually normalized with a white light source with a known spectral energy distribution. White light is produced by adding the three primary sources. Letting w_k represent the quantity of the kth primary source required to match the reference white source, then the quantities $T_k(C) = \beta_k / w_k$, k = 1, 2, 3 are called the tristimulus values of color C (Jain, 1989). Chromaticities of a color are defined in equation 3.14.4

$$t_k = \frac{T_k}{T_1 + T_2 + T_3} \qquad (3.14.4).$$

There are several systems for describing color. People normally describe color in terms of brightness, hue, and saturation. The sensation of brightness describes the light intensity, or gray level (black, gray, or white). Hue describes the dominant wavelength which defines the color as red, blue, green, etc (Benson, 1986). Saturation describes the amount of white which gives sensation of the color's strength or purity (pale, pastel, vivid, strong). A common color system is called the C.I.E. XYZ system formed by the Commission Internationale de L'Eclairage (International Committee on Color Standards). The C.I.E. XYZ system is based on an imaginary set of primary sources so that all colors can be reproduced with it. A plot of the x and y chromaticities, called chromaticity diagram is shown in figure 3.14.1.

Only physically realizable colors would fall in the shaded area of the chromaticity diagram shown in figure 3.14.1. Distances between two points within the C.I.E. XYZ chromaticity diagram do not well quantify perceived difference in color. The ellipses shown in the chromaticity diagram are called MacAdam ellipses (Jain, 1989). Colors within each of these ellipses are not distinguishable to the average viewer. The C.I.E. XYZ color system can be transformed to other color systems, such as the modified Uniform Chromaticity Scale (UCS) system or C.I.E. L* a* b* system, where distance between two points in the chromaticity diagram quantitatively corresponds to perceived difference in color (Jain, 1989).

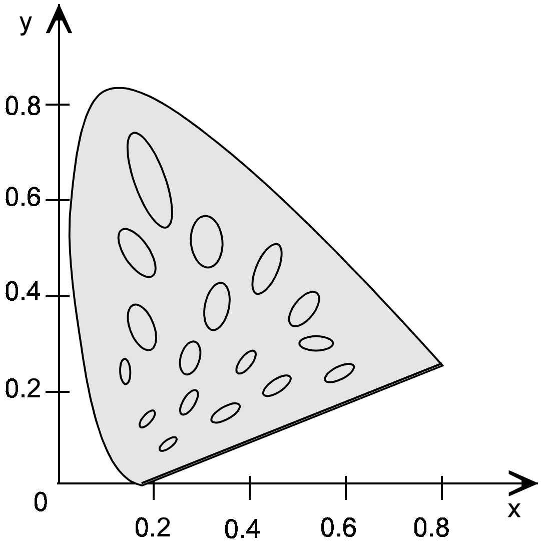


Figure 3.14.1. Chromaticity diagram for the C.I.E. XYZ color system, from (Jain, 1989).

In the C.I.E. L* a* b* system, L* indicates lightness, a* indicates level of red or green, and b* indicates the level of yellow or blue. An a* b* chromaticity diagram is shown in figure 3.14.2. As can be seen from this figure, a high positive a* value indicates red colors, a highly negative a* value indicates green colors, a high positive b* value indicates yellow colors and a highly negative b* value indicates blue. The C.I.E. L* a* b* values can be computed from the C.I.E. XYZ system with equations 3.14.1 through 3.14.3

$$L^* = 116\sqrt{\frac{Y}{Y_n}} \qquad (3.14.1)$$

$$a^* = 500\left[\left(\frac{X}{X_n}\right)^{1/3} - \left(\frac{Y}{Y_n}\right)^{1/3}\right] \qquad (3.14.2)$$

$$b^* = 200\left[\left(\frac{Y}{Y_n}\right)^{1/3} - \left(\frac{Z}{Z_n}\right)^{1/3}\right] \qquad (3.14.3)$$

where X_n, Y_n, and Z_n are the tristimulus values for the reference white (Jain, 1989).

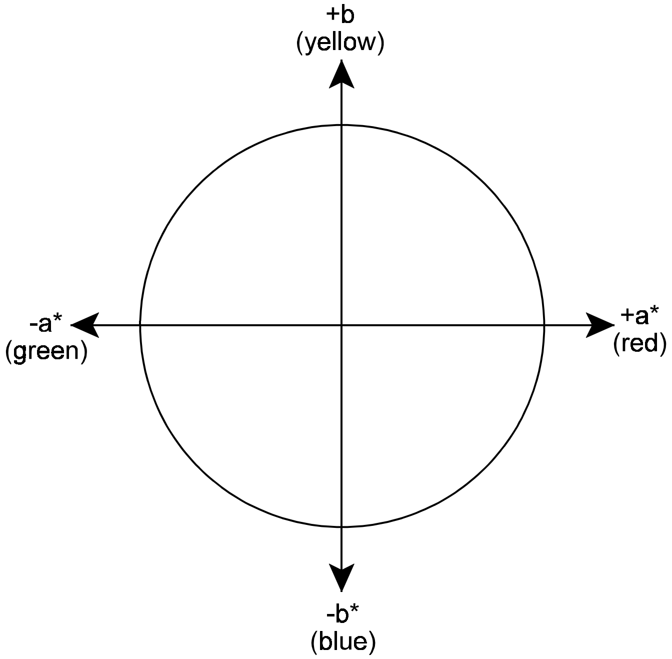


Figure 3.14.2. Chromaticity diagram for the C.I.E. L* a* b* system.

4. MATERIALS AND METHODS

4.1. Field Experiments

Effect of added moisture on incidence of concealed damage of windrowed nuts and nuts scattered on the orchard floor was studied in two different orchards at harvest time. Nonpareil almonds were sampled from the first orchard on August 13, 1996 in the Wolfskill Agricultural Experiment Station in Winters, California. Mission almonds were sampled from the second orchard on September 20, 1996 in the Dewy Farms orchard near Woodland, California. Upon harvest (tree shaking), samples weighing approximately 300 g were collected from the orchard floor for moisture measurement. Moisture was measured in a vacuum oven (National Appliance, Portland, OR, model 5851) by the drying method with 30 psi vacuum, 50°C temperature, and 24 hours. These parameters were determined by trial and error.

Three groups of approximately 2 kg in-shell nuts with hulls (in each group) were collected from the orchard floor after tree shaking and condensed into an area approximately one square meter for moisture and temperature monitoring. Some nuts were touching each other but there was only one single layer. Likewise, another three groups, each comprising 2 kg nuts, were raked along with leaves and other orchard floor material into windrows approximately 1 m long and 20 cm high. A windrow height of 20 cm was typical of adjacent orchards for this harvest year. One pair of windrow and scattered nuts was treated to sprinkle irrigation for 45 minutes which amounted to 1 cm water coverage. Another pair of windrow and scattered nuts was sprinkle irrigated for 135 minutes which amounted to 3 cm water coverage. The third pair of scattered and windrow nuts was left as a control set and not treated with any irrigation. A four channel temperature recorder (Hanna Instruments, Italy, model HI 9274C) was used to monitor the temperature of one nut in each windrow treatment as well as the ambient temperature. The nut was then positioned at the center of the windrow, about 10 cm from the ground. The probe from the temperature recorder was inserted through the hull, shell, and into the kernel. The temperature of each channel was recorded every thirty minutes for the first three days of the experiment. Approximately 300 g of nuts were sampled from each treatment each day for five days for moisture content measurement. After drying to determine moisture content on the fifth day, Mission almonds were split at the suture and visually inspected for presence of concealed damage. Since the nuts were dried in the vacuum oven at 50°C for 24 hours, internal browning was already present. So, no further cooking was required to induce concealed damage.

4.2. Chemical tests

a. Overview

Chemical properties of almonds was studied before they were exposed to moisture, after exposure to moisture, after drying, and after cooking. Whole Mission almonds kernels from the 1996 harvest were used for this entire experiment. Almonds were exposed to three different moisture treatments (none, short, long), two different drying treatments (low temperature and high temperature) and all cooked together at the same temperature and for the same time. Details of the moisture, drying and cooking treatments will follow later in this discussion. Before exposure to moisture, after moisture exposure, after drying, and after cooking, batches of nuts were removed for chemical analysis. Sets of ten batches, with each batch comprising 30 almonds, were analyzed for glucose, sucrose, color, moisture, oil refractive index and soluble solids. Basic amino acids were measured before moisture treatment and after cooking.

The short moisture treatment involved soaking the nuts in water for 30 minutes then transferring them to a 95% relative humidity environment at 22°C for 12 hours. The long moisture treatment involved soaking the nuts in water for 10 minutes then transferring them to a 95% relative humidity environment at 22°C for 72 hours. These two treatments were found to raise the kernel moisture content to approximately 15% d.b. for the short treatment and 22% for the long treatment. These moisture contents were similar to the 1 cm and 3 cm irrigation treatments used in the field experiments.

After moisture treatment, nuts were dried to their original mass at one of two different drying temperatures, 55°C or 110°C, in an air convection dehydrator (Proctor, Horsham, PA, #062). The nuts were then cooked in a gravity convection oven at 135°C for 90 minutes.

A total of 12 sets of ten batches were analyzed since there were two different moisture treatments (as well as the controls) and two different drying treatments. The sample sets comprising ten batches of nuts were removed for analysis after each treatment. Table 4.2.1 displays all of the treatment combinations used and what measurements were taken for a given treatment combination. Figure 4.2.1 displays a flow chart of the experimental procedure.

Table 4.2.1. Chemical analysis performed for a given moisture, drying and cooking treatment. An "X" means that the analysis was performed for the treatment combination.

Treatments			Analysis							
moisture	dry temp (°C)	cooked?	moisture	glucose	sucrose	soluble solids	basic amino acids	oil refractive index	color	concealed damage incidence
none	none	no	X	X	X	X	X	X	X	X
short	none	no	X	X	X	X		X	X	
long	none	no	X	X	X	X		X	X	
short	55	no	X	X	X	X		X	X	
short	110	no	X	X	X	X		X	X	
long	55	no	X	X	X	X		X	X	
long	110	no	X	X	X	X		X	X	
none	none	yes	X	X	X	X	X	X	X	X
short	55	yes	X	X	X	X	X	X	X	X
short	110	yes	X	X	X	X	X	X	X	X
long	55	yes	X	X	X	X	X	X	X	X
long	110	yes	X	X	X	X	X	X	X	X

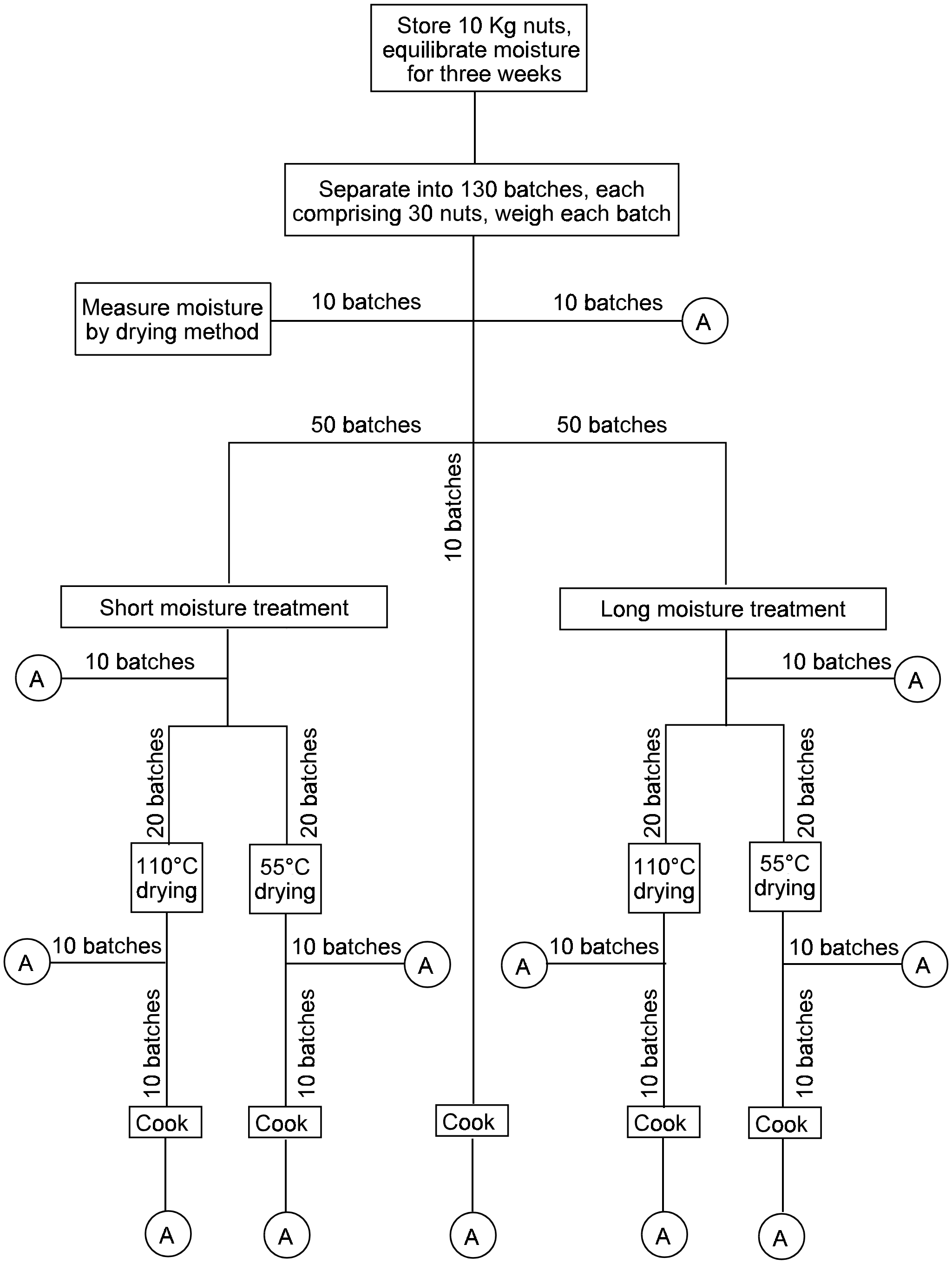


Figure 4.2.1. Flow chart of experimental procedure. The "A" enclosed in a circle indicate nuts were removed for analysis. Refer to table 4.2.1 for the specific analysis performed at each stage.

b. Moisture measurement

Prior to subjecting the nuts to any treatment, all of the nuts were stored in a sealed plastic container for three weeks in order for their moisture to equilibrate. After three weeks, the moisture content of ten batches of thirty nuts was measured by drying in a vacuum oven (National Appliance, Portland, OR, model 5851) with 30 psi vacuum, 50°C temperature, for 24 hours (AOAC, 1980). Moisture of the treated nuts was computed using the change of mass of the batch and assuming all batches started with the same initial moisture as the vacuum oven dried samples.

c. Sugar assay

The sugar analysis was performed by an adaptation of a glucose oxidase method described by Anonymous (1995b). Each almond batch was ground in a blender with nearly equal proportions of dry ice and nuts to prevent caking. After two minutes of grinding, the resulting powder was passed through a #30 mesh screen. The larger particles not passing through the screen were ground again for two minutes and sifted again. This procedure was repeated until all the almond grindings were able to pass freely through the screen. The sugars were extracted by placing 200 mg of almond powder in a test tube with 4 ml of water for one hour at room temperature. The sample was found, by trial and error, to reach an asymptotic sugar concentration in one hour. It is desirable to keep the extraction time and temperature at a minimum to prevent sucrose inversion during extraction. During extraction, the test tube contents were mixed with a vortex mixer every 15 minutes for 30 seconds. After the extraction period, the test tube was centrifuged for 5 minutes at 4500 RPM. Next, 2 ml of supernatant was removed and placed into a clean test tube and centrifuged again for 5 minutes at 4500 RPM.

For glucose determination, 1 ml of supernatant was mixed with 5 ml of combined enzyme - color reagent solution (1.6 ml o-dianisidine dihydrochloride reagent, Sigma, St. Louis, MO, #510-50) mixed with 100 ml glucose oxidase solution (Sigma, St. Louis, MO, #510-6). A standard glucose assay was prepared by adding 0.075 ml glucose standard solution (Sigma, St. Louis, MO, #635-100) to 0.925 ml water and 5 ml of the enzyme color reagent solution. All samples, including the standard, were incubated at

37° C for 20 minutes. Thereafter, 2 ml of each sample was transferred to a cuvet and light absorbance was measured at 440 nm with a spectrometer (Ocean Optics, #PC1000, Dunedin, FL). The glucose concentration of the almond powder was then computed using equation 4.2.1.

$$Glucose\ (mg/g_{almond}) = (A_{sample})(1.5)/A_{standard}) \qquad (4.2.1)$$

For the sucrose assay, 0.075 ml sugar extract supernatant was added to 0.425 ml invertase solution containing 40 mg class IV invertase (Sigma, St. Louis, MO, #I-4504) dissolved in 50 ml water. The standard comprised 0.075 ml glucose standard solution with 0.425 ml invertase solution. The dilution of the invertase solution was found, by trial and error, to completely invert dilutions of pure sucrose of comparable concentration to the almond sugar extract. Each sample, and the standard was added to 5 ml of combined enzyme - color reagent solution. All samples were incubated at 37°C for 20 minutes and absorbance was measured at 440 nm. This reading determines the total invert glucose plus resident glucose before analysis. Sucrose concentration of the almond powder was computed using equation 4.2.2. All glucose and sucrose assays for each sample were performed in duplicate and the average of the two measurements were used. Duplication of the assays was used to check if an error was made during the experiment. When the experiment was properly performed, the two assays would not deviate more than 1% from each other.

$$Sucrose\ (mg/g_{almond}) = (A_{sample})(40)/A_{standard}) - 2{*}Glucose\ (mg/g_{almond}) \qquad (4.2.2)$$

d. Water soluble solids measurement

The water soluble solids of the sugar extract solution used for the glucose and sucrose assays were measured with an Abbe refractometer (American Optical Corporation, Buffalo, NY, #10450). Three separate refractometer measurements were taken for each sample and the average was used.

e. Oil refractive index measurement

Oil from the almond powder was extracted by placing approximately 5 g of almond powder in a Buchner type funnel (Coors, Golden, CO, #60240) fitted with filter paper (Watham, Maidston, England, #50,). Approximately 10 ml hexane was slowly poured over the powder and pulled through the filter paper by a vacuum on the opposite end of the funnel. The filtered fluid was transferred to a clean test tube and placed in a water bath at 50°C. A flow of gaseous nitrogen was directed into each test tube to shield the extracted oil in order to prevent oxidation. Under these conditions, the hexane completely evaporated in less than ninety minutes. Duplicate readings of refractive index of the extracted oil were obtained with an Abbe refractometer (American Optical Corporation, Buffalo, NY, #10450). As with the sugar assay, the duplicate readings were used as a check. When the experiment was performed properly, the refractive index readings from the same sample would fall within 0.2% of each other.

f. Amino acid assay

The amino acid assay, adapted from Hurrell et al. (1979), was performed to quantify amino acid degradation that might occur during the browning reaction associated with concealed damage. This assay can be used to determine all three basic amino acids, histidine, arginine, and lysine and with an additional step, lysine alone can be measured. Only the procedure to measure all three basic amino acids was used as almonds contain small amounts of lysine and preliminary tests did not show a correlation between lysine concentration and concealed damage. The dye solution used for the assay contained: 3.89 mmol dye/l, 1.36 g acid orange 12, 20 g oxalic acid dihydrate, 3.4 g potassium dihydrogen phosphate, 60 ml glacial acetic acid and water to bring the volume to one liter. The basic amino acid assay procedure was as follows: 100 mg of almond powder (estimated to contain 3.25 mg basic amino groups @ 32.5 mg/g) was mixed with 1 ml sodium acetate solution (16.4%w/v), and vibrated for 20 minutes. Afterwards, 4 ml dye solution was added and the mixture vibrated for 2 hours. The mixture was then centrifuged for 10 minutes at 4000 rpm and the supernatant diluted 50 fold with buffer solution. The buffer solution contained 0.1M sodium bicarbonate with the pH adjusted to

8.5, as measured by litmus paper (Fisher Scientific, Pittsburgh PA, #A979), with sodium hydroxide. A dye solution (with no sample) was diluted 50 fold as well as a standard. The diluted mixture was transferred to a cuvet and the absorbance was measured at 480 nm with a spectrometer (Ocean Optics, Dunedin, FL, #PC1000). The standard dye solution has a dye concentration of 3.89 mmol/l. The concentration of dye in the sample solutions is less as the dye binds to the amino groups of the basic amino acid side chains on a one to one basis. Thus, the concentration of basic amino acids is approximated by the difference in absorbance readings between the sample and standard as shown in equation 4.2.3

$$\textit{Basic amino acid concentration (mmol/l)} = 3.89(A_{std}\text{-}A_{sample})/A_{std} \quad (4.2.3).$$

g. Color and concealed damage measurements

The color of the almond powder was measured with a colorimeter (Minolta, Japan, #CR-200). The almond powder was held in a clear plastic, 20 mm diameter, cylindrical vial with a flat bottom and the powder was compressed with a 0.5 kg cylindrical lead weight from the top. The diameter of the weight was slightly less than the inside diameter of the vial. The vial was then set on top of the colorimeter for measurement. Three measurements for each sample were taken and averaged. The colorimeter was re-calibrated with a white standard between each sample.

After cooking each batch of nuts, the incidence of concealed damage was measured by splitting each nut in the batch at the suture and visually classifying the nut into one of two classes, concealed damaged or good. A nut was considered concealed damaged if more than 50% of the kernel cross-sectional area appeared dark brown as shown in figure 1.1. This is the approximate criteria used by the almond industry (Stoddard, 1995).

h. Post-dry water soluble solids and post-cook color

Based on soluble solid content results from the batch tests, soluble solids in dried nuts before cooking might be used to predict severity of concealed damage after cooking. This was investigated further with 24 individual nuts. Nuts were exposed to moisture by soaking them in water for 30 minutes and then holding them in a 95% relative humidity environment at 22° C for 36 hours. Afterwards, the nuts were dried at 110°C to their original bulk mass. After drying, individual nuts were split in half at the suture. Water soluble solids were extracted and measured from one half per the procedure used in the sugar assays in section 4.2b. The other half of the nut was cooked at 135° C for 40 minutes. The color of the cooked half was measured with a colorimeter (Minolta, Japan, #CR-200) on a 6.35 mm diameter spot at approximately the widest part of the nut.

4.3 Physical and spectral property measurement

a. Overview

Physical properties and light transmission spectra were measured on individual nuts which were treated with one of three different moisture treatments (none, short or long) and one of two different drying treatments. These moisture treatments were slightly different than those used in the chemical tests. The reason for the change in moisture treatments was to induce a higher incidence of nuts with concealed damage. The long moisture treatment comprised soaking the nuts in water for 60 minutes, then transferring them to a 95% relative humidity environment for 60 hours. This treatment raised to moisture content of the almond kernels to approximately 44% d.b. The short moisture treatment comprised soaking the nuts in water for 30 minutes then transferring them to a 95% relative humidity environment for 30 hours. This treatment raised to moisture content of the almond kernels to approximately 23% d.b. Drying was performed in an air convection dehydrator (Proctor, Horsham, PA, #062). The following eight moisture, drying, and storage treatment combinations were used**:**

(1) long moisture and 110°C convection dry,
(2) long moisture and 55°C convection dry,
(3) short moisture and 110°C convection dry,
(4) short moisture and 55°C convection dry,
(5) no moisture treatment and no drying,
(6) no moisture treatment and 55°C convection dry,
(7) no moisture treatment and 110°C convection dry,

(8) long moisture treatment and 110°C convection dry, then stored for one year at 90% relative humidity at 4°C.

The nuts that were dried but received no moisture treatment were dried for the same length of time as the short moisture treatment nuts. Eighty one Mission variety almonds, 1996 harvest, were exposed to each moisture and drying treatment combination. Before any treatment was performed, nuts to be used in this experiment were stored together in a sealed container for three weeks to equilibrate the moisture. A sample comprising 30 g of nuts (approximately 33 nuts) was removed and their moisture measured by the vacuum drying method (National Appliance, Portland, OR, model 5851) with 30 psi vacuum, 50°C temperature, for 24 hours. The cooking treatment was 135°C for 90 minutes in a convection oven (Lab-Line Instruments, Inc., Melrose Park, IL, Imperial IV). The pre-cook physical properties measured of each sample were: pre-moisture treatment mass, post-dry mass, post-dry thickness, post-dry volume by x-ray imaging, transmission spectra from 700 to 1000 nm before treatment, post-dry transmission spectra from 700 to 1400 nm. In addition, the moisture treated nuts were visually inspected for mold on the kernel surface after moisture exposure, a wet appearance after moisture exposure, and if the nut was from a double kernel. The post-cook physical properties measured were mass and color of the kernel interior with a colorimeter (Minolta, Japan, #CR-200). In addition to these measurements, the effect of a seven month storage on severity of concealed damage was studied (section 4.3e).

b. Thickness and volume measurement

Nut thickness was measured, to the nearest 0.25 mm, at its widest point perpendicular to the suture plane with handheld calipers (General National, Berne, Switzerland). Nut volume was measured by obtaining a film x-ray image of nuts lying flat, with their suture plane parallel to the image plane. The film x-ray was created with an x-ray source (Faxitron X-ray, Buffalo Grove, IL, #4380N) at a voltage of 30 KeV, current of 30 ma and exposure time of 100 s. The x-ray film (Eastman Kodak, Rochester, NY, Industrex B film) was developed with an automated developing system (Eastman Kodak, Rochester, NY, X-Omat M35). The x-ray operating parameters were established by trial and error to give optimum contrast determined by manual inspection. A square piece of wood was placed in each image as a standard volume. Wood was used as it's molecular content and atomic absorptions are similar to almonds. All images were taken within one hour to avoid moisture changes in the wood standard. The film x-ray images were digitized at 8 bit resolution, 58 pixels per centimeter with an automated film scanner (Lumisys Inc., Sunnyvale, CA, #0068-255). The digital images were thresholded to set the background intensity to zero then logarithmically transformed. The log transform makes the intensity of each pixel proportional to the thickness of the nut at that pixel location, not counting internal voids. The intensities of all pixels comprising only the nut were integrated as well as the wood standard. This integration is proportional to the volume of the nut. To test the correlation of pixel integration and nut volume, x-ray images of 49 control nuts and 49 moisture treated and dried nuts (long moisture exposure, 110° C dry) were obtained. Afterwards, the volume of the nut was measured by buoyancy in toluene.

c. Light transmission spectra measurement

The light transmission spectra of shelled whole natural Mission almonds were obtained with two different fiber optic spectrometers. A silicon photodiode array sensor based spectrometer (Ocean Optics, Dunedin, FL, #PC1000) was used to obtain the spectrum from 700 to 1000 nm, and an InGaAS photodiode array spectrometer (Control Development, South Bend, IN, #OSC/256L-1.7T1-250A/0.9-1.7/3.2) was used to measure the spectrum from 950 nm to 1400 nm. The spectral resolution, or interval between absorbance points, of the silicon spectrometer and InGaAs spectrometer were 0.48 nm and 3.2 nm respectively. The optical resolution for both spectrometers had a full width half maximum of approximately 3 nm. This means that if a purely monochromatic light source was measured with the spectrometer, then the measured spectrum would not appear as a single point but would appear as a gaussian distribution with a width of width of 3 nm at its half maximum. The apparatus used for obtaining the transmission spectra is displayed in figure 4.3.1. Each spectrometer sampled ten complete transmission spectra and stored the average. The integration time of each photodiode element on the silicon spectrometer and InGaAs spectrometer was 0.5 s and 1.0 s respectively. The integration times and number of samples to average were obtained by trial and error.

Approximately 15 seconds were required to acquire spectra for one nut with these parameters. The time that the nut was exposed to the light source had to be minimized. If the nut was exposed to the light source for more than one minute, heating and scorching would sometimes begin due to the intense light from the light source. The light source was a 100 W quartz tungsten halogen lamp (Oriel, Stratford, CT, #77501). The light transmission spectra of each nut was measured at approximately the thickest point perpendicular to the suture plane. Transmitted light through the nut was split and directed to each of the two spectrometers through fiber optic cables. This facilitated acquisition of spectra by both spectrometers at the same time. A light standard and dark standard were obtained between sampling each nut. The dark standard was obtained by blocking the light source with a steel shutter. The light standard was obtained by placing a glass neutral density filter with a transmission of 0.1% (Ealing, Holliston, MA, #35-5941) in place of the sample.

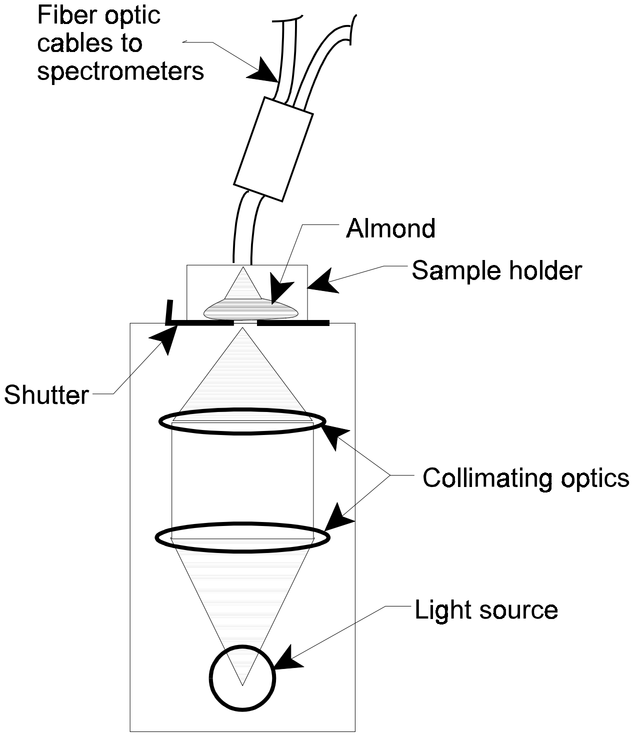


Figure 4.3.1. Schematic of light transmission spectra hardware.

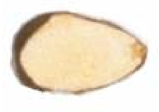
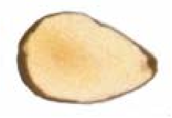
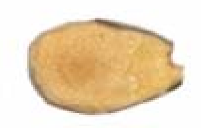
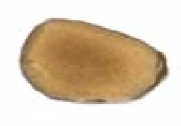
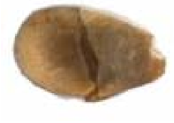
Both of the spectrometers stored each spectrum from a single nut in a separate data file. All the spectral data files from each spectrometer were consolidated into one large comma delimitated ASCII file using the programs listed in appendices A and B. Afterwards, the spectra was filtered using a 19 point Savitzky - Golay 2nd order smooth performed by the program listed in appendix D. Using the program listed in appendix C, the combined spectra were converted from comma delimitated format to a standard JCAMP-DX format, as described by McDonald and Wilks, Jr. (1988), for importing into spectral analysis software.

d. Scoring concealed damage severity

After cooking, nuts were split at the suture and their color measured on a 6.35 mm diameter circle of the nut interior at approximately the same location as where the transmission spectrum was obtained. A colorimeter (Minolta, Japan, #CR-200) was used to measure the color in C.I.E. L* a* b* color space.

The cooked nuts were evaluated for severity of concealed damage using a five level visual scoring system, table 4.3.1. Nuts from each representative group were used as references. Two people performed the scoring together as a team and a score was assigned only after the two people agreed on the assigned score.

Table 4.3.1. Visual concealed damage scoring criteria.

Score	Description	Example
1	no perceivable browning	
2	minor light browning, covering less than 50% of kernel cross-sectional area	
3	minor, light browning, covering more than 50% of the kernel cross-sectional area, OR minor browning covering less than 50% of kernel area if a darker brown spot exists	
4	darker browning covering at least 50% of the kernel area	
5	darker browning covering at least 50% of kernel area and at least 10% of kernel area appears very dark or burnt.	

Photographic slides (Eastman Kodak Co., Rochester, N.Y., ASA 200, Ektachrome) were obtained of all split open almond kernels after cooking. All 81 kernels from one treatment group were arranged side by side in a small square array and photographed all at once. Thus, one slide captured the images of all nuts belonging to one treatment group. Two paint chips, rawhide brown and almond color (Behr Process Corp., Santa Ana, CA), were included in each image as standard colors. The rawhide paint chip approximated the darkest brown that may appear in concealed damaged almonds while the almond color paint chip approximates the lightest color of non-damaged almond kernels. The slides were digitized with a 35 mm film scanner (Nikon, Shinagawa-ku, Japan, LS-1000). The resolution of the digital images was 1296 pixels x 1944 pixels which corresponded to approximately 150 pixels per inch in real space. The color intensity resolution was 24 bit (8 bits per color channel).

The digital images were further processed with a shareware software package (Image PC, Scion Corp, Frederick, MD). First, they were converted to grayscale, then linearly histogram stretched so that the minimum and maximum corresponded to the mean intensity of the rawhide and almond paint chips, respectively. This assured that the intensity scale for all images were consistent. The gray level of the rawhide paint chip ranged from 30 to 43 and the gray level of the almond paint chip ranged from 226 to 234. Most almond images had a dark edge around the perimeter of the nut caused by the brown skin and shadows adjacent to the nut. These areas were manually set to zero intensity with an eraser tool. About 10% of the almonds had cracks in the middle of the kernel, giving a darker appearance than the surrounding kernel tissue. The cracked areas were manually set to zero intensity as well, with an eraser tool. The background was set to zero and the mean pixel intensity of non-zero pixels was computed. The percentage of pixels with an intensity greater than zero and less than 120 was also computed. The gray level threshold of 120 was found, by trial and error, to contain the darker brown regions used to characterize nuts with visual scores of four or five as discussed in table 4.3.1. Each stage of the image processing procedure is shown in figure 4.3.2.

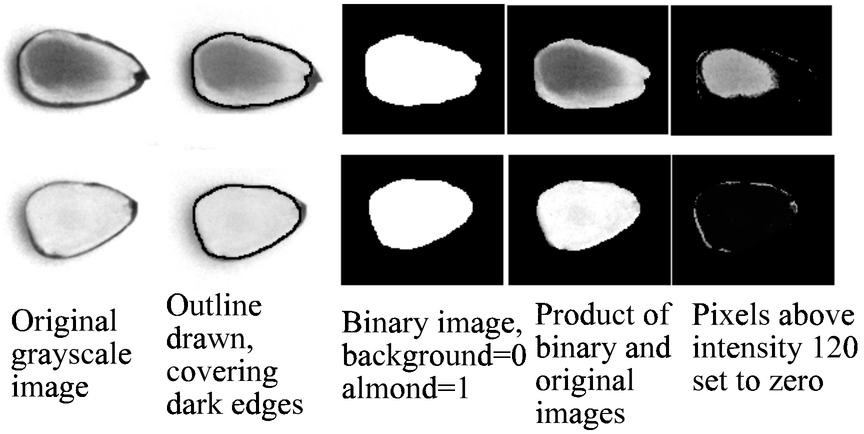

Original grayscale image
Outline drawn, covering dark edges
Binary image, background=0 almond=1
Product of binary and original images
Pixels above intensity 120 set to zero

Figure 4.3.2 Image processing of cooked almond halves. The upper almond is concealed damaged, the lower is normal.

e. Storage effects

The effect of dry storage on the incidence and severity of concealed damage was studied by storing nuts in a controlled atmosphere maintained at 10°C and 45%±5% relative humidity after exposure to the following five moisture and drying treatment combinations:

(1) long moisture and 110°C convection dry,
(2) long moisture and 55°C convection dry,
(3) short moisture and 110°C convection dry,
(4) short moisture and 55°C convection dry,
(5) no moisture treatment and no drying (control).

The long and short moisture treatments, as well as the drying treatments, were the same as discussed in section 4.3a. Batches of 81 Mission almonds were used for each moisture and drying treatment combination. These nuts were from the same group used for the spectral property measurement as discussed in section 4.3a. Nuts were stored for seven months after the moisture and drying treatments. After seven months of storage in the controlled atmosphere, nuts were removed and cooked with the same cooking treatment as discussed in section 4.3a: 135°C for 90 minutes in a convection oven (Lab-Line Instruments, Inc., Melrose Park, IL, Imperial IV). After cooking, nuts were split open at the suture, photographed, and image scored for concealed damage severity as discussed in section 4.3d.

f. Prediction of concealed damage from post-dry spectra, overview

A primary objective of this study was to determine what, if any, spectral transmission features of the pre-cooked almonds could be used to predict if a nut will become concealed damaged after cooking. Features of the pre-cook transmission spectra were selected based on their ability to correctly classify nuts as concealed damaged or normal. The mean gray level obtained from the kernel images after cooking has a high correspondence to the visual concealed damage scores as can be seen by figure 5.3.1. For the data calibration and classification of nuts into two categories, concealed damaged or normal, the mean gray level was used rather than the concealed damage scores. Nuts with a mean gray level above 160 were classified as normal and nuts with a mean gray level below 160 were classified as concealed damaged. The mean gray level of 160 was chosen as the division between concealed damage and normal nuts because this level will included all nuts with visual scores of 4 and 5 as concealed damaged. It also includes the brownest of the nuts with a visual score of three as concealed damaged. Some browning of almond kernels, typical of all almonds with visual scores of two and most with visual scores of three, are not considered a problem as these nuts do not develop bitter flavors and the degree of discoloration does not result in a negative reaction by the consumer (Stoddard, M. 1995). Bitter flavors are found in nuts with visual scores of four and five, and perhaps some of the darkest nuts with a concealed damage score of three (Stoddard, 1995).

Due to wide variations in skin quality, nut thickness, and nut shape, absorbance values were normalized by dividing each absorbance value with the mean of all values in the sampled spectrum. This data treatment helped to cancel out the effect of nut thickness, skin chips, and skin condition.

g. Prediction based on stepwise discriminant analysis

Features to classify samples into groups with discriminant analysis can be selected in a stepwise manner similar to stepwise selection for regression variables. However, when high multicollinearity exists between the features, the stepwise procedure may not select the best group of features to perform the classification. Principle component analysis was performed on the spectra to overcome this problem while developing classification models using large portions of the spectrum. Principle component analysis was performed on the entire normalized absorbance, first derivative, and second derivative spectra from 700 to 1300 nm, from 700 to 975 nm, and from 1000 to 1300 nm. The spectra used was 19 point Savitzky - Golay 2nd order smoothed and with data spaced in 5 nm increments. Absorbance values from the Ocean Optics spectrometer, between 700 and 975nm were not equally spaced. Equally spaced data in 5 nm increments was computed by 19 point interpolation. Absorbance values were normalized by dividing each absorbance value with the mean of all values in the sampled spectrum to cancel out the effect of nut thickness, skin chips, and skin condition. First derivatives were computed using the forward difference method with a 10 nm gap. Second derivatives were computed using the central difference with a 10 nm gap. Statistical analysis was performed with the SAS statistical package (SAS Institute Inc., Cary, NC) to perform a two way classification based on the mean gray level of the almond kernel. Stepwise discriminant variable selection (sle = 0.05, sls = 0.05) of the principle components was then used to choose a relatively small set of variables to be used in the discriminant function. Discriminant analysis in SAS was trained with the selected principle components on half of the data (odd numbered samples) and validated with the other half (even numbered samples). Equal a priori probabilities were used for normal and concealed damaged groups. The pool = test option was used to determine equivalence of covariance matrices. The model selected by the stepwise selection procedure was checked for over fitting on the validation set. For each model, a plot was constructed of the number of variables in the model, in the order selected, versus the error rate. One of these plots, for the model using absorbance, first derivative and second derivative data between 1000 and 1300 nm is shown in figure 5.3.3. It can be seen from this plot that removing the last variable selected will slightly improve the classification error rate in the validation set. For many of the models, the error rate of the validation set could be improved by removing one or two of the last variables selected during the stepwise selection procedure on the calibration set. The error rate is the percent number of incorrectly classified nuts in the validation set.

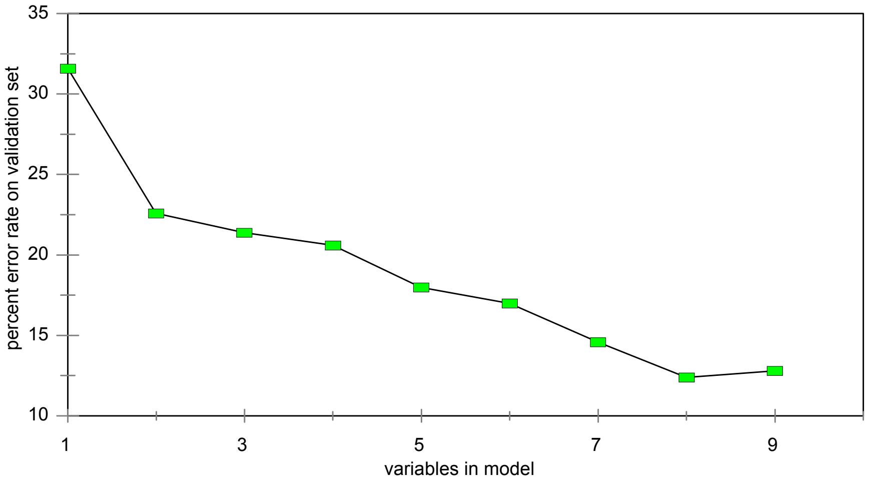


Figure 4.3.3. Example plot of percent error in validation for a discriminant model with a range of variables.

h. Prediction with discriminant analysis using all combinations of three features

The PCR and PLS methods may not select the best combinations of variables even though they take measures to eliminate multicollinearity. Discriminant or prediction models based from these whole spectrum methods may still be effected by interfering variables. Also, the design of a real-time sorting device to inspect almonds at a high speed would be simplified if only a few discrete wavelengths were needed rather than a whole spectrum. Selection of a small number of spectrum variables is difficult because of the high multicollinearity existing in spectra data. Another approach of selecting variables, or features, for a discriminant model is to evaluate all possible combinations of variables. A program, listed in appendix F, was written to perform discriminant analysis using three features to classify nuts into one of two groups. Feature selection was based on the error rate of all possible combination of three features. Nuts were considered concealed damaged if their mean gray level was below 160. One half (odd numbered samples) of the 648 nut data set were used for calibration, and the other half (even numbered samples) used for validation. Equal a priori probabilities were used for normal and concealed damaged groups. The covariance matrices for the two groups were pooled as tests performed with SAS (proc discrim pool = test) indicated that the variance of the two groups were equivalent. This analysis was performed on the portion of the spectrum from 700 nm to 975 nm with normalized absorbance values available in 5 nm increments. All combinations of features within this range were used. Normally this required about 50 hours of processing time (using a Sun Sparc station 10 computer) which necessitated the limitation to test a maximum of only three features. The analysis was performed on five different data sets: (1) normalized absorbances, (2) all first derivatives with a central difference gap less than 25 nm, (3) all second derivatives with a central difference gap less than 50 nm, (4) all possible combinations of ratios of two normalized absorbance values, (5) all possible combinations of continuous areas in the normalized absorbance spectra less than 100 nm wide. The set of three features from each data set that obtained the lowest classification error rate on the validation set was recorded. The derivatives were computed by the finite difference method with absorbance values separated by a certain wavelength gap. Equation 4.3.1 shows the computation of, A_x', the first derivative at wavelength *x*, using the normalized absorbance values and equation 4.3.2 shows the computation of the second derivative, A_x'', at wavelength *x* with a given gap

$$A_x' = \frac{\left(A_{x-gap} - A_{x+gap}\right)}{gap} \qquad (4.3.1)$$

$$A_x'' = \frac{\left(A_{x-gap} - 2A_x + A_{x+gap}\right)}{(gap)^2} \qquad (4.3.2).$$

First and second derivatives were computed with gaps ranging from 5 to 50 nm, in 5 nm increments.

i. Prediction with partial least squares regression

Partial least squares (PLS) regression was performed with the mean gray level as the dependent variable and normalized absorbance spectra as independent variables. The PLS software used, NSAS (NIRSystems, Shingle Springs, MD), required that the data set contain less than 500 samples. Thus, only the first 62 samples from each treatment were used, resulting in a data set of 496 total samples. Half of the samples in the data set were randomly selected and used for calibration, the other half were later used for validation. The calibration procedure used four internal cross validation subsets to check for over fitting. The classification performance of the PLS regression equations to classify nuts as concealed damaged or normal was tested on the validation set. Nuts were classified as concealed damaged if the predicted mean gray level was less than 160. The PLS predicted class was compared with the actual class.

j. Prediction using simulated LED absorbances

For detecting concealed damaged nuts at a rapid rate and low cost, it is most desirable to use a small number of spectral features. Preferably, the spectral features would be obtained from wide segments of the spectrum to improve signal strength. Whole spectrum methods using principle components or partial least squares require the capture of a complete spectrum, or large portions of a spectrum. However, hardware to obtain and process a whole spectrum rapidly would be more expensive than a system using a small number of light bands based on LEDs. Low cost LEDs for use in a high speed sorter were selected by examining the specific wavelengths in the three variable discriminant models (table 5.3.7).

The feasibility of using light emitted by LEDs to rapidly detect nuts with concealed damage was studied using the acquired transmission spectra and emission spectra from several infrared LEDs. To simulate the total light absorbance associated with a LED, normalized emission spectra from several LEDs were multiplied and integrated, one LED at a time, with each sample absorbance spectra. The LEDs used in this simulation had peak emission wavelengths of 700, 830, 840, 850, 880, 890, 940, 950 nm. The part number and manufacturer of these 8 LEDs are listed in table 4.3.3. The peak emission wavelengths of these 8 LEDs comprise all that were readily available after surveying several LED manufacturers. Other peak emission wavelengths between 700 and 950 nm are reportedly available but not readily and not without a bulk order. All of the LEDs used for the simulation had half peak bandwidths of 40 to 60 nm with approximately gaussian shaped emission spectra. The actual spectra used for the simulation were obtained from the manufactures. For this feasibility study, no effort was made to check the LED wavelength correspondence, or calibration, with the spectrometer used to obtain the almond transmission spectra. In not performing this check, correspondence to a real system may suffer. However, the objective of this feasibility study was to determine if a sorting device could be constructed from LEDs, not to select the ideal set of LEDs to be used.

Table 4.3.3. Sources of LEDs used in the simulation experiment.

LED peak wavelength (nm)	manufacturer part number	manufacturer
700	VL700XA	UDT Sensors, Inc., Hawthorne, CA
830	L3989-01	Hamamatsu, Hamamatsu City, Japan
840	IR-840T5	UDT Sensors, Inc., Hawthorne, CA
850	IR-850T5	UDT Sensors, Inc., Hawthorne, CA
880	L2791-02	Hamamatsu, Hamamatsu City, Japan
890	L2690-02	Hamamatsu, Hamamatsu City, Japan
940	L2388-01	Hamamatsu, Hamamatsu City, Japan
950	LN54	Panasonic, Osaka, Japan

Principle components were computed for the responses of all LEDs and all possible ratios of LEDs and selected by the stepwise discriminant analysis procedure (sle = 0.05 and sls = 0.05). Redundant ratios were not used, resulting in a total of 28 ratios. Nuts were classified into one of two groups, normal or concealed damaged if their mean gray level was above or below 160, respectively. Odd numbered samples were used for calibration and even numbered samples used for validation. Stepwise discriminant analysis was performed on principle components of all LEDs, and on principle components of all variable sets leaving one, two, or three LEDs left out.

4.4 Real-time sorting device

a. Prototype design

A prototype device to detect almonds with concealed damage after drying but before cooking was developed. A constraint of the prototype design was that it had to be able to inspect nuts at a rate of 40 nuts per second which is comparable to the rates of automated color sorters at almond processing plants. To achieve this rate with a single channel system, almonds need to travel in a single file stream at a speed of approximately 1.0 m/s. To maintain a sorting rate of 40 nuts per second, a maximum of 25 ms can be used to acquire the necessary information, process it, and activate an air nozzle to divert the nut from the stream if it is determined to contain concealed damage. Normal photodiode array spectrometers cannot currently acquire a full transmission spectrum of a whole almond in 25 ms using a 100 W halogen illumination source..

The inspection device measures transmitted light from six different LEDs as shown in figure 4.4.1. Table 4.4.1 lists the part number and manufacturer of the six different LEDs used while figure 4.4.2 displays the actual normalized emission spectra from each of the LEDs measured by a spectrometer (Ocean Optics, #PC1000, Dunedin, FL). The LEDs used for the prototype had emission wavelengths of 660, 830, 880, 890, 940, and 950 nm. These six LEDs were chosen from the set of LEDs studied in the simulated LED experiment, section 5.3j (table 4.3.3). The 660 nm LED was not studied in section 4.3j because the transmission spectra was low pass filtered at 700 nm. However, the 700 nm LED was not readily available so the 660 nm LED was added as it has a closer peak emission wavelength than any near infrared LED that was readily available. The 840 nm and 850 nm LEDs, studied in section 4.3j, did not emit light with sufficient power to transmit through whole almonds. A light intensity modulation/demodulation scheme diagramed in figure 3.12.1 and discussed in section 3.12 was used for the prototype design. The light intensity of each LED was sine wave modulated at a different frequency. A 50 mm diameter plano-convex lens (Edmund Scientific Co., Barrington, NJ, #E32,970) focused the light emitted from all the LEDs onto a short single, 3 mm diameter, fiber optic cable, (Edmund Scientific Co., Barrington, NJ, #P38659). Th fiber optic cable was used to diffuse the light and deliver it to the nut. The light emitted from all LEDs needed to be diffused so that the path length that light travels through a nut from each LED would be identical. The transmitted light through the nut was detected by an avalanche photodiode module (Hamamatsu, Hamamatsu City, Japan, #C5460). The dimensions of the sensor in this avalanche photodiode was a 1.5 mm diameter circle. The photodiode module was not temperature compensated but the temperature in the room where all testing was performed was maintained at 25°C ± 2°C. The signal from the avalanche photodiode was input to a digital signal processing board (DSP) (Dalanco Spry, Rochester, NY, #310). The DSP board was equipped with a TMS320C31 digital signal processor and mounted into an ISA slot of a 60 MHz Pentium personal computer

(Micron Electronics, Inc., Nampa, ID). The DSP board performed a discrete Fourier transform on the input signal from the avalanche photodiode. The demodulated response from each LED was transferred to the PC where the response from each LED was used to classify the nut.

A schematic of the driver circuit for the LEDs is shown in figure 4.4.3. The modulating sine waves were generated with precision, serially programable, sine wave generators (Micro Linear, San Jose, CA, ML2035). The sine wave generators output a sine wave at a frequency determined by an input clock frequency and a 16 bit digital word serially programed into the sine wave generator. The sine wave frequency, f_{out}, was given by equation 4.4.1 which was provided by the sine wave generator manufacturer

$$f_{out} = \frac{(f_{clock})(x)}{2^{23}} \qquad (4.4.1)$$

where x was the decimal equivalent to the 16 bit word and f_{clock} was the input clock frequency. The maximum sine wave frequency for the ML2035 was 25KHz. To obtain the maximum frequency, per equation 4.4.1, a minimum input clock frequency of 3.2MHz must be provided. A 3.6864MHz TTL input clock (SaRonix, Palo Alto, CA, #S1500) was used as this was one of the lowest frequency clocks, readily available. This input clock frequency gave the sine wave generator a resolution of 0.44Hz. The 16 bit word, to determine the output of a sine wave generator, was output by a PC digital interface card (Keithly Metrabyte, Taunton, MA, PIO-24). The program controlling the digital output card to send it the proper 16 bit number is listed in appendix G. The 16 bit word is transmitted serially to the sine wave generator from least significant bit first to most significant bit last. Sine wave modulating frequencies of 9KHz, 12KHz, 15KHz, 18KHz, 21KHz, and 24KHz were used. These frequencies were chosen because their periods were all integer multiples of the sampling period, 1.67 ms. The DSP board was programed to sample 300 points at 180KHz which requires 1.67ms. Furthermore, the resolution of the discrete Fourier transform (DFT) with these sampling parameters was 600Hz. Therefore, each peak in the frequency spectrum due to a modulation frequency coincided with a computed point in the DFT.

The magnitude of the sine wave driving the LEDs was from -2.5 volts to +2.5 volts. The cathodes of all LEDs were connected to a -5 volt source, resulting in a voltage differential across the LED ranging from 7.5 volts to 2.5 volts. The minimum of a 2.5 volt differential across the LEDs was required to prevent to the LED from turning completely off and losing the sine wave shape of the emitted light intensity. The combined light from all LEDs transmitted through almonds caused to avalanche photodiode output signal to range from 0.5 volts to 3.0 volts. The input voltage range of analog to digital (A/D) converter on the DSP board was 0.0 to 5.0 volts. The A/D converter had a resolution of 14 bits so the digital output of the A/D converter ranged from approximately 1640 to 8200 (i.e. between 12 or 13 bits).

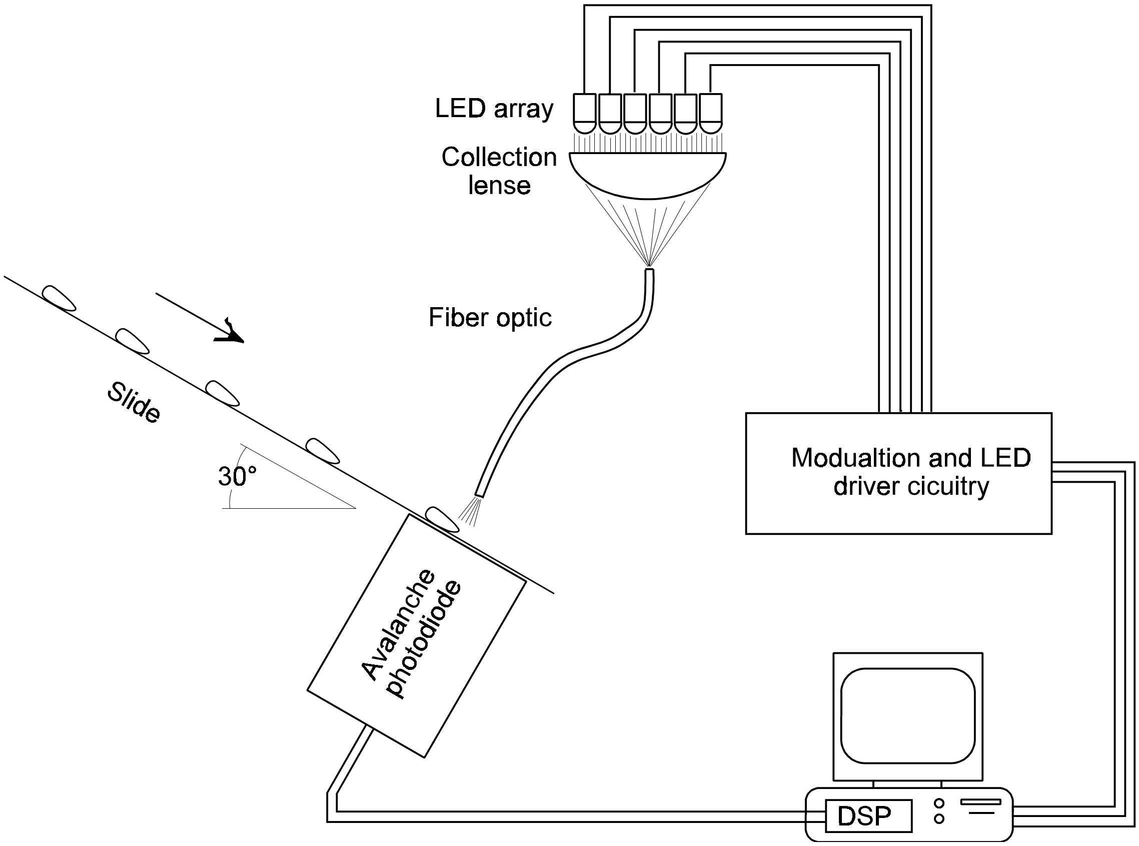


Figure 4.4.1. Schematic of sorting machine.

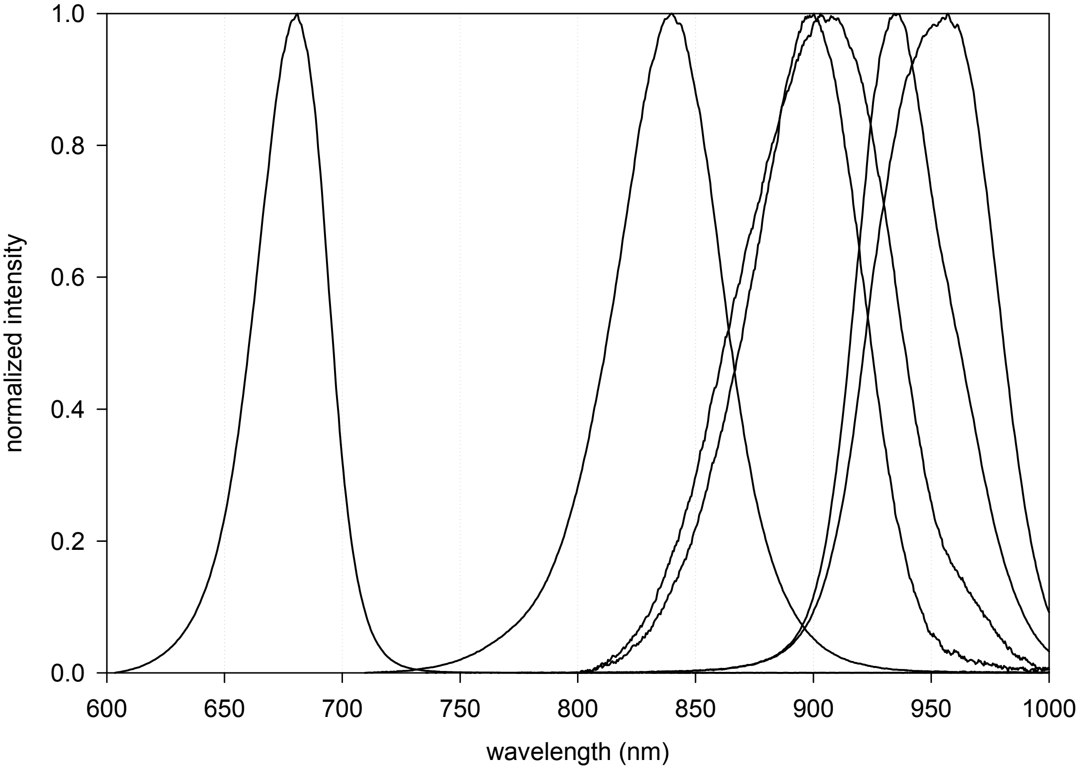


Figure 4.4.2. Normalized emission spectra from the six LEDs used on the sorting machine. Spectra obtained after LEDs were burned in for 140 hours.

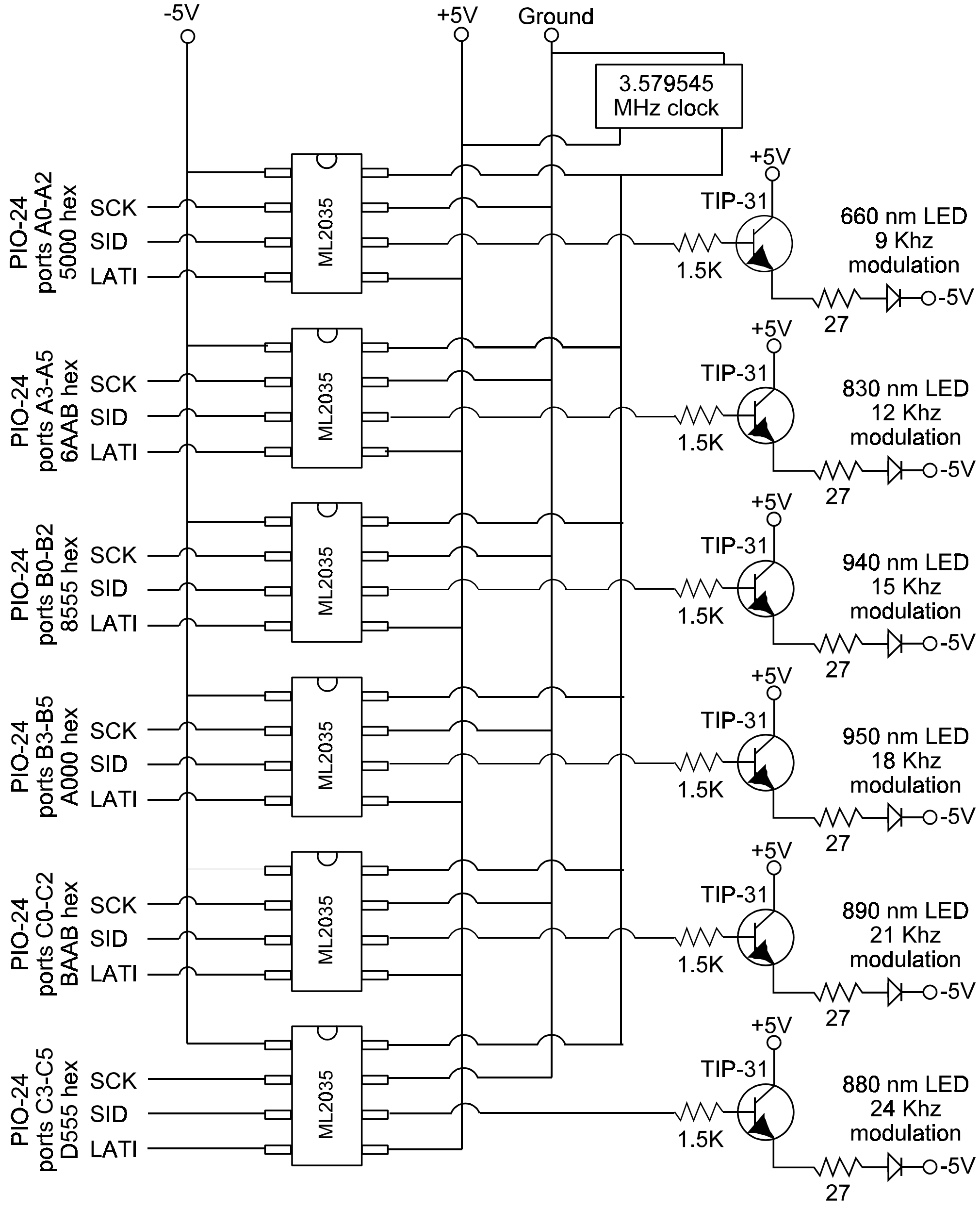


Fig ure

4.4.3. LED driver circuit for automated sorting machine.

Table 4.4.1. Sources of LEDs used for the prototype and modulation frequencies.

LED peak wavelength (nm)	modulation frequency (KHz)	manufacturer part number	manufacturer
660	9	SL660WCT3	UDT Sensors, Inc., Hawthorne, CA
830	12	L3989-01	Hamamatsu, Hamamatsu City, Japan
880	21	L2791-02	Hamamatsu, Hamamatsu City, Japan
890	24	L2690-02	Hamamatsu, Hamamatsu City, Japan
940	15	L2388-01	Hamamatsu, Hamamatsu City, Japan
950	18	LN54	Panasonic, Osaka, Japan

To minimize analysis time, the DFT of the avalanche photodiode output signal was computed for only the six modulating frequencies of the LEDs. This method allowed computation of the six DFT points concurrently with the data acquisition. The trigonometric form of the DFT equation (3.8.6) is shown here again as equation 4.4.2

$$\widetilde{F}(\frac{n}{NT_s}) = \sum_{k=0}^{N-1}\left[f(kT_s)w(kT_s)*\Delta'(kT_s)\right]\left[\cos(2\pi nk/N) - j\sin(2\pi nk/N)\right] \qquad (4.4.2)$$

where $\widetilde{F}(\frac{n}{NTs})$ is the discrete Fourier transform at frequency n/NT_s, N is the total number of samples (300), T_s is the time interval between samples, n is an integer set to determine the frequency of the specific DFT point, $f(t)$ is the signal from the avalanche photodiode, $w(t)$ is the window function, $\Delta'(t)$ is the sampling function, and k is an integer from 0 to N-1 defining the sample number in sequential order. The real and imaginary components of equation 4.4.2 can be computed each time a sample is acquired and a running sum of these quantities can be stored. Thus, upon acquiring N number of samples, the value of equation 4.4.2 only needs to be computed for k=N-1 and added to the running sum to obtain the DFT at frequency n/NT_s. The PC program listed in appendix H computes the Hanning window value for each k, the cos($2\pi nk/N$), and sin($2\pi nk/N$) values for each k at each of the six modulation frequencies specified by n. These parameters are loaded to the DSP memory and called by the DSP when needed. The program listed in appendix I is an assembly language program to control the analog to digital conversion and perform the DFT on the DSP. When a nut is not present, the photodiode saturates, resulting in a DC signal with near zero frequency response at the modulating frequencies. When the photodiode signal drops below a set threshold, then the presence of a nut is detected and the DSP computes twelve consecutive DFTs. Computation of twelve DFTs requires approximately 20 ms, which is within the time frame allowed to inspect 40 nuts per second. Occasionally, a nut would pass by the photodiode in less than 20 ms, either because the nut was of a short length, or it was sliding faster than normal. When this happens, the avalanche photodiode would saturate and output a DC signal at approximately five volts. The PC program (appendix H) checks that all magnitudes at the modulation frequencies in each of the twelve DFTs exceed a preset threshold to assure that the photodiode is not saturated. If twelve good DFTs were acquired, they would be stored. If twelve good DFTs are not acquired, the PC will flag the user and the nut would be run again until twelve good DFTs are acquired.

The signal from the avalanche photodiode was very noisy due to the high gain required to detect the transmitted light through whole almonds. It was found that use of a Hanning window function, rather than a rectangular window function, applied over the sampling period significantly reduced the effect of noise. To test the noise effects on the resulting DFT, an almond was held over the photodiode and 30 responses, each comprising the average of 12 DFTs, from each LED were recorded. With a rectangular sampling window function, the highest coefficient of variance of the mean LED response was 11%. In contrast, when a Hanning sampling window function was used, the highest coefficient of variance of the mean LED response was found to be 0.6%.

b. Prototype testing

The prototype was evaluated using a sample of 324 Mission almonds from the 1997 harvest. Nuts were exposed to the long moisture treatment as described in section 4.3a. Half of these nuts were then dried to their original bulk mass in an air convection dryer at 55°C and the other half dried at 110°C as described in section 4.3a. These nuts and 81 control nuts, or nuts not exposed to moisture and drying treatments, were individually inspected by the prototype device. Nuts were inspected in separate batches of 81 nuts. Before and after inspecting a batch, LED light emission standards were measured by placing a 0.1% transmission neutral density filter (Ealing, Holliston, MA, #35-5941) between the photodiode and the fiber optic and the photodiode signal was sampled as if a nut were present. The average of the two LED emission standards were used to normalize the absorbance for each nut in the batch. The LED emissions measured before and after each batch never deviated more than 1% from each other. The 12 DFTs obtained while the nuts were sliding by the photodiode were stored for analysis. After obtaining the LED light transmission for all nuts, they were cooked for 90 minutes at 135°C in a convection oven as described earlier in this section. After cooking, the nuts were split at the suture, photographed, and the mean gray level of the kernel half was measured as discussed earlier in section 4.3d.

In addition to testing the prototype with almond samples, the consistency of the 12 DFTs were tested by sliding a small rectangular piece of Teflon through the device. When no nut is present between the photodiode and fiber optic, the photodiode is saturated by the unobstructed light from the LEDs. The step response of the photodiode in going from a saturated state to accurately measuring the modulated LED signal when a nut suddenly blocks the light incident on the photodiode needed to be determined. As a nut passed by the photodiode, twelve DFTs were taken. This, in effect, samples each LED twelve times across the length of the nut. In an ideal case, the twelve LED samples for each individual LED should be the same when the Teflon piece is inspected by the prototype. However, due to frequency response limitations of the photodiode, this may not be the case. The main objective of this test is to determine if the first few DFTs are equivalent to the rest of the DFTs. The Teflon piece (McMaster-Carr, Los Angeles, CA, #873K13) was 6.35 mm thick, 38.1 mm long, and 12.5 mm wide. The Teflon piece was passed through the prototype 60 times and the 12 DFTs acquired for each pass were saved in sequential order. This resulted in a data set containing 60 sets of twelve DFTs numbered, one through twelve, in the order taken. Afterwards, Tukey's Studentized Range Test (SAS Institute Inc., Cary, NC, proc glm), with $\alpha = 0.05$, was used to test the equivalence of the mean DFT value for each of the twelve sequential samples for each individual LED. The means of the twelve DFT values were tested for all six LEDs. The means of the twelve samples for a particular LED determined to be not significantly different from each other at the $\alpha = 0.05$ level were averaged and used for prediction of concealed damage.

c. Prediction of concealed damage with stepwise discriminant analysis

A discriminant function was developed to classify nuts into one of two categories, concealed damaged or normal with LED absorbance values, ratios of absorbance values, and differences in absorbance values. The discriminant analysis was performed with the SAS proc discrim and proc princomp (SAS Institute Inc., Cary, NC). Each LED absorbance value was normalized by the mean absorbance values of all six LEDs. All normalized LED absorbance values, all possible ratios of normalized absorbance values and all possible differences of two normalized absorbance values were computed. Redundant ratios and differences were not used, leaving a total of 15 each. Table 4.4.2 shows the ratios and differences used in the analysis. Three sets of principle components were computed, one for the normalized absorbance values, a second set for ratios, and a third for the differences. Principle components were selected for classifying nuts as concealed damaged or normal with stepwise discriminant analysis using a significance for entry and elimination from the model of 0.05. Equal a priori probabilities for normal and concealed damage were used. Covariance matrices were tested for equivalence using the pool = test option in SAS. The stepwise selection was trained using odd numbered samples only. Nuts in this training set were considered concealed damaged if their mean gray level, measured from imaging after cooking, was below 160. The even numbered samples were used as a validation set. Discriminant analysis was performed using all variables selected by the stepwise procedure. After performing discriminant analysis with all variables selected by the stepwise procedure, the least significant principle component, determined from the stepwise procedure, was eliminated and discriminant analysis was performed again. This was repeated until the error rate of the validation set reached a minimum.

Table 4.4.2. List of ratios and differences used for analysis. Subscript denotes peak emission wavelength of the LED in nm.

ratios	differences
A_{830}/A_{660}	A_{830}-A_{660}
A_{880}/A_{660}	A_{880}-A_{660}
A_{890}/A_{660}	A_{890}-A_{660}
A_{940}/A_{660}	A_{940}-A_{660}
A_{950}/A_{660}	A_{950}-A_{660}
A_{880}/A_{830}	A_{880}-A_{830}
A_{890}/A_{830}	A_{890}-A_{830}
A_{940}/A_{830}	A_{940}-A_{830}
A_{950}/A_{830}	A_{950}-A_{830}
A_{890}/A_{880}	A_{890}-A_{880}
A_{940}/A_{880}	A_{940}-A_{880}
A_{950}/A_{880}	A_{950}-A_{880}
A_{940}/A_{890}	A_{940}-A_{890}
A_{950}/A_{890}	A_{950}-A_{890}
A_{950}/A_{840}	A_{950}-A_{840}

d. Prediction of concealed damage with regression analysis

A prediction equation was developed with mean gray level of the cooked almond kernels as the dependent variable and principle components of the LED data as independent variables. As with the discriminant analysis procedure in the previous section, three sets of principle components were computed, one for normalized absorbance values, a second set for ratios of normalized absorbance values, and a third for differences between normalized absorbance values. The same 15 ratios and 15 differences shown in table 4.4.2 were used for this analysis as well. Principle components were selected for classifying nuts as concealed damaged or normal with by comparing the adjusted R^2 of all possible combinations of principle components. The odd numbered samples were used for calibration while the even numbered samples were used as a validation set. After model selection by the adjusted R^2 procedure on the calibration set, the last principle component, determined from the stepwise procedure on the validation set, was eliminated and regression was performed again. This was repeated until the standard error of prediction of the validation set reached a minimum. For classification, nuts were considered concealed damaged if their predicted mean gray level was below 160.

For comparison purposes, regression analysis was also performed with the mean gray level as the dependent variable and the six absorbance, fifteen ratio and fifteen difference values as independent variables without transforming them into principle components. Regression models were selected by comparing the model adjusted R^2 of all possible combinations of ten or less variables. To speed the computation of adjusted R^2 for the models, the effect of multicollinearity was reduced by eliminating highly correlated ($|r|>0.9$) independent variables. A correlation matrix for all independent variables was computed. If two independent variables were highly correlated ($|r| > 0.9$), then the variable with the lower correlation with mean gray level was eliminated. When no independent variables remained with $|r| > 0.9$, then adjusted R^2 model selection was performed on the remaining variable set.

5. RESULTS AND DISCUSSION

5.1 Field Experiment Results

The mean temperature, across time, of the nuts from all windrow treatments remained within 0.2°C of the mean ambient temperature for the duration of the three day temperature monitoring. The minimum and maximum ambient temperature always deviated from the minimum and maximum temperature of the nuts, respectively, in the windrows by 2°C to 4°C. This suggests that respiration of the wet almonds and leaves in the windrow is not raising the temperature of nuts within the windrow. The average ambient temperature during the Nonpareil experiment was 25.2°C and the average ambient temperature for the Mission almond experiment was 19.2°C. The temperature of the nuts from each treatment always remained within 1°C of each other. The moisture contents for each day for all treatments of Mission almonds are shown in figure 5.1.1. Similar moisture trends were seen with the Nonpareil almonds except that their initial kernel moisture content was 12.6% d.b. The 3 cm irrigated windrow samples reached 21.7% moisture after four days. The incidence of almonds with concealed damage, observed after 5 days in the orchard, increased with more severe irrigation treatments and incidence was higher among nuts held in windrows compared to scatted nuts, as can be seen in table 5.1.1.

Table 5.1.1. Incidence of concealed damaged nuts observed after drying samples removed after five days in the orchard.

irrigation	scattered nuts with concealed damage (%)	windrowed nuts with concealed damage (%)
none	0	0
1 cm	4	12
3 cm	3	16

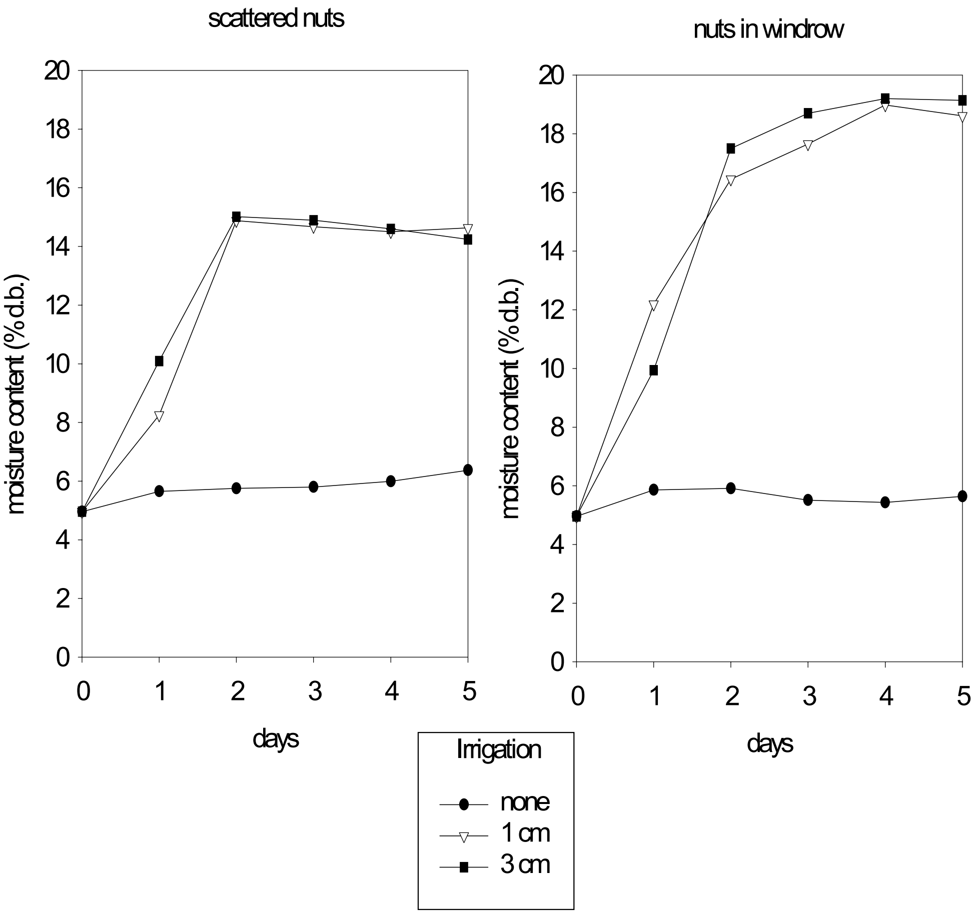


Figure 5.1.1. Moisture content changes for windrowed and scattered nuts for treatments of 0 cm, 1 cm, and 3 cm sprinkle irrigation.

5.2 Chemical tests

a. Batch tests

The results of the chemical assay tests are shown in table 5.2.1. The results of the color and concealed damage incidence tests are shown in table 5.2.2. A correlation matrix of all measured properties after cooking is displayed in table 5.2.3. Tables 5.2.4 through 5.2.8 show the results of Tukey's Studentized Range (HSD) Test, α = 0.05, on means of the chemical and color measurements after each treatment. Table 5.2.4 lists the means test results on the chemical measurements after nuts were moisture treated but before drying. Tables 5.2.5 and 5.2.6 list the results of the means tests on the chemical and color measurements obtained after drying but before cooking. For table 5.2.5, the results of the different drying treatments, measured after drying, are pooled and the means from the different moisture treatments are tested. Conversely, in table 5.2.6, the results of the different moisture treatments, measured after drying, are pooled and the means from the different drying treatments are tested. The means test results for chemical and color measurement obtained after cooking are shown in table 5.2.7 for the different moisture treatments, with drying effects pooled; and, in table 5.2.8 for the different drying treatments, with the moisture effects pooled.

It appears that most of the carbonyl group reactants are formed during exposure to moisture. After both the long and short moisture treatments, there was a significant decrease ($\alpha \leq 0.05$) in sucrose and increase in glucose between the control and moisture treated samples, table 5.2.4. The loss of sucrose and increase in glucose suggests invertase activity is elevated when dormant nuts become rehydrated. This has also been observed in almonds by Reil et al. (1996) and in macadamia nuts by Prichavudhi and Yamamoto (1965). Also after both the long and short moisture treatments, there was a significant decrease ($\alpha \leq 0.05$) in oil refractive index, table 5.2.4. The mean oil refractive index from the short moisture treatment was significantly less than the control and the mean oil refractive index from the long moisture treatment was significantly lower than the short moisture treatment mean, table 5.2.4. This suggests increased oil oxidation at higher moisture contents. The oxidized fatty acid can form into a carbonyl compound and bind to proteins just as sugars do.

As can be seen from tables 5.2.2 and 5.2.6, the color of almond kernels can change during the high temperature drying process. The mean C.I.E. L* and a* values of the high temperature drying treatment are significantly different ($\alpha \leq 0.05$) than the control and low temperature drying treatments. This suggests that the Maillard reaction proceeds into the intermediate stage during high temperature drying. A characteristic of the Maillard reaction intermediate stage is a slight yellowing of the food (Hodge and Osman, 1976). Also after drying, the mean soluble solids content and sucrose for the low and high temperature drying treatments, as well as the two moisture treatments are significantly lower ($\alpha \leq 0.05$) than the controls, tables 5.2.5 and 5.2.6. The loss of soluble solids can be due to sugars binding to proteins and losing their solubility. Also, protein folding during the intermediate stage of the Maillard reaction reduces soluble solids content.

Basic amino acid concentration measured after cooking had the highest correlation ($r = -0.84$) with incidence of concealed damage observed after cooking. Of the color measurements, C.I.E. L* value of cooked nuts had the highest correlation ($r = -0.75$) to the observed incidence of concealed damage. The lower the C.I.E. L* value, the darker the cooked almond powder appeared to the eye. Possibly, cracks on the sliced surface of the kernel added error into the C.I.E. L* measurement. Also, the C.I.E. a* chromaticity of cooked nuts had the highest correlation ($r = -0.77$) with the basic amino acid concentration.

The incidence of concealed damage is significantly reduced for the lower temperature drying experiments. None of the control nuts had concealed damage, 1.2% of the low temperature dried nuts had concealed damage, while 15.0% and 9.6% of the high temperature dried nuts had concealed damage with long moisture exposure and short moisture exposure respectively. As can be seen from table 5.2.8, the mean C.I.E. L* value, mean basic amino acids and mean incidence of concealed damage for the low temperature drying treatment are not significantly different than the control at the 0.05 level. In contrast, mean C.I.E. L* value, mean basic amino acids and mean incidence of concealed damage for the high temperature drying treatment are significantly different, at the 0.05 level, than the control and low temperature drying treatments. These results suggest that lower drying temperatures can prevent browning due to concealed damage. This observation will be supported further in the discussion of the individual nut physical property tests, section 5.3b.

An ANOVA was performed with concealed damage incidence as the dependent variable and the moisture treatment (none, short, long), drying treatment (none, low, high), and the interaction of moisture and drying treatments as the independent variables. The R^2 for the model was 0.95. Moisture and drying treatment were significant at the 0.0001 level while moisture-drying interaction was significant at the 0.001 level.

The water soluble solids content measured after cooking had a high correlation (r = -0.77) with incidence of concealed damage. From table 5.2.1, the loss of water soluble solids appears to occur during the drying process. The loss of water soluble products is expected in the early and intermediate stages of the Maillard reaction. In the early stage, sugars lose solubility as they bind with proteins. In the intermediate stage, protein folding causes additional loss of soluble solids (Hodge and Osman, 1976).

Table 5.2.1. Mean and standard deviation of all chemical assays.

moist treatment	dry temp (°C)	cooked?	final moisture % dry basis		glucose mg/g dry matter		sucrose mg/g dry matter		% soluble solids in dry matter		basic amino acids mmol/g dry matter		oil refractive index	
			mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
none	none	no	6.0%	0.0%	0.09	0.06	28.50	0.14	30.1	0.7	0.201	0.003	1.4697	0.0001
short	none	no	18.8%	1.2%	1.44	0.21	25.58	0.33	29.3	1.0	n/a		1.4642	0.0002
long	none	no	15.7%	0.3%	1.58	0.19	25.29	0.46	29.2	0.5	n/a		1.4637	0.0001
short	55	no	6.0%	0.0%	0.55	0.06	25.32	0.28	27.9	0.4	n/a		1.4620	0.0001
short	110	no	6.0%	0.0%	0.06	0.03	25.26	0.29	27.7	0.4	n/a		1.4640	0.0005
long	55	no	6.0%	0.0%	0.48	0.05	25.19	0.30	27.9	0.2	n/a		1.4625	0.0002
long	110	no	6.0%	0.0%	0.04	0.01	25.24	0.13	26.7	0.6	n/a		1.4640	0.0001
none	none	yes	3.1%	0.1%	0.00	0.00	25.30	0.25	28.4	0.6	0.190	0.003	1.4679	0.0001
short	55	yes	3.7%	0.2%	0.00	0.00	25.26	0.21	29.7	0.3	0.200	0.003	1.4644	0.0001
short	110	yes	4.1%	0.1%	0.00	0.00	25.07	0.17	27.4	0.5	0.183	0.002	1.4652	0.0001
long	55	yes	4.1%	0.1%	0.00	0.00	25.25	0.14	31.2	0.5	0.198	0.007	1.4648	0.0002
long	110	yes	3.4%	0.2%	0.00	0.00	25.11	0.20	26.7	0.4	0.173	0.004	1.4648	0.0001

Table 5.2.2. Color and concealed damage incidence results.

			CIE color space						incidence	
moist	dry		L*		a*		b*		of CD	
treatment	temp (°C)	cooked?	mean	std	mean	std	mean	std	mean	std
none	none	no	63.4	0.9	3.3	0.3	11.5	0.4	0.0%	0.0%
short	none	no	57.2	1.5	3.3	0.3	10.9	0.5	n/a	
long	none	no	56.2	0.3	3.3	0.1	10.5	0.3	n/a	
short	55	no	61.0	0.8	3.8	0.2	11.4	0.4	n/a	
short	110	no	60.6	0.4	4.0	0.1	12.4	0.3	n/a	
long	55	no	64.9	0.4	3.1	0.1	11.9	0.1	n/a	
long	110	no	58.4	0.5	5.0	0.1	11.7	0.3	n/a	
none	none	yes	60.3	0.7	4.7	0.1	12.1	0.2	0.0%	0.0%
short	55	yes	62.2	0.7	3.9	0.2	12.2	0.2	1.2%	1.6%
short	110	yes	56.0	0.5	5.2	0.1	12.4	0.2	9.6%	1.6%
long	55	yes	58.5	0.6	4.7	0.2	11.3	0.4	1.2%	1.3%
long	110	yes	56.9	0.4	5.2	0.1	12.4	0.2	15.0%	2.1%

Table 5.2.3. Correlation matrix of measured chemical and physical properties on nuts after cooking. Correlation values with a magnitude greater than 0.5 are listed in bold.

	final moisture content	glucose	sucrose	CIE L* value	CIE a* value	CIE b* value	oil refractive index	% soluble solids	basic amino acids
final moisture content									
glucose	**0.81**								
sucrose	**0.90**	**0.87**							
CIE L* value	0.49	**0.56**	**0.66**						
CIE a* value	**-0.68**	**-0.66**	**-0.77**	**-0.94**					
CIE b* value	-0.45	-0.26	-0.45	0.34	0.46				
oil refractive index	**0.62**	**0.71**	**0.82**	**0.59**	**-0.58**	0.31			
% soluble solids	0.42	0.29	0.36	**0.55**	**-0.60**	**-0.72**	0.17		
basic amino acids	0.46	0.34	0.45	**0.75**	**-0.77**	**-0.53**	0.33	**0.82**	
concealed damage incidence	-0.30	-0.30	-0.39	**-0.75**	**0.70**	**0.54**	-0.48	**-0.77**	**-0.84**

Table 5.2.4. Moisture treatment means test results for chemical measurements after moisture treatment but before drying. Nuts were not exposed to any drying or cooking treatments. Means with the same group letter are not significantly different at the 0.05 level determined by Tukey's Studentized Range (HSD) Test.

	glucose mg/g dry matter		sucrose mg/g dry matter		soluble solids (%)		oil refractive index	
moisture treatment	group	mean	group	mean	group	mean	group	mean
none	A	0.09	A	28.50	A	30.14	A	1.4697
short	B	1.44	B	25.56	A	29.30	B	1.4642
long	B	1.58	B	25.30	A	29.20	C	1.4637

Table 5.2.5. Moisture treatment means test results for chemical and color measurements taken after drying but before cooking. Data from the different drying treatments was pooled. Means with the same group letter are not significantly different at the 0.05 level determined by Tukey's Studentized Range (HSD) Test.

	glucose mg/g dry matter		sucrose mg/g dry matter		soluble solids (%)		oil refractive index		CIE L*		CIE a*		CIE b*	
moisture treatment	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean
none	A	0.09	A	28.50	A	30.14	A	1.470	A	63.44	A	3.34	A	11.52
short	C	0.31	B	25.29	B	27.83	B	1.463	B	61.63	B	3.92	A	11.90
long	B	0.24	B	25.22	B	27.33	B	1.463	B	60.82	B	4.01	A	11.82

Table 5.2.6. Drying treatment means test results for chemical and color measurements taken after drying but before cooking. Data from the different moisture treatments was pooled. Means with the same group letter are not significantly different at the 0.05 level determined by Tukey's Studentized Range (HSD) Test.

	glucose mg/g dry matter		sucrose mg/g dry matter		soluble solids (%)		oil refractive index		CIE L*		CIE a*		CIE b*	
drying treatment	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean
none	A	0.09	A	28.50	A	30.14	A	1.470	A	63.44	A	3.34	A	12.04
low	B	0.52	B	25.26	B	27.92	B	1.464	A	62.94	A	3.43	A	11.68
high	A	0.03	B	25.25	B	27.24	C	1.462	B	59.51	B	4.50	A	11.52

Table 5.2.7. Moisture treatment means test results for measured chemical and color measurements taken after cooking. Data from the different drying treatments was pooled. Means with the same group letter are not significantly different at the 0.05 level determined by Tukey's Studentized Range (HSD) Test.

	soluble solids (%)		oil refractive index		CIE L*		CIE a*		CIE b*		basic amino acids mmol/g dry matter		concealed damage incidence (%)	
moisture treatment	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean
none	A	28.38	A	1.468	A	60.28	AB	4.68	AB	12.1	A	0.190	A	0.0
short	A	28.54	B	1.465	AB	59.07	B	4.55	A	12.33	A	0.192	B	5.4
long	A	28.95	B	1.465	B	57.73	A	4.94	B	11.85	A	0.185	C	8.1

Note: means of glucose and sucrose, not shown in this table, were not significantly different for all three drying treatment levels.

Table 5.2.8. Drying treatment means test results for measured chemical and color measurements taken after cooking. Data from the different moisture treatments was pooled. Means with the same group letter are not significantly different at the 0.05 level determined by Tukey's Studentized Range (HSD) Test.

	soluble solids (%)		oil refractive index		CIE L*		CIE a*		CIE b*		basic amino acids mmol/g dry matter		concealed damage incidence (%)	
drying treatment	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean	group	mean
none	B	28.38	A	1.468	A	60.28	A	4.68	AB	12.1	AB	0.190	A	0.0
low	A	30.45	B	1.465	A	60.33	B	4.30	B	11.75	A	0.199	A	1.2
high	C	27.04	C	1.465	B	56.47	C	5.19	A	12.43	B	0.178	B	12.3

Note: means of glucose and sucrose, not shown in this table, were not significantly different for all three drying treatment levels.

b. Post-dry water soluble solids content and post-cook color

A good correlation between post-cook color and post-dry soluble solids was found from the experiment measuring soluble solids contained after drying but before cooking and internal kernel color after cooking. The post-cook C.I.E. a* value had the highest correlation, r = -0.87, with post-dry soluble solids while the post-cook C.I.E. L* value had a correlation, r = 0.74, with post-dry soluble solids. A plot of the C.I.E. a* chromaticity measured on the cooked kernel half and water soluble solids measured from the half before cooking is shown in figure 5.2.1.

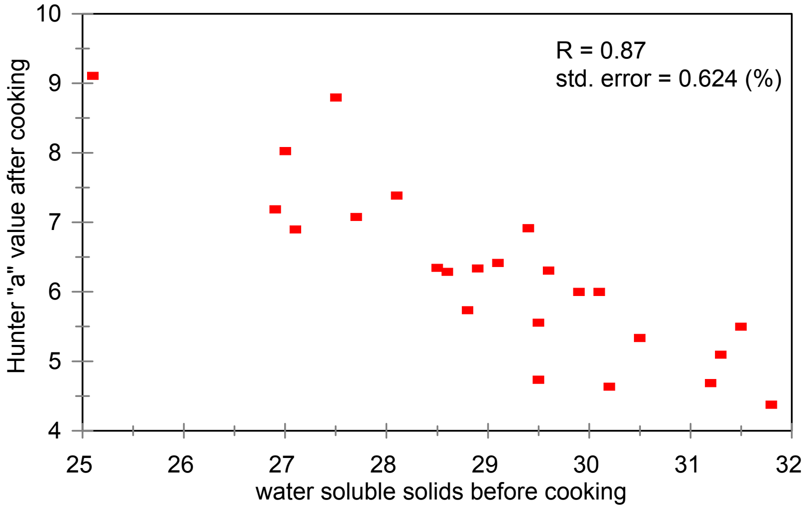


Figure 5.2.1. Plot of pre-cook water soluble solids versus post-cook C.I.E. a* value.

5.3. Physical and spectral property tests

a. Scoring concealed damage severity

An image based scoring system was developed with the mean pixel intensity and percent of pixels below intensity 120. The image based scoring was performed with discriminant analysis using the mean pixel intensity and percent of pixels darker than 120 as features and the visual scores as the classes. The classes defined by the discriminant analysis training step were defined as the image based concealed damage scores. The relationship between the visual concealed damage scores, described in table 4.3.1, and the image-based concealed damage scores is shown in table 5.3.1.

Table 5.3.1 Correspondence between visual and image-based concealed damage scores.

Visual concealed damage score	Image-based concealed damage score				
	1	2	3	4	5
1	81.25%	17.97%	0.78%	0.0%	0.0%
2	30.53%	50.44%	19.03%	0.0%	0.0%
3	0.0%	20.37%	71.6%	8.02%	0.0%
4	0.0%	0.94%	31.13%	37.74%	30.19%
5	0.0%	0.0%	0.0%	19.05%	80.95%

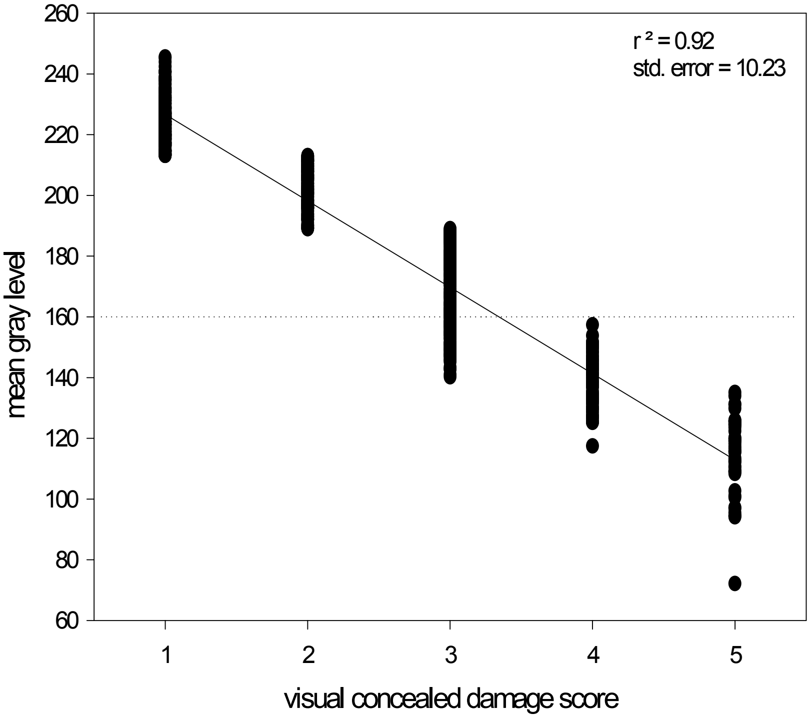


Figure 5.3.1. Correspondence between mean gray level of kernel images and visual concealed damage score.

b. Effect of moisture, drying, and storage treatments

Using data from the moisture and drying treatments discussed in section 4.3a, and storage treatments described in sections 4.3a and 4.3e, an ANOVA was performed with drying method (none, low temperature, high temperature), moisture exposure (none, short, long), and storage (short, medium, long) as independent variables, and image based concealed damage score as the dependent variable. The medium storage treatment is the seven month storage at 10°C and 45% relative humidity (section 4.3e) while the long storage treatment is the year long storage at 4°C and 90% relative humidity (section 4.3a). The ANOVA table is shown below in table 5.3.2. Drying, moisture exposure, and the interaction between drying and moisture treatments are all significant at the 0.0001 level. Storage was not significant at the 0.05 level.

Table 5.3.2 ANOVA results for treatment effect on concealed damage score.

```
                                       Sum of            Mean
Source                     DF         Squares          Square   F Value      Pr > F

Model                       6    414.79475309     69.13245885     89.51      0.0001

Error                     641    495.09259259      0.77237534

Corrected Total           647    909.88734568

                     R-Square            C.V.        Root MSE         IM_SCORE Mean

                     0.455875        35.97562       0.8788489             2.4429012


Source                     DF       Type I SS     Mean Square   F Value      Pr > F

MOIST                       2    208.21656379    104.10828189    134.79      0.0001
DRY                         2    129.80658436     64.90329218     84.03      0.0001
DRY*MOIST                   2     76.77160494     38.38580247     49.70      0.0001

Source                     DF     Type III SS     Mean Square   F Value      Pr > F

MOIST                       2    105.54444444     52.77222222     68.32      0.0001
DRY                         2    104.66872428     52.33436214     67.76      0.0001
DRY*MOIST                   2     76.77160494     38.38580247     49.70      0.0001
```

Table 5.3.3 shows the fraction of nuts for each moisture/drying treatment classified into each of the five image-based concealed damage classes. For the long moisture treated nuts dried at 110°C, 9.4% of the nuts were classified with a concealed damage score of one or two and 92.6% were given a score of 3, 4 or 5. In contrast, of the long moisture treated nuts dried at 55°C, 79.0% were given a score of one or two and only 21.0% were given a score of 3, 4 or 5. Similar results were obtained for the short moisture treated nuts. However, there was very little difference in concealed damage scores between the control nuts (no moisture treatment and no drying) and nuts that were not exposed to moisture to a drying treatment. This confirms the results from the batch tests that concealed damage can be reduced by use of lower drying temperatures. From these observations and the chemical assays, it appears that carbonyl compounds are formed in almonds during exposure to moisture. The Maillard reaction can proceed at low moisture contents if its first stage is completed with Amadori product formation. However, the first stage of the Maillard reaction will be inhibited at water activities below 0.4, or 4% to 5% moisture for almonds. If the first stage of the Maillard reaction does not complete before the moisture level drops, then the later stages will be inhibited. It could be that the lower temperature drying lowers the moisture of almonds before significant Amadori product formation. The heat from higher temperature drying accelerates the Amadori product formation and allows the later stages of the Maillard reaction to occur during roasting. Further experimentation would be needed to confirm this hypothesis.

The incidence of concealed damage scores of 4 and 5, as shown in table 5.3.3, are considerably higher for the high temperature drying treatment than the concealed damage incidences observed in the chemical tests shown in table 5.2.2. The results for table 5.2.2 were based on visual inspection while the table 5.3.3 are based on image analysis. Another possible reason for the different results is that the moisture treatments used for the chemical tests were less severe than the individual nut tests used to obtain the table 5.3.3 results.

Table 5.3.3. Percentage of nuts from each treatment combination classified into the five image based concealed damage score levels. Treatments are ordered from lowest incidence of nuts with a concealed damage score of one to the highest.

moisture treatment	drying treatment	storage treatment	image based concealed damage score 1	2	3	4	5
long	110°C	long	0.00%	6.33%	35.44%	45.57%	12.66%
long	110°C	medium	0.00%	7.41%	46.91%	39.51%	6.17%
long	110°C	none	1.23%	6.17%	48.15%	40.74%	3.70%
short	110°C	medium	1.23%	16.86%	46.91%	32.84%	2.53%
short	110°C	none	2.47%	16.05%	41.98%	34.57%	4.94%
short	55°C	none	15.19%	43.04%	27.85%	11.39%	2.53%
short	55°C	medium	16.05%	40.34%	30.86%	11.43%	1.23%
long	55°C	medium	27.16%	51.25%	20.23%	0.00%	1.28%
long	55°C	none	30.86%	48.15%	19.75%	0.00%	1.23%
none	55°C	none	33.33%	45.68%	20.99%	0.00%	0.00%
none	none	medium	33.33%	54.32%	11.11%	1.23%	0.00%
none	none	none	37.04%	55.56%	7.41%	0.00%	0.00%
none	110°C	none	38.27%	50.62%	9.88%	1.23%	0.00%

Note: The long storage treatment, described in section 4.3a, was for 12 months at 4°C and 90% relative humidity; the medium storage treatment, described in section 4.3e, was for 7 months at 10°C and 45% relative humidity.

As can be seen in table 5.3.3, the treatment with the most concealed damaged nuts exposed to the long moisture, 110°C dry, year storage treatment combination. While the storage was at 4°C, the humidity in this particular environment was approximately 90%. These are not normal storage conditions for almonds. The humidity would normally be lower. For the nuts exposed to seven month storage at 45% relative humidity and 10°C (section 4.3e), the effect of storage on severity of concealed damage is not noticeably different compared to the concealed damage scores of nuts not stored.

c. Physical property test results

The presence of mold after moisture exposure, wet appearance after moisture exposure, or type of nut kernel (single or double) have no apparent effect on severity of concealed damage after cooking. The nuts exposed to a short or long moisture treatment had a mean gray level of 179.4 with a standard deviation of 34.7. There were 14 nuts that developed mold during moisture exposure. The moldy nuts had a mean gray level of 176.5. There were 12 nuts that had a wet appearance after moisture exposure and these had a mean gray level of 181.8. And, the double kernel nuts had a mean gray level of 182.6.

For volume measurement by x-ray imaging, a linear relation between the measured volume and pixel integration is shown in figure 5.3.2, the R^2 is 0.87. Data from this plot is from the test set containing 49 control and 49 moisture treated nuts. The measured volume of the wood standards in all images were within 0.5% of each other so it was not used to adjust the pixel integration figure. Most of the discrepancy between measured volume and pixel integration may be due to internal voids inside the kernel. The buoyancy method will include the internal voids with the nut volume, the x-ray integration method will not include internal void volume.

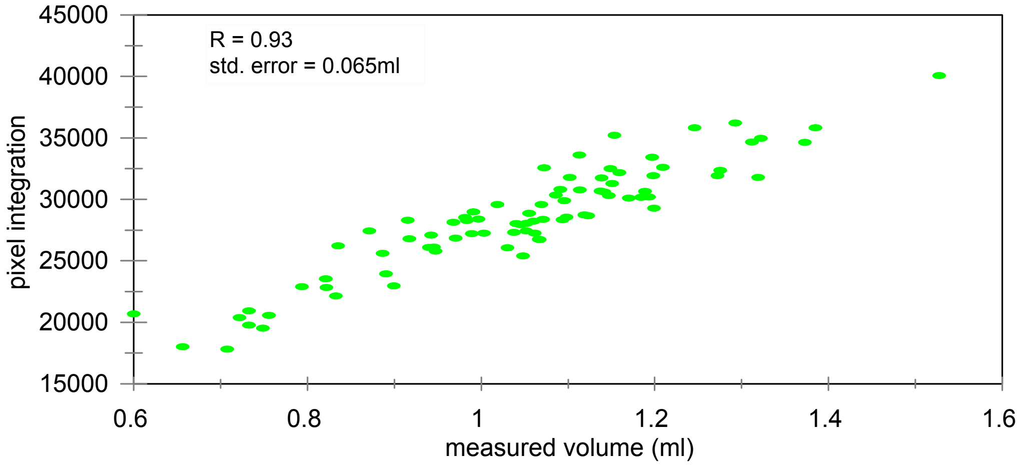


Figure 5.3.2. Graph of pixel integration versus measured nut volume by buoyancy.

A correlation matrix of all measured physical properties on individual nuts is shown in table 5.3.4. There was no large correlation between any of the pre-cook properties measured and image based concealed damage score. Soluble solids were not measured on these nuts. The mass change in nuts during the combined moisture treatment and drying, computed with equation 5.3.1, had no correlation with concealed damage. The loss of mass between drying and cooking, as measured by the mass loss ratio given by equation 5.3.2, had a correlation of -0.34 with concealed damage. This weak correlation may indicate that concealed nuts have a very mild tendency to lose less moisture during cooking than normal nuts. The late stages of the Maillard reaction react with water. This could be a reason for this weak relation between mass loss through cooking and concealed damage; however, more experimentation would be needed to confirm this.

$$\textit{mass loss ratio, drying} = [(\textit{initial mass}) - (\textit{post-dry mass})] / (\textit{initial mass}) \qquad (5.3.1)$$

$$\textit{mass loss ratio, cooking} = [(\textit{initial mass}) - (\textit{post-cook mass})] / (\textit{initial mass}) \qquad (5.3.2).$$

Table 5.3.5 shows the means of all the measured physical properties separated into the five image based concealed damage groups. The means were tested for equivalence at $\alpha \leq 0.05$ with Tukey's Studentized Range Test. For the five concealed damage scores, the initial mass, post-dry mass, and post-cook mass are not significantly different at the 0.05 level. Physical properties that had significantly different means between nuts with an image score of 4 or 5 and nuts with image scores of 1 or 2 were: mass loss ratio (drying), post-dry volume measured by x-ray imaging, post-cook C.I.E. L* a* b* values, post-cook mean gray level, and post-cook percent pixels less than 120. Note that higher concealed damage scores tend to have significantly higher mean kernel volumes. Post-cook C.I.E. L* value and mean gray level had significantly different means for all five image based concealed damage scores. Note that nuts with concealed damage scores of four and five had significantly lower mass loss ratios through drying than nuts with concealed damage scores of one and two. This indicates that the measured post-dry mass of nuts with concealed damage was close to their initial mass. In contrast, the post-dry mass of normal almonds tends to be slightly greater than their original mass.

Table 5.3.4. Correlation matrix of individual nut properties. Correlation values with a magnitude greater than 0.5 are shown in bold.

	initial mass	P-D mass	P-C mass	m_{loss}, drying	m_{loss}, cooking	P-D density	P-D volume	P-D thickness	P-C L* value	P-C a* value	P-C b* value	P-C mean gray	#pixels < 120	image CD score
P-D mass	**1.00**													
P-C mass	**1.00**	**1.00**												
m_{loss}, drying	-0.14	-0.19	-0.17											
m_{loss}, cooking	-0.11	-0.14	-0.16	**0.64**										
P-D density	0.09	0.09	0.09	0.04	-0.01									
P-D volume	**0.87**	**0.87**	**0.87**	-0.13	-0.07	-0.32								
P-D. thickness	0.33	0.31	0.31	0.37	0.35	0.02	0.27							
P-C L* value	-0.03	-0.03	-0.05	-0.02	0.24	0.07	-0.11	-0.05						
P-C a* value	0.09	0.09	0.11	-0.02	-0.35	-0.08	0.15	-0.03	-0.89					
P-C b* value	0.02	0.02	0.02	-0.02	-0.03	0.05	-0.04	-0.14	0.45	-0.09				
P-C mean gray	-0.04	-0.04	-0.05	0.01	0.34	0.07	-0.10	-0.03	**0.85**	**-0.83**	0.31			
# pixels < 120	0.02	0.03	0.03	-0.16	-0.23	-0.04	0.06	-0.04	**-0.65**	**0.52**	**-0.53**	**-0.72**		
image CD score	0.10	0.10	0.11	-0.05	-0.34	-0.04	0.14	0.03	**-0.82**	**0.80**	-0.31	**-0.94**	**0.75**	
visual CD score	0.08	0.08	0.09	-0.01	-0.31	-0.06	0.14	0.05	**-0.86**	**0.83**	-0.32	**-0.86**	**0.62**	**0.84**

Note: P-D stands for post-dry, P-C stands for post-cook, m_{loss} stands for mass loss ratio, and CD stands for concealed damage.

Table 5.3.5. Means of the measured physical properties for each of the five image based concealed damage scores. Means with the same group letter are not significantly different at the 0.05 level determined by Tukey's Studentized Range (HSD) Test.

image based CD score	initial mass (g)	P-D mass (g)	P-C mass (g)	m_{loss}, drying	m_{loss}, cooking	P-D volume (pixel integration)	P-D thickness (mm)	P-C L* value	P-C a* value	P-C b* value	P-C mean gray level	percent pixels < 120	visual CD score
1	1.117 A	0.839 A	1.070 A	0.247 A	0.042 AB	75377 A	9.52 AB	79.43 A	4.02 A	22.27 A	224.4 A	0.06 A	1.40 A
2	1.129 A	0.936 A	1.083 A	0.171 AB	0.040 B	76281 A	9.37 B	76.35 B	5.72 B	23.39 B	202.0 B	0.30 A	2.05 B
3	1.146 A	1.079 A	1.097 A	0.057 BC	0.043 A	79820 AB	9.75 A	70.72 C	7.88 C	22.62 AB	168.6 C	3.10 B	2.95 C
4	1.170 A	1.167 A	1.122 A	0.003 C	0.041 AB	82673 B	9.61 AB	64.42 D	9.96 D	20.26 C	138.6 D	27.61 C	3.87 D
5	1.148 A	1.147 A	1.101 A	0.001 C	0.042 AB	81691 B	9.27 B	61.78 E	10.00 D	18.22 D	115.9 E	61.8 D	4.34 A

Note: P-D stands for post-dry, P-C stands for post-cook, m_{loss} stands for mass loss ratio, and CD stands for concealed damage.

d. Spectral property results

The normalized mean transmission spectra and the second derivative spectra (10 nm gap) from 700 nm to 1400 nm, obtained after drying but before cooking, for nuts with a concealed damage score of 1 and 5 are shown in figure 5.3.4. The normalized spectra is computed by dividing each absorbance value of an individual nut's spectrum by the mean of all absorbance values in the spectra for that nut. Normalization is required to compensate for variations in nut thickness and kernel skin condition. The second derivative spectra is computed from the raw transmission spectra, not the normalized spectra. The mean normalized transmission spectra from 700 nm to 970 nm, obtained after drying but before cooking, for nuts from all 5 concealed damage scores are shown in figure 5.3.7. As can be seen from figure 5.3.4, the largest differences between means of the concealed damage groups are at 700 nm, 930 nm and 1120 nm. Figure 5.3.5 shows the t-values at all wavelengths for the difference between means of nuts with concealed damage scores of 1 and 5. The program listed in appendix E was used to compute the t-values. Equal variances of the two groups was not assumed so the variances of the two groups were not pooled when computing t-values. The approximate t-value under the assumption of unequal variances was computed, as recommenced by SAS (1994), using equation 5.3.1

$$t = \frac{\overline{x}_1 - \overline{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}} \qquad (5.3.1)$$

where $\overline{x_a}$ is the mean of group a, s_a is the standard deviation of group a, and n_a is the number of samples in group a. The largest t-value magnitudes between concealed damage scores 1 and 5 are -0.7 at 1120 nm, followed by 0.42 at 930 nm and 0.34 at 700 nm. The absorbance peak at 930 nm and possibly 1120 nm can be attributed to oil (Tkachuk, 1987). The concealed damaged nuts have less absorbance in this area, possibly due to oxidation during moisture exposure. Figure 5.3.8 shows the absorbance spectrum of pure almond oil with a peak at 930 nm. More research would be needed to confirm this trend for different lengths of storage time.

Figure 5.3.6 shows the change in absorbance spectra of normal and concealed damaged nuts during the moisture and drying treatments. These plots were made by subtracting the normalized absorbance spectrum obtained before any treatment was applied to the nuts from normalized absorbance spectrum obtained after drying. The curves in figure 5.3.6 represent the mean of these differences in absorbance values for concealed damaged and normal nuts. For this graph, concealed damaged nuts are those with an image based concealed damage score of 4 or 5 and normal nuts are those with an image based score of 1 or 2. The concealed damaged nuts show a higher increase in absorbance around 700 nm after drying than normal nuts. In contrast, normal nuts show a higher increase absorbance around 930 nm after drying in the region than concealed damaged nuts.

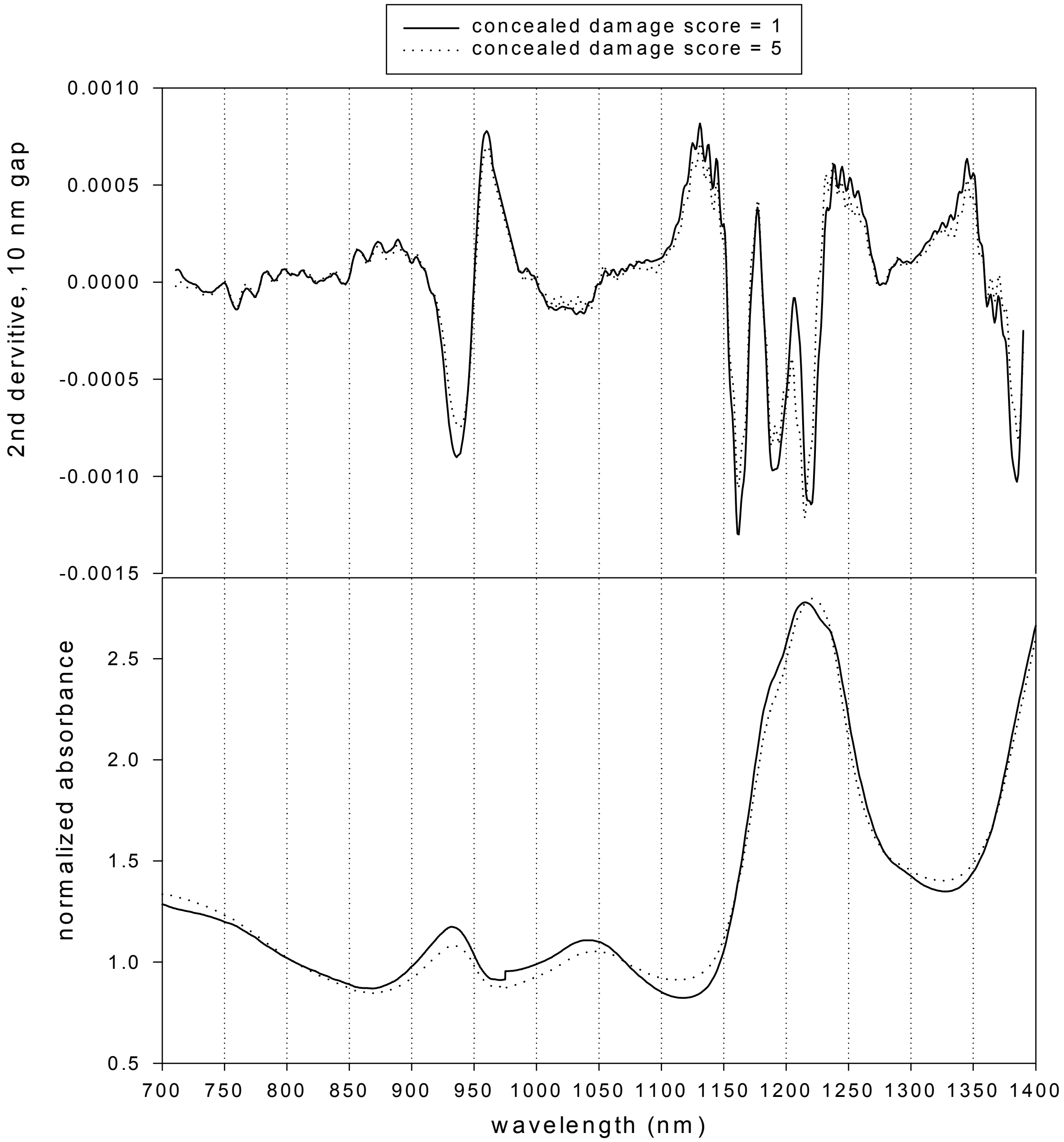


Figure 5.3.4. Mean normalized absorbance spectra from 700 nm to 1400 nm of almonds measured after drying with concealed damage scores of 1 and 5 measured after cooking. The second derivative spectra was computed from the raw absorbance spectra using a 10 nm gap.

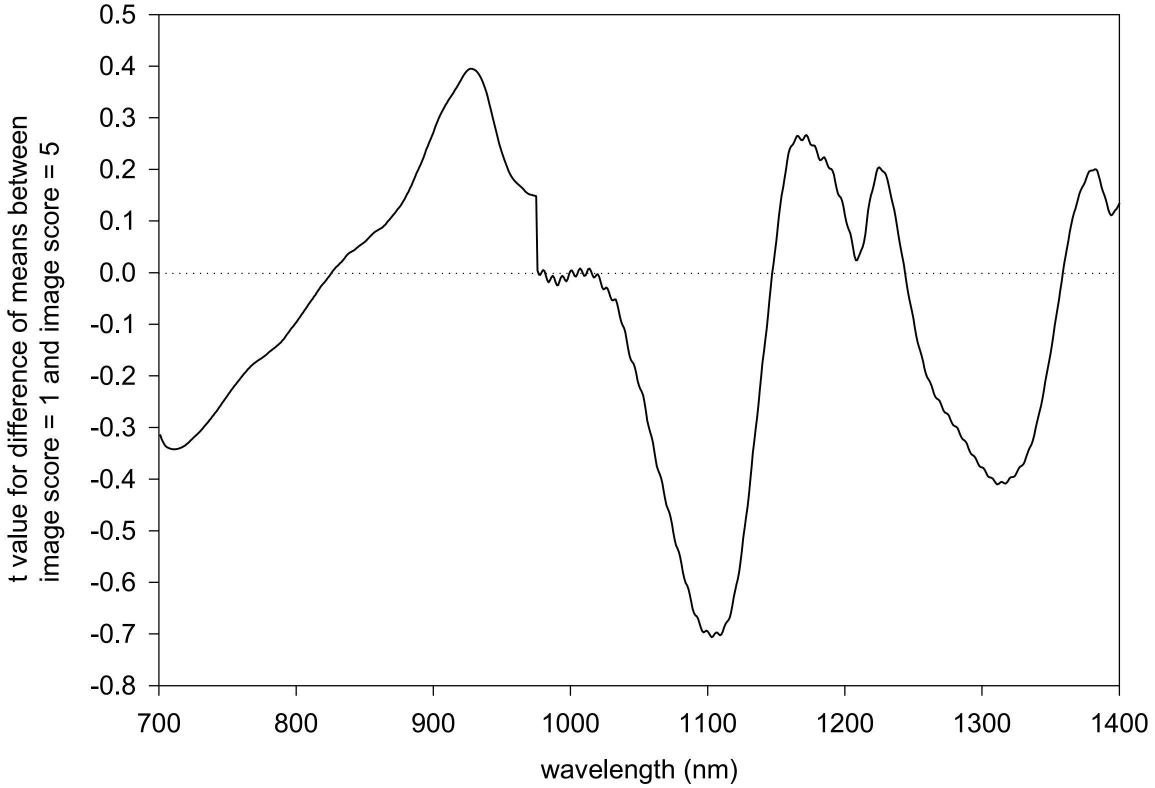


Figure 5.3.5. Statistical t-value for difference between normalized absorbance means between almonds with a concealed damage score of 1 and a concealed damage score of 5 at each wavelength.

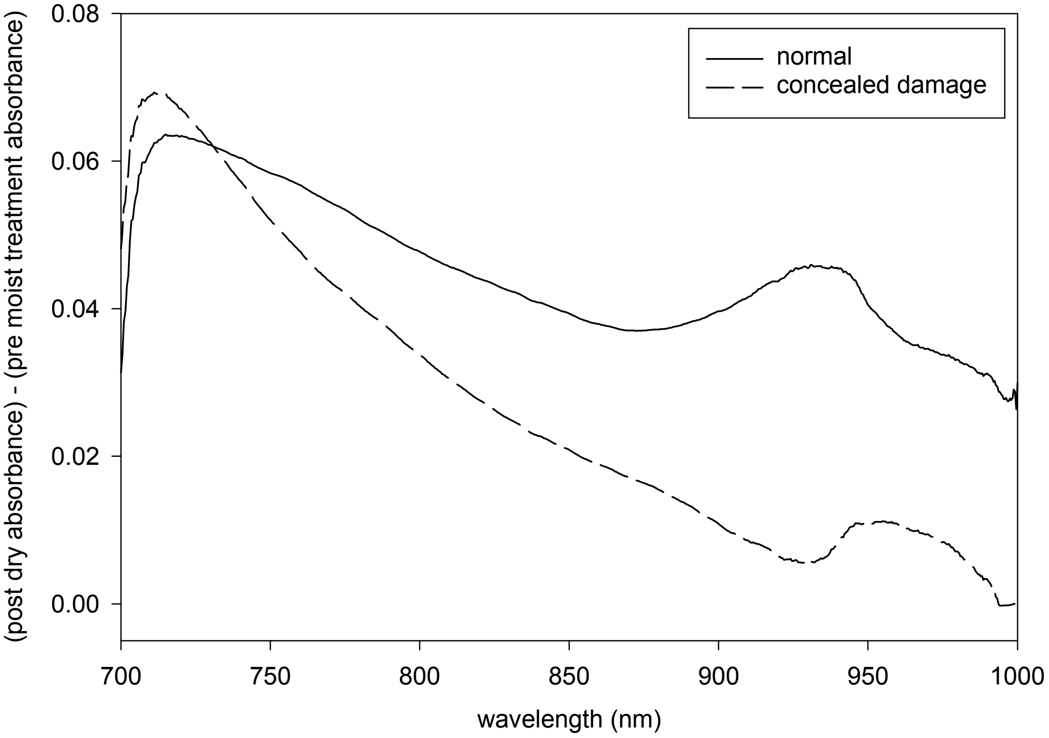


Figure 5.3.6 Difference in mean normalized absorbance spectra before and after moisture treatments and drying of concealed damaged nuts and normal nuts. Concealed damaged nuts had image-based concealed damage scores of 4 or 5 and the normal nuts had concealed damage scores of 1 or 2.

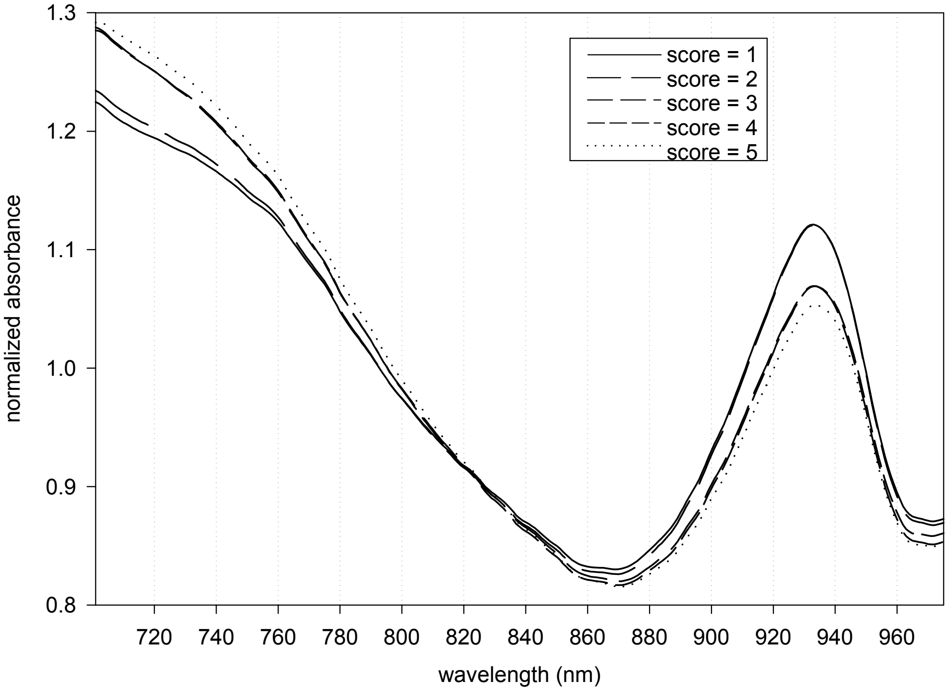


Figure 5.3.7. Graph of mean spectrum from each of the 5 image based groups.

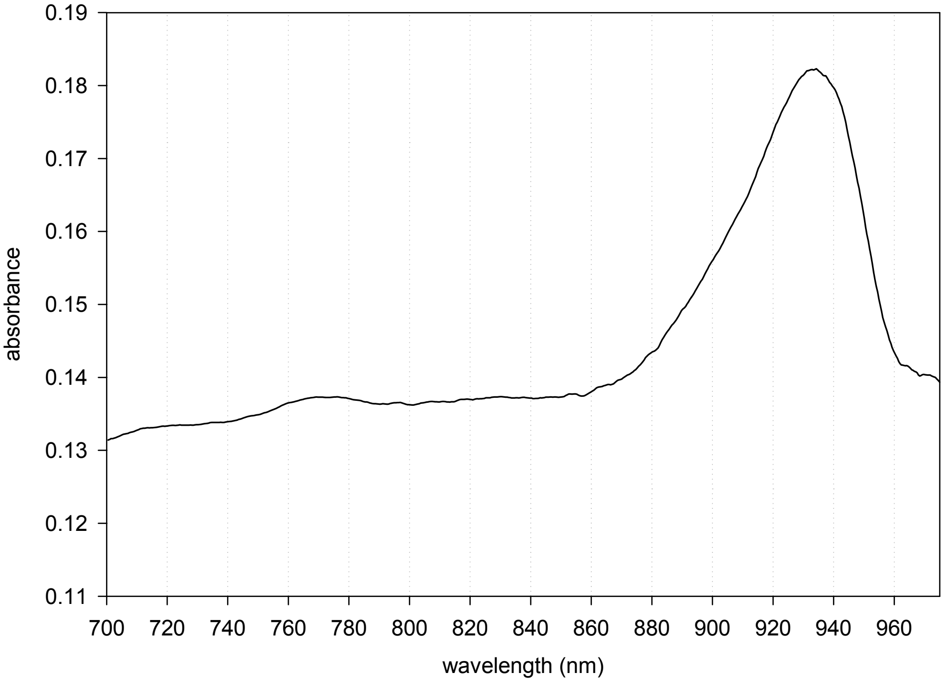


Figure 5.3.8. Absorbance spectrum of pure almond oil.

e. Prediction based on stepwise discriminant analysis

Results of the discriminant analysis using principle components of the normalized absorbance, first derivative, and second derivative spectra are tabulated in Table 5.3.6. The error rate is the percentage of nuts in the validation set incorrectly classified. The false positive error rate is defined as the percentage of nuts, in the validation set, with an image based concealed damage score of 1 or 2 being classified as concealed damaged. The false negative error rate is defined as the percentage of nuts, in the validation set, having an image based concealed damage score of 4 or 5 being classified as normal.

Table 5.3.6. Results of discriminant analysis using principle components of different portions on the spectra. Results are ordered from lowest error rate to highest. Abbreviations: A=absorbance, A' = first derv.,A" = second derv.

spectra	wavelength range (nm)	total error rate (%)	false positives (%)	false negatives (%)	number of principle components used
absorbance & 1st derv & 2nd derv.	1000 to 1300	12.4	1.4	11.1	5 A, 1 A', 2 A"
1st derivative	1000 to 1300	14.2	1.4	15.9	8 A'
absorbance & 1st derv & 2nd derv.	700 to 975	15.0	5.4	11.1	1 A, 3 A', 1 A"
absorbance	1000 to 1300	15.4	1.4	12.7	10 A
1st derivative	700 to 975	15.6	2.0	11.1	7 A'
absorbance	700 to 975	16.0	2.7	12.7	5 A
absorbance & 1st derv & 2nd derv.	700 to 1300	16.0	2.7	12.7	3 A, 2 A', 2 A"
absorbance	700 to 1300	16.6	4.1	12.7	8 A
2nd derivative	1000 to 1300	18.3	0.7	11.5	8 A"
1st derivative	700 to 1300	18.6	1.4	20.6	7 A'
2nd derivative	700 to 1300	20.8	1.4	15.9	11 A"
2nd derivative	700 to 975	27.5	5.4	23.8	7 A"

The lowest total error rate, 12.4%, obtained by this method was using principle components of the absorbance, first derivative, and second derivative spectra between 1000 and 1300 nm. Most errors are false positives. This classification model had false positive and false negative error rates of 1.4% and 11.1%, respectively. Comparable false positive and false negative error rates are obtained using only first derivative spectra between 700 and 975 nm. This is an important result because silicon light detectors, sensitive between 700 and 975 nm, are less expensive, for a given signal to noise ratio, than detectors sensitive to light between 1000 and 1300 nm.

f. Prediction with discriminant analysis using all combinations of three features

The five best models, based on error rate, for the discriminant analysis performed using all possible combinations of three variables are shown in Table 5.3.7. The lowest error rate achieved with this method was 17.6% using first derivative parameters at 932.5 nm with a 25 nm gap, 745 nm with a 50 nm gap, and 932.5 nm with a 5 nm gap. Recall that the analysis was performed on data containing normalized absorbances, all first derivatives with forward difference gaps between 5 and 50 nm, all second derivatives with a central difference gaps between 5 and 50 nm, all possible combinations of ratios of two absorbance values, and all possible combinations of continuous areas less than 100 nm wide.

Table 5.3.7. Top five classification models using three variables.

feature type	error rate	specific wavelengths
first derivative: $A_{wave(gap)}$	17.6%	$A_{932.5(25)}$ $A_{745(50)}$ $A_{932.5(5)}$
	18.8%	$A_{932.5(25)}$ $A_{712.5(5)}$ $A_{930(40)}$
	19.1%	$A_{930(40)}$ $A_{862.5(45)}$ $A_{862.5(5)}$
	23.1%	$A_{922.5(25)}$ $A_{720(20)}$ $A_{737.5(15)}$
	23.5%	$A_{912.5(45)}$ $A_{712.5(5)}$ $A_{735(30)}$
ratio: A_{wave1}/A_{wave2}	18.2%	A_{920}/A_{945} A_{760}/A_{785} A_{910}/A_{920}
	18.5%	A_{910}/A_{950} A_{710}/A_{750} A_{890}/A_{900}
	19.1%	A_{910}/A_{950} A_{710}/A_{725} A_{780}/A_{810}
	20.7%	A_{910}/A_{945} A_{710}/A_{735} A_{790}/A_{805}
	21.0%	A_{900}/A_{955} A_{710}/A_{730} A_{780}/A_{810}
second derivative: $A_{wave(gap)}$	18.8%	$A_{900(50)}$ $A_{780(50)}$ $A_{790(10)}$
	19.1%	$A_{900(50)}$ $A_{780(45)}$ $A_{790(10)}$
	19.4%	$A_{900(50)}$ $A_{710(10)}$ $A_{790(10)}$
	19.8%	$A_{900(50)}$ $A_{710(5)}$ $A_{790(10)}$
	20.1%	$A_{900(50)}$ $A_{710(5)}$ $A_{780(20)}$
areas: A_{wave}	21.6%	A_{835} thru A_{850}, A_{730}, A_{855}
	22.2%	A_{835} thru A_{850}, A_{705} thru A_{710}, A_{865},
	22.5%	A_{830} thru A_{850}, A_{705} thru A_{710}, A_{845} thru A_{865}
	23.1%	A_{825} thru A_{840}, A_{715} thru A_{720}, A_{930} thru A_{935}
	23.5%	A_{825} thru A_{840}, A_{715} thru A_{720}, A_{930} thru A_{935}

g. Prediction with partial least squares regression

Table 5.3.8 shows the wavelength ranges used and the PLS regression results for predicting mean gray level of the cooked nuts on the calibration data. The classification performance, based on the predicted gray level, on the validation set, of the top six PLS equations, based on R value from the calibration set, are shown in Table 5.3.9. Again, the error is dominated by false negatives.

Table 5.3.8. PLS calibration results from different wavelength ranges.

absorbance spectra range (nm)	number of PLS factors used	R	standard error of calibration (gray level)
705 - 760, 870 - 970	4	0.80	19.7
705 - 970	4	0.79	20.2
870 - 970	3	0.78	20.8
840 - 960	3	0.77	21.2
830 - 950	4	0.75	21.6
705 - 760, 840 - 960	3	0.74	22.1
705 - 760, 880 - 970	3	0.73	22.3
705 - 1300	5	0.72	23.8
1020 - 1200	1	0.42	30.7

Table 5.3.9. Validation set classification performance of PLS equations.

absorbance spectra range (nm)	false positives (%)	false negatives (%)	overall error rate (%)
705 - 760, 870 - 970	3.0	48.8	18.1
705 - 970	3.0	50.0	18.5
870 - 970	3.0	52.4	19.3
840 - 960	3.0	56.4	20.6
830 - 950	3.0	58.8	21.4
705 - 760, 840 - 960	3.0	60.0	21.8

The error rates using stepwise discriminant analysis on principle components of large portions of the spectra between 700 and 1300 nm, shown in Table 5.3.6, range from 12.4% to 27.5%, are somewhat better than the three variable discriminant models obtained by selecting individual features within the spectrum where the error rates ranged from 17.6% to 23.5%. However, the three variable discriminant models only included the portion of the spectrum between 700 and 970 nm. The classification error rates using the PLS regression equation obtained from absorbance spectra range from 18.1% to 21.8%. Note that the PLS regression was not performed using first and second derivative data, only absorbance data was used. When only principle components of the absorbance spectra were provided to the stepwise discriminant analysis procedure, the classification error rate was 16% using the interval between 700 nm and 975 nm. This is comparable to the PLS results.

h. Prediction with simulated LED absorbances

The simulated LED transmitted light responses yielded classification results comparable to the methods previously discussed. The lowest classification error rate on the validation set, 17.7%, was obtained using principle components of 700 nm, 830 nm, 880 nm, 890 nm, 940 nm, and 950 nm LED absorbances. Principle components of the absorbance ratios were not chosen by the stepwise selection procedure. A majority of the classification errors, 55%, were made on nuts with a concealed damage score of three. These nuts were on the border between concealed damaged and normal. Of the nuts with image based concealed damage scores of 4 and 5, 20.6% were false negatives, or classified as normal. Of the nuts with image based concealed damage scores of 1 and 2, only 3.3% were classified as concealed damaged.

Classification errors based on the simulated LED responses are not related to the moisture treatment or drying temperature applied to the nuts. Most of the classification errors, 74%, were false negatives. For the two long moisture treatments, 50% of the concealed damaged nuts from both the high and low temperature dried batches were classified as normal. For the two short moisture treatments, 43% of the concealed damaged nuts from the high drying temperature batch were classified as normal and 40%

of the concealed damaged nuts from the low drying temperature batch were classified as normal.

5.4. Real-time sorting device

a. Repeatability of DFT values

From the tests of sliding the rectangular Teflon piece through the prototype, it was determined by Tukey's Studentized Range Test, that the mean intensity of the first two DFTs was significantly different, at the 0.05 level, than the mean intensity of the last ten DFTs. This was true for all six LEDs. The mean intensity of the last ten DFTs were determined equivalent at the 0.05 level. Because of these results, data from the first two DFTs were not be used for prediction of concealed damage. The average of the last ten DFT values was used for prediction of concealed damage.

b. Classification based on discriminant analysis

Using the data from the prototype system and classification with discriminant analysis, the minimum error rate of the validation set was 20.4% using three principle components, two from the LED absorbance set and one from the LED difference set. The stepwise selection of principle components procedure was repeated for every model containing only five LEDs. The error rates of all the five LED models were at least 5.5% higher than the model containing all six LEDs.

From a processor’s perspective, it is desired to minimize the number of false positives, or nuts classified as concealed damaged when, in fact, they are normal. Classification from a discriminant function is based on the a posteriori probability. The false positives can be reduced by increasing the normal nut a priori probability. For example, a normal nut a priori probability of zero results in a 100% false positive error rate while a normal nut a priori probability of one results in a zero false positive error rate. However, raising the normal nut a priori probability increases the false negative error rate. Table 5.4.1 shows the false positive, false negative and total error rates of the validation set for a range of normal nut a priori probabilities. The total error rate is the total number of incorrect classifications divided by the total number of nuts in the validation set. Using a normal nut a priori probability of 0.7, there are 9.0% false positives and 28.8% false negatives giving a total error rate of 14.3%. Most of the incorrectly classified nuts are on the border between actually being considered concealed damaged or normal. Of the 9.0% false positives, 83% have a mean gray level below 190 and would have a concealed damage score of 3. Of the 28.8% false negatives, 56% have a mean gray level above 130 and also would also have a concealed damage score of 3. In this case, only 1.5% of the almond samples showing little browning, or having a concealed damage score of 1 or 2, were classified as concealed damaged. Also, 16.2% of almond samples with severe concealed damage, or having concealed damage scores of 4 or 5, were classified as normal. The errors from the two drying treatments used in this experiment are equally divided. The high temperature drying treatment comprise 55% percent of the classification errors and the remaining 45% are from the low temperature drying treatment.

Table 5.4.1. Error rates at different posterior probability thresholds for classifying nuts as normal or concealed damaged.

a priori probability for normal almond q_{normal}	false positive error rate	false negative error rate	total error rate
0.0	100.0%	0.0%	73.5%
0.1	54.9%	1.9%	40.8%
0.2	42.4%	1.9%	31.6%
0.3	33.3%	7.7%	26.5%
0.4	28.5%	9.6%	23.5%
0.5	22.2%	15.4%	20.4%
0.6	17.4%	19.2%	17.9%
0.7	9.0%	28.8%	14.3%
0.8	7.6%	32.7%	14.3%
0.9	2.8%	42.3%	13.3%
1.0	0.0%	100.0%	26.5%

c. Classification based on regression analysis

A correlation matrix of the light absorbances measured with the prototype system before cooking and mean gray level measured after cooking are shown in table 5.4.2. The absorbance from the 890 nm LED has the highest correlation, at r = 0.24, with mean gray level.

Table 5.4.2. Correlation matrix of the absorbance from the six LEDs with mean gray level.

		LED emission peak wavelength (nm)					
		660	830	880	890	940	950
LED peak emission wavelength (nm)	830	-0.59					
	880	-0.75	0.80				
	890	-0.46	-0.20	-0.15			
	940	-0.86	0.17	0.46	0.55		
	950	-0.84	0.17	0.41	0.51	0.95	
	gray level	-0.15	0.09	0.12	0.24	0.04	-0.02

Three regression models were developed to predict the mean gray level of almonds after cooking. One regression model used principle components of the absorbance, ratio and differences of LED light absorbance values. Another regression model used the raw LED absorbance, ratio and difference values after removing highly correlated values. A third regression model comprised only the six normalized LED absorbance values. Nuts were classified as concealed damaged or normal if their predicted mean gray level was below or above 160, respectively. For each of these three models, the adjusted R^2 of the validation set and classification error rates on the validation set are listed in table 5.4.3. The total error rate is the percent nuts incorrectly classified. The false positive error rate is the percentage of normal nuts classified as concealed damaged. The false negative error rate is the percentage of concealed damaged nuts classified as normal.

For the regression procedure using raw LED absorbance, ratio and difference values after removing highly correlated independent variables, only twelve of the original 36 independent variables were left after removing all the highly correlated independent variables. The remaining variables were: absorbance values from the 660, 830, 880, 890 and 940 nm LEDs, the ratio between 830 and 890 nm LEDs, and the differences between two LEDs having the following peak emission wavelengths (in nm): 660-880, 830-940, 830-950, 940-950, 950-880, 950-890. The model with the highest adjusted R^2, 0.35, contained the following eight variables: absorbance from the 660 nm LED, ratio between 830 and 890 nm LEDs, and differences between two LEDs having the following peak emission wavelengths (in nm): 660-880, 830-950, 830-950, 940-950.

The regression model comprising only the six LED absorbance values had an adjusted R^2 of 0.40. Dropping any one of the LED absorbance values resulted in large drop of the model adjusted R^2. The highest adjusted R^2 using only five LED absorbance values was 0.29.

Table 5.4.3. Comparison of the three regression models used to classify nuts as concealed damage or normal.

model	adjusted R^2	total classification error rate	false positive error rate	false negative error rate
principle components	0.46	20.0%	9.1%	39.4%
LED absorbance, ratio and difference values	0.35	22.9%	11.2%	43.8%
LED absorbance values	0.40	22.1%	10.7%	42.6%

Classifying nuts as concealed damaged or normal based on the predicted gray value by principle component regression yields similar classification results obtained with the discriminant analysis procedure. The maximum adjusted R^2 for the regression model was 0.46 on the validation set. A scatter plot of the actual gray level versus the predicted gray level from the validation set is shown in figure 5.4.1. This was obtained using five principle components, two from the LED absorbance set and three from the ratio set. The total classification error rate was 20.0%. Only 9.1% of the normal nuts were classified as concealed damaged. Most of the errors arose from classifying concealed damaged nuts as normal with 39.4% of the concealed damaged nuts being incorrectly classified as normal. As with the discriminant analysis procedure, most of the classification errors involve nuts with gray levels near 160, the division between normal and concealed damage. Of the concealed damaged nuts incorrectly classified as normal, 70% have a mean gray level above 140 and have an image based concealed damage score of three. For normal nuts incorrectly classified as concealed damaged, 86% have a mean gray level below 180 and have image based concealed damage scores of three. Table 5.4.4 shows the fraction of nuts, from each image based concealed damage score, that are classified as concealed damaged using the given gray level threshold. Nuts with a predicted gray level below the given gray level threshold are classified as concealed damaged. As can be seen from table 5.4.4, with a gray level threshold set at 160, all nuts with an image based concealed damage score of one are classified as normal and all nuts with a concealed damage score of five are classified as concealed damaged. Only 2.0% of the nuts with an image based concealed damage score of one or two are classified as concealed damaged. Only 11.8% of the nuts with an image based concealed damage score of four or five are classified as normal. If a gray level threshold of 150 is used to classify nuts as concealed damaged or normal, then no nuts with an image based concealed damage score of 1 or 2 are classified as concealed damaged. However, 51% of the nuts having image based concealed damage scores of four or five will be classified as normal.

The classification errors from the two drying treatments used in this experiment are not as equally divided as with the discriminant analysis procedure. No errors were made in classifying the control nuts. None of the controls were concealed damaged. Nuts from the high temperature drying treatment comprise 58% percent of the classification errors, while the remaining 42% of the classification errors were from the low temperature drying treatment. This may be due to the larger fraction of classification errors involving concealed damaged nuts.

Table 5.4.4. Fraction of nuts classified as concealed damaged for various predicted gray level thresholds. Percentages based on predicted gray levels from the validation set.

image based concealed damage score	Predicted gray level threshold to classify nuts as concealed damage or normal						
	130	140	150	160	170	180	190
1	0.0%	0.0%	0.0%	0.0%	0.0%	25.0%	71.4%
2	0.0%	0.0%	0.0%	2.9%	28.2%	57.1%	89.9%
3	0.4%	2.3%	17.8%	31.0%	51.4%	75.3%	94.9%
4	9.6%	14.2%	49.4%	78.3%	90.2%	97.6%	100.0%
5	33.2%	54.5%	78.3%	100.0%	100.0%	100.0%	100.0%

These results are comparable to the results of the simulated LED experiment. In the simulated LED experiment, section 5.3h, the minimum total error rate was 15.7% but of the nuts with image based concealed damage scores of 4 and 5, 20.6% were classified as normal. Of the nuts with image based concealed damage scores of 1 and 2, only 3.3% were classified as concealed damaged. This was obtained using simulated LEDs with peak emission wavelengths of 700, 830, 880, 890, 940, and 950 nm. There are two important differences between these simulated LEDs and the actual ones used on the prototype. The first difference is that the simulated LED set used a 700 nm LED while the prototype used a 660 nm LED. The other important difference is that the simulated LED experiment used emission spectra supplied by the manufacturer. However, the measured peak emission spectra of all LEDs, figure 4.4.2, differ somewhat from the manufacturer specifications. Table 5.4.5 shows the manufacturer specified and measured peak emission wavelength.

Finally, there is no known method to sort almonds with concealed damage so the sorting results obtained through this study cannot be compared with other methods. However, the classification error rates are comparable to other nut sorting operations. For example, in sorting pistachio nuts for shell stains a false positive error rate of 1% and a false negative error rate of 10% was obtained (Pearson, 1996). These error rates were an improvement over what the industry was obtaining with automatic color sorters to remove stained nuts. The minimum false positive rate of 2.0% achieved with the prototype sorter would not be acceptable if all almonds needed to be inspected by this sorter. Fortunately, only those batches of nuts that were exposed to rainfall during harvest and show significant levels of concealed damage, through quality checks, need to be inspected.

Table 5.4.5. Manufacturer specified and measured peak emission spectra of LED used on the prototype.

manufacturer specified peak emission wavelength (nm)	measured peak emission wavelength (nm)
660	680
830	840
880	895
890	905
940	930
950	960

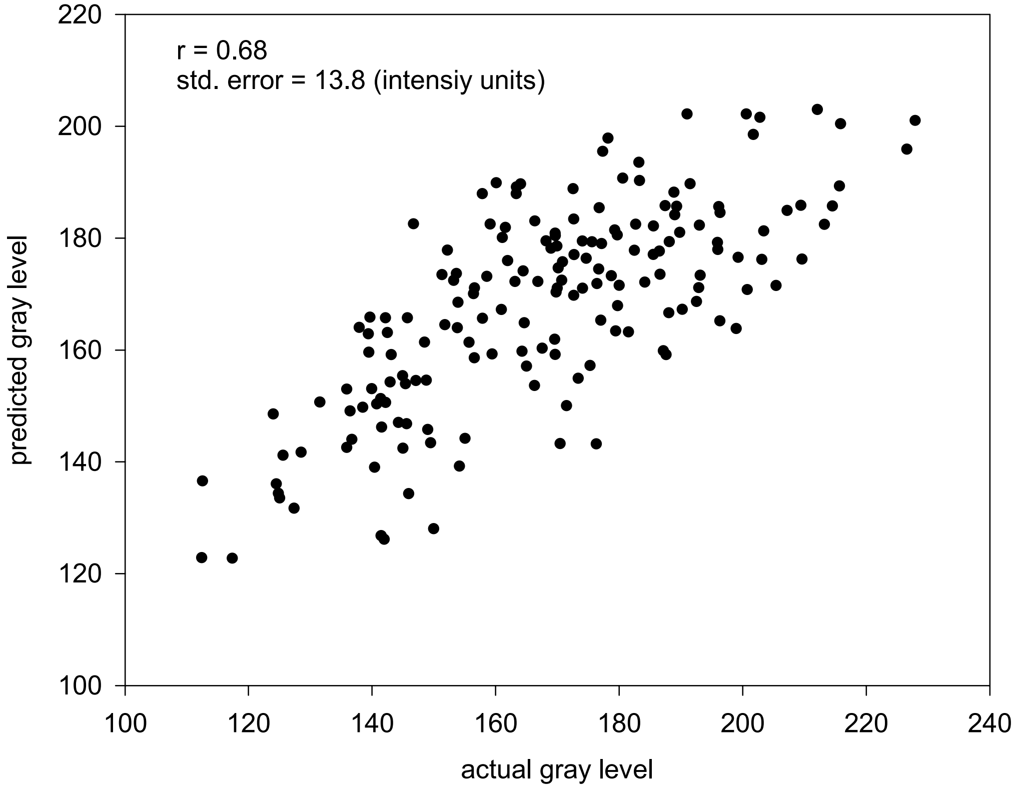


Figure 5.4.1. Graph of actual gray level versus predicted gray level from principle component regression.

6. CONCLUSION

Results from the field tests indicate that concealed damage can be reduced, but not eliminated, if nuts are left in a scattered arrangement on the orchard floor during exposure to moisture, such as rain. From the simulated rainfall experiments on scattered nuts and nuts in windrows, the incidence of concealed damage among scattered nuts was less than 25% of the concealed damage incidence of nuts held in windrows. For the same amount of simulated rainfall, scattered nuts absorbed 29% less moisture than nuts held in windrows over a five day period. Nevertheless, 4% of the scattered nuts exposed to 1 cm simulated rainfall and held in the orchard for five days contained concealed damage.

Results from the batch tests show that as almonds increase in moisture, sucrose will break down into glucose and fructose. The quantity of reducing sugars formed during exposure to moisture does not appear to be different between the two moisture treatments, long and short, used in this experiment. This indicates that breakdown of sucrose might be due to enzymatic activity. However, this was not confirmed. Also during exposure to moisture, refractive index of extracted oil indicates that fatty acid oxidation occurs.

After high temperature, 110°C, drying, the reducing sugars mostly disappear, presumably because they become bound to amino acids in the Maillard reaction. Only about two thirds of the reducing sugars disappear during low temperature, 55°C, drying. The refractive index of extracted oil after drying does not indicate that more oil oxidation occurs during drying. However, the refractive index after moisture exposure and drying compared with the refractive index of the control still indicates some oil oxidation occurred during moisture exposure. A significant ($\alpha \leq 0.05$) decrease in mean C.I.E. L* value after drying was observed for nuts dried at 110°C but not at 55°C . A reduction in water soluble solids was observed after nuts were dried. A correlation of -0.87 between water soluble solids measured after drying and the C.I.E. a* chromaticity after cooking was seen. The oxidation of oil and formation of reducing sugars followed by a reduction of soluble solids, presumably due to protein folding, and slight color changes well before browning occurs are all characteristic of the Maillard reaction proceeding through the first and intermediate stages.

Use of a lower temperature drying treatment can significantly reduce the incidence of concealed damage. For the long moisture treated nuts dried at 110°C, 1.28% of the nuts had no browning and 44.44% had severe browning due to concealed damage. In contrast, 30.86% of the long moisture treated nuts dried at 55°C had no browning and only 1.23% had severe browning due to concealed damage. Similar results were obtained for the short moisture treated nuts. Storage of almond kernels at, with initial moisture content at 6% d.b., for seven months at 10°C and 45% relative humidity did not appear to effect the incidence or severity of concealed damage.

All of the physical properties measured on individual nuts (mass, thickness, volume, presence of mold, wet appearance after moisture exposure, or if the nut was a double kernel), had very low correlations with post-cook color or concealed damage score. The mass loss during cooking had the highest correlation, at -0.34, with concealed damage score.

Comparing post-dry transmission spectra of concealed damaged and normal almonds, it was observed that nuts with concealed damage have less absorbance in the oil absorption band at 930 nm and increased absorbance in the region between 700 and 750 nm. These spectral features were used to predict, before cooking, if the nuts would develop concealed damage after cooking. The difference between absorbance spectra obtained before moisture treatment and after drying of individual nuts indicate that the changes in the absorbance occurs sometime during moisture treatment and/or drying.

Several methods were used to distinguish concealed damaged from normal almonds based on the transmission spectra obtained after drying. The lowest classification error rate on the validation set, 12.4%, was obtained by discriminant analysis using principle components of the absorbance, first derivative and second derivative spectra between 1000 and 1300 nm. The lowest classification error rate on the validation set, 15.0%, using spectra between 700 and 1000 nm was also obtained by discriminant analysis using principle components of the absorbance, first derivative and second derivative spectra.

Spectra obtained from the manufacturer of eight different infrared LEDs were multiplied and integrated with the transmission absorbance spectra of each individual almond. It was found that using principle components of the integration from six different simulated LEDs and principle components from all possible ratios of these six simulated LED integrations could classify almonds as concealed damaged or normal with an error rate of 17.7% on a validation set.

A prototype to automatically detect almonds with concealed damage was built that could operate at commercial speeds, about 40 nuts per second. The automated inspection device detected transmitted light through whole almonds from six different near infrared LEDs. Each LED was modulated at a different frequency so that the light from all LEDs was transmitted through the almond at the same time. The demodulation of the signal from the photodiode was performed by a digital signal processor in real-time by computing a discrete Fourier transform at the six modulating frequencies. Multiple linear regression and discriminant analysis to classify nuts as concealed damaged or normal was performed on principle components of all normalized LED absorbance values, all possible ratios of two normalized absorbance values, and all possible differences in two normalized absorbance values. A classification error rate of 20.4% on the validation set was obtained with three principle components selected in a stepwise discriminant analysis procedure. A classification error rate of 20.0% was obtained with seven variables selected in a stepwise regression analysis procedure. Most of the incorrectly classified nuts were on the border between actually being considered concealed damaged or normal. For the regression procedure, only 2.0% of the almond samples showing little or no browning were classified as concealed damaged. Also, 11.8% of almond samples with severe concealed damage were classified as normal. Similar results were obtained with the discriminant analysis procedure.

There are two recommendations for further research. Through this study, it has been shown that promising sorting results might be obtained with the modulated LED design. However, more research is needed to determine the optimal set of LEDs to use for a commercial sorter. Several other LEDs are available but were not used in this study due to time limitations. Furthermore, the sorting machine needs more thorough testing and the issue of different nut storage conditions, storage duration, and nuts from different growing conditions need to be addressed. The second recommendation for further research, on a more fundamental level, would be to perform a more in depth study on the NIR spectral properties of almond oil. In this study, it was observed that almonds with concealed damage show reduced absorbance at 930 nm. This spectral feature was used to help predict if the nut would become concealed damaged after cooking. It was hypothesized that the reason for the reduced absorbance at 930 nm was due to oil oxidation. This was supported by oil refractive index measurements. However, more research is required to confirm this hypotheses.

7. REFERENCES

Adhikari, H.R. and A.L. Tappel. 1973. Fluorescent products in a glucose- glycine browning reaction. Journal of Food Science. 38: 486-488.

AOAC. 1980. Official methods of analysis, 13th ed. Association of Official Analytical Chemists, Washington, DC.

Anonymous. 1984. Composition of Foods: Nut and Seed Products - Raw, Processed, Prepared. USDA Agricultural Handbook 8-12.

Anonymous. 1994a. Statistical Tables. Almond Board of California, Modesto, CA.

Anonymous. 1994b. Light sources, monochromators and spectrographs, detectors and detection systems, fiber optics, volume 2. Oriel Corporation, Stratford, CT.

Anonymous. 1995a. Disk of Knowledge. Blue Diamond Corporation. Sacramento, CA.

Anonymous. 1995b. Procedure number 510. Sigma Diagnostics, St. Louis, MO.

Anonymous. 1996. Solid State emitters. Hamamatsu Photonics K.K., Hamamatsu City, Japan.

Anonymous. 1997. California Fruit and Nut Review, vol. 17, no. 9. California State Statistics Service.

Bendat, J. S. and A. G. Piersol. 1986. Random Data Analysis and Measurement Procedures, 2nd edition. John Wiley and Sons. New York, NY.

Benson, K.B. 1986. Television Engineering Handbook. McGraw-Hill. New York, NY.

Birth, G.S. and H.G. Hecht. 1987. The physics of near infrared reflectance. In Near-Infrared Technology in the Agricultural and Food Industries. P. Williams and K. Norris, editors. American Association of Cereal Chemists, Inc. St. Paul, MN.

Booth, V.H. 1971. Problems in the determination of FDNB-available lysine. Journal of the Science of Food and Agriculture. 22: 658-666.

Bringham, E. O., 1988. The Fast Fourier Transform and its Applications. Prentice-Hall, Inc. Englewood Cliffs, NJ.

Broughton, C. 1996. Almonds enhance food value. Cereal Foods World. 41(6): 458-462.

Bushnell, P.G. and G.A. King. 1986 The Domestic and Export Markets for California Almonds. Giannini Research Report #334. Agriculture and Natural Resources Publications, Oakland, CA.

Bushuk, W. and J.W. Lee. 1978. Biochemical and functional changes in cereals: maturation, storage and germination. In Postharvest Biology and Biotechnology. H.O. Hultin and M. Milner, editors. Food and Nutrition Press, Inc. Westport, CN.

Cheftel, J.C., L.L. Cuq and D Lorient. 1985. Amino acids, peptides, and proteins. In Food Chemistry 2nd ed. Fennema, O.R., editor. Marcel Dekker, Inc. New York, NY.

Cooley, W.L., and J. Belina. 1996. Circuit Principles. In Electronic Engineers Handbook 4th edition. Donald Christiansen, editor. McGraw-Hill, New York, NY.

Delwiche, S.R. 1995. Single wheat kernel analysis by near-infrared transmittance: protein content. Cereal Chemistry. 72(1): 11-16.

Eichner, K., R. Laible and W. Wolf. 1985. The influence of water content and temperature on the formation of Maillard reaction intermediates during drying of plant products. In Properties of Water in Foods. D. Simatos and J.L. Multon, editors, Martinus Nijhoff Publishers, Dordrecht, The Netherlands.

Eskin, N.A. 1990. Biochemistry of Foods, 2nd edition. Academic Press, San Diego, CA.

Gibson, J.J., L.J. Thorpe, R.A. Kupnicki, G. Snell, M. O. Felix, L.H. Hoke, Jr., and S. Urban. 1996. Video and facsimile systems. In Electronic Engineers Handbook 4th edition. Donald Christiansen, editor. McGraw-Hill, New York, NY.

Haaland, D.M. and E.V. Thomas. 1988. Partial least-squares methods for spectral analysis. 1. Relation to other quantitative calibration methods and the extraction of qualitative information. Analytical Chemistry. 60: 1193-1202.

Hodapp, M. 1996. Light emitting diodes. In Electronic Engineers Handbook 4th edition. Donald Christiansen, editor. McGraw-Hill, New York, NY.

Hodge, J. E. and E. M. Osman. 1976. Carbohydrates. In Principles of Food Science, Part 1, Food Chemistry. O.R. Fennema, editor. Marcel Dekker, Inc. New York, NY.

Horowitz, P. and W. Hill. 1989. The Art of Electronics. Cambridge University Press, London, England.

Hruschka, W.R. 1987. Multivariate calibration and data compression. In Near-Infrared Technology in the Agricultural and Food Industries. P. Williams and K. Norris, editors. American Association of Cereal Chemists, Inc. St. Paul, MN.

Huberty, C.J. 1994. Applied Discriminant Analysis. John Wiley and Sons. New York, NY.

Hunt, W.H., M.H. Neustadt, A.A. Shurkus, and L. Zeleny. 1951. Simple iodine number refractometer for testing flaxseed and soybeans. Journal of American Oil Chemists. 28(1): 5-8.

Hurrell, R.F., P. Lerman, and K.J. Carpenter. 1979. Reactive lysine in foodstuffs as measured by a rapid dye-binding procedure. Journal of Food Science. 44: 1221-1227.

Hurrell, R.F. and K.J. Carpenter. 1981. The estimation of available lysine in foodstuffs after the Maillard reactions. Progress in Food and Nutrition Science. 5(2): 159-176.

Jain, A.K. 1989. Fundamentals of Digital Image Processing. Prentice Hall, Englewood Cliffs, N.J.

Kader, A.A. and J.F. Thompson. 1992. Postharvest handling systems: Tree nuts in Postharvest Technology of Horticultural Crops. A.A. Kader, editor. University of California Division of Agriculture and Natural Resources, publication 3311. pp. 254.

Labuza, T.P. and W.M. Baisier. 1992. The kinetics of nonenzymatic browning. In Physical Chemistry of Foods. H.G. Schwartzberg and R.W. Hartel, editors. Marcel Dekker, Inc. New York, NY.

Lamb, D.T. and C.R. Huburgh Jr. 1991. Moisture determination in single soybean seeds by near-infrared transmittance. Transactions of the ASAE. 34(5): 2123-2129.

Lee, F.A. 1983. Basic Food Chemistry. AVI Publishing Company, Westport CT. pp. 291-305.

Lea, C.H. and R.S. Hannan. 1949. Studies of the reaction between proteins and reducing sugars in the "dry" state. I. The effect of water, pH and of temperature on the primary reaction between casein and glucose. Biochim. Biophys. Acta. 3: 313.

Letellier, M.N. and J.L. Cuq. 1991. In Vitro evaluation of available lysine in cereal products by near infrared reflectance spectroscopy. Lebensmittel-Wissenschaft und -Technologie. 24(1): 17-22.

Leung, H.K. 1987. Influence of water activity on chemical reactivity. In Water Activity: Theory and Applications to Food, L.B. Rockland and L.R. Beuchat, editors. Marcel Dekker, Inc. New York, NY.

MacAdam, D.L. 1970. Sources of Color Science. MIT Press. Cambrige, MA.

McDonald, R.S. and P.A. Wilks, Jr. 1988. JCAMP-DX: A standard form for exchange of infrared spectra in computer readable form. Applied Spectrascopy. 24(1): 151-162.

Martens, H. and T. Naes. 1987. Multivariate calibration and data compression. In Near-Infrared Technology in the Agricultural and Food Industries. P. Williams and K. Norris, editors. American Association of Cereal Chemists, Inc. St. Paul, MN.

Mark, H.L. and D. Tunnell. 1985. Qualitative near-infrared reflectance analysis using Mahalanobis distances. Analytical Chemistry. 57(7): 1449-1456.

Murray, I. and P.C. Williams. 1987. Chemical principles of near-infrared technology. In Near-Infrared Technology in the Agricultural and Food Industries. P. Williams and K. Norris, editors. American Association of Cereal Chemists, Inc. St. Paul, MN.

Orman, B.A. and R.A. Schumann Jr. 1992. Nondestructive single-kernel oil determination of maize by near infrared transmission spectroscopy. Journal of American Oil Chemists Society. 69(10): 1036-1038.

Palais, J.C. 1988. Fiber Optic Communications, 3rd edition. Prentice Hall, Englewood Cliffs, NJ.

Pearson, T.C. 1996. Machine vision system for sorting stained pistachio nuts. Lebensmittel-Wissenschaft und -Technologie 29: 203-209.

Pike, O.A. 1994. Fat Characterization. In Introduction to the Chemical Analysis of Foods. S.S. Nielsen, editor. Jones and Bartlet Publishers, Boston, MA.

Pokorny, J. 1981. Browning from lipid-protein interactions. Progress in Food and Nutrition Science. 5(3): 421-428

Pomeranz, Y., and C.E. Meloan. 1994. Food Analysis, Theory and Practice. Chapman and Hall. New York, NY.

Prichavudhi, K., and H. Y. Yamamoto. 1965. Effect of drying temperature on chemical composition and quality of macadamia nuts. Food Technology. 1153: 129-132.

Rackis, J.J. 1978. Biochemical changes in soybeans: maturation, postharvest storage and processing, and germination. In Postharvest Biology and Biotechnology. H.O. Hultin and M. Milner, editors. Food and Nutrition Press, Inc. Westport, CT.

Reil, W, J.M. Labavitch, D. Holmberg. 1996. Harvesting. In Almond Production Manual. W.C. Micke, editor. University of California, Division of Agriculture and Natural Resources (publication 3364). Oakland, CA. pp. 260-264.

Rhee, K.S. and K.C. Rhee. 1981. Nutritional evaluation of the protein in oilseed products heated with sugars. Journal of Food Science. 46: 164-168.

Richardson, T. and D.B. Hyslop. 1985. In Food Chemistry 2nd ed. O.R. Fennema, editor, Marcel Dekker, Inc. New York, NY.

Rosengarten Jr., F. 1984. The Book of Edible Nuts. Walker and Company, New York, NY.

Rubenthaler, G.L. and B.L. Bruinsma. Lysine estimation in cereals by near-infrared reflectance. Crop Science. 18: 1039-1042.

SAS Institute Inc. 1994. SAS/STAT User's guide, Version 6, Fourth edition, Volume 2, SAS Institute Inc., Cary, NC.

Shallenberger, R.S. 1974. Browning reactions, nonenzymatic. In Encyclopedia of Food Technology. A.H. Johnson and M.S. Peterson, editors. The AVI Publishing Company, Inc. Westport, CT.

Scherz, H. and G. Kloos. 1981. Food Composition and Nutrition Tables 1981/82. Wissenschaftliche Verlagsgesellschaft mbH, Stuttgart, Germany. pp. 1122-1123.

Soler, L., J. Canellas, F. Saura-Calixto. 1989. Changes in carbohydrate and protein content and composition of developing almond seeds. Journal of Agriculture and Food Chemistry. 37: 1400-1404.

Stoddard, M. 1995. Personal Communication. Blue Diamond Growers. Sacramento, CA.

Thompson, J.F., T.R. Rumsey, J.H. Connell. 1996. Drying. In Almond Production Manual. W.C. Micke, editor. University of California, Division of Agriculture and Natural Resources (publication 3364). Oakland, CA. pp. 268-273.

Tkachuk, R. 1987. Analysis of whole grains by near-infrared reflectance. In Near-Infrared Technology in the Agricultural and Food Industries. P. Williams and K. Norris, editors. American Association of Cereal Chemists, Inc. St. Paul, MN.

Tseng, H.F., J.R. Ambrose, and M. Fattahi. 1985. Evolution of the solid state image sensor. Journal of Imaging Science. 29(1): 1-7.

Wettlaufer, S.A. and C. Leopold. 1991. Relevance of Amadori and Maillard products to seed deterioration. Plant Physiology. 97: 165-169.

Whistler, R.L. and J.R. Daniel. 1985. Carbohydrates. In Food Chemistry 2nd Ed. Fennema, O.R., editor. Marcel Dekker, Inc. New York, NY.

Woodroof, J.G. 1979. Tree Nuts: Production, Processing, Products, 2nd. Ed. AVI Publishing Company, Inc. Westport, CT.

Young, T. 1802. On the theory of light and colors. Philosophical transactions of the Royal Society of London. 92: 20-71.

APPENDIX A

```
/* cd_parse_cd.c  Tom Pearson   Jan 14, 1997

Reads a data file generated by Control Development spectrometer.  Strips data
above 1500nm.  Puts remaining data in array with the top row being the wave-
length and the subsequent rows being the spectra from one sample. First two
columns contain the treatment code and sample number provided that the sample
number starts at element 6 of the spectrum filename and the treatment code is
element 4.
*/


#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{

int i, j, sample, dummywave;
float specdata[100][1300], wave[1300];
char outfile[25], newline, tab, comma, space, treatment;
FILE *finSSM, *finCD, *fout;


printf("\n  output data file to store data: ");
scanf("%s",outfile);
fout = fopen(outfile,"w");



/************************* TREATMENT A DATA *************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_a_ssm_data", "r");
        if(finSSM == NULL)
                {
                printf("\n\n ***** file not found (SSM)*****\n\n");
                exit(-1);
                }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_a_cd_data", "r");
        if(finCD == NULL)
                {
                printf("\n\n ***** file not found (CD) *****\n\n");
                exit(-1);
                }
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&wave[j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        fscanf(finSSM,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finSSM,"%s",&treatment);
        fscanf(finSSM,"%c",&space);
        fscanf(finSSM,"%d",&sample);
        fscanf(finSSM,"%c",&space);
        printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&specdata[sample][j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        }
fclose(finSSM);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%d",&dummywave);
                    wave[j] = dummywave * 1.0;
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        fscanf(finCD,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finCD,"%s",&treatment);
        fscanf(finCD,"%c",&space);
        fscanf(finCD,"%d",&sample);
        fscanf(finCD,"%c",&space);
        printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%f",&specdata[sample][j]);
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        }
fclose(finCD);

/*********** STORE DATA *******************/

for(j=1; j<=1264; j++)
        {
        fprintf(fout, "%4.1f, ",wave[j]);
        }
fprintf(fout, "\n");

for(i=1; i<=81; i++)
        {
        fprintf(fout, "\n%c  %d ", treatment, i);
        for(j=1; j<=1264; j++)
                {
                fprintf(fout, "%4.1f, ",specdata[i][j]);
                }
        }
fprintf(fout, "\n");
```

```
/************************* TREATMENT B DATA *************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_b_ssm_data", "r");
        if(finSSM == NULL)
                {
                printf("\n\n ***** file not found (SSM)*****\n\n");
                exit(-1);
                }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_b_cd_data", "r");
        if(finCD == NULL)
                {
                printf("\n\n ***** file not found (CD) *****\n\n");
                exit(-1);
                }
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&wave[j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        fscanf(finSSM,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finSSM,"%s",&treatment);
        fscanf(finSSM,"%c",&space);
        fscanf(finSSM,"%d",&sample);
        fscanf(finSSM,"%c",&space);
        printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&specdata[sample][j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        }
fclose(finSSM);
```

```
    for(j=665; j<=1264; j++)
        {
        fscanf(finCD,"%d",&dummywave);
            wave[j] = dummywave * 1.0;
        fscanf(finCD,"%c",&comma);
        fscanf(finCD,"%c",&space);
        }
    fscanf(finCD,"%c",&newline);
    fscanf(finCD,"%c",&newline);
    sample = 1;
    for(i=1; i<=81; i++)
    {
    fscanf(finCD,"%s",&treatment);
    fscanf(finCD,"%c",&space);
    fscanf(finCD,"%d",&sample);
    fscanf(finCD,"%c",&space);
    printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
    for(j=665; j<=1264; j++)
        {
        fscanf(finCD,"%f",&specdata[sample][j]);
        fscanf(finCD,"%c",&comma);
        fscanf(finCD,"%c",&space);
        }
    fscanf(finCD,"%c",&newline);
    }
fclose(finCD);
/*********** STORE DATA ******************/

for(i=1; i<=81; i++)
    {
    fprintf(fout, "\n%c  %d ", treatment, i);
    for(j=1; j<=1264; j++)
        {
        fprintf(fout, "%4.1f, ",specdata[i][j]);
        }
    }
fprintf(fout, "\n");
```

```
/************************ TREATMENT C DATA ************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_c_ssm_data", "r");
        if(finSSM == NULL)
                {
                printf("\n\n ***** file not found (SSM)*****\n\n");
                exit(-1);
                }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_c_cd_data", "r");
        if(finCD == NULL)
                {
                printf("\n\n ***** file not found (CD) *****\n\n");
                exit(-1);
                }
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&wave[j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        fscanf(finSSM,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finSSM,"%s",&treatment);
        fscanf(finSSM,"%c",&space);
        fscanf(finSSM,"%d",&sample);
        fscanf(finSSM,"%c",&space);
        printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&specdata[sample][j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        }
fclose(finSSM);
```

```
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%d",&dummywave);
                        wave[j] = dummywave * 1.0;
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        fscanf(finCD,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finCD,"%s",&treatment);
        fscanf(finCD,"%c",&space);
        fscanf(finCD,"%d",&sample);
        fscanf(finCD,"%c",&space);
        printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%f",&specdata[sample][j]);
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        }
fclose(finCD);

/*********** STORE DATA *******************/

for(i=1; i<=81; i++)
        {
        fprintf(fout, "\n%c  %d ", treatment, i);
        for(j=1; j<=1264; j++)
                {
                fprintf(fout, "%4.1f, ",specdata[i][j]);
                }
        }
```

```
fprintf(fout, "\n");

/************************ TREATMENT D DATA ************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_d_ssm_data", "r");
        if(finSSM == NULL)
                {
                printf("\n\n ***** file not found (SSM)*****\n\n");
                exit(-1);
                }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_d_cd_data", "r");
        if(finCD == NULL)
                {
                printf("\n\n ***** file not found (CD) *****\n\n");
                exit(-1);
                }
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&wave[j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        fscanf(finSSM,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finSSM,"%s",&treatment);
        fscanf(finSSM,"%c",&space);
        fscanf(finSSM,"%d",&sample);
        fscanf(finSSM,"%c",&space);
        printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&specdata[sample][j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        }
```

```
fclose(finSSM);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%d",&dummywave);
                        wave[j] = dummywave * 1.0;
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        fscanf(finCD,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finCD,"%s",&treatment);
        fscanf(finCD,"%c",&space);
        fscanf(finCD,"%d",&sample);
        fscanf(finCD,"%c",&space);
        printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%f",&specdata[sample][j]);
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        }
fclose(finCD);

/*********** STORE DATA *******************/

for(i=1; i<=81; i++)
        {
        fprintf(fout, "\n%c  %d ", treatment, i);
        for(j=1; j<=1264; j++)
                {
                fprintf(fout, "%4.1f, ",specdata[i][j]);
                }
```

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



/************************ TREATMENT E DATA ************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_e_ssm_data", "r");
    if(finSSM == NULL)
        {
        printf("\n\n ***** file not found (SSM)*****\n\n");
        exit(-1);
        }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_e_cd_data", "r");
    if(finCD == NULL)
        {
        printf("\n\n ***** file not found (CD) *****\n\n");
        exit(-1);
        }
    for(j=1; j<=664; j++)
        {
        fscanf(finSSM,"%f",&wave[j]);
        fscanf(finSSM,"%c",&comma);
        fscanf(finSSM,"%c",&space);
        }
    fscanf(finSSM,"%c",&newline);
    sample = 1;
    for(i=1; i<=81; i++)
    {
    fscanf(finSSM,"%s",&treatment);
    fscanf(finSSM,"%c",&space);
    fscanf(finSSM,"%d",&sample);
    fscanf(finSSM,"%c",&space);
    printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
    for(j=1; j<=664; j++)
        {
        fscanf(finSSM,"%f",&specdata[sample][j]);
        fscanf(finSSM,"%c",&comma);
        fscanf(finSSM,"%c",&space);
        }
    }
fclose(finSSM);
    for(j=665; j<=1264; j++)
        {
        fscanf(finCD,"%d",&dummywave);
            wave[j] = dummywave * 1.0;
        fscanf(finCD,"%c",&comma);
        fscanf(finCD,"%c",&space);
        }
    fscanf(finCD,"%c",&newline);
    fscanf(finCD,"%c",&newline);
    sample = 1;
    for(i=1; i<=81; i++)
    {
    fscanf(finCD,"%s",&treatment);
    fscanf(finCD,"%c",&space);
    fscanf(finCD,"%d",&sample);
    fscanf(finCD,"%c",&space);
    printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
    for(j=665; j<=1264; j++)
        {
        fscanf(finCD,"%f",&specdata[sample][j]);
        fscanf(finCD,"%c",&comma);
        fscanf(finCD,"%c",&space);
        }
    fscanf(finCD,"%c",&newline);
    }
fclose(finCD);

/*********** STORE DATA *******************/

for(i=1; i<=81; i++)
    {
    fprintf(fout, "\n%c  %d ", treatment, i);
    for(j=1; j<=1264; j++)
        {
                fprintf(fout, "%4.1f, ",specdata[i][j]);
                }
        }
fprintf(fout, "\n");



/************************ TREATMENT F DATA ************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_f_ssm_data", "r");
        if(finSSM == NULL)
                {
                printf("\n\n ***** file not found (SSM)*****\n\n");
                exit(-1);
                }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_f_cd_data", "r");
        if(finCD == NULL)
                {
                printf("\n\n ***** file not found (CD) *****\n\n");
                exit(-1);
                }
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&wave[j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        fscanf(finSSM,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finSSM,"%s",&treatment);
        fscanf(finSSM,"%c",&space);
        fscanf(finSSM,"%d",&sample);
        fscanf(finSSM,"%c",&space);
        printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&specdata[sample][j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        }
fclose(finSSM);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%d",&dummywave);
                        wave[j] = dummywave * 1.0;
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        fscanf(finCD,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finCD,"%s",&treatment);
        fscanf(finCD,"%c",&space);
        fscanf(finCD,"%d",&sample);
        fscanf(finCD,"%c",&space);
        printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%f",&specdata[sample][j]);
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        }
fclose(finCD);

/*********** STORE DATA *******************/

for(i=1; i<=81; i++)
        {
        fprintf(fout, "\n%c  %d ", treatment, i);
```

```
          for(j=1; j<=1264; j++)
                    {
                    fprintf(fout, "%4.1f, ",specdata[i][j]);
                    }
          }
fprintf(fout, "\n");



/************************* TREATMENT G DATA *************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_g_ssm_data", "r");
          if(finSSM == NULL)
                    {
                    printf("\n\n ***** file not found (SSM)*****\n\n");
                    exit(-1);
                    }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_g_cd_data", "r");
          if(finCD == NULL)
                    {
                    printf("\n\n ***** file not found (CD) *****\n\n");
                    exit(-1);
                    }
          for(j=1; j<=664; j++)
                    {
                    fscanf(finSSM,"%f",&wave[j]);
                    fscanf(finSSM,"%c",&comma);
                    fscanf(finSSM,"%c",&space);
                    }
          fscanf(finSSM,"%c",&newline);
          sample = 1;
          for(i=1; i<=81; i++)
          {
          fscanf(finSSM,"%s",&treatment);
          fscanf(finSSM,"%c",&space);
          fscanf(finSSM,"%d",&sample);
          fscanf(finSSM,"%c",&space);
          printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
          for(j=1; j<=664; j++)
```

```
                {
                fscanf(finSSM,"%f",&specdata[sample][j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        }
fclose(finSSM);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%d",&dummywave);
                        wave[j] = dummywave * 1.0;
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        fscanf(finCD,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finCD,"%s",&treatment);
        fscanf(finCD,"%c",&space);
        fscanf(finCD,"%d",&sample);
        fscanf(finCD,"%c",&space);
        printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
        for(j=665; j<=1264; j++)
                {
                fscanf(finCD,"%f",&specdata[sample][j]);
                fscanf(finCD,"%c",&comma);
                fscanf(finCD,"%c",&space);
                }
        fscanf(finCD,"%c",&newline);
        }
fclose(finCD);

/*********** STORE DATA ******************/

for(i=1; i<=81; i++)
```

```
        {
        fprintf(fout, "\n%c  %d ", treatment, i);
        for(j=1; j<=1264; j++)
                {
                fprintf(fout, "%4.1f, ",specdata[i][j]);
                }
        }
fprintf(fout, "\n");



/************************ TREATMENT H DATA ************************/
finSSM = fopen("/home/tpearson/almond/post_dry_spectra/dry_h_ssm_data", "r");
        if(finSSM == NULL)
                {
                printf("\n\n ***** file not found (SSM)*****\n\n");
                exit(-1);
                }
finCD = fopen("/home/tpearson/almond/post_dry_spectra/dry_h_cd_data", "r");
        if(finCD == NULL)
                {
                printf("\n\n ***** file not found (CD) *****\n\n");
                exit(-1);
                }
        for(j=1; j<=664; j++)
                {
                fscanf(finSSM,"%f",&wave[j]);
                fscanf(finSSM,"%c",&comma);
                fscanf(finSSM,"%c",&space);
                }
        fscanf(finSSM,"%c",&newline);
        sample = 1;
        for(i=1; i<=81; i++)
        {
        fscanf(finSSM,"%s",&treatment);
        fscanf(finSSM,"%c",&space);
        fscanf(finSSM,"%d",&sample);
        fscanf(finSSM,"%c",&space);
```

```
    printf("treatment = %c, sample = %d  SSM DATA\n",treatment, sample);
    for(j=1; j<=664; j++)
        {
        fscanf(finSSM,"%f",&specdata[sample][j]);
        fscanf(finSSM,"%c",&comma);
        fscanf(finSSM,"%c",&space);
        }
    }
fclose(finSSM);
    for(j=665; j<=1264; j++)
        {
        fscanf(finCD,"%d",&dummywave);
            wave[j] = dummywave * 1.0;
        fscanf(finCD,"%c",&comma);
        fscanf(finCD,"%c",&space);
        }
    fscanf(finCD,"%c",&newline);
    fscanf(finCD,"%c",&newline);
    sample = 1;
    for(i=1; i<=81; i++)
    {
    fscanf(finCD,"%s",&treatment);
    fscanf(finCD,"%c",&space);
    fscanf(finCD,"%d",&sample);
    fscanf(finCD,"%c",&space);
    printf("treatment = %c, sample = %d  CD DATA\n",treatment, sample);
    for(j=665; j<=1264; j++)
        {
        fscanf(finCD,"%f",&specdata[sample][j]);
        fscanf(finCD,"%c",&comma);
        fscanf(finCD,"%c",&space);
        }
    fscanf(finCD,"%c",&newline);
    }
fclose(finCD);
```

```
/*********** STORE DATA *******************/

for(i=1; i<=81; i++)
            {
            fprintf(fout, "\n%c  %d ", treatment, i);
            for(j=1; j<=1264; j++)
                        {
                        fprintf(fout, "%4.1f, ",specdata[i][j]);
                        }
            }
fprintf(fout, "\n");

fclose (fout);

printf("\n\ndata from %4.1f to %4.1f stored\n\n",wave[1], wave[1264]);


} /* end of main */
```

APPENDIX B

```
/*  parse_ssm.c  Tom Pearson   Jan 14, 1997

Reads a SSM data file generated by Ocean Optics spectrometer.  Strips data
below 690nm and above 1000nm.  Puts remaning data in array with the top row
being the wave-length and the subsequent rows being the spectra from one sample.
The first two columns are the treatment code and sample number provided that the
treatment code is fourth element of the spectrum filename and the sample number
starts with the fith element.
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{

int i, j, filecount, sample_number[1000], sample;
float specdata[100][1100], wave[1090], dummy;
char outfile[25], infile[25], filelist[25], comma, treatment[1000];
char sample_char_1, sample_char_10;
FILE *fin, *fout, *flist;

printf("\n list of data files to parse: ");
scanf("%s",filelist);
flist = fopen(filelist,"r");

filecount = 81;

for(i=1; i<=filecount; i++)
{

fscanf(flist, "%s", infile);
```

```
     fin = fopen(infile, "r");
          if(fin == NULL)
          {
          printf("\n\n ***** file not found *****\n\n");
          exit(-1);
          }

        treatment[i] = infile[4];
        sample_number[i] = atoi(&infile[5]);
        sample = sample_number[i];

        for(j=0; j<=1040; j++)   /* read data from input file */
                {
                fscanf(fin,"%f",&wave[j]);
                fscanf(fin,"%c",&comma);
                fscanf(fin,"%f",&specdata[sample][j]);
                fscanf(fin,"%c",&comma);
                fscanf(fin,"%f",&dummy);   /* wavelength below 400nm */
                fscanf(fin,"%c",&comma);
                fscanf(fin,"%f",&dummy);   /* spectra below 400nm */
                }

fclose(fin);

}

printf("\n  output data file to store data: ");
scanf("%s",outfile);
fout = fopen(outfile,"w");

for(j=374; j<=1037; j++)
        {
        fprintf(fout, "%4.1f, ",wave[j]);
        }
fprintf(fout, "\n");

for(i=1; i<=filecount; i++)
```

```
        {
        fprintf(fout, "\n%c  %d ", treatment[i], i);
        for(j=374; j<=1037; j++)
                {
                fprintf(fout, "%4.1f, ",specdata[i][j]);
                }
        }
fprintf(fout, "\n");

fclose (fout);



for(j=374; j<=1037; j++)
        {
        printf("%4.1f, ",wave[j]);
        }
printf("\n");

for(i=1; i<=filecount; i++)
        {
        printf("\n%c  %d ", treatment[i], i);
        for(j=374; j<=1037; j++)
                {
                printf("%4.1f, ",specdata[i][j]);
                }
        }
printf("\n");

printf("\n\ndata from %4.1f to %4.1f stored\n\n",wave[374], wave[1037]);

} /* end of main */
```

APPENDIX C

```
/**********************************************************************
                        tom pearson
                        raw2jcmp.cpp
                        May 1, 1997

                        this program read a data file of spectra parameters, chemical data
                        then a spectrum for as many samples as specified in variable filecount.
                        the data is deliminated by a ", " that is a comma and space.  There is
                        a newline at the end of each sample.  The sample information variables
                        come before the spectra data.  This particular program reads 701 data
                        points in the spectra portion of the data file,  This can be changed
                        by adjusting the j parameter in the for loops.  The program outputs the
                        data in JCAMP-DX format for reading in software packages.
************************************************************************/



#include<stdio.h>
#include <stdlib.h>
#include <math.h>
void main()
{

int i,j,k, filecount, cd_score, sample, im_score;
float data[720], dummy, mean, pix, volume;
FILE *fout, *fin;
char comma, space, treatment, newline;

filecount = 648;
/* filecount = 12;  */

 fin = fopen("/home/tpearson/almond/test", "r");
        if(fin == NULL)
                {
                printf("\n\n*****input file not found ******\n\n");
                exit(-1);
                }

 fout = fopen("/home/tpearson/almond/specdata.dx", "w");
        if(fin == NULL)
        {
        printf("\n\n*****output file open not succesful******\n\n");
        exit(-1);
        }

fprintf(fout, "##TITLE= ALMOND TRANSMISION DATA\n");
fprintf(fout, "##JCAMP-DX= 4.24\n");
fprintf(fout, "##DATA TYPE= LINK\n");
fprintf(fout, "##BLOCKS= %d\n",filecount);

        /***** read data, one spectrum at a time ****/


    for(i=1; i<=filecount; i++)
    {
        fscanf(fin,"%c", &treatment);
        fscanf(fin, "%c", &comma);
        fscanf(fin, "%c", &space);

        fscanf(fin,"%d", &sample);
        fscanf(fin, "%c", &comma);
        fscanf(fin, "%c", &space);

                for(k=1; k<=7; k++)
                    {
                    fscanf(fin,"%f", &dummy);
                    fscanf(fin, "%c", &comma);
                    fscanf(fin, "%c", &space);
                    }
        fscanf(fin,"%d", &cd_score);
        fscanf(fin, "%c", &comma);
        fscanf(fin, "%c", &space);

        fscanf(fin,"%d", &im_score);
        fscanf(fin, "%c", &comma);
        fscanf(fin, "%c", &space);
```

```
                fscanf(fin,"%f", &volume);
                fscanf(fin, "%c", &comma);
                fscanf(fin, "%c", &space);

                fscanf(fin,"%f", &mean);
                fscanf(fin, "%c", &comma);
                fscanf(fin, "%c", &space);

                fscanf(fin,"%f", &pix);

                for(j=1; j<=700; j++)
                        {
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);
                        fscanf(fin,"%f", &data[j]);
                        }
fscanf(fin, "%c", &newline);



/*** write data, one spectrum at a time  ***/

fprintf(fout, "##TITLE= %c %d\n",treatment,sample);
fprintf(fout, "##JCAMP-DX= 4.24\n");
fprintf(fout, "##DATA TYPE= NIR ABSORBANCE SPECTRUM \n");
fprintf(fout, "##ORIGIN= USDA-ARS-WRRC\n");
fprintf(fout, "##OWNER= TOM PEARSON\n");
fprintf(fout, "##CONCENTRATIONS=\t(Name,\t\tConcentration,\tUnits)\n");
fprintf(fout, "\t\t\t(CD_SCORE  ,\t%d,\tn.a.)\n", cd_score);
fprintf(fout, "\t\t\t(IM_SCORE  ,\t%d,\tn.a.)\n", im_score);
fprintf(fout, "\t\t\t(PIXELS    ,\t%3.2f,\tn.a.)\n", mean);
fprintf(fout, "##SAMPLING PROCEDURE=\n");
fprintf(fout, "\tdetector mode = transmittance\n");
fprintf(fout, "##DATA PROCESSING=\n");
fprintf(fout, "\tmath treatment= 19 pnt 2nd order S-G smoothing\n");
fprintf(fout, "##XUNITS= NANOMETERS\n");
fprintf(fout, "##YUNITS= ABSORBANCE\n");
fprintf(fout, "##XFACTOR= 1.0\n");
fprintf(fout, "##YFACTOR= 1.0\n");
fprintf(fout, "##FIRSTX= 701\n");
fprintf(fout, "##LASTX=1400\n");
fprintf(fout, "##NPOINTS= 700\n");
fprintf(fout, "##FIRSTY= %f\n",data[1]);
fprintf(fout, "##DELTAX= 1.0\n");
fprintf(fout, "##XYDATA= (X++(Y..Y))\n");

        for(j=1; j<=700; j=j+7)
            {
            fprintf(fout,"%d",j+700);
                    for(k=0; k<=6; k++)
                    {
                    fprintf(fout," %f",data[j+k]);
                    }
            fprintf(fout,"\n");
            }
        fprintf(fout, "##END=\n\n");
    } /** end of filecount loop **/
fclose(fin);
fclose(fout);
} /** end of main **/
```

APPENDIX D

```
/*  spec_calc.c  Tom Pearson   April 22, 1997


    peforms 19 point Savitzky - Golay filter
    reads reference spectrum and computes absorbances
    normalizes spectra with area
    interpolates Ocean Optics data at 1 nm intervals
    includes routine to compute mean at each wavelength for each concealed damage score

*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{

int i, j, k, w, sample, dummywave, count, filecount,  cd_score[700];
int score_count[6], score, df[6][1270];
float sum_y, sum_xy, sum_xxy, a0[1270], a1[1270], a2[1270], x;
float score_sum[6][1270], score_mean[6][1270], score_sum2[6][1270];
float volume[650], specdata[650][1270], wave[1270], hue[700], n,;
float initial_mass[700], post_dry_mass[700], post_cook_mass[700], post_dry_thickness[700];
float color_L[650], color_a[650], color_b[650], delta_x, dummy_data[1270],reference[650][1270];
char outfile[25], newline, tab, comma, space, treatment[700],chardummy;
double double_ratio, score_var[6][1270];
float areaCD, areaSSM;
FILE *finSPEC, *fout, *finDATA;

filecount = 648;  /** there are 648 total spectrum samples  ***/


/************************* READ SPECTRA DATA FROM HARD DISK *************************/
finSPEC = fopen("/home/tpearson/almond/post_dry_spectra/all_spec_data", "r");
        if(finSPEC == NULL)
                {
                printf("\n\n ***** file not found (SPEC)*****\n\n");
                exit(-1);
                }

        for(j=1; j<=1264; j++)
                {
                fscanf(finSPEC,"%f",&wave[j]);
                fscanf(finSPEC,"%c",&comma);
                fscanf(finSPEC,"%c",&space);
                }
```

```
    for(j=665; j<=1264; j++)
        {
        wave[j] = wave[j] + 20;
        }

    fscanf(finSPEC,"%c",&newline);
    treatment[0] = 'x';

    for(i=1; i<=filecount; i++)
        {
        fscanf(finSPEC,"%s",&chardummy);
            treatment[i] = chardummy;
            if(chardummy != treatment[i-1])
                {
                printf("\nREADING TREATMENT %c SPECTRA",chardummy);
                }
        fscanf(finSPEC,"%c",&space);
        fscanf(finSPEC,"%d",&sample);
        fscanf(finSPEC,"%c",&space);

        for(j=1; j<=1264; j++)
            {
            fscanf(finSPEC,"%f",&specdata[i][j]);
            fscanf(finSPEC,"%c",&comma);
            fscanf(finSPEC,"%c",&space);
            }
        }

fclose(finSPEC);

/************** INPUT PROPERTY DATA FROM HARD DISK  ***********************/

finDATA = fopen("/home/tpearson/almond/post_dry_spectra/ind_prop_data.txt", "r");
    if(finSPEC == NULL)
        {
        printf("\n\n ***** file not found (DATA)*****\n\n");
        exit(-1);
```

```
                }
        printf("\n\nREADING PHYSICAL PROPERTY DATA\n\n");
        for(i=1; i<=filecount; i++)
                {
                fscanf(finSPEC,"%s",&chardummy);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%d",&sample);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&initial_mass[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&post_dry_mass[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&post_cook_mass[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&post_dry_thickness[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&color_L[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&color_a[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&color_b[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%d",&cd_score[i]);
                fscanf(finSPEC,"%c",&tab);
                fscanf(finSPEC,"%f",&volume[i]);
                fscanf(finSPEC,"%c",&newline);
                hue[i] = atan(color_b[i]/color_a[i]);
                }

fclose(finDATA);

/******************** READ REFERENCE SPECTRA **************************/

finDATA = fopen("/home/tpearson/almond/post_dry_spectra/reference.csv", "r");
        if(finDATA == NULL)
                {
                printf("\n\n ***** file not found (REFERENCE)*****\n\n");
                exit(-1);
                }
        printf("\n\nREADING REFERENCE SPECTRUM\n\n");

        for(i=1; i<=filecount; i++)
                {
                for(j=1; j<=1264; j++)
                        {
                        fscanf(finSPEC,"%f",&reference[i][j]);
                        fscanf(finSPEC,"%c",&newline);
                        }
                }

fclose(finDATA);

/********************** DATA CALCULATIONS *********************************/

        /********************* compute absorbance with reference spctrum ********************/

        for(j=1; j<=1264; j++)
                {
                if(reference[j] < 1.0) reference[j] = 1.0;
                reference[j] = reference[j]*10.0;
                }

        printf("\nCOMPUTING ABSORBANCES\n");
        for(i=1; i<=filecount; i++)
                {
                        for(j=1; j<=1264; j++)
                                {
                                if(specdata[i][j] < 1.0) specdata[i][j] = 1.0;
                                 = fabs(reference[j]/specdata[i][j]);
                                specdata[i][j] = log(double_ratio);
                                }
                }
```

```
/******** Normalize intensity (or absorbance) values with spectrum area  ***********/

printf("\nNORMALIZING DATA\n");
for(i=1; i<=filecount; i++)
        {
        areaSSM = 0.0;
        areaCD = 0.0;
        wave[0] = 690.0;
        for(j=1; j<=664; j++)
                {
                areaSSM = areaSSM + fabs((wave[j]-wave[j-1])*specdata[i][j]);
                }

                areaSSM = areaSSM/309.2;
                for(j=1; j<=664; j++)
                        {
                        specdata[i][j] = specdata[i][j]/areaSSM;
                        }

        for(j=666; j<=1264; j++)
                {
                areaCD = areaCD + fabs((wave[j]-wave[j-1])*specdata[i][j]);
                }

                areaCD = areaCD/600.0;
                for(j=665; j<=1264; j++)
                        {
                        specdata[i][j] = specdata[i][j]/areaCD;
                        }
        }



/**** perform Savitzki - Golay polynomial fit using 19 points, and 2nd order on SSM data only*****/

printf("\nPERFORMING SAVITZKI - GOLAY SMOOTHING AND INTERPOLATION\n");
for(i=1; i<=filecount; i++)
```

```
        {
        for(j=10; j<=655; j=j++)
            {
            sum_y = 0.0;
            sum_xy = 0.0;
            sum_xxy = 0.0;
            for(k=j-9; k<=j+9; k=k++)
                {
                sum_y = sum_y + specdata[i][k];
                sum_xy = sum_xy + (k-j)*specdata[i][k];
                sum_xxy = sum_xxy + (k-j)*(k-j)*specdata[i][k];

                a2[j] = (sum_xxy - 30.0*sum_y)/13566.0;
                a1[j] = sum_xy/570.0;
                a0[j] = (sum_y - 570.0*a2[j])/19.0;
                }
            }
        for(w=700; w<=975; w=w++)
            {
            j = 1;
            while (wave[j] <= w*1.0) j=j+1;
            x = (w*1.0 - wave[j-1])/(wave[j] - wave[j-1]);
            specdata[i][w-699] = a0[j] + a1[j]*x +a2[j]*x*x;
            }
        }

    for(i=1; i<=filecount; i++)
        {
        for(j=740; j<=1165; j=j++)
            {
            specdata[i][j-464] = specdata[i][j];
            }
        }


/******************************* compute spectra means for each cd_score  ***********/
```

```
printf("\nCOMPUTING SCORE MEANS AT EACH WAVELEGNTH DATAPOINT\n");
for(k=1; k<=5; k++)
        {
        for(j=1; j<=701; j++)
                {
                score_sum[k][j] = 0.0;
                score_sum2[k][j] = 0.0;
                }
        score_count[k] = 0;
        }

for(i=1; i<=filecount; i++)
        {
        score = cd_score[i];
        score_count[score] = score_count[score] + 1;
        for(j=1; j<=701; j++)
                {
                score_sum[score][j] = specdata[i][j] + score_sum[score][j];
                score_sum2[score][j] = specdata[i][j]*specdata[i][j] + score_sum2[score][j];
                }
        }
for(k=1; k<=5; k++)
        {
        for(j=1; j<=701; j++)
                {
                n = score_count[k] * 1.0;
                score_mean[k][j] = score_sum[k][j]/n;
                score_var[k][j] = (n*score_sum2[k][j] -
score_mean[k][j]*score_mean[k][j])/(n*(n-1));
                }
        }



/*********** STORE DATA *******************/

printf("\n  output data file to store data: ");
```

```
scanf("%s",outfile);
fout = fopen(outfile,"w");


count = 1;
for(i=1; i<=filecount; i++)
        {
        fprintf(fout, "\n%c, %d, %f, %f, ", treatment[i], count, initial_mass[i], post_dry_mass[i]);
        fprintf(fout, "%f, %f, %f, %f, %f, %d", post_cook_mass[i], post_dry_thickness[i], color_L[i],
color_a[i], color_b[i], cd_score[i], volume[i]);
        count = count + 1;
        if(count > 81)
                {
                count = 1;
                printf("\nTREATMANT %c DATA STORED specdata[81][1264] = %f \n",
treatment[i-2],specdata[i][1264]);
                }
        for(j=1; j<=701; j=j++)
                {
                fprintf(fout, ", %f",specdata[i][j]);
                }

        }
fprintf(fout, "\n");


fclose(fout);

} /* end of main */
```

APPENDIX E

```
/****************************************************************************


                t_value.c
                Tom Pearson
                April 18, 1997

                Reads Savitzki - Golay smoothed data generated by spec_calc.c and stored in
file
                "s_g_spec.csv.  Includes routine to compute second derivative by central
                difference method.  Computes t_values for different means for nuts with an image
                score of 1 and 5.
                While computing t_values, unequal variances are assumed for the two groups.

***************************************************************************************/

#include<stdio.h>
#include <stdlib.h>
#include <math.h>
void main()
{

int i,j,k, filecount, cd_score[650], sample[650], image_score[650];
float data[650][720], gray_mean[650], thresh_pix[650], volume[650];
float initial_mass[650], dry_mass[650], cook_mass[650], thick[650];
float L_value[650], a_value[650], b_value[650];
float n1, n2, w1, w2, t_1[750], score_var[6][750], score_sum2[6][750];
FILE *fout, *fin;
char comma, space, treatment[650], newline;
int score, score_count[6];
float score_mean[6][720], n, df, dummy, derv2_spec[650][720];


filecount = 648;

 fin = fopen("/home/tpearson/almond/s_g_spec.csv", "r");
```

```
            if(fin == NULL)
                        {
                        printf("\n\n*****spectra file not found ******\n\n");
                        exit(-1);
                        }



 fout = fopen("/home/tpearson/almond/Dtvalues.csv", "w");
            if(fout == NULL)
                        {
                        printf("\n\n*****output file open not succesful******\n\n");
                        exit(-1);
                        }
 for(i=1; i<=filecount; i++)
            {

/** read physical property data and spectra **/

                        fscanf(fin,"%c", &treatment[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%d", &sample[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &initial_mass[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &dry_mass[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &cook_mass[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);
                        fscanf(fin,"%f", &thick[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &L_value[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &a_value[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &b_value[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%d", &cd_score[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%d", &image_score[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &volume[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &gray_mean[i]);
                        fscanf(fin, "%c", &comma);
                        fscanf(fin, "%c", &space);

                        fscanf(fin,"%f", &thresh_pix[i]);

                        for(j=1; j<=700; j++)
                                {
                                fscanf(fin, "%c", &comma);
                                fscanf(fin, "%c", &space);
                                fscanf(fin,"%f", &data[i][j]);
                                }
                        fscanf(fin, "%c", &newline);

                } /** end of filecount loop **/

                    /******* compute second derivitive *************/

                    printf("\nCOMPUTING DERIVATIVES\n");
                    for(i=1; i<=filecount; i++)
                        {
                        for(j=11; j<=690; j++)
                        {
                        derv2_spec[i][j] = (data[i][j+10] - (2.0*data[i][j]) + data[i][j-10])/(100);
                        }
                        }
                    for(i=1; i<=filecount; i++)
                        {
                        for(j=11; j<=690; j++)
                        {
                        data[i][j] = derv2_spec[i][j];
                        }
                        }



                /************* compute spectra means for each cd_score *********/

                for(k=1; k<=5; k++)
                    {
                    score_count[k] = 0;
                    for(j=1; j<=700; j=j+1)
                        {
                        score_mean[k][j] = 0.0;
                        score_sum2[k][j] = 0.0;
```

```
                }
        }

for(i=1; i<=filecount; i++)
        {
        score = image_score[i];
        score_count[score] = score_count[score] + 1;
        for(j=1; j<=700; j=j+1)
                {
                score_mean[score][j] = score_mean[score][j] + data[i][j];
                score_sum2[score][j] = data[i][j]*data[i][j] + score_sum2[score][j];
                }
        }

for(k=1; k<=5; k++)
        {
        for(j=1; j<=700; j=j+1)
                {
                n = score_count[k]*1.0;
                score_mean[k][j] = score_mean[k][j]/n;
                score_var[k][j] = (n*score_sum2[k][j] -
score_mean[k][j]*score_mean[k][j])/(n*(n-1));
                }
        }


/*************************** compute t values for unequal means *************************/
        /** assume unequal variances so use t' = (Xbar1 - Xbar2)sqrt(var1/n1 + var2/n2)  **/
        /** reference SAS Users Guide, vvol. 2, pg 1636  **/

printf("\nCOMPUTING T VALUES\n");


        for(j=1; j<=700; j++)
                {
                n1 = score_count[1]*1.0;
                n2 = score_count[5]*1.0;
```

```
                    w1 = score_var[1][j]/n1;
                    w2 = score_var[5][j]/n2;
                    t_1[j] = (score_mean[1][j] - score_mean[5][j])/sqrt(w1+w2);
                    }


/*** OUTPUT DATA  ***/

/*********** output t value data ***********/

for(j=1; j<=700; j=j+1)
          {
fprintf(fout, "%d, %f\n",j+700, t_1[j]);
          }



/*********** output score mean data ***********/
/*
for(j=1; j<=700; j=j+1)
          {
fprintf(fout, "%d, %f, %f, %f, %f, %f\n",j+700,
score_mean[1][j],score_mean[2][j],score_mean[3][j],score_mean[4][j],score_mean[5][j]);
          }

*/
/**************** OUTPUT ALL READ DATA *************/
/*
for(i=1; i<=filecount; i++)
          {
fprintf(fout, "%c, %d, %f, %f, ", treatment[i], sample[i], initial_mass[i], dry_mass[i]);
fprintf(fout, "%1.4f, %1.4f, %2.4f, %2.4f, ", cook_mass[i], thick[i], L_value[i], a_value[i]);
fprintf(fout, "%2.4f, %d, %d, ", b_value[i], cd_score[i], image_score[i]);
fprintf(fout, "%f, %f, %f",volume[i], gray_mean[i], thresh_pix[i]);
                    for(j=1; j<=700; j=j+1)
                              {
                              fprintf(fout,", %f",data[i][j]);
                              }
                    fprintf(fout,"\n");
          }


fclose(fin);
fclose(fout);
} /** end of main **/
```

APPENDIX F

```
/******** discrimD.c       Tom Pearson           May 15, 1997 ***************/
/****** computes best three variable dicriminate models comprised of absorbance areas**********/
/******    computes best three variable dicriminate models, the models can be any combination
           of two variables desired, such as derivatives, ratios, etc.
           The best three variable discriminant models are computed using
           spectra from 700 nm to 970nm.  Computes discriminant functions for one group,
           labled 1, comprising only of nuts having a mean gray level greater than 160 (good nuts),
           and another discriminant function, labled 2, for nuts with a gray mean below 160
           (concealed damaged)  These functions are developed using odd numbered samples only.
           Classification is performed on the even numbered samples using the two discriminant
           function.  A nut is classified into a group, either “1" or “2" based on having
           a lower D^2 value.  Error is tracked by summing the absolute value of
           the difference between the actual score and the classification score.
.

#include<stdio.h>
#include <stdlib.h>
#include <math.h>
void main()
{

int i,j,k, filecount, cd_score[650], sample[650], image_score[650];
float data[650][60], gray_mean[650];
float L_value[650], a_value[650], b_value[650];
FILE *fout, *fin;
char comma, space, treatment[650], newline;
int score, score_count[6];
float score_mean[6][60], derv[650][60];

int x, y, z, error, Dcd, Dg, class_score, countg, countcd;
int minx, miny, minz, min2x, min2y, min2z, minerr, min2err;
double g_cov_xx, g_cov_xy, g_cov_xz, g_cov_yy, g_cov_yz, g_cov_zz;
double total_df, cd_count, mag, Cg, Cgx, Cgy, Cgz, Ccd, Ccdx, Ccdy, Ccdz;
double cd_cov_xx, cd_cov_xy, cd_cov_xz, cd_cov_yy, cd_cov_yz, cd_cov_zz, g_mean[60],
g_count;
double pcov_xx, pcov_xy, pcov_xz, pcov_yx, pcov_yy, pcov_yz, pcov_zx, pcov_zy, pcov_zz;
double inv_pcov_xx, inv_pcov_xy, inv_pcov_xz, inv_pcov_yx;
double inv_pcov_yy, inv_pcov_yz, inv_pcov_zx, inv_pcov_zy, inv_pcov_zz;
double g_mean_X, cd_mean_X, g_sum_X, cd_sum_X, X[650];
int X1, X2, min2_X1, min2_X2, min_X1, min_X2, damage[650];
double g_mean_Y, cd_mean_Y, g_sum_Y, cd_sum_Y, Y[650];
int Y1, Y2, min2_Y1, min2_Y2, min_Y1, min_Y2, area, width;
double g_mean_Z, cd_mean_Z, g_sum_Z, cd_sum_Z, Z[650];
int Z1, Z2, min2_Z1, min2_Z2, min_Z1, min_Z2, stop_X, stop_Y, stop_Z;

filecount = 648;
minerr = 1000;
min2err = 1000;

 fin = fopen("/home/tpearson/almond/spectra.csv", "r");
        if(fin == NULL)
                {
                printf("\n\n*****spectra file not found ******\n\n");
                exit(-1);
                }



 fout = fopen("/home/tpearson/almond/derv2_results", "w");
        if(fout == NULL)
                {
                printf("\n\n*****output file open not succesful******\n\n");
                exit(-1);
                }
 for(i=1; i<=filecount; i++)
        {

/** read physical property data and spectra **/


                fscanf(fin,"%c", &treatment[i]);
                fscanf(fin, "%c", &comma);
                fscanf(fin, "%c", &space);

                fscanf(fin,"%d", &sample[i]);
                fscanf(fin, "%c", &comma);
                fscanf(fin, "%c", &space);

                fscanf(fin,"%d", &cd_score[i]);
                fscanf(fin, "%c", &comma);
                fscanf(fin, "%c", &space);

                fscanf(fin,"%d", &image_score[i]);
                fscanf(fin, "%c", &comma);
                fscanf(fin, "%c", &space);

                fscanf(fin,"%f", &gray_mean[i]);


                for(j=1; j<=54; j++)
                    {
                    fscanf(fin, "%c", &comma);
                    fscanf(fin, "%c", &space);
                    fscanf(fin,"%f", &data[i][j]);
                    }
                fscanf(fin, "%c", &newline);

} /** end of filecount loop **/



/************ assign scores based on gray level mean *********/

        for(i=1; i<=filecount; i++)
                {
                damage[i] = 1;
                if (gray_mean[i] <= 160.0) damage[i] = 2;
                }
```

```
/********** find best three variable discriminant models *****/



for(X1=1; X1<=54; X1=X1+2)
{
stop_X = X1+10;
if(stop_X >= 54) stop_X = 54;
for(X2=X1+1; X2<=stop_X; X2=X2+1)
{
printf("X1 = %d  X2 = %d\n",X1, X2);

        countg = 0;
        countcd = 0;
        cd_sum_X = 0.0;
        g_sum_X = 0.0;
        width = X2-X1;
        for(i=1; i<=filecount; i++)
                {
                X[i] = (data[i][X1-width]-2*data[i][X1]+data[i][X1+width])/(width*width);
                }
        for(i=1; i<=filecount; i=i+2)
                {
                if ( damage[i] == 1 )
                        {
                        g_sum_X = g_sum_X + X[i];
                        countg = countg + 1;
                        }

                if ( damage[i] == 5 )
                        {
                        cd_sum_X = cd_sum_X + X[i];
                        countcd = countcd + 1;
                        }
                }

        cd_count = countcd*1.0;
```

```
          g_count = countg*1.0;
          g_mean_X = g_sum_X/g_count;
          cd_mean_X = cd_sum_X/cd_count;
          total_df = g_count+cd_count - 1.0;


for(Y1=3; Y1<=53; Y1=Y1+2)
{
stop_Y = Y1+10;
if(stop_Y >= 54) stop_Y = 54;
for(Y2=Y1+1; Y2<=stop_Y; Y2=Y2+1)
{

          if(Y2==X2 && Y1 == X1) Y2=Y2+1;

          cd_sum_Y = 0.0;
          g_sum_Y = 0.0;
          width = Y2-Y1;
          for(i=1; i<=filecount; i=i+2)
                    {
                    Y[i] = (data[i][Y1-width]-2*data[i][Y1]+data[i][Y1+width])/(width*width);
                    }
          for(i=1; i<=filecount; i=i+2)
                    {
                    if ( damage[i] == 1 ) g_sum_Y = g_sum_Y + Y[i];
                    if ( damage[i] == 5 ) cd_sum_Y = cd_sum_Y + Y[i];
                    }
          g_mean_Y = g_sum_Y/g_count;
          cd_mean_Y = cd_sum_Y/cd_count;


for(Z1=5; Z1<=53; Z1=Z1+2)
{
stop_Z = Z1+10;
if(stop_Z >= 54) stop_Z = 54;
for(Z2=Z1+1; Z2<=stop_Z; Z2=Z2+1)
{
```

```
          if(Z2==X2 && Z1 == X1) Z2=Z2+1;
          if(Z2==Y2 && Z1 == Y1) Z2=Z2+1;

          cd_sum_Z = 0.0;
          g_sum_Z = 0.0;
          width = Z2-Z1;
          for(i=1; i<=filecount; i=i+2)
                    {
                    Z[i] = (data[i][Z1-width]-2*data[i][Z1]+data[i][Z1+width])/(width*width);
                    }
          for(i=1; i<=filecount; i=i+2)
                    {
                    if ( damage[i] == 1) g_sum_Z = g_sum_Z + Z[i];
                    if ( damage[i] == 5 ) cd_sum_Z = cd_sum_Z + Z[i];
                    }
          g_mean_Z = g_sum_Z/g_count;
          cd_mean_Z = cd_sum_Z/cd_count;

/*************** compute covariance matricies for three variables from each group ***********/


g_cov_xx = 0.0;
g_cov_xy = 0.0;
g_cov_xz = 0.0;
g_cov_yy = 0.0;
g_cov_yz = 0.0;
g_cov_zz = 0.0;

cd_cov_xx = 0.0;
cd_cov_xy = 0.0;
cd_cov_xz = 0.0;
cd_cov_yy = 0.0;
cd_cov_yz = 0.0;
cd_cov_zz = 0.0;

for(i=1; i<=filecount; i=i+2)
          {
  if(damage[i] == 1)
              {

              g_cov_xx = g_cov_xx + (X[i] - g_mean_X)*(X[i] - g_mean_X);
              g_cov_xy = g_cov_xy + (X[i] - g_mean_X)*(Y[i] - g_mean_Y);
              g_cov_xz = g_cov_xz + (X[i] - g_mean_X)*(Z[i] - g_mean_Z);

              g_cov_yy = g_cov_yy + (Y[i] - g_mean_Y)*(Y[i] - g_mean_Y);
              g_cov_yz = g_cov_yz + (Y[i] - g_mean_Y)*(Z[i] - g_mean_Z);

              g_cov_zz = g_cov_zz + (Z[i] - g_mean_Z)*(Z[i] - g_mean_Z);
              }

  if(damage[i] == 5)
              {
              cd_cov_xx = cd_cov_xx + (X[i] - cd_mean_X)*(X[i] - cd_mean_X);
              cd_cov_xy = cd_cov_xy + (X[i] - cd_mean_X)*(Y[i] - cd_mean_Y);
              cd_cov_xz = cd_cov_xz + (X[i] - cd_mean_X)*(Z[i] - cd_mean_Z);

              cd_cov_yy = cd_cov_yy + (Y[i] - cd_mean_Y)*(Y[i] - cd_mean_Y);
              cd_cov_yz = cd_cov_yz + (Y[i] - cd_mean_Y)*(Z[i] - cd_mean_Z);

              cd_cov_zz = cd_cov_zz + (Z[i] - cd_mean_Z)*(Z[i] - cd_mean_Z);
              }
       }

/******* pool covariance maticies ***********/


pcov_xx = (g_cov_xx + cd_cov_xx)/total_df;
pcov_xy = (g_cov_xy + cd_cov_xy)/total_df;
pcov_xz = (g_cov_xz + cd_cov_xz)/total_df;

pcov_yx = pcov_xy;
pcov_yy = (g_cov_yy + cd_cov_yy)/total_df;
pcov_yz = (g_cov_yz + cd_cov_yz)/total_df;
```

```
pcov_zx = pcov_xz;
pcov_zy = pcov_yz;
pcov_zz = (g_cov_zz + cd_cov_zz)/total_df;

/*
printf("\n\npooled within-class covariance matrix\n");
printf("%f  %f  %f",pcov_xx, pcov_xy, pcov_xz);
printf("\n%f  %f  %f",pcov_yx, pcov_yy, pcov_yz);
printf("\n%f  %f  %f\n\n",pcov_zx, pcov_zy, pcov_zz);
*/

/*********** invert covariance matrix ***************/

mag = pcov_xx*pcov_yy*pcov_zz + pcov_xy*pcov_yz*pcov_zx;
mag = mag + pcov_yx*pcov_zy*pcov_xz - pcov_zx*pcov_yy*pcov_xz;
mag = mag - pcov_yx*pcov_xy*pcov_zz - pcov_zy*pcov_yz*pcov_xx;

inv_pcov_xx = (pcov_yy*pcov_zz - pcov_zy*pcov_yz)/mag;
inv_pcov_xy = (pcov_zy*pcov_xz - pcov_xy*pcov_zz)/mag;
inv_pcov_xz = (pcov_xy*pcov_yz - pcov_yy*pcov_xz)/mag;

inv_pcov_yx = inv_pcov_xy;
inv_pcov_yy = (pcov_xx*pcov_zz - pcov_zx*pcov_xz)/mag;
inv_pcov_yz = (pcov_yx*pcov_xz - pcov_xx*pcov_yz)/mag;

inv_pcov_zx = inv_pcov_xz;
inv_pcov_zy = inv_pcov_yz;
inv_pcov_zz = (pcov_xx*pcov_yy - pcov_yx*pcov_xy)/mag;
/*
printf("\n\ninverted pooled within-class covariance matrix\n");
printf("%f  %f  %f",inv_pcov_xx, inv_pcov_xy, inv_pcov_xz);
printf("\n%f  %f  %f",inv_pcov_yx, inv_pcov_yy, inv_pcov_yz);
printf("\n%f  %f  %f\n\n",inv_pcov_zx, inv_pcov_zy, inv_pcov_zz);
*/

/*************** compute linear discriminant coeficients: Cf = invS*X *********/
/******** and compute linear discriminant function consant C = -0.5*X'*invS*X  **********/
```

```
Cgx = g_mean_X*inv_pcov_xx +  g_mean_Y*inv_pcov_xy +  g_mean_Z*inv_pcov_xz;
Cgy = g_mean_X*inv_pcov_yx +  g_mean_Y*inv_pcov_yy +  g_mean_Z*inv_pcov_yz;
Cgz = g_mean_X*inv_pcov_zx +  g_mean_Y*inv_pcov_zy +  g_mean_Z*inv_pcov_zz;

Cg = -0.5*(Cgx*g_mean_X + Cgy*g_mean_Y + Cgz*g_mean_Z);

Ccdx = cd_mean_X*inv_pcov_xx +  cd_mean_Y*inv_pcov_xy +  cd_mean_Z*inv_pcov_xz;
Ccdy = cd_mean_X*inv_pcov_yx +  cd_mean_Y*inv_pcov_yy +  cd_mean_Z*inv_pcov_yz;
Ccdz = cd_mean_X*inv_pcov_zx +  cd_mean_Y*inv_pcov_zy +  cd_mean_Z*inv_pcov_zz;

Ccd = -0.5*(Ccdx*cd_mean_X + Ccdy*cd_mean_Y + Ccdz*cd_mean_Z);



/*************** classify each nut, count errors, keep track of two best models ******************/

error = 0;
for(i=2; i<=filecount; i=i+2)
        {
        Dg = Cg + Cgx*X[i] + Cgy*Y[i] + Cgz*Z[i];
        Dcd = Ccd + Ccdx*X[i] + Ccdy*Y[i] + Ccdz*Z[i];

        class_score = 1;
        if(Dcd > Dg) class_score = 2;
        error = error + abs(damage[i] - class_score);
        }

if(error < minerr)
        {
        min2err = minerr;
                min2_X1 = min_X1;
                min2_X2 = min_X2;
                min2_Y1 = min_Y1;
                min2_Y2 = min_Y2;
                min2_Z1 = min_Z1;
                min2_Z2 = min_Z2;
        minerr = error;
                min_X1 = X1;
                min_X2 = X2;
                min_Y1 = Y1;
                min_Y2 = Y2;
                min_Z1 = Z1;
                min_Z2 = Z2;

        fprintf(fout,"minerr = %d  %d:%d  %d:%d
%d:%d\n",minerr,min_X1,min_X2,min_Y1,min_Y2,min_Z1, min_Z2);

        fprintf(fout,"min2err = %d  %d:%d  %d:%d
%d:%d\n",min2err,min2_X1,min2_X2,min2_Y1,min2_Y2,min2_Z1, min2_Z2);
        }

} /***end of Z2 variable loop *******/
} /***end of Z1 variable loop *******/
/** printf("Y1 = %d  Y2 = %d\n",Y1, Y2);  */

} /***end of Y2 variable loop *******/
} /***end of Y1 variable loop *******/


printf("minerr = %d  %d:%d  %d:%d  %d:%d\n",minerr,min_X1,min_X2,min_Y1,min_Y2,min_Z1,
min_Z2);
printf("min2err = %d  %d:%d  %d:%d
%d:%d\n",min2err,min2_X1,min2_X2,min2_Y1,min2_Y2,min2_Z1, min2_Z2);


} /***end of X2 variable loop *******/
} /***end of X1 variable loop *******/



fclose(fin);
fclose(fout);
} /** end of main **/
```

APPENDIX G

```
/********** Program for Micro Linear ML2035 sine wave gernators *********/

/*  SINEPRG.C  Tom Pearson   July 1, 1997    */
/* programs Micro Linear ML2035 with 16 bit number in
serial fashion.  Controls pins SCK SID and LATI
with Kiethly Metrobyte PIO-24 digital output card*/

#include <stdio.h>
#include <dos.h>    /* for outportb()  */


void
main(void)
{
int quit,i,j;
unsigned int pause;

outportb(0x303, 0);   /* sets ports to output only mode */

/**
program the ML2035: three bits are required.  In the following lines
the MSB is used to clock the serial port data (SCK).  Data is latched on
falling  edges of SCK.  The second MSB is used for data (SID).  It stays
at a low (0) or high (1) value for two SCK cycles in order to to be at the
proper state during a falling edge of SCK.  16 bit ata is fed serially to
the ML2035, data LSB first to data MSB last.  After sixteen bits of data
are fed, the LSB (of the three) is changed from low to high then to low
to latch the 16 bits of data fed to the ML2035.  This corresponds to LATI
on the Ml2035.  SCK is kept low during this cycle.  After data is transfered,
the ML2035 latches onto it and no further data transmissions are needed.

**/
/*********** program 9 kHz through ports PA 0-2 ***/
```

```
outport(0x300, 0);
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0); /** start data input first bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);  /** second bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);  /** third bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);  /** fourth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);  /** fith bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);  /** sixth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);   /** seventh bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);   /** eigth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);   /** nineth bit **/
```

```
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);   /** tenth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);   /** eleventh bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 0);   /** twelveth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 2);   /** thirteenthth bit **/
j = i * 3;
outport(0x300, 6);
j = i * 3;
outport(0x300, 0);   /** fourteenth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 2);   /** fifteenth bit **/
j = i * 3;
outport(0x300, 6);
j = i * 3;
outport(0x300, 0);   /** sixteenth bit **/
j = i * 3;
outport(0x300, 4);
j = i * 3;
outport(0x300, 1);  /** latch data **/
j = i * 3;
outport(0x300, 1);
j = i * 3;
outport(0x300, 0);
j = i * 3;
```

```
/*********** program 21 kHz through ports PC 0-2 ***/


outport(0x302, 0);
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2); /** start data input first bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 2);  /** second bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 0);  /** third bit **/
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2);  /** fourth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 0);  /** fith bit **/
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2);  /** sixth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 0);   /** seventh bit **/
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2);   /** eigth bit **/
j = i * 3;
```

```
outport(0x302, 6);
j = i * 3;
outport(0x302, 0);   /** nineth bit **/
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2);   /** tenth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 0);   /** eleventh bit **/
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2);   /** twelveth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 2);   /** thirteenthth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 2);   /** fourteenth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 0);   /** fifteenth bit **/
j = i * 3;
outport(0x302, 4);
j = i * 3;
outport(0x302, 2);   /** sixteenth bit **/
j = i * 3;
outport(0x302, 6);
j = i * 3;
outport(0x302, 1);  /** latch data **/
j = i * 3;
outport(0x302, 1);
```

```
j = i * 3;
outport(0x302, 0);
j = i * 3;

/*********** program 15 kHz through ports PB 0-2 ***/


outport(0x301, 0);
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2); /** start data input first bit **/
j = i * 3;
outport(0x301, 6);
j = i * 3;
outport(0x301, 0);  /** second bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2);  /** third bit **/
j = i * 3;
outport(0x301, 6);
j = i * 3;
outport(0x301, 0);  /** fourth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2);  /** fith bit **/
j = i * 3;
outport(0x301, 6);
j = i * 3;
outport(0x301, 0);  /** sixth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2);   /** seventh bit **/
j = i * 3;
```

```
outport(0x301, 6);
j = i * 3;
outport(0x301, 0);   /** eigth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2);   /** nineth bit **/
j = i * 3;
outport(0x301, 6);
j = i * 3;
outport(0x301, 0);   /** tenth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2);   /** eleventh bit **/
j = i * 3;
outport(0x301, 6);
j = i * 3;
outport(0x301, 0);   /** twelveth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 0);   /** thirteenthth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 0);   /** fourteenth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 0);   /** fifteenth bit **/
j = i * 3;
outport(0x301, 4);
j = i * 3;
outport(0x301, 2);   /** sixteenth bit **/
j = i * 3;
outport(0x301, 6);
```

```
j = i * 3;
outport(0x301, 1);  /** latch data **/
j = i * 3;
outport(0x301, 1);
j = i * 3;
outport(0x301, 0);
j = i * 3;

/********** program 12 kHz through ports PA 3-5 ***/


outport(0x300, 0);
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16); /** start data input first bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 16);  /** second bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 0);  /** third bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16);  /** fourth bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 0);  /** fith bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16);  /** sixth bit **/
j = i * 3;
```

```
outport(0x300, 48);
j = i * 3;
outport(0x300, 0);   /** seventh bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16);   /** eigth bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 0);   /** nineth bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16);   /** tenth bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 0);   /** eleventh bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16);   /** twelveth bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 0);   /** thirteenthth bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 16);   /** fourteenth bit **/
j = i * 3;
outport(0x300, 48);
j = i * 3;
outport(0x300, 16);   /** fifteenth bit **/
j = i * 3;
outport(0x300, 48);
```

```
j = i * 3;
outport(0x300, 0);   /** sixteenth bit **/
j = i * 3;
outport(0x300, 32);
j = i * 3;
outport(0x300, 8);  /** latch data **/
j = i * 3;
outport(0x300, 8);
j = i * 3;
outport(0x300, 0);
j = i * 3;


/*********** program 18 kHz through ports PB 3-5 ***/


outport(0x301, 0);
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0); /** start data input first bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);  /** second bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);  /** third bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);  /** fourth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);  /** fith bit **/
```

```
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);  /** sixth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** seventh bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** eigth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** nineth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** tenth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** eleventh bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** twelveth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 0);   /** thirteenthth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 16);   /** fourteenth bit **/
j = i * 3;
```

```
outport(0x301, 48);
j = i * 3;
outport(0x301, 0);   /** fifteenth bit **/
j = i * 3;
outport(0x301, 32);
j = i * 3;
outport(0x301, 16);   /** sixteenth bit **/
j = i * 3;
outport(0x301, 48);
j = i * 3;
outport(0x301, 8);  /** latch data **/
j = i * 3;
outport(0x301, 8);
j = i * 3;
outport(0x301, 0);
j = i * 3;

/*********** program 24 kHz through ports PC 3-5 ***/


outport(0x302, 0);
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16); /** start data input first bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);  /** second bit **/
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);  /** third bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);  /** fourth bit **/
```

```
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);  /** fith bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);  /** sixth bit **/
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);   /** seventh bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);   /** eigth bit **/
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);   /** nineth bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);   /** tenth bit **/
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);   /** eleventh bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);   /** twelveth bit **/
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);   /** thirteenthth bit **/
j = i * 3;
```

```
outport(0x302, 48);
j = i * 3;
outport(0x302, 0);   /** fourteenth bit **/
j = i * 3;
outport(0x302, 32);
j = i * 3;
outport(0x302, 16);   /** fifteenth bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 16);   /** sixteenth bit **/
j = i * 3;
outport(0x302, 48);
j = i * 3;
outport(0x302, 8);  /** latch data **/
j = i * 3;
outport(0x302, 8);
j = i * 3;
outport(0x302, 0);
j = i * 3;

} /** end of main **/
```

APPENDIX H

```
/*********** PC program to interface with DSP in real time ******************/

/* FTPC.CPP     Tom Pearson     8-18-97
   this program sends hanning window and discrete Fourier transorm
   parameters to DSP to perform a 300 point DFT at 9, 12, 15, 18, 21,
   and 24 KHz.  The DSP performs the computations between sampling
   points. This program sends a flag to DSP to start sampling, data is
   stored once 300 points are sampled and transformed, DSP sends
   another flag to notify PC that 12 DFT have been obtained and stored.
   PC collects data and flags DSP to start sampling again once another
   nut is in hte field of view.  Use with DSP program FTDSP.ASM

*/

#include <stdio.h>
#include <conio.h>   /* for outpw and inpw */
#include <dos.h>     /* for outportb and inportb  and outport and inport*/
#include <math.h>

void main(){

        unsigned int low16, high16, lowbyte, dummy, quit;
        long int real, imag;
        long int data, ldata;
        int i, count, saturate;
        double mag[7], data, pi, i_data[7], r_data[7];
        double sum[7], data1[400];

        FILE *dataf;
        char datafile[12];

        printf("\n  filename to write data to: ");
        scanf("%s",datafile);
        dataf = fopen(datafile, "w");
```

```
outportb(0x316,0x0);   /* sets 64k pg address to location 0 hex on 310 board */

/* send Hanning window & FT multipliers to DSP memory address 400H*/
        pi = 3.14159265;
        outport(0x312,0x400);  /* set address location on DSP */

        for(i=0;i<=299;i++)
                {

                /* 9 K Hz multipliers */
                real = 65535*(cos(pi/2.0 - i*pi/299.0))*cos(0.31415927*i)*1;
                imag = 65535*(cos(pi/2.0 - i*pi/299.0))*sin(0.31415927*i)*1;
                outpw(0x310, real);    /* write 16 LSB to DSP */
                real = real >> 16;
                outpw(0x310, real);    /* write 16 MSB  (sign) to DSP */
                outpw(0x310, imag);    /* write 16 LSB to DSP */
                imag = imag >> 16;
                outpw(0x310, imag);    /* write 16 MSB  (sign) to DSP */

                /* 12 K Hz multipliers */
                real = 65535*(cos(pi/2.0 - i*pi/299.0))*cos(0.41887902*i)*1;
                imag = 65535*(cos(pi/2.0 - i*pi/299.0))*sin(0.41887902*i)*1;
                outpw(0x310, real);    /* write 16 LSB to DSP */
                real = real >> 16;
                outpw(0x310, real);    /* write 16 MSB  (sign) to DSP */
                outpw(0x310, imag);    /* write 16 LSB to DSP */
                imag = imag >> 16;
                outpw(0x310, imag);    /* write 16 MSB  (sign) to DSP */

                /* 15 K Hz multipliers */
                real = 65535*(cos(pi/2.0 - i*pi/299.0))*cos(0.52359878*i)*1;
                imag = 65535*(cos(pi/2.0 - i*pi/299.0))*sin(0.52359878*i)*1;
                outpw(0x310, real);    /* write 16 LSB to DSP */
                real = real >> 16;
                outpw(0x310, real);    /* write 16 MSB  (sign) to DSP */
                outpw(0x310, imag);    /* write 16 LSB to DSP */
```

```
            imag = imag >> 16;
            outpw(0x310, imag);    /* write 16 MSB  (sign) to DSP */


            /* 18 K Hz multipliers */
            real = 65535*(cos(pi/2.0 - i*pi/299.0))*cos(0.62831853*i)*1;
            imag = 65535*(cos(pi/2.0 - i*pi/299.0))*sin(0.62831853*i)*1;
            outpw(0x310, real);    /* write 16 LSB to DSP */
            real = real >> 16;
            outpw(0x310, real);    /* write 16 MSB  (sign) to DSP */
            outpw(0x310, imag);    /* write 16 LSB to DSP */
            imag = imag >> 16;
            outpw(0x310, imag);    /* write 16 MSB  (sign) to DSP */

            /* 21 K Hz multipliers */
            real = 65535*(cos(pi/2.0 - i*pi/299.0))*cos(0.73303829*i)*1;
            imag = 65535*(cos(pi/2.0 - i*pi/299.0))*sin(0.73303829*i)*1;
            outpw(0x310, real);    /* write 16 LSB to DSP */
            real = real >> 16;
            outpw(0x310, real);    /* write 16 MSB  (sign) to DSP */
            outpw(0x310, imag);    /* write 16 LSB to DSP */
            imag = imag >> 16;
            outpw(0x310, imag);    /* write 16 MSB  (sign) to DSP */

            /* 24 K Hz multipliers */
            real = 65535*(cos(pi/2.0 - i*pi/299.0))*cos(0.83775804*i)*1;
            imag = 65535*(cos(pi/2.0 - i*pi/299.0))*sin(0.83775804*i)*1;
            outpw(0x310, real);    /* write 16 LSB to DSP */
            real = real >> 16;
            outpw(0x310, real);    /* write 16 MSB  (sign) to DSP */
            outpw(0x310, imag);    /* write 16 LSB to DSP */
            imag = imag >> 16;
            outpw(0x310, imag);    /* write 16 MSB  (sign) to DSP */
            }


/***************  base adress 310 DATA ACQUISITION ******************/
```

```
/* set up infinite loop that only is broken with keyboard entry */


printf("\n type q to quit\n ");
printf("\n9 KHz\t12 KHz\t15 KHz\t18 KHz\t21 KHz\t24 KHz\n");
quit = 0;
count = 0;
while(quit == 0)
{
        saturate = 0;
        count++;
        /** send value 1 to DSP to tell it to start collecting data **/

        outport(0x312,0x809800);  /* set address location on DSP */
        outpw(0x310, 0x1);   /* write number 1 to DSP to start sampling */
        outpw(0x310, 0x0);

        /** wait for DSP to get 300 samples, a value of zero will
                appear at DSP memory location 0x809800 after
                300 samples are aquired **/

        outport(0x312,0x809800);  /* set address location on DSP */
        lowbyte = inpw(0x310); /* read 2 bytes from 310 memory */
        dummy = inpw(0x310);  /*DSP increments one address space */

        while ( lowbyte == 1)
                {
                outport(0x312,0x809800);  /* set address location on DSP */
                lowbyte = inpw(0x310); /* read 2 bytes from 310 memory */
                dummy = inpw(0x310);  /*DSP increments one address space */

                /** allow escape from program here **/
                if(kbhit())
                        {
                        if(getch() == 'q')
                                {
                                quit = 1;
                                printf("\n\n****** program terminated ******");
                                }
                        }
                }

        /* read 12 DFT's from DSP memory */
        outport(0x312,0x110);  /* set address location on DSP */
        for(j=1;j<=12;j++)
                {
                for(i=1;i<=6;i++)
                        {
                        low16 = inpw(0x310); /* read 16 LSB from DSP memory */
                        high16 = inpw(0x310);  /*read 16 MSB from DSP memory  */
                        Rdata = high16;
                        Rdata = Rdata << 16;    /* shift 16 MSB to proper place */
                        r_data[j][i] = (Rdata + low16)/100000.0;  /* add in 16 LSB */

                        low16 = inpw(0x310); /* read 16 LSB from DSP memory */
                        high16 = inpw(0x310);  /*read 16 MSB from DSP memory  */
                        Idata = high16;
                        Idata = Idata << 16;    /* shift 16 MSB to proper place */
                        i_data[j][i] = (Idata + low16)/100000.0;  /* add in 16 LSB */
                        }
                }

                for(i=1;i<=6;i++)
                        {
                        sum[i] = 0.0;
                        }
        for(j=1;j<=12;j++)
                {
                for(i=1;i<=6;i++)
                        {
                        mag[j][i] = sqrt(r_data[j][i]*r_data[j][i] + i_data[j][i]*i_data[j][i]);
                        sum[i] = sum[i] + mag[j][i];
                        If (mag[j][i] < 500) then
                                {
```

```
                printf( “\nphotodiode saturated before 12 DFT’s aquired\n”);
                saturate = 1;
            }

        /***** write data to hard drive and monitor ****/
        if(saturate == 0)
        {
        printf("%d\t %4.0f\t%4.0f\t%4.0f\t%4.0f\t%4.0f\t%4.0f\n",count,
        sum[1],sum[2],sum[3],sum[4],sum[5],sum[6]);
        fprintf(dataf"%d\t%4.0f\t%4.0f\t%4.0f\t%4.0f\t%4.0f\t%4.0f\n",count,sum[1],sum[2]
        ,sum[3],sum[4],sum[5],sum[6]);
        }

}  /** end of keyboard interupt while loop **/

fclose(dataf);

} /** end of main **/
```

APPENDIX I

```
************************* TMS320 ASSEMBLY PROGRAM FOR REAL TIME DFT **************

; FTDSP.ASM          Tom Pearson           8-8-97

; Uses 180KHz external clock (connected to lower pin of J40, J42 in the B
; position) as the the ConvertStart signal.

; Uses the onboard 5 mhz oscillator as the serial port clock signal
; Uses xf1 to turn oscillator ON/OFF. The oscillator can be turned off
; to reduce noise when the serial port is not used.
;
; samples 300 data points and performs hanning window and discrete
; Fourier Transform at 9, 12, 15, 18, 21, 24 KHz.  Outputs magnetude
; of the transform at these frequencies.  Detects precence of nut by
; decline in average of 50 samples.  Computes 12 DFT's for every nut.
; Flags PC when 12 DFT's are obtained and stored.
; Use this program in conjunction with the C program FTPC.CPP

        .text
RESET   .word   StartPt


;
IOF_SET_XF1     .word   62H
IOF_RESET_XF1   .word   22H
CTRL            .word   808000H ; Base addr. of onchip peripherals
TIMGB0CONHI     .word   6H      ; Word to make tclk0 HI
TIMGB0CONLO     .word   2H      ; Word to make tclk0 LO
SERGLOBA        .word   150144H ; Ser. Pt. Global Control Reg.
SERGLOB0        .word   0C150144H ; Ser. Pt. Global Control Reg.
SERPRTX0        .word   111H    ; Ser. Pt. transmit pin particulars
SERPRTR0        .word   111H    ; Ser. Pt. receive pin particulars
SERTIM0         .word   03CFh   ; Ser. Pt. timer control
SERTIM0VAL      .word   01h     ; Ser. Pt. timer period
DATA_TO_HOST    .word   0110H   ; Memory loc.
```

```
MAX_MEM        .word   1A0H    ; maximum memory location  for 12 DFT's
LATCH_VAL      .word   0H     ; Value to write to Model 310 Latch
                                ; sets gain and A/D mux channel
LATCH_AREA     .word   0FFFFFFH ; Address of Latch in TMS320 memory
HANNING        .word   400H ; memory address of Hanning window
FT_PARAMS      .word   400H   ; memory location of first FT parameters
FT_STORE       .word   809C00H ; location of 24 KHZ complex FT data
COMP           .word   0FFF00000H      ; used to get 12 MSB
NUT            .word   0400H       ; used to determine presence of nut
        .text
StartPt:
;
;
        LDP     0h                  ; Set Data Page
        LDI     1800H,ST              ; Initialize status reg.
        LDI     0985H,SP              ; Initialize stack ptr

        LDI     @IOF_SET_XF1,R1         ; Turn oscillator on
        LDI     R1,IOF                ; to enable serial port
                                        ; transmissions

        LDI     @CTRL,AR0


        LDI     @LATCH_AREA,AR3         ; Set the A/D Channel
        LDI     @LATCH_VAL,R0          ; and Gain=8 channel = 0
        STI     R0,*+AR3(0)


        LDI     @SERGLOBA,R0    ; Program the Serial
        STI     R0,*+AR0(64)    ; Port and its
        LDI     @SERTIM0VAL,R0  ; Timer
        STI     R0,*+AR0(70)
        LDI     @SERTIM0 ,R0
        STI     R0,*+AR0(68)
        LDI     @SERPRTX0,R0
        STI     R0,*+AR0(66)
```

```
          LDI    @SERPRTR0,R0
          STI    R0,*+AR0(67)
          LDI    @SERGLOB0,R0
          STI    R0,*+AR0(64)

          LDI    @TIMGB0CONHI,R0 ; ConvertStart HI
          STI    R0,*+AR0(20H)

          LDI    18H,R0         ; Set Wait States in TMS320 to ZERO
          STI    R0,*+AR0(64H)   ; (See Chap. 7 in TMS320C3x Guide)

          LDI    11H, R3        ; store a non-zero at address 809800
          STI    R3, @809800H    ; to prevent DSP from sampling
                                 ; until PC tells it to
          LDI    @COMP,AR7       ; used to get 12 MSB

initial:
          LDI    0,R4           ; load 0 into maximum sample register
                                 ; for checking presence of nut
          LDI    @809800H, R1      ; read poll value from 809800h
          CMPI   0,R1           ; compares 0 to value at 809800h
          BNE    wait_for_nut    ; get sample in not equal to zero
          BR     initial         ; keep waiting if equal to zero

  ; sample 50 points to and check for presence of nut
wait_for_nut:                ; Has data been received ?
          LDI    *+AR0(64),R2    ; Check status
          AND    01H,R2
          BZ     wait_for_nut     ; Branch if not received

          LDI    *+AR0(76),R3    ; Read value from serial port
          AND    AR7,R3   ; get 12 MSB and shift into lower two bytes
          LSH    -20,R3         ; of the 32 bit memory word
          ADDI   01H,R1         ; add to sample index
          CMPI   R4,R3          ; compare new sample with maximum
          BGT    new_maximum
          CMPI   32H,R1         ; see if 50 samples has been taken
```

```
         BGT    check          ; if 50 sample received, check for nut
         BR     wait_for_nut   ; otherwise go get another sample
new_maximum:
         LDI    R3,R4          ; store new maximum value
         BR     wait_for_nut

check:
  ; check if nut is present
                 STI    R4,@109H     ;TEST  TEST
         LDI    R4,R3         ; reset maximum register
         LDI    0,R4
         CMPI   @NUT,R3        ; check if nut was present
         BGT    wait_for_nut   ; go back if no nut present

  ; reset data to host memory adress if no nut present
         LDI    0110H,R3
         STI    R3, @DATA_TO_HOST    ;reset location of memory storage

mem_reset:
         LDI @HANNING,AR2       ; set location of Hanning window
         LDI 0H,R0       ; 9 KHz real data
         LDI 0H,R1       ; 9 KHz imaginary data
         LDI 0H,R4       ; 12 KHz real data
         LDI 0H,R5       ; 12 KHz imaginary data
         LDI 0H,R6       ; 15 KHz real data
         LDI 0H,R7       ; 15 KHz imaginary data
         LDI 0H,AR1       ; 18 KHz real data
         LDI 0H,AR3       ; 18 KHz imaginary data
         LDI 0H,AR4       ; 21 KHz real data
         LDI 0H,AR5       ; 21 KHz imaginary data
         LDI 0H,AR6       ; 24 KHz real data
         LDI 0H,R3        ; 24 KHz imaginary data
         STI R3,@FT_STORE    ; 24 KHz imaginary data  (on chip memory)

get_sample:                 ; Has data been received ?
         LDI    *+AR0(64),R2    ; Check status
         AND    01H,R2
```

```
        BZ      get_sample      ; Branch if not received

        LDI     *+AR0(76),R3    ; Read value from serial port
        AND     AR7,R3   ; get 12 MSB and shift into lower two bytes
        LSH     -20,R3          ; of the 32 bit memory word


; hanning window and Fourier Transform
; 9 KHz

        MPYI     *+AR2(0),R3,R2    ; multiply hanning FT window  REAL
        ADDI    R2,R0          ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address
        MPYI     *+AR2(0),R3,R2    ; multiply hanning FT window  IMAGINARY
        ADDI    R2,R1          ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address

; 12 KHz

        MPYI     *+AR2(0),R3,R2    ; multiply hanning FT window  REAL
        ADDI    R2,R4          ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address
        MPYI     *+AR2(0),R3,R2    ; multiply hanning FT window  IMAGINARY
        ADDI    R2,R5          ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address

; 15 KHz

        MPYI     *+AR2(0),R3,R2    ; multiply hanning FT window  REAL
        ADDI    R2,R6          ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address
        MPYI     *+AR2(0),R3,R2    ; multiply hanning FT window  IMAGINARY
        ADDI    R2,R7          ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address

; 18 KHz
```

```
        MPYI    *+AR2(0),R3,R2   ; multiply hanning FT window  REAL
        ADDI    R2,AR1           ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address
        MPYI    *+AR2(0),R3,R2   ; multiply hanning FT window  IMAGINARY
        ADDI    R2,AR3           ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address

 ; 21 KHz

        MPYI    *+AR2(0),R3,R2   ; multiply hanning FT window  REAL
        ADDI    R2,AR4           ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address
        MPYI    *+AR2(0),R3,R2   ; multiply hanning FT window  IMAGINARY
        ADDI    R2,AR5           ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address

 ; 24 KHz

        MPYI    *+AR2(0),R3,R2   ; multiply hanning FT window  REAL
        ADDI    R2,AR6           ; add to Fourier transform total
        ADDI    01H,AR2          ; increment hanning window address
        MPYI    *+AR2(0),R3,R2   ; multiply hanning FT window  IMAGINARY
        LDI     @FT_STORE,R3     ; get data from on chip memory
        ADDI    R2,R3            ; add to Fourier transform total
        STI     R3,@FT_STORE     ; send this to on chip memory
        ADDI    01H,AR2          ; increment hanning window address


; check if 300 data points have been sampled
        LDI     AR2, R3          ; allow 300 sample to be taken
        SUBI    1210H,R3         ; 1210h - 400h = 3600dec
        BNZ     get_sample       ; go get next sample

  ; send data to DSP memory so PC can access
        LDI @DATA_TO_HOST,AR2    ; DSP memory location for PC to read data
        ; 9 KHz
        STI     R0,*+AR2(0)      ; Send to memory for viewing
```

```
          ADDI   01H,AR2         ; increment memory address
          STI    R1,*+AR2(0)     ; Send to memory for viewing
          ; 12 KHz
          ADDI   01H,AR2         ; increment memory address
          STI    R4,*+AR2(0)     ; Send to memory for viewing
          ADDI   01H,AR2         ; increment memory address
          STI    R5,*+AR2(0)     ; Send to memory for viewing
          ; 15 Khz
          ADDI   01H,AR2         ; increment memory address
          STI    R6,*+AR2(0)     ; Send to memory for viewing
          ADDI   01H,AR2         ; increment memory address
          STI    R7,*+AR2(0)     ; Send to memory for viewing
          ; 18 KHz
          ADDI   01H,AR2         ; increment memory address
          STI    AR1,*+AR2(0)     ; Send to memory for viewing
          ADDI   01H,AR2         ; increment memory address
          STI    AR3,*+AR2(0)     ; Send to memory for viewing
          ; 21 KHz
          ADDI   01H,AR2         ; increment memory address
          STI    AR4,*+AR2(0)     ; Send to memory for viewing
          ADDI   01H,AR2         ; increment memory address
          STI    AR5,*+AR2(0)     ; Send to memory for viewing
          ; 24 KHz
          ADDI   01H,AR2         ; increment memory address
          STI    AR6,*+AR2(0)     ; Send to memory for viewing
          ADDI   01H,AR2         ; increment memory address
          LDI    @FT_STORE,R3    ; get data from on chip memory
          STI    R3,*+AR2(0)     ; Send to memory for viewing

          ADDI   01H,AR2         ; increment memory address
          STI    AR2, @DATA_TO_HOST  ; store memory location

          CMPI   @MAX_MEM,AR2    ; see if 12 DFT's have been acquired
          BGT    send_data
          BR     mem_reset       ; sample another set if nut was present

      send_data:
```

```
LDI     0H,R3           ; send 0h to memory address 809800h
STI     R3, @809800H    ; to tell PC that data has been collected
BR      initial         ; Go wait for PC command to sample again

            .end
```