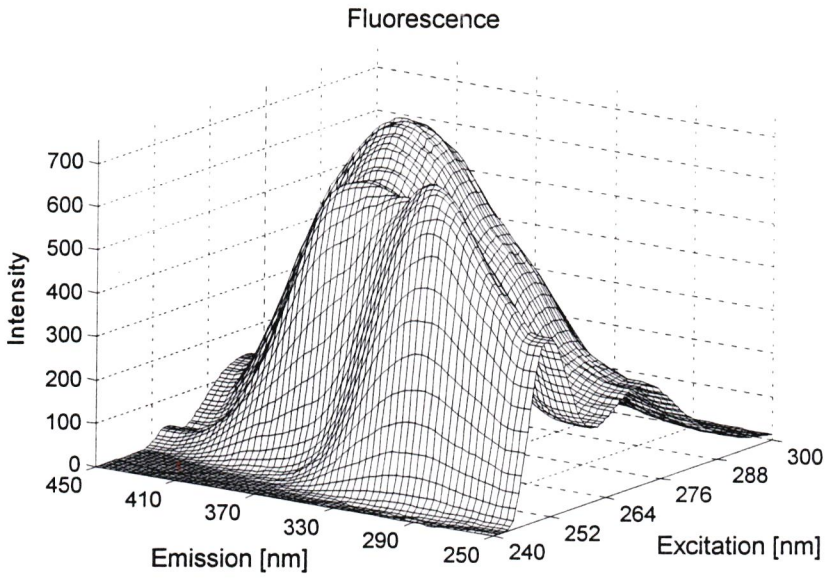
Flow Injection Analysis, Fluorometry and Chemometrics



Final thesis
Autumn 1995

Claus A. Andersson
C888031

Department of Biotechnology,
Technical University of Denmark

Food Technology, Department of Dairy and Food Science,
Royal Veterinary and Agricultural University

Flow Injection Analysis, Fluorometry and Chemometrics

Final thesis
Autumn 1995

Claus A. Andersson

Tutors:

Ass. professor, Ph.D Jens Frisvad
Department of Biotechnology,
Technical University of Denmark

Professor, dr. phil. Lars Munck
Food Technology, Department of Dairy and Food Science,
Royal Veterinary and Agricultural University

Abstract

Methodologies for the development of general *n*-way algorithms for PARAFAC and TUCKER models and *n*-way partial least squares (PLS) calibration are presented. The methods have been implemented in the Matlab programming language and the source code is listed. A new procedure for rotating TUCKER estimates to obtain a desired core structure is proposed. The findings of a literature search on the process of sugar production and the connected colour forming reactions are reported. From the literature search six chemical components are chosen to make a model system of thick juice. The model system is examined by spectrofluorometric analysis using a diode-array in conjunction with fibre optics. Excitation-emission matrices with 28 excitation wavelengths and 64 emission wavelengths have been measured for 18 samples at 3 different pH levels. The resulting data is an array with a 4-way structure having dimensions (18×3×28×64). The data are used for validating the developed Matlab library being simultaneously subject to analysis. The *n*-way PARAFAC, TUCKER and PLS algorithms are applied for calibration, furthermore, the PARAFAC and TUCKER models are used for a brief explorative approach. Titration curves from titration of thick juice samples are investigated for prediction of quality parameters of sugar.

Foreword and acknowledgements

The time spent with the instruments and on the development of new chemometric algorithms has convinced me that the Chemometrics Group at Food Technology, KVL was the right place to write the thesis. The work with instruments has imparted hands-on experience with front-end technology. The chemometrics side of the project has mainly dealt with developing the necessary mathematical basis on which true *n*-way algorithms could be based. As a whole, the project has given me lots of experience in both fields. However, this would not have been the case if not for the people at Food Technology. I owe Carsten Ridder thanks for giving me the first introduction to PCA.

I must thank professor Lars Munck for giving, sometimes overwhelmingly, many ideas for experiments that could turn out to give interesting and useful results. Lars Nørgaard was kind to keep my focus on real-world applications, not just tampering new algorithms. Rasmus Bro has given many enlightening and very yielding comments and was always ready to answer any questions. Hopefully, we will continue our competition to make the most general, the smartest and, most of all, fastest algorithms. The Chemometrics Group at KVL is a very special place were the ideas are nourished by a certain straight-forward approach.

Contents

1. Introduction

The primary aim of this project, 'Flow injection analysis, fluorometry and chemometrics', has been to collect knowledge and experience on fast fluorometric methods for screening analysis. Another important aim has been to develop and evaluate chemometric algorithms and methods for analysis of such data. This report is a continuation of a preliminary thesis, Andersson (1995), and the two reports should be regarded as a whole.

The experiences will be used for finding appropriate equipment and techniques for extracting relevant information from *intermediate* products from the production of sugar. The information from the screening methods will be used in predictive models forecasting the quality parameters of the final sugar product. Such predictive models could bring about an understanding of the chemical and physical mechanisms acting upon the intermediary products between the different unit operations. Fulfilling this goal would give insight that could reach into other areas of applied chemical analysis and perhaps even contribute to the basic knowledge of the mechanisms giving rise to formation of colour, i.e. caramelization, polymerization, oxidation and structural transformations. However, by establishing a fast, robust and otherwise feasible method for collecting such high quality information is only the first step. A more detailed chemical understanding will come in time when other analytical instrumental analyses have been applied to the same samples the screening methods are used on. The more elaborate and time consuming analyses currently planned to supplement the fast fluorometric investigations are high performance liquid chromatography (HPLC), thin layer chromatography (TLC), low pressure liquid chromatography (LPLC), supercritical fluid chromatography (SFC), Fourier transformed near infra-red (FT-NIR) and Raman spectroscopy. The chromatographic methods will most likely enable resolution of pure substances from the composite sugar mixtures. Knowing the composition of sugar juice enables qualified interpretation of the information collected from screening measurements. The screening analyses are mainly concentrated on fluorometry due to the arguments above; it is fast, robust and have shown to give information of good quality, Andersson (1995).

Investigating a fluorescent model system will allow assessment of the possibilities for using mathematical resolution of the spectra of the pure substances from composite samples. Composing such a model system requires basic knowledge of what fluorescent species have been found in the thick juices. A study of literature have been conducted to find such fluorescent components. However, in order to keep focus on the present problem the search was narrowed to fluorophoric components. The search results made it possible to make a simple chemical model with thick-juice-like fluorescent behaviour.

A new instrument have been built to acquire the fastest possible screening results. The apparatus makes use of light sources giving high intensity ultra violet (UV) radiation (wavelengths from 200 nm to 400 nm), and visible (VIS) radiation (from 400 nm to 800 nm). These wavelengths are quite ordinary for fluorescence measurements, but the detector is based on a diode array (DA). The use of DA detectors for fluorometry is new, and the instrument designates the start of a new era. An era that, by the use of the robust diode arrays will move fluorometry from the laboratories to the fabrication halls. With this instrumental development many processes involving fluorescent intermediary products are enabled to make use of

fluorometry as a tool for process control, quality control and as an input for predictive models based on chemometrics.

When information has been gathered it must be analysed. For this purpose robust and accurate data analysis algorithms and procedures must be available. In the modern instrumental analysis multi-linear data structures occur more and more often. Examples are excitation-emission matrices from fluorometry, time-absorbency profiles from chromatography, NIR analysis of images and so on. Such multi-way data structures call for multi-way algorithms that can provide models for analysis and calibration. Hence, such *n*-way algorithms have been developed for selected multi-way models for use on data to be collected in near future.

2. Problem description

A description of the aims of the project is appropriate in order to account for the working program.

Instrumental analysis

The new system for fluorometric analysis must be set up and be fine-tuned for spectrofluorometric analysis in the flow injection analysis (FIA) system. This part will include optimization of the components in the system. An adjustment of the software may be necessary to achieve a proper functionality. The instrument should be optimized for use as a spectrofluorometric analysis system in-line with flow injection analysis. After the system has been set up properly, is should be possible to collect multi-way data from analysis on a model system and eventually thick juice. A model system must be comprised from species that have been established as being present in thick juice or other technical sugar juices. This requires a literature search on the subject.

Chemometrics

The chemometric part of the work will require development of the necessary algorithms for analysis and calibration of the measured *n*-way data structures. General *n*-way algorithms for PARAFAC, TUCKER and PLS modelling must be made for future use on *n*-way data structures. A theoretical base for the development of *n*-way algorithms must be established. The true *n*-way algorithms will be applied to analyse the collected data. This will combine the evaluation of the algorithms with the analysis of data.

3. Formation of colour during sugar production

The sugar beet itself contains no coloured components. However, it contains a variety of species capable of *forming* coloured components. In order to obtain the purest possible final sugar product it is desirable to suppress formation of colour during the process. In this chapter it will be discussed where and how colour is formed during the process. This will make it possible to evaluate the predictive quality of the intermediate products with regards to quality parameters of the final sugar.

Some clarification on the term *colour* seems to be appropriate. One of the historical ways to quantify the purity of the sugar is to evaluate the colour by simple visual inspection. It is not possible to define what the colour of white sugar is - since white has no colour. On the other hand the *browning* can easily be quantified as absorption of electromagnetic radiation at different wavelengths. In sugar production as well in the general area of food technology, the colouration or *browning*, being complementary to whiteness, is defined as the absorption at 420 nm, see Macrae et al. (1993). In order to have a simple quantitative expression for the degree of browning the DDS factories use the absorbance at 420 nm of the final sugar product in aqueous solution.

At first the process is described in terms of operational conditions such as temperature, pH and time of exposure to the given operations. After introducing the key component in sugar, *sucrose*, and the precursors for formation of colour, *reducing sugars,* it will be shown that two basically different mechanisms give rise to formation of colour, these are *enzymatic* and *non-enzymatic browning* mechanisms.

3.1 The process of sugar production

In order to acquire knowledge concerning the aspects of colour formation in the sugar juices during the production, it is appropriate to start the discussion by sketching the process. The following summary on the process is supplemented by a detailed discussion by DDS (1985).

The beets are delivered at the factory without top and they are stored outdoor up to 40 hours before processing. The beets are transported from the depot to the washing facility, see fig. 3.1 (A), in floating channels. The washer stirs the beets around in a tank since this makes the beets rub soil and dirt of each other. It is important to keep the beets whole and undamaged to avoid sugar diffusing into the washing water where it will be lost. Also, unwanted oxidative browning is avoided by keeping the sugar beets intact as long as possible. The washed, and more or less dry, beets enter the slicing machine (B) where the beets are cut into rectangular strips at approximately 1 x 1 cm with varying heights. The slicing must give clean edges without mashing the peel since this would cause the peel to release pectin and starch. The strips enter the diffusion tank (C) where the sugar juice is extracted by diffusion with water at 70°C. This temperature has been found to denaturate the cell membranes to such an extent that the sucrose molecules can diffuse out of the cells, whilst the larger molecules

such as polypeptides and pectin are hindered. The efficiency of the diffusive process is increased by slowly moving the strips by spiral transporters while washing with the hot water in counter current flow. After extraction the strips leave the process as waste. The water, now containing the extracted sugar, is called the *diffusion juice*.

During preliming (D) milk of lime, $Ca(OH)_2$, is added to the diffusion juice until pH is approximately 11. Preliming is done at 40 °C and lasts about 20-30 min. The purpose of preliming is to precipitate non-sugars including proteins and pectines as far as the non-sugars form insoluble complexes with the Ca^{2+} ions. Another purpose is to coagulate suspended colloids in the highly polar juice-water suspension. Hereafter lime is again added (E) to increase pH to approximately 12. In heaters (F) the temperature is raised to 85 °C before the juice enters the hot liming tank (G). Hot liming is used to allow precipitation and to stabilise the juice since the vigorous conditions to some extent denaturate amino acids, amides, proteins, enzymes, pectin and bacteriological matter. The hot liming tank is designed to have a retention time of approximately 20 min. Addition of CO_2 in the 1. saturation tank (H) lowers the pH to about 10.8 and makes the surplus of Ca-ions precipitate as $CaCO_3$. The juice is filtered in bag filters (I) where precipitated material is removed. From this stage a stream leads sludge back to the preliming tank (D) since sludge increases the precipitative properties during saturation. The saturation is repeated in the 2. saturation tank (J) where pH is lowered to ensure optimal conditions for precipitation of certain colloids. The precipitate is removed by filtration (K) in which a finer filter than in filtration (I) is used. The second saturation adjusts pH to 9.2 and is mainly used to ensure removal of colloids along with the surplus of Ca^{2+} ions.

Later in the process water has to be removed from the juice and this involves heating. Since heating promotes discolouration through oxidative pathways and cross linkage SO_2 is added (L) to avoid this problem. Water is removed (M) by increasing the temperature stepwise from 80 °C to 130 °C. If insufficient SO_2 have been added discolouration will occur early in this process. The product leaving the evaporators is called *thick juice*. Thick juice is added product streams from stages later in the process which still contain high levels of sugar - the resulting stream is called *standard liquor*. Since the backfeed streams have been heated several times they contain coloured components. The standard liquor is heated under vacuum (N) to decrease the necessary temperature for the removal of water. This massecuite is centrifuged (O) to separate the sugar crystals from the *syrup 1*. The sugar from this (first) boiling is called *sugar 1* - this is the final product. A small stream of syrup 1 is recycled to the inlet of the centrifuge for use as dilution if the standard liquor is too viscid for the centrifuge. Syrup 1 is heated (P) and centrifuged (Q) again to give *sugar 2* which is of lower quality than is sugar 1. The syrup 2 is boiled (R) and centrifuged (S) in order to remove the *molasses* from the sugar still present in the stream. The product from this operation is called *sugar 3 remelt*. Sugar 3 remelt is mixed with a small stream of the sulphurated thin juice and a small amount of crystalline sugar and is returned to the thick juice for use as inoculation for formation of crystals in step (N).

As can be seen from the flowsheet in fig. 3.1, there are many refluxed streams in the process. However, the refluxing ensures a certain averaging giving a more uniform product being independent of fluctuations in the quality of the raw beets and the parameters of the

process. The primary aim of the recycling is of course to obtain the highest possible yield of sugar.

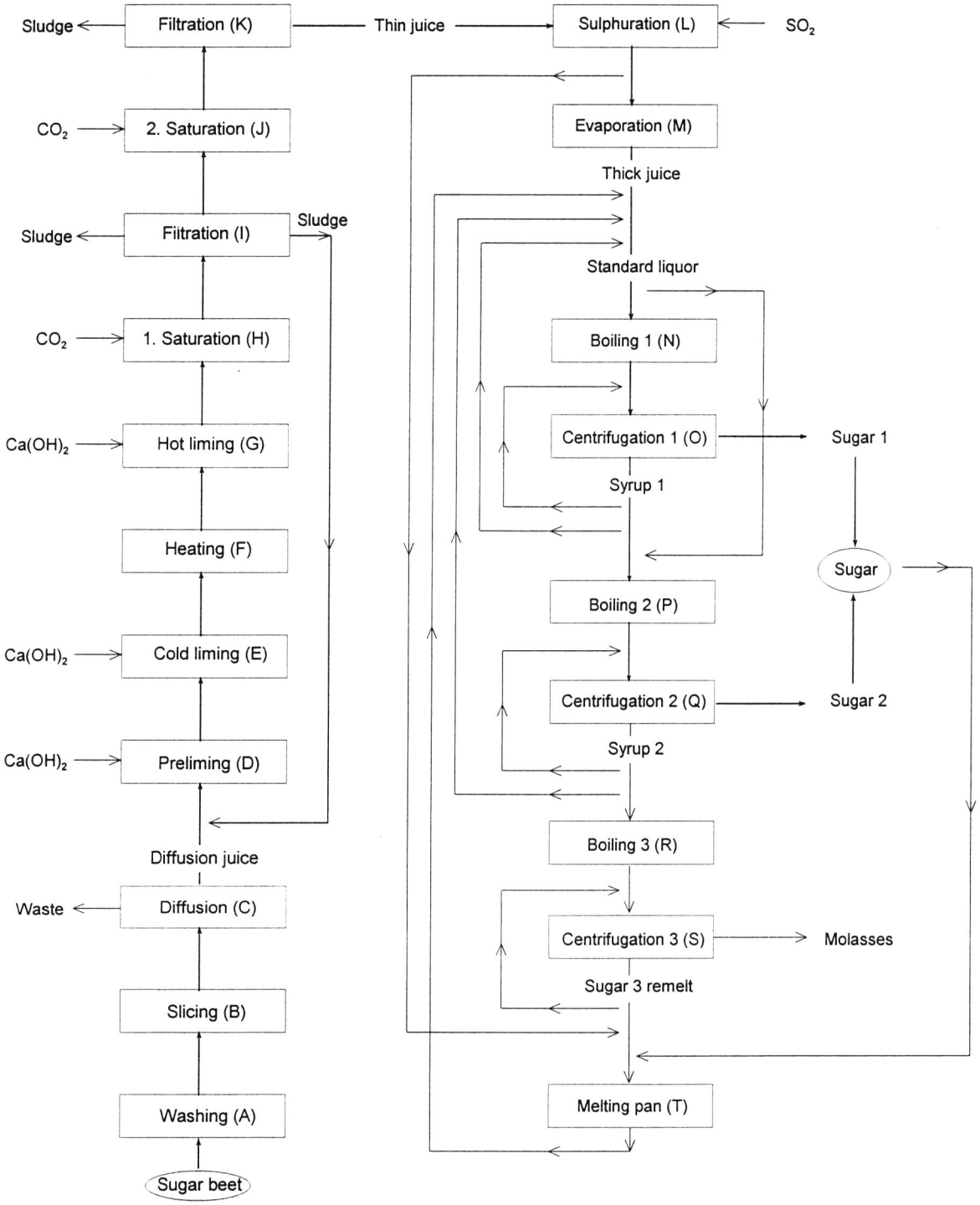


Figure 3.1: Schematics of the sugar production. The process involves many back feeding streams which allow colorants to circulate among the various stages in the process. See table 3.1 for a summary of the unit operations and conditions.

Step	T [°C]	pH	t [min.]	Description
A				Washer. Stirred washing tank.
B			10	Cutter. The whole beets are cut into 1 cm slices.
C	70	≈6	5	Diffusor. Sugar juice is extracted from the slices.
D	40	11	20-30	Preliming. $Ca(OH)_2$ is added. Precipitation of insoluble Ca-salts.
E	40	12	20-45	Cold liming. Additional $Ca(OH)_2$ is added to ensure precipitation. Higher alkalinity will form limestone in the apparatus.
F	85	12	≈5	Heaters. The strong alkaline solution is heated to 85 °C in order to induce insoluble Ca-salts. Invert sugar is degraded.
G	85	12	15-20	Reaction tank. Allows the species in solution to react and precipitate, hence pH decreases.
At this point the juice is partly free from insoluble acids, pectin and invert sugar. The heating and pH has denatured the quaternary structures of proteins. Enzymatic activity is neglectable from this point on.				
H	>80	10.8	-	1. Saturation. CO_2 is bubbled into the alkaline solution to lower the pH and precipitate the formed salts and the surplus of Ca^{2+}.
I	>80	10.8	-	Bag filtration. Precipitated compounds are removed by filtration.
J	>80	9.2	-	2. Saturation. pH-controlled addition of CO_2.
K	>80	9.2	-	Bag filtration. These filters are finer than the 1. saturation filters.
At this point the juice is, or should be, free from albumen/proteins, enzymes, pectin, Ca-salts, insoluble species and insoluble acids. Also invert sugar and reducing sugars should have been degraded to some extent.				
L	>80	8.5	-	Sulphuration. SO_2 is added to prevent discolouration in the subsequent boiling. Also pH is properly adjusted.
M	130	-	-	Evaporation. The product is called *thick juice*.
-	130	-	-	Sulphuration. SO_2 is added only if required in this stage.
N	≈80	-	120	Evaporation. Crystals are formed during removal of water.
O	≈40	-	-	Centrifugation. Separation of the stream into a non-sugar fraction, *syrup 1*, and a product stream called *sugar 1*.
P	≈80	-	240	Evaporation. Sugar crystals are precipitated from syrup 1.
Q	≈40	-	-	Centrifugation. Sugar crystals are removed by centrifugation. Produced sugar is *sugar 2*. Syrup 2 proceeds.
R	≈80	-	480	Evaporation. Crystals are formed from syrup 2.
S	≈40	-	12	Centrifugation. The non-sugar fraction is called *molasses* and is wasted. The sugar-containing fraction continues to the pan.
T	-	-	-	Melting pan. The sugar containing sludge is blended with thin juice and inoculated with sugar crystals before being recycled.

Table 3.1: Process parameters of main unit operations from production of sugar, DDS (1985), Larsson (1989) and Madsen et al. (1978). Compare to figure 3.1.

3.2 Sucrose and reducing sugars

The main sugar component, *sucrose*, is brought into focus when discussing the different types of species present in technical sugar juices. Sucrose is a disaccharide, implying that it is made up of two *mono*saccharides; *glucose* and *fructose*.

Sucrose is extracted from the beets in the diffusion process. However, at pH lower than 7-8 sucrose starts to decompose into *glucose* and *fructose*. An equimolar solution of these two monosaccharides is termed *invert sugar* within the field of sugar production. High temperatures and also bacteriological activity can promote decomposition of sucrose into invert sugar, DDS (1985).

As can be seen in fig. 3.2 there are important differences between sucrose and invert sugar. Sucrose itself is only weakly chemically reactive whereas both glucose and fructose contain a highly reactive carbonyl group. The invert sugar is, due to the carbonyl groups, responsible for formation of many coloured species. Besides formation of colour also loss of sucrose is what makes invert sugar unwanted in sugar production. Glucose and fructose are socalled *reducing sugars*. The term *reducing sugar* covers saccharides capable of reducing free aqueous Cu^{2+} to Cu^{+}, Stryer (1988). The carbonyl groups of the reducing sugars can react with amines to form Maillard products. It is also possible for the carbonyl group of a reducing sugar to react with other sugars to form polymers, Morrison and Boyd (1987). Both of these reactions form coloured products. A way to circumvent the problem of colour formation from the reactive carbonyl groups is by adding SO_2 which inhibits the reactive carbonyl groups. The chemical addition of SO_2 during sulphuration (fig. 3.1, L) must be sufficient to saturate the carbonyl groups currently present in the sugar juice, but there should also be sufficient SO_2 to saturate the carbonyls of invert sugar formed during the operations to follow. The addition of SO_2 to the carbonylic groups is irreversible under the conditions in the sugar process. Coloured species already formed are not affected by addition of SO_2, thus, the saturation with SO_2 must be done early in the process.

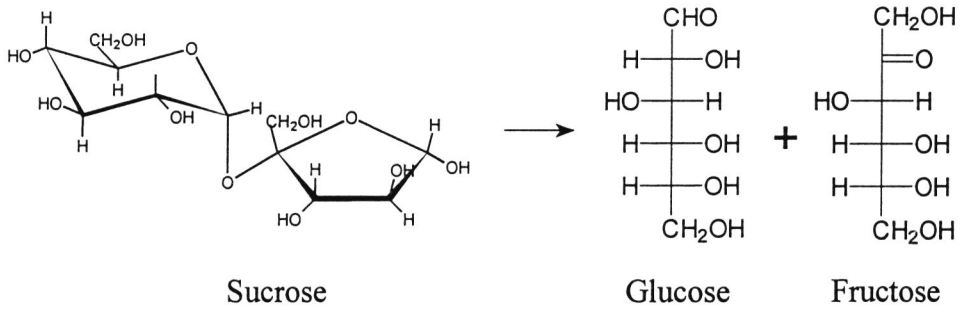


Figure 3.2: The 3 main saccharides in sugar juices. At pH lower than 7-8 sucrose decomposes into equal moles of glucose and fructose and the product is called *invert sugar* as a whole. Invert sugar is unwanted since it is precursory for formation of colour.

3.3 Fluorophores in technical sugar juices

Knowledge of the composition of the diffusion juice can give understanding of the subsequent reactions leading to formation of colour. Also the behaviour of the sugar juices when exposed to changes in pH can be understood when the different types of species are known. Numbers in square brackets refers to structures in appendix A.

Maag et al. (1972) and Schneider et al. (1966) report findings of all 23 *amino acids* in the diffusion juice. Drewnowska (1979) confirms nineteen of these amino acids and additionally two amides. Even though not all are fluorophores they can all subsequently act as reactants in the Maillard reaction producing a variety of coloured species. There are 3 fluorophoric amino acids; these are tryptophane[1], tyrosine[2] and phenyl-alanine[3]. Especially tryptophane should be noted since it has a significantly high fluorescence intensity due to a high absorptivity coefficient and a high quantum yield, see Ewing (1985) and Schulman (1979).

Winstrøm-Olsen et al. (1979) reports 21 *phenolic components* of varying concentrations of which tyrosine[2], DOPA[4], DOPAmine[5], cathecol[6] and L-noradrenaline[7] remains in almost the same concentrations throughout the purification process. Madsen et al. (1978) found presence of betalaine[8], betanine[9] and betalamic acid[10]. Hardegger (1952) reports presence of oleanolic-acid-glucoronid[11] which is a saponine causing foam in the diffusion juice.

McGhie (1993) reports findings of 20 *flavonoids* in sugarcane. Flavonoids generally absorb in the low UV with no absorbance in the VIS region. The structural flavonoidic base is depicted in appendix A as structure [27]. This group of compounds is expected to be present in sugar beets since flavonoids are secondary metabolites known to be present in most root plants.

Of non-fluorescing components phosphoric acid, sulphuric acid, oxalic acid, citric acid, lactic acid and other organic and inorganic acids are reported by Schneider et al. (1966). Kaipainen and Laitinen (1994) identify 19 pyrazines[12], 9 acids, 4 alcohols, 1 ester, 4 carbonylic compounds and 11 ethers and nitriles. Some of these reactive components are able to participate in the various reactions leading to formation of colour as will be shown later.

3.4 Non-enzymatic browning mechanisms

The non-enzymatic browning in sugar is accounted for by four mechanisms: Formation of Maillard products, cross linkage, polymerisation and caramelization. The Maillard reaction causes formation of melanoidines by reactions between carbonylic groups and amino groups. Cross linkage and polymerisation are two aspects of the same phenomenon: clustering of species in the sugar juices. The issue of caramelization has to do with thermal decomposition of sugars. Macrae et al. (1993), Madsen et al. (1979), Namiki et al. (1993) and Schneider et al. (1966) provides detailed discussion on the browning mechanisms.

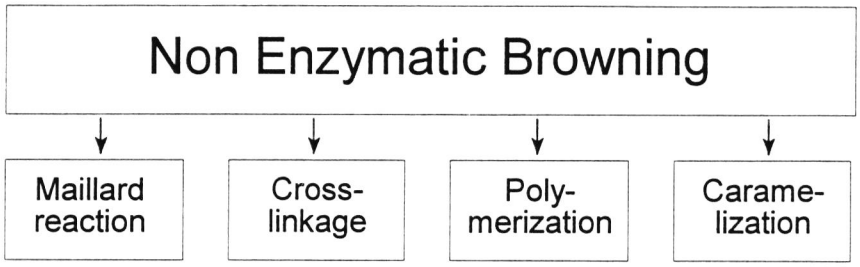


Figure 3.3: The non enzymatic browning is explained by four mechanisms, see Macrea et al.(1993), Madsen et al. (1978) and Schneider (1966)

3.4.1 Maillard reactions - formation of melanoidines

The Maillard reaction path is a composite reaction scheme involving formation of several complex intermediates. The final products from the Maillard reactions are termed *melanoidines* as a whole. The reaction offspring is the reaction between amines and carbonylic groups. Amines in the technical sugar juices are present as free amino acids and as parts of proteins. In the very composite sugar juices the potential products are numerous due to the large number of amines and carbonyls present.

The reaction scheme for the Maillard path is depicted in fig. 3.4. The amine and the carbonylic oxygen combines to form a Schiff's base[13]. This base is readily converted into an Amadori-compound[14]. Amadori compounds are through heating and dehydration converted into several different products. Among these products the very reactive 2,3-dicarbonyls[15] and 1,2-dicarbonyls[16] are of particular interest since they, having carbonylic groups, can react with free amines thereby restarting the reaction path with other reactants giving new products. However, a relatively small part of the total dicarbonyls are dehydrated by heating to form cyclic components, e.g. furfurals[17]. Dehydration and heating of the 2,3-dicarbonyls can liberate CO_2 giving shorter-chained carbonyls. According to Macrea et al. (1993) sole amino acids can in alkaline solution degrade by the Strecker degradation path and recombine to form new types of premelanoidines, subsequently new types of melanoidines.

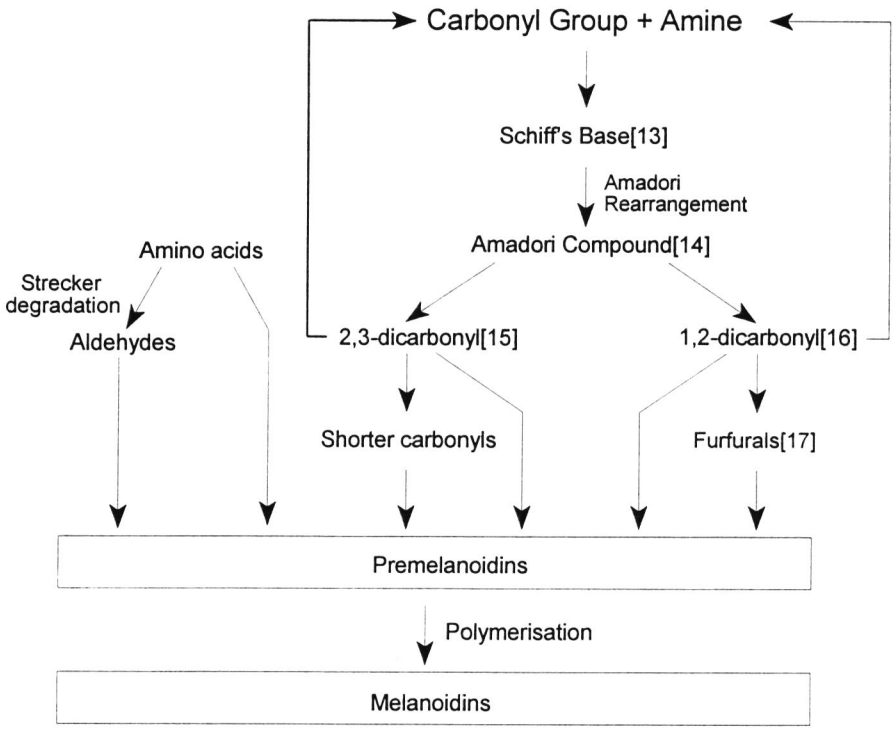


Figure 3.4: Schematic overview of the Maillard reaction path.

The melanoidines in general show strong absorption in the UV and low VIS regions.

3.4.2 Cross linkage, polymerization and caramelization

The highly reactive carbonyl groups in the reducing sugars can react to form glucose-fructose dimers, glucose-glucose dimers and fructose-fructose dimers. The resulting products are unstable and readily decompose into cyclic molecules giving rise to strong absorptions in the UV area. At high concentrations and temperatures, like in the final boiling where the crystals are formed, sucrose, glucose and fructose can liberate water and form caramels, see Schallenberger and Birch (1975). Due to this dehydration a new class of cyclic species can be formed giving rise to strong absorbance in the high UV and low VIS area.

3.5 Enzymatic browning mechanisms

An intact and functioning cell has membranes separating the different compartments. However, when the cell is disrupted, e.g. when slicing a sugar beet, the cell constituents are all mixed. Thereby proteins and enzymes are allowed to interfere with species they would not normally be in contact with. The resulting mixture consists of strong enzymatic and non-enzymatic oxidants and reductants which all are highly reactive. Several components

belonging to this group are reported by Schneider (1966); vitamins B_1, B_2 and B_6 along with enzymes *phosphorylases*, *dehydrogenases*, *oxidases* and *reductases*.

Especially one class of enzymes is of interest when discussing enzymatic browning in sugar juices and that is *polyphenol oxidases* (PPO). The PPO group reacts with the naturally occurring phenols in the sugar juices. Winstrøm-Olsen (1981) and (1982) have in-depth discussion of the physical-chemical properties of the PPO's.

3.5.1 Formation of melanines

The PPO group designates a group of mono-, di- and tri-phenol oxygenases, Schneider (1966). The PPO enzymes convert mono-, di- and tri-phenols such as tyrosine[2], DOPA[4], DOPAmine[5], cathecol[6], noradrenaline[7], olenanolic acid[11], phenol[18] and gallol[19] into highly reactive aromatic carbonylic compounds, socalled *quinones*.

The PPO enzyme group is capable of converting mono-phenols into di-phenols and successively di-phenols into di-ketones, e.g. a compound like 5,6-dihydroxy 2-indolic acid [22] is converted into 5,6-diketo 2-indolic acid [23]. The relatively high concentration of different substituted phenols in the diffusion juice, e.g. components [4,5,6,7], see Winstrøm-Olsen et al. (1979), is reflected in the high concentration of di-quinones [20, 21]. The carbonylic diquinones are very reactive in the Maillard reaction due to the two carbonylic groups, hence the enzymatically produced di-quinones can participate in the Maillard reaction via reactions with amino acids as discussed in a previous chapter. It has been shown that copper is a cofactor for the enzymes, see Madsen et al. (1979). Besides participation in the non-enzymatic Maillard cycle the various di-quinones produced by the PPO's can chemically form polymers as depicted in appendix A structure [24]. The polymers/complexes that are formed from the enzymatically produced quinones are called *melanines*. Heimdahl (1995) suggests melanine structure [25] to be likely in the case of PPO's converting dehydro ascorbic acid [26] into diphenols with successive polymerization into melanines. When the variety of possible substituents on the aromatic ring of phenols naturally occurring in the beet juices is considered, it must be concluded that a broad variety of melanines is likely to be present in the sugar juices.

However, in the process of sugar production any enzymatic activity of non-thermophile enzymes, like the PPO's, must take place before the vigorous conditions during cold liming are applied and certainly before the hot liming, see fig. 3.1, operations E and F. These vigorous conditions with temperatures over 80°C and pH higher than 11 will disrupt quaternary and tertiary structures of nearly all enzymes. This leaves the enzymes about 40 minutes to react with the phenols in the juice. On the other hand it must be excluded that all enzymatic activity stops as early as during the diffusion process where the slices are heated to approximately 70 °C. Winstrøm-Olsen (1982) reports that the enzymatic activity causing oxidation of the amino acid tyrosine[2] to DOPA[4] first stops at approximately 82 °C. The process parameter resulting in the highest degree of inhibition is pH. Winstrøm-Olsen (1981) found that above pH 8 no enzymatic formation of colour appears. Regardless if the reaction time is 10 or 40 minutes enzymatic activity must be considered as having a high colour-

forming potential since enzymes in most cases are extremely efficient due to their inherent catalytic properties.

According to Schneider et al. (1966) the diphenols produced during the oxidative process of the PPO's can react with iron ions present in the solution. These iron-diphenol complexes absorb in the VIS region and their absorbance have been shown to be strongly pH dependent with an optimum at around 8. However, if there are active PPO's present in the sugar juices, it is more likely that the polyphenols are converted into quinones. The reactive quinones will then follow the Maillard reaction paths outlined above producing melanoidines. Since the PPO's convert mono-phenols into poly-phenols and since there are free iron ions in solution due to the natural occurrence of iron in soil the conditions for iron-polyphenol complexes exist.

The plane oxygen rich structures of melanines depicted in appendix A as [24] and [25] clearly fulfill the main requirements for strong fluorescence. The most important characteristic is the presence of the non-bonding π-electrons in the rings. Secondly, the rigid structures are not able to compensate structurally for the absorbed energy. I. e., such structures can not transform the absorbed energy into structural bonding energy but must emit equally more. Lastly, electron rich atoms like oxygen, containing 2 lone-pair electrons, are present. The melanine group has a high probability for being one of the many fluorescent species detected in the sugar juices.

3.6 Summary on browning mechanisms in sugar juices

As a summary on the preceding discussion four main classes of sugar juice colorants have shown to be relevant. Clarke et al. (1989), Schneider et al. (1966) and Madsen et al. (1979) agrees more or less to the classes listed in table 3.2.

Melanoidines(NE)	Large polymers and complexes formed by the Maillard reaction. Absorb in both UV and VIS regions.
Flavonoides(NE)	Initially present in the beet juice. Absorb in the UV region.
Caramels(NE)	Thermal decomposition of sucrose, glucose and fructose followed by polymerization. Absorbance in the low VIS region.
Melanines(E)	Enzymatic oxidation of mono- and poly phenols into large polymers, see appendix A structure [24]. Absorbance in the VIS region.

Table 3.2: A summary on the four main classes of coloured species formed during sugar production. NE designates non-enzymatic reaction paths and E designates enzymatic reaction paths.

4. The fluorescent model system

A model system of thick juice was constructed so that it complied with the findings from a literature search and the fluorescent behaviour of thick juice. The usefulness of chemometric models will be assessed by using them for analysis on the results from spectrofluorometric analysis of the model system. Experiences from this analysis will be used in the future when the pure substances of thick juice will be obtained by chromatography (HPLC and LPLC) to calculate their concentrations. Another investigation of equal priority is to find the *limits* of the existing mathematical algorithms for resolving pure spectra. However, such limits are necessarily influenced by the performance of the apparatus in use.

The model system is based on a set of chemical species that were selected due to their known presence or due to their structural, hence fluorescent, likeness with other known components in thick juice. Since the fluorescence landscapes are strongly pH dependant, it was found that measuring landscapes at different pH levels could yield extra information. Hence, buffer systems capable of providing nearly non-quenching and non-fluorescent buffering have been chosen for this purpose.

4.1 Selecting modelling components

From the literature a set of chemical species have been chosen for making a model system of the fluorescent behaviour of the thick juice. However, far from all of the components found in thick juice can be bought commercially. Even if they could, they had to be synthesized with an enormous cost as a consequence. Instead, 9 fluorescent components were selected for various reasons: their presence in thick juice is established, their presence in thick juice seems likely or finally they have structural resemblance with components that have been found in thick juice. In table 4.1 a list of the selected components is shown. The reason for choosing each of the components is given as well as the wavelengths for excitation and emission for the fluorescence intensity maximum. The intensity maxima have been identified in non-buffered aqueous solutions on a reference spectrofluorometer and is given as (Ex_{max},Em_{max}) in nm.

Chemical component	Cause for selection
Tryptophane[1]	Amino acid, indolic, likely substrate for PPO's. Drewnovska (1979), Maag et al. (1972), Schneider et al. (1966) (285 nm, 390 nm)
Tyrosine[2]	Amino acid, phenolic, likely substrate for PPO's. Winstrøm-Olsen et al. (1979) (280 nm, 320 nm).
DOPA[4]	Intermediary product from enzymatic oxidation. Winstrøm-Olsen et al. (1979) (288 nm, 340 nm)
Phenol[18]	Phenolic, substrate for PPO's. (274 nm, 336 nm)
Catechol[6]	Phenolic, di-phenol, substrate for PPO's. Winstrøm-Olsen et al. (1979) (280 nm, 348 nm)
Pyrogallol[19]	Phenolic, tri-phenol, substrate for PPO's. (260 nm, 386 nm)
Hydroquinone[30]	Phenolic, di-phenol. Quinone-precursor. Substrate for PPO's. Schneider et al. (1966) (306 nm, 343 nm)
Furan-2-carboxylic acid[31]	Indolic. Schneider et al. (1966) (334 nm, 400 nm)
Indol[32]	Indole-base. Characteristic for indole based components. (305 nm, 345 nm)

Table 4.1: Selected fluorescent components for use in aqueous solutions. The structures are given in appendix B

However, as it turned out, the instrumental parameters would not allow detection of furan-2-carboxylic acid, pyrogallol and tyrosine. Thus, these 3 components had to be discarded in the final model system.

4.2 The effect of changes in pH on fluorophoric species

During fluorometric measurements the chemical species in a sample are excited by the excitation radiation. The energy in the excitation radiation is absorbed by the electron systems of the chemical substances in the solution. The fluorescence intensity is then determined as the energy, hence the wavelength, of the emitted radiation when the electron systems gain their normal energy levels. However, an excited system of electrons will try to compensate for the absorbed energy by spreading the concentration of high-energy electrons on all the neighbouring bonds, water molecules (H_2O) and protonated and non-protonated ions (e.g. H_3O^+ or OH^-) in the surroundings. In other words, the ease of which the excited electron system can compensate for the absorbed energy is dependent on the polarity and ionic composition of the solution. Since there are fewer fluorophore-solvent dipole bonds in non-

polar organic solvents these solvents can often provide higher fluorometric sensitivity. The excited species cannot use the surrounding solvent molecules to compensate for the higher energy level, but instead are forced to intensify the emission of energy. For a mathematical in-depth discussion on the energetics of the fluorescence mechanisms see Schulman (1977).

By applying different levels of proton (H^+) activity it is possible to force the species in the solution to undertake protonated or non-protonated forms. Since protonated and non-protonated species in most cases show different fluorophoric behaviour, due to the energetic properties mentioned, this feature can be used for extracting pH-dependent fluorescence landscapes for each sample.

A well known and (in food chemistry) often encountered fluorophoric group is the aromatic ring. The ring fluoresce due to the conjugated electrons making up the 1½-bond π-type skeleton. A group of fluorophoric aromatic compounds are the phenols which are known to be sensitive to changes in pH, Schulman (1979). The non-substituted phenols are weak acids, Morrison and Boyd (1987). Regular phenols without sidechains have pK_a's around 9.9, i. e. 9.89 for phenol[18] and 9.85 for cathecol[6], see Handbook(1989). Substituted benzene rings have more varying pK_a's, e.g. 9.38 for tryptophane[1], 8.40 for tyrosine[2] and 9.24 for phenylalanine[3].

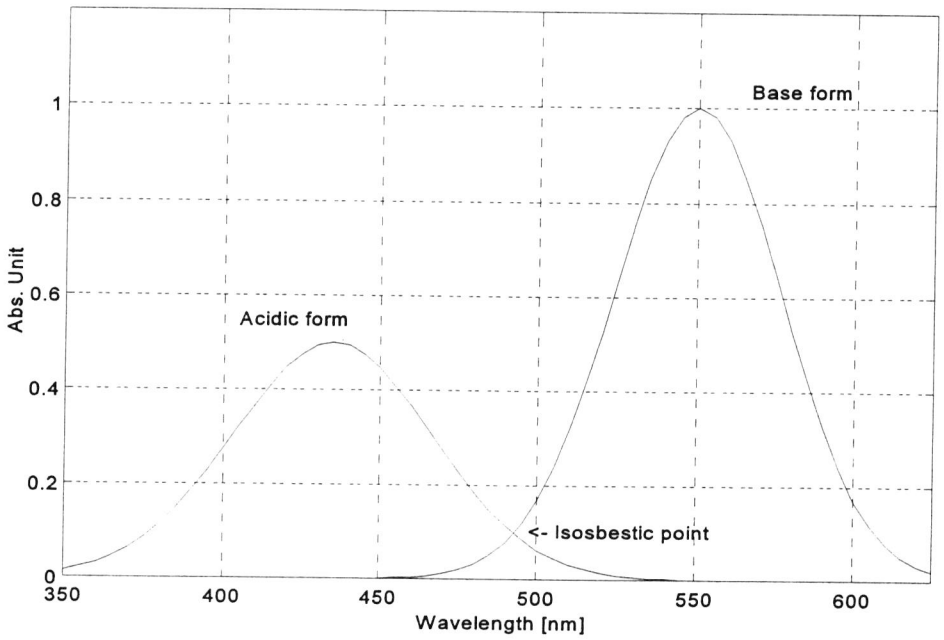


Figure 4.1: Example of isosbestic behaviour of phenol red, Ewing (1985).

Another example of the pH dependency of the fluorescence intensity is benzeneazodiphenylamine, also known as phenol red, see Ewing (1985). This well known indicator has two fluorophoric sites responsible for the absorption of light. In low pH solutions the acidic site dominates and the basic site is suppressed due to protonisation. In high pH solution the reversed is observed, this is depicted in fig. 4.1.

4.3 Buffers for controlling pH in fluorescent systems

A few buffers where chosen for the purpose of controlling the pH of samples being measured. Initially four levels of pH were chosen, i. e. 4, 6, 8 and 10. In table 4.2 the recipes for the four buffers are listed. For most solutions in fluorometry it generally applies that the analytes are present in suitably low concentrations giving no significant pH-effects - this also applies to thick juice, Andersson (1995). From table 4.2 it is apparent that the pH 8.0 buffer has a low capacity due to the high ratio between the concentrations of the acid and the base. Preliminary measurements showed that the capacity of the pH 8 buffer was too low to be used with satisfactory pH stabilizing effects. See Perrin and Dempsey (1974) for other suggestions for applicable buffers. The pK_a values are listed in table 4.2.

Citric acid, $(HOOC)C(OH)(CH_2COOH)_2$	$pK_{a,1}$	3.13
	$pK_{a,2}$	4.76
	$pK_{a,3}$	6.40
Glycine, $(NH_3)CH_2(COOH)$	$pK_{a,1}$	2.35
	$pK_{a,2}$	9.78

Table 4.2: Values of pK_a for the buffer components. Values taken from Handbook (1989)

After preparation the adjusted buffers were degassed using ultrasonics to avoid formation of air bubbles in the FIA tubing. The buffers in table 4.3 were all measured before use to verify that no measurable fluorescent contaminants were present. Aqueous solutions differ markedly in their contents of free oxygen (O_2) with and without citric acid, Handbook (1987). This feature can be expected to diminish quenching from free oxygen. Schulman (1979) argues that the content of free O_2 in polar solvents has a potential as a quencher of most fluorescent species.

pH	Preparation	I [M]
4.0	32.0 ml 0.1 M Citric acid stock solution and 18.0 ml 0.1 M Tri-sodium citrate stock solution diluted to 100 ml. Adjusted with HCl and/or NaOH.	0.059
6.0	9.0 ml 0.1 M Citric acid stock solution and 41.0 ml 0.1 M Tri-sodium citrate stock solution diluted to 100 ml. Adjusted with HCl and/or NaOH.	0.206
8.0	2.0 ml 0.1 M Citric acid stock solution and 48.0 ml 0.1 M Tri-sodium citrate stock solution diluted to 100 ml. Adjusted with HCl and/or NaOH.	0.295
10.0	25.0 ml 0.2 M Glycine stock solution and 16.0 ml 0.2 M NaOH diluted to 100 ml. Adjusted with HCl and/or NaOH.	0.057

Table 4.3: Recipes for buffers. The buffering components have been proven not to inflict on the fluorescence of thick juice, Andersson (1995). Stock solutions are defined in appendix B. I designates the ionic strength of the buffer, see Atkins (1989)

In table 4.3 the recipes for the buffers are listed. Also the ionic strength in the final buffer is shown. This will not differ markedly from the ionic strength when the analytes have been added. The concentrations of the various poly-valency ions used for calculating the ionic

strength have been found by solving the equilibrium equations for the buffer system, applying a zero-net-charge constraint to the mathematical solution. For this purpose MathCad have been used. A discussion of the importance of ionic strength will not be given here. However, the issue can be of great importance for the fluorescence intensities, see Schulman (1979), and as such the issue needs further investigation.

4.4 Sample preparation and measurement

The model system was made from 6 of the 9 chemical species. These were in alphabetical order: Catechol[6] (CATE), DOPA[4], hydroquinone[30] (HQUI), indol[32] (INDO), phenol[18] (PHEN) and tryptophane[1] (TRYP). The reason for disregarding furan-2-carboxylic acid[31], pyrogallol[19] and tyrosine[2] was the surprisingly low sensitivity of the instrument in the low UV range where these components have fluorescence peaks. Buffers for pH-levelling at 4, 6 and 10 were chosen. The pH 8 buffer had to be discarded due to lack of satisfactory buffering capacity as stated earlier.

The samples were prepared by adding certain volumes of the six stock solutions shown in appendix B. The samples were all diluted to a total volume of 100 ml with each of the 3 buffers. The volumes of added stock solution can be found in table 4.4

Volume of stock solution [ml], diluted to a total volume of 100 ml						
#	CATE	DOPA	HQUI	INDO	PHEN	TRYP
1	0.300					
2		1.800				
3			0.020			
4				1.750		
5					0.600	
6						2.000
7	0.300	0.600				
8	0.300			1.600		
9		0.800			0.600	
10		0.500				1.800
11	0.250		0.025		0.600	
12	0.100		0.030		0.400	
13			0.020	1.750	0.800	
14	0.400	0.700		1.600		1.800
15		0.500	0.020	1.800	0.900	2.000
16		0.700	0.030	1.600	0.900	1.600
17	0.300	0.600	0.030	1.400	0.800	1.400
18	0.400	0.800	0.030	1.600	0.800	1.800

Table 4.4: Added volumes of stock solution for the six components.

Using the buffers as solvents for the samples ensured the highest possible buffering capacity. The pH level of the measured samples was measured after each measurement to ensure that the buffer had the necessary capacity. All samples had the same pH as the buffer after measurement, hence, the capacities were sufficient for all 3 buffers.

The samples were, via the FIA pumping station, pumped through the flowcuvette while measuring. Since each of the 18 samples in table 4.4 gave rise to 3 different levels of pH, a total of 54 landscapes were measured. Due to the high intensity of the monochromator, it was found that the samples should not be exposed to stopped-flow measurements. By keeping a constant flow in the system the sample in the cuvette was exchanged continuously. The landscapes were made up of 28 excitation wavelengths (270 nm - 324 nm, 2 nm intervals), each consisting of 256 readouts (306 nm - 1137 nm, 3.2 nm intervals). It should be noted that not all readouts from the apparatus contained relevant information, hence, data were reduced before analysis, as shown in chapter 7.

In table 4.5 the resulting concentrations of the 6 species are shown. The concentration can be verified by multiplying the volumes from table 4.4 by the concentrations of the stock solutions in appendix B, correcting for dilution to 100 ml.

	Concentration [M·10^6] of fluorophore in sample					
#	CATE	DOPA	HQUI	INDO	PHEN	TRYP
1	300					
2		180				
3			20.0			
4				17.5		
5					600	
6						2.0
7	300	60				
8	300			16.0		
9		80			600	
10		50				1.8
11	250		25.0		600	
12	100		30.0		400	
13			20.0	17.5	800	
14	400	70		16.0		1.8
15		50	20.0	18.0	900	2.0
16		70	30.0	16.0	900	1.6
17	300	60	30.0	14.0	800	1.4
18	400	80	30.0	16.0	800	1.8

Table 4.5: The final concentrations in the 18 samples. Each of the 18 samples have been diluted to 100 ml using 3 different buffers at pH 4, 6 and 10.

The resulting 54 landscapes are investigated by 2-, 3- and 4-way methods in chapter 7. The concentrations in table 4.5 are considered to be the true values of the concentrations for use in the calibration models.

5. The instrumental analysis

The use of diode array (DA) detectors for fluorescence intensity measurements is quite new. The development of new and more sensitive DA's has made the application possible. The first DA's lacked the required sensitivity compared to the classic photomultipliers. In this application a DA has been used to measure emission spectra from fluorescent species in aqueous solution by the use of a flowcell. Due to the novelty of such an application and due to the time spent getting the system set up and making it work properly, it will be appropriate to report on the experiences from this work.

5.1 System components and specifications

The components are presented individually, each with a short description of the mechanics and the construction of it. The presentation is ended with suggestions for optimization. Some suggestions are minor, but other will greatly improve the performance of the system and are considered to be necessary to obtain a satisfactory instrument for the screening of thick juice.

The overall system

The hardware used in this context is intended for fast fluorometric analysis. It is the aim that it will eventually be sufficiently sensitive as well as sufficiently fast for use in automated in-line/at-line experimental setups. The key components in the system are the ultra fast monochromator and the highly sensitive diode array (DA), see figure 5.1. Since the system is a prototype it was expected that some time had to be used for fine-tuning the system for its first application: Flow injection analysis. The FIA system has been presented in the preliminary thesis, Andersson (1995).

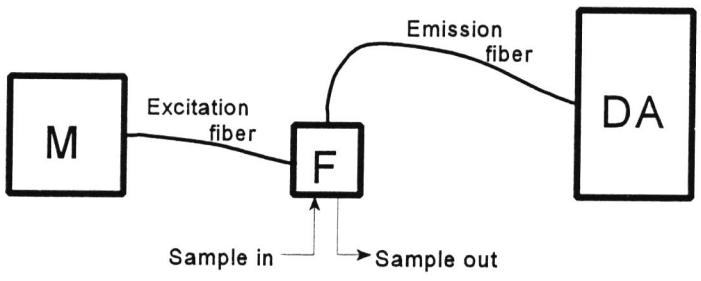


Figure 5.1: The system layout of monochromator (M), flowcuvette (F) and diode array spectrometer (DA).

The idea of putting together a uniquely powerful monochromator with the latest devel-

opment within fast spectral detectors was proposed by prof. Lars Munck. The suggested components were an ultra fast monochromator from T. I. L. L. Photonics Gmbh, Germany, in conjunction with a highly sensitive DA from Zeiss, Germany. The DA was furthermore equipped with Peltier cooling. The electronic and optical engineering as well as the software development have been entrepreneured by J&M, Aalen, Germany.

The monochromator

In figure 5.2 the pathway of the light in the monochromator is shown. The light from the xenon-gas lamp (L) is reflected by the concave mirror (M1) and is lined up by a slit (S) placed between mirror (M1) and mirror (M2). From mirror (M2) it is directed towards the grating (G) which disperses the light to a spectrum. The dispersed light is collected by mirror (M3) and is directed into the excitation fibre (F). The distance from the grating (G) to the mirror (M3) to the fibre (F) will result in spreading the dispersed light to a spectrum with a width of approx. 100 mm. A small part, 1.2 mm of the 100 mm, is directed into the fibre. This makes the light monochromatic. A key component not shown on figure 5.2 is the shutter placed at (F).

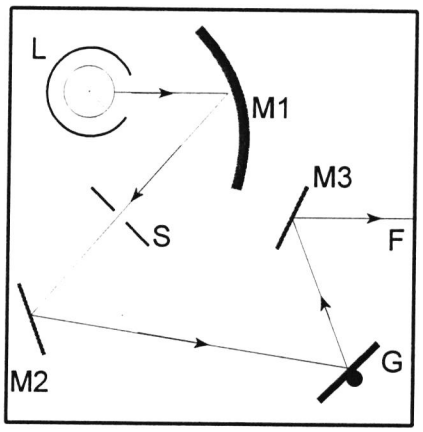


Figure 5.2: Layout of the fast monochromator. The light leaving the monochromator and entering the fibre at F should be nearly monochromatic.

The grating (G) is mounted on the axis of a powerful galvanomotor. The angle between the surface of the grating and the light from mirror (M2) is determined by the voltage applied to the galvanomotor. As will be discussed later, the wavelength calibration for a DA is very stable, hence the DA can be used for establishing a calibration table that relates the voltage to the excitation wavelength. During calibration the light from the monochromator is directed directly in to the DA. A fully automated calibration routine will apply a voltage to the galvanomotor while identifying the precise location, in nm, of the resulting peak. The location of the peak, and the voltage send to the monochromator are stored in a calibration file. This means that the calibration only takes about 5 minutes and that it requires no special equipment. It may be necessary, depending on the lamp, to introduce an element that can reduce the

light from the monochromator to avoid overloading the DA. It will not be possible to identify the exact location of the excitation peak if the DA is overloaded. After having established a calibration file, the software can look up the desired wavelength in the table and send a signal to the power control of the monochromator to apply the necessary voltage to the galvanic element.

The most interesting feature of the monochromator is the speed of which it is able to change wavelength. The galvanomotor makes it possible to run through 1200 different wavelengths per second. The high speed is facilitated by the circumstance that the grating is mounted *directly* on the axis of a highly powered high performance galvanomotor. The reported sweep rate is dependent on the weight of the grating mounted on the axis of the galvanomotor. The heavier grating, the more momentum has to be dealt with every time the motor moves. The only feasible solution when using a heavy grating is to built in a time delay to ensure that the axis is really in position to give the calibrated wavelength. Since the D/A converter is operating with 16 bit conversion, the voltage applied to the galvanomoter cannot undertake more than $2^{16} = 65536$ different values ranging from -10 to +10 V. This is more than sufficient when considering that the wavelengths used for excitation typically ranges from 200 nm to no more than 800 nm. Under these circumstances the resulting accuracy of the voltage will be ±0.4 mV, giving a theoretically accuracy of the excitation light at about ±0.01 nm. However, even though the D/A converter is more than sufficiently precise, the power supply has to be able to provide a very steady voltage for the galvanomotor. The power supply must be able to supply a steady voltage, otherwise it influences on the measurement by changing the excitation wavelength during the measurement.

The monochromator is flexible since it is possible to change the inner components for the purpose of optimizing the light source for a given application. The lamp can, as long as the basic fittings comply, be interchanged with any arc-type lamp, e.g. xenon lamps, deuterium lamps and combined mercury-xenon lamps, depending on the application. After changing the lamp, the concave mirror (M1) (figure 5.2) must be optimized to focus as much light as possible through the slit (S) and thereby onto the grating. When shifting to another lamp it is necessary to consider the choice of grating. The grating is the one component that reduces the power of the light most. Each grating is characterized by a 'blaze' wavelength and the number of grated lines per mm. The choice of grating must agree with the spectral characteristics of the lamp in such a way that the grating has its 'blaze' in the desired wavelength range. In order to optimize the output of the monochromator for future use a new mercury-xenon lamp has been purchased. The mercury light is characterized by having high intensity in the low UV area (below 300 nm). For the purpose of having full use of such a lamp, a new grating with a blaze around 230 nm have been ordered. The lamp and grating can simply replace the existing components in the monochromator to allow for optimization for the present spectrofluorometric application.

Fibres

The system components are connected by two 1 m optical fibres. The fibre used between the monochromator and the sampling unit is a 1.2 mm singlemode monofibre. The term *monofibre* designates that the cable contains one fibre in contrast to multifibres which are built up by many smaller fibres. The fibres used here are socalled *singlemode* fibres. The mode of a fibre characterizes how the optical core of the fibre is built up. A singlemode fibre consists of quartz material with the same refractive index throughout the radius of the fibre. A multimode fibre utilizes several layers of quartz alloys with different refractive indexes to make the light travel in a more uniform pattern. Whether the fibre is of one or the other mode does not influence on the signals from fluorescence measurements. The fibre leading the emitted light from the sample to the detector is a 0.6 mm singlemode monofibre.

It has been experienced that the fibres must be fixed in the experimental setup. The signal from the fibres depends strongly on the curvature of the fibre. The fibres are causing a minimum of loss in the intensity of the light, hence there is no cause for replacing the present fibres with shorter ones. The fibres cannot be shortened since this would impose restrictions of the versatility of the system in different setups.

Flowcuvette

The fluorescence/transmittance flowcuvette is the point where the sample is being analysed, and as such the flowcuvette is important for the performance of the whole system. This is reflected in the many little details built into the cuvette aiming to improve the light input and output. The cuvette is equipped with optics so that the excitation light from the fibres is focused on the middle of the 8 µl sample compartment. The emitted light is collected and focused to the end of the 0.6 mm fibre leading to the detector. The optics are made of quartz to allow for the use of light in the UV area. However, the optics are only *necessary* on the emission side to collect the strayed light into the narrow 0.6 mm fibre, but since the loss of light in the one lense is neglectable, the optics have been mounted on all 3 measuring sites. This allows for smarter designs of the instrumental setups.

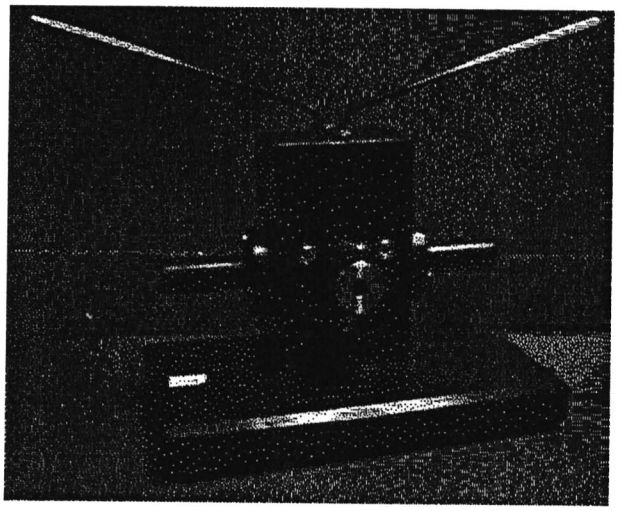


Figure 5.3: The flowcuvette. Note the 3 couplings that allow for simultaneous measurements of fluorescence as well as absorbance/transmittance.

On the back wall, opposite to the front coupling in figure 5.3, a mirror is mounted. The mirror is not, as could be expected, coated with glass or quartz, since the shift of refractive index in such a coating would change the wavelength of the emitted light. Thus, the mirror is made of specially grinded aluminium metal. According to the manufacturers the mirror improves the intensity of the emitted light with up to 40 %. A drawback is the high fragility of the mirror; it must not be touched, not even with a piece of lense cloth.

During operation it is crucial to avoid formation of gas bubbles in the fluid. The gas-liquid interface mirrors the excitation light directly into the emission detector. Such a strong signal will disrupt the measurement. The air bubbles tends to grow on non-polar sites in the tubing until they are torn of by the stream and passes through the flowcuvette. The problem can partly be accommodated for by degassing the solutions by using ultra sonics. This approach have been used with good results.

There are no suggestions to the improvement of the flowcuvette. A very little crack has been identified in the lense of one of the fibre couplings. By comparison, the crack is found to reduce the excitation light by less than 1%. This is neglectable in comparison to the overall influence of the flowcell to the reduction of the light.

Diode array detector

The core of the detector is depicted in figure 5.4. It consists of a fibre, a grating (G) and a DA. The light enters the core through a short fibre. After passage of the slit (S) the grating (G) disperses the light on to the diodes in the array (DA). The 256 diode readouts are send to the interface through the electronic port (E). The DA is specially designed for optimum sensitivity. This is done by choosing fewer diodes to maximize the area of each diode and by mounting a Peltier cooling system. The benefits from using a DA instead of a classic photo-multiplier detector is the scan time. A DA detector measures on all its diodes

simultaneously, hence no second monochromator is necessary to select the wavelengths for the detector. Unfortunately the sensitivity is approx. 100 times lower on a typical DA than on a classic photomultiplier system. A strong feature of DA's in general is the wavelength calibration. Once a DA has been exposed to a calibrated light source, one knows what wavelength range each individual diode covers. A DA contains no mechanical parts since all the components of the detector are cemented to a robust housing.

The 256 diodes in the array should, according to the specifications, facilitate detection from 300 nm to 1150 nm. With 256 diodes this gives an average span of each diode of approx. 3.3 nm. The grating has been designed to have a blaze at approx. 340 nm.

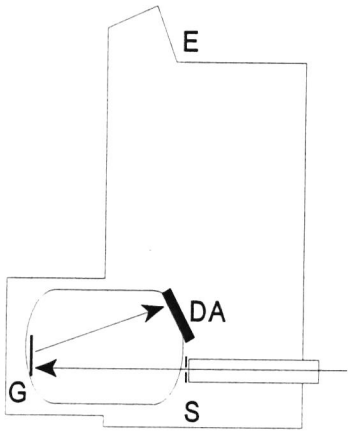


Figure 5.4: The mounting of the diode array in the detector. Scale approximately 1:1.

Just like a photomultiplier, the DA itself is adding noise to the measured signal. Each of the 256 diodes in the DA are given a constant voltage. A foton hitting a diode will reduce the current over that diode by making the silicium surface conductive. The decrease in voltage will, due to the supervising electronics, increase a counter for that diode by one. Then, after the desired integration time, the counters are read and the values are send to the computer through the interface (E). Before every measurement the counters are reset. However, since the applied voltage is optimized to give a high sensitivity it must be set to about the limit where the diode will start being conductive without the influence of incoming fotons. Sometimes, due to small instabilities of the current and due to thermal fluctuations, the current can occasionally decrease due to random discharges. Such discharges are causing artefactual counts giving rise to the socalled *dark current* (DC). Since it is independent of the light from the analytical application, it can be estimated by measuring the number of counts when the DA is unexposed to light. This is done automatically by use of the shutter in the monochromator. The software simply shuts off the light by activating the shutter, measures the number of counts for the same time as the measurement will take and the DC is known. Then the shutter opens and the counts are integrated for the desired time. The readout presented to the user are then the difference between the total number of counts and the DC. DC is very dependent of the integration time. This is caused by the rise in temperature of the DA when measuring for a longer period. Hence, the DC must be measured as long time as the

real measurement will take. It is not necessary to measure DC before every sampling. In order to improve the signal to noise ratio (S/N) a Peltier cooling system is mounted directly on the sides of the core. The Peltier cooling allows the core to be cooled down to -15 °C. The average level of DC over the diodes is approximately 620 counts per diode during an integration period of 500 ms at 25 °C. At 10 °C the number is 580 counts and at 0 °C the number is 500 counts. A minimum is reached at approx. -12 °C where the DC has lowered to approx. 400 counts. Using lower temperatures will not reduce the DC significantly. It should be noted that since the DC is a function of the integration time, the Peltier cooling system will not improve short-time measurements. The limit where it is possible to see the difference is about 500 ms - 700 ms when comparing room temperature to -10°C. For fluorescence measurements the cooling is necessary since these often last longer than 500 ms. The vendor and other experts regards Peltier cooling elements as being slowly consumed during operation. They can not be considered to provide the same cooling efficiency eternally since they are worn in time.

The detector is capable of measuring with integration times between 0.8 ms and 100000 seconds. The lower limit is set as a compromise between having as wide a working range as possible while still being able to download the spectra to the computer. Measurements lasting only 0.8 ms results in the fabrication of 1250 spectra each second. Each spectrum consists of (256*4=) 1024 bytes. This means that the instrument can produce up to 1.28 MB data every second. In order to facilitate a speedy data transfer the instrument makes use of a transputer board. This is mounted inside a computer. The link between the transputer board and the counter registers is a 16 pin RS422 interface. This interface allows the instrument to store the counts from the registers directly in the memory of the transputer board. The transputer board itself is connected to the PC via a 32 bit PCI connection allowing for fast downloading of data to the ordinary computer memory.

As will be discussed later, the effective range of DA showed not to meet the specifications. This gave severe problems for the analysis of the composite solutions which were designed to make full use of the specified range of the DA. The lower bound on the effective range of the instrument was found to start at 340 nm instead of the 300 nm specified by the vendor. In addition to this, the blaze of the grating was found to be very dominating in the emission spectra. Due to this, almost any analyte emitting between 340 nm and 360 nm would have a peak at 340 nm. This of course will have influence on the mathematical resolution of true underlying spectra. Individual weighing of the diodes, by multiplying each of the measured intensities in the low-sensitivity area with a factor to get a higher response from these diodes, was investigated. By doing this, the low signal-to-noise ratio (S/N) will remain the same since the noise is enlarged equally. It was concluded that a correction of the spectra was necessary since a correction would allow to investigate if the weighing of the diodes could higher the quality of the response in the range 306 nm - 360 nm. In order to determine the correction weights a sample of thick juice was measured on a Perkin-Elmer LS50B and the TIDAS-MMS1. Thick juice provides a wide peak, ranging from 290 nm to approx. 520 nm when excitated at 285 nm. The wide emission peak makes thick juice almost ideal for comparative measurements. The spectrum from the PE LS50B is automatically corrected since the instrument is a double-beam reference instrument with two photomultipliers. The

measured spectrum from the TIDAS was then divided point-by-boint by the reference spectrum to give the weights. These are shown in fig. 5.5. After applying the weights in fig. 5.5 it had to be concluded that the flaws of the DA could not be mended by applying correction weights to the diodes. This is due to the low S/N for the diodes in the range 306 nm to 340 nm.

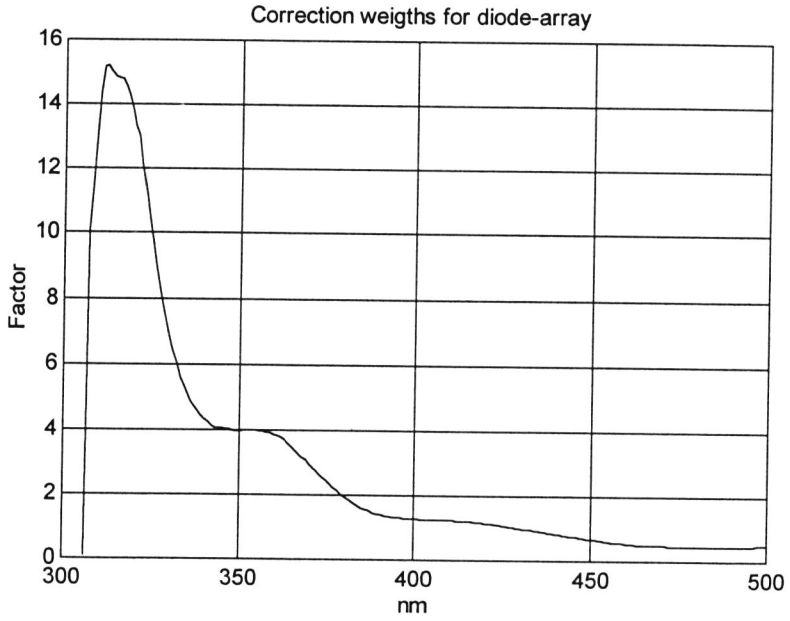


Figure 5.5: It was investigated if correction of the spectra could, at least partly, compensate for the low-sensitivity areas of the diode array. Note the high correction factors necessary to compensate for the low sensitivity of the instrument in the 300 nm - 400 nm range.

Note that the correction factors for the diodes in the range 300 nm - 360 nm are high. This indicates that the sensitivity of the instrument is correspondingly low. The reason for the factors going towards zero at about 310 nm is that the signals from these diodes are without any spectral information. Instead of having random readouts from these diodes their weights were set to zero to discard their counts.

Software

The monochromator and the DA requires controlling from a powerful computer and appropriate software. The development of the software has by far been the most troublesome and time consuming part of this project. A period of 5 months, with weekly contact to the programmers at the vendor, gave rise to 5 versions of the controlling software. Currently a version of the software, FL3095 Vers. 1.2.1 is being used. It is operational but it still contains minor flaws. The one main problem concerns storage of data. If the parameters are set to measure a landscape with, say, 28 different excitation wavelengths the software often records a lower number of spectra. This means that after each scan one has to leave the measurement mode, enter plotting mode, and check if the proper number of spectra have been recorded. If

not, one has to repeat the measurement from the start. As a consequence, the software does not provide a base on which an automated FIA measuring station can be build.

Even when the bug described above is fixed, there are still many ways to improve the software. A solution which includes programming software of our own has also been considered. This will possibly be the last resort from the software problems. However, it will enable us to obtain a tailor-made solution for each application.

Future activities - a summary

The monochromator will be optimized by installing a high yield UV-lamp and an appropriate grating in the monochromator. The spectrometer needs a new detector core, with much more sensitivity in the region below 350 nm. This requires a new instrument or a replacement of the old core with a new type which will fulfill the original specifications: Suitable sensitivity to measure fluorescence in the range 300 nm - 950 nm. In order to accommodate for the high DC arising during the relatively long fluorescence measurements, a new core will require Peltier cooling enabling temperatures below -10 °C. Finally the software needs thorough revision. If necessary, a new software will be programmed to control the instruments.

6. Algorithms for *n*-way modelling

In this chapter terminology, procedures for data storage and handling and also implementations of general *n*-way algorithms for PARAFAC- and TUCKER-models will be presented. A library of Matlab® programs capable of performing decomposition on general *n*-way data structures have been developed. These programs extend the collection of procedures for 3-way analysis made during the preliminary work, Andersson (1995). In the preliminary report, chapter 5, the models for PARAFAC, TUCKER and PLS are introduced. The following discussion is an extension of the former work rather than a repetition.

6.1 *n*-way terminology and notation

In the following scalars are represented by lowercase italics, e.g. n. Vectors are indicated by lowercase bold, e.g. $\mathbf{r}$. Matrices will be written as upper-case bold, e.g. $\mathbf{X}$, whereas *n*-way data arrays are written as underlined upper-case bold, e.g. $\underline{\mathbf{X}}$. Individual elements in vectors and matrices will be referred to by the use of suffixes, e.g. $\mathbf{r}=[r_1\ r_2\ r_3\ \cdots\ r_n]$. The dimensions of variables will be specified in parenthesis only at the introduction. The number of ways of the *n*-way array $\underline{\mathbf{X}}$ is represented by the scalar n. The dimension of each of the ways in $\underline{\mathbf{X}}$ $(r_1 \times r_2 \times \cdots \times r_n)$ is described by elements in a vector $\mathbf{r}$ (n). Hence, a 4-way structure could have $\mathbf{r}$=[10 3 6 8] designating that $\underline{\mathbf{X}}$ consists of 10 observations in the first mode, 3 in the second and so on. The number of factors to derive from the data will be contained in the vector $\mathbf{w}$ (n). When decomposition is done using the TUCKER model a different number of factors can be derived in each of the n ways, e. g. $\mathbf{w}$=[3 3 2 3]. The elements of $\mathbf{w}$ must be the same in the PARAFAC case, e.g. [2 2 2 2]. Vector notation is preferred for the sake of generality. ⊗ designates the kronecker product, Henrion (1994).

In order to provide a familiar, straight forward and yet syntactically correct base for discussion of algorithms *pseudo*-Matlab commands will be used to illustrate the implementation of operations on data structures and vectors. In some procedures Matlab notation like `r(1:n)` is used. This refers to a subvector of the vector $\mathbf{r}$ which is constructed from elements number 1 to n. It is necessary to define an operation that will allow for the rearrangement of data during calculations. If $\mathbf{X}$ $(i \times j)$ is a matrix, then the Matlab expression `x=X(:)'` will return a vector $\mathbf{x}$ $(i \cdot j)$ with the elements of $\mathbf{X}$ taken columnwise, stringing these elements out to a vector, i. e. $\mathbf{x}=[x_{1,1}\ x_{2,1} \cdots x_{i,1}\ x_{2,1}\ x_{2,2} \cdots x_{i,2} \cdots x_{1,j} \cdots x_{i,j}]$. The prime designates transposition of the matrix. The inverse function returning a matrix $\mathbf{Y}$ $(k \times l)$ from an argument matrix $\mathbf{X}$ $(i \times j)$, with $k \cdot l = i \cdot j$, is formulated in Matlab notation as `Y=reshape(X,k,l)`. The defined operations will be used for the discussion of algorithms later in the text. In the equations to follow, the error terms are neglected. Of course error is present, but including error terms does not facilitate the essential issue; how to derive *n*-way algorithms giving the solutions to *n*-way PARAFAC and TUCKER models and PLS regression models for truly any n.

6.2 Storage and handling of *n*-way data structures

In order to store and manipulate data in most popular mathematical programming environments (Matlab®, Mathematica®, MathCad® and others) the contents of the *n*-way array $\underline{\mathbf{X}}$ need to take form of a matrix or a vector. For the purpose of establishing general *n*-way algorithms the matrix format have been preferred over vector format. This is mostly due to the circumstance that Matlab specifically has been optimized for matrix manipulations. Also, authors in general tend to use the matrix form for storing *n*-way data, Geladi (1990) and Henrion (1994).

When the *n*-way data array is transformed into a 2-way matrix, this is referred to as *unfolding*. When unfolding an *n*-way structure, the way representing the rows of the matrix can be arbitrarily chosen among the *n* ways. The columns will then have to be formed by the remaining (*n*-1) ways. For the purpose of optimizing the numerical aspects of the algorithms, a special method should be applied when transforming *n*-way data structures into 2-way data structures. The recipe for unfolding, which I call *systematic unfolding methodology* (SUM), ensures that it is possible to shift successively between the different unfoldings during the iterations. An example of the SUM principle for unfolding is given in Table 6.1. In the 4-way case the structure $\underline{\mathbf{X}}$ is unfolded into matrices $\mathbf{X}^1$, $\mathbf{X}^2$, $\mathbf{X}^3$ and $\mathbf{X}^4$ where the superscript denotes the number of the way that is represented by the rows in the $\mathbf{X}$-matrices.
The dimensions of the unfolded matrix $\mathbf{X}^k$ are given in equation (6.1) as

$$\left(r_k \times \frac{1}{r_k} \prod_{l=1}^{n} r_l \right) \tag{6.1}$$

For way number *i* running down through rows, the unfolded matrices are made of distinct submatrices with elements having observation number *n*-*i*+1 in common. Again, these submatrices are made of distinct submatrices from the same observation number *n*-*i*. This system is repeated down to column level. This will be exemplified: Observe $\mathbf{X}^3$, that is *i*=3. Due to the relation just mentioned, the elements from the same observation in way number (*n*-*i*+1 = 4-3+1=) 2 are forming distinct submatrices in the unfolded matrix. In each of these submatrices the elements from the preceding way number (*n*-*i*=4-3=) 1 are forming submatrices. This can be verified from table 6.1 where the four possible SUM unfolding matrices for the 4-way case are shown. The SUM principle holds for any *n*.

$$X^1 = \begin{bmatrix} x_{1,1,1,1} & x_{1,2,1,1} & x_{1,1,2,1} & x_{1,2,2,1} & x_{1,1,1,2} & x_{1,2,1,2} & x_{1,1,2,2} & x_{1,2,2,2} \\ x_{2,1,1,1} & x_{2,2,1,1} & x_{2,1,2,1} & x_{2,2,2,1} & x_{2,1,1,2} & x_{2,2,1,2} & x_{2,1,2,2} & x_{2,2,2,2} \end{bmatrix}$$

$$X^2 = \begin{bmatrix} x_{1,1,1,1} & x_{2,1,1,1} & x_{1,2,1,1} & x_{2,2,1,1} & x_{1,1,2,1} & x_{2,1,2,1} & x_{1,2,2,1} & x_{2,2,2,1} \\ x_{1,1,1,2} & x_{2,1,1,2} & x_{1,2,1,2} & x_{2,2,1,2} & x_{1,1,2,2} & x_{2,1,2,2} & x_{1,2,2,2} & x_{2,2,2,2} \end{bmatrix}$$

$$X^3 = \begin{bmatrix} x_{1,1,1,1} & x_{1,1,1,2} & x_{2,1,1,1} & x_{2,1,1,2} & x_{1,2,1,1} & x_{1,2,1,2} & x_{2,2,1,1} & x_{2,2,1,2} \\ x_{1,1,2,1} & x_{1,1,2,2} & x_{2,1,2,1} & x_{2,1,2,2} & x_{1,2,2,1} & x_{1,2,2,2} & x_{2,2,2,1} & x_{2,2,2,2} \end{bmatrix}$$

$$X^4 = \begin{bmatrix} x_{1,1,1,1} & x_{1,1,2,1} & x_{1,1,1,2} & x_{1,1,2,2} & x_{2,1,1,1} & x_{2,1,2,1} & x_{2,1,1,2} & x_{2,1,2,2} \\ x_{1,2,1,1} & x_{1,2,2,1} & x_{1,2,1,2} & x_{1,2,1,2} & x_{2,2,1,1} & x_{2,2,2,1} & x_{2,2,1,2} & x_{2,2,2,2} \end{bmatrix}$$

Table 6.1: The four resulting matrices of a 4-way data array $\underline{\mathbf{X}}$ (2×2×2×2) with systematic unfolding methodology (SUM) applied to the unfolding.

From table 6.1 it is readily seen that $\mathbf{X}^k$ can be derived from $\mathbf{X}^{k-1}$ by applying procedure 6.1. The existence of this fast and simple procedure for obtaining each of the necessary $\mathbf{X}$ matrices is a great advantage when large data structures, as often in n-way analysis, are to be analysed.

It is assumed that $\underline{\mathbf{X}}$ contains an n-way structure and that $1 \le k \le n$. The contents of $\underline{\mathbf{X}}$ is stored in a matrix $\mathbf{X}^{k-1}$ according to the SUM principle. The vector $\mathbf{r}$ hold the dimensions of the array.

1. Calculate $i = r_k \quad , \quad j = \prod_{l=1}^{n} r_l$

2. If k=1
   $\mathbf{X}^k$=reshape($\mathbf{X}^n$,j/i,i)'
   else
   $\mathbf{X}^k$=reshape($\mathbf{X}^{k-1}$,j/i,i)'
   end

Procedure 6.1: Procedure for changing between successive unfolding matrices when they apply to the SUM unfolding. This procedure will change the unfolding from $\mathbf{X}^{k-1}$ to $\mathbf{X}^k$. The procedure is cyclic, that is, after n rearrangements the original arrangement, $\mathbf{X}^1$, is obtained.

Due to procedure 6.1 it is possible to keep all data in one array instead of making all the unfolding matrices of $\underline{\mathbf{X}}$ before the iterations start. Making all the unfoldings of the original matrix simultaneously would demand n times as much memory, but now, keeping data in the same array, merely undertaking different structures, it is possible to require memory only double to that of the original array (double due to the reshaping routine making a copy of the array while working). Procedure 6.1 must be applied successively to the unfolded matrices, hence it is not possible to jump directly from $\mathbf{X}^2$ to $\mathbf{X}^5$. This can present a minor time expense in the case of data preprocessing where only certain modes are to be scaled or centred. With the presentation of procedure 6.1 the discussion of how to obtain the desired forms of data is

ended. Procedure 6.1 is one of the cornerstones in the true *n*-way algorithms due to the low requirement of memory and the simplicity of the equations as will be shown later.

In order to make *n*-way algorithms it is essential to find systematics in the way the formulas are set up. Here, *one* simple equation is defined to represent all of the model equations. It is shown in equation (6.2).

$$\mathbf{X}^k = \mathbf{F}^k\mathbf{M}^k \tag{6.2}$$

In equation (6.2) the *k*th unfolding of $\underline{\mathbf{X}}$ is related to a matrix containing the w_k factors in the *k*th way stored columnwise, $\mathbf{F}^k$ ($r_k \times w_k$), and a multiplicator matrix, $\mathbf{M}^k$ which is formed to make the equation valid. The dimensions of $\mathbf{X}^k$ are given in equation (6.1). The dimensions of $\mathbf{M}^k$ are given in equation (6.3) as

$$\left(w_k \times \frac{1}{r_k} \prod_{l=1}^{n} r_l \right) \tag{6.3}$$

The algorithms developed here are iterative, that is, they solve for the factors in one way at a time, using the current states of the other factors. The update is performed for each way and is repeated until convergence is obtained for the total fit of the model. However, other convergence criteria can be used. Since the structure of $\mathbf{M}^k$ is dependent on the model, we will postpone the discussion hereof. Now, we will turn to the solution of equation (6.2), assuming that $\mathbf{M}^k$ is arranged properly so that equation (6.2) can be used to solve the PARAFAC and TUCKER models. *N*-way PLS is based on either the PARAFAC or the TUCKER algorithm, thus it is left out of the discussion for the time being.

Unconstrained factors

The most common approach for the initial analysis of data is to solve for totally unconstrained factors giving a least squares fit to the model. Additionally, if one is set out for a true explorative analysis one often wishes to remove all other constraints on the solutions than the ones the choice of model imposes.

Equation (6.4) shows the updating step for each of the *n* ways with $1<k<n$ giving the least squares fit of $\mathbf{F}^k$ to $\mathbf{X}^k$ and $\mathbf{M}^k$ in equation (6.2).

$$\mathbf{F}^k = \left(\mathbf{X}^k\mathbf{M}^{k\prime}\right)\left(\mathbf{M}^k\mathbf{M}^{k\prime}\right)^{-1} \tag{6.4}$$

The term *unconstrained* means that the derived factors do not have to comply with any external constraints.

Constraining factors to non-negativity

In addition to the derivation of unconstrained factors, equation 6.4, it can sometimes be instructive to derive factors that are constrained. One constraint with great potential for analysis of results from spectral analysis is the non-negativity constraint (NNC). It is possible to force the factors $\mathbf{F}^k$, to give least squares fit to equation (6.2) under the constraint that they must be non-negative. This strong assumption of the behaviour of reality is applicable to a broad span of data arising from spectral chemical analysis. However, being careful not to force the reality to undertake constraints based on erroneous deductions, such constraints should be used late in the progress of data analysis. For obvious reasons explorative data analysis should include a comparison between the constrained solutions and the unconstrained solutions in order to draw conclusions on the validity of the assumed underlying mathematics. The theory of non-negativity regression algorithms will not be discussed here due to the rather extensive theory, see Lawson and Hanson (1974). In this coherency the implementation of NNC is based on a modified version of the Matlab built-in procedure, slightly optimized for speed. NNC is implemented in the PARAFAC as well as the TUCKER algorithms as a call to this modified routine. The Matlab routine performing the non-negativity least squares regression is called CNNLS. It is listed in appendix C. The function call to solve equation (6.2) under NNC with regards to $\mathbf{F}^k$ is as described in equation (6.5). It is shown here so that it can be recognized in the procedures.

$$\mathbf{F}^k = \mathrm{CNNLS}\left(\mathbf{X}^k, \mathbf{M}^{k\prime}\right) \tag{6.5}$$

Constraining factors to orthogonality

The last constraint to be presented here is orthogonality. It is based on the assumption that the factors should be orthogonal within one or more ways. The orthogonality update used in this context is shown in equation (6.6). The equation will give factors that are orthogonal in $\mathbf{F}^k$ while giving the least squares fit of equation (6.2).

$$\mathbf{F}^k = \mathbf{X}^k\mathbf{M}^{k\prime}\left(\mathbf{M}^k\mathbf{X}^{k\prime}\mathbf{X}^k\mathbf{M}^{k\prime}\right)^{-1/2} \tag{6.6}$$

Equation (6.6) has been implemented for obtaining orthogonal solutions in both the PARAFAC and TUCKER procedures shown in appendix C. Equation (6.6) is discussed in Harshman and Lundy (1994). Orthogonality constraints can sometimes be used to derive solutions to problems that would give degenerate solutions using unconstrained factors, Mitchell and Burdick (1994).

Equation (6.2) is a somewhat trivial matrix equation but the interesting part is to arrange $\mathbf{M}^k$ in such a way that a general model can be obtained. But before entering that discussion a structure that can hold the derived factors (solutions) must be constructed. The common way to store factors from PARAFAC and TUCKER modelling is to use matrix structures. E.g., for

a fourway PARAFAC or TUCKER model the loadings of the four modes are held in the matrices **A**, **B**, **C** and **D**, where column l in the matrix **A** holds the lth factor in the first way. In a similar manner the factors from the second, third and fourth ways are stored columnwise in matrices **B** ($r_2 \times w_2$), **C** ($r_3 \times w_3$) and **D** ($r_4 \times w_4$). For the purpose of making one general structure that can hold all the factors from all of the ways, the vector **fac** is introduced. In **fac** all factors are stored in one long structure. Thus, **fac** contains the number of elements given in equation (6.7).

$$\sum_{l=1}^{n} w_l r_l \tag{6.7}$$

If **fac** is to be constructed from the solutions in each of the n steps during the iterations, it is done as illustrated in pseudo-Matlab notation in equation (6.8). This corresponds to $\mathbf{f}^1$, $\mathbf{f}^2$, $\mathbf{f}^3$ and $\mathbf{f}^4$ being equal to **A**, **B**, **C** and **D**.

$$\mathbf{fac} = [\ \mathbf{f}^1(:)'\ \mathbf{f}^2(:)'\ \cdots\ \mathbf{f}^n(:)'\ ] \tag{6.8}$$

From matrices **A**, **B**, **C** and **D** the vector **fac** is constructed as in equation (6.9).

$$\begin{aligned}\mathbf{fac} = [\ & a_{1,1}\ a_{2,1}\ \cdots\ a_{r_1,1}\ a_{1,2}\ a_{2,2}\ \cdots\ a_{r_1,2}\ a_{1,n}\ a_{2,w_1}\ \cdots\ a_{r_1,w_1}\ \cdots \\ & b_{1,1}\ b_{2,1}\ \cdots\ b_{r_2,1}\ b_{1,2}\ b_{2,2}\ \cdots\ b_{r_2,2}\ b_{1,w_2}\ b_{2,w_2}\ \cdots\ b_{r_2,w_2}\ \cdots \\ & c_{1,1}\ c_{2,1}\ \cdots\ c_{r_3,1}\ c_{1,2}\ c_{2,2}\ \cdots\ c_{r_3,2}\ c_{1,w_3}\ c_{2,w_3}\ \cdots\ c_{r_3,w_3}\ \cdots \\ & d_{1,1}\ d_{2,1}\ \cdots\ d_{r_4,1}\ d_{1,2}\ d_{2,2}\ \cdots\ d_{r_4,2}\ d_{1,w_4}\ d_{2,w_4}\ \cdots\ d_{r_4,w_4}\ ]\end{aligned} \tag{6.9}$$

In order to access the ith element of the jth factor in the kth way one must use the sth element of **fac**. The element number is calculated as in equation (6.10)

$$s(i,j,k) = i + (j-1)w_1 \qquad \text{for } 1 \le j \le w_1\ ,\ 1 \le i \le r_1\ ,\ k=1, \tag{6.10a}$$

and

$$s(i,j,k) = i + (j-1)w_k + \sum_{l=1}^{k-1} w_l r_l \qquad \text{for } 1 \le j \le w_k\ ,\ 1 \le i \le r_k\ ,\ 2 \le k \le n, \tag{6.10b}$$

From the same equations it can be shown that the matrices **A**, **B**, **C**, **D** and so on, can be derived from **fac** by the following Matlab code as:

```
A=reshape(fac(s(1,1,1):s(r(1),w(1),1)),r(1),w(1))
B=reshape(fac(s(1,1,2):s(r(2),w(2),2)),r(2),w(2))
C=reshape(fac(s(1,1,3):s(r(3),w(3),3)),r(3),w(3))
D=reshape(fac(s(1,1,4):s(r(4),w(4),4)),r(4),w(4))
```

At this point the general equation (6.2) and the different ways to derive constrained solutions have been presented. Also, the structure of the vector containing the derived solutions, **fac**, has been described. These operations give a base upon which the *n*-way algorithms for PARAFAC, TUCKER and PLS modelling can be build.

6.3 General *n*-way PARAFAC modelling

The PARAFAC model is probably the simplest model for analysis of *n*-way structures. For a general introduction to the model see Andersson (1995) page 22. PARAFAC has a strong characteristic; due to the model structure there is no problem with rotation of factors and as a consequence hereof the solutions to the PARAFAC model resembles the true behaviour of the observations provided that the choice of model is correct with regards to the nature of the data. This makes the model almost ideal for modelling data with origin in chemical analysis. The PARAFAC model has already been proven to be able to resolve the true spectra from fluorescence measurements of 3 component mixtures, Andersson (1995). The matrix equation for the PARAFAC model is given in equation (6.11). References to factors stored columnwise in matrices **B**, **C**, and **D** are represented as vectors **b**, **c** and **d** with the column number given as subscript.

$$\mathbf{X}^1 = \mathbf{A}\mathbf{M}^1 = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_{w_1} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1' \otimes \mathbf{c}_1' \otimes \mathbf{d}_1' \\ \mathbf{b}_2' \otimes \mathbf{c}_2' \otimes \mathbf{d}_2' \\ \vdots \\ \mathbf{b}_{w_2}' \otimes \mathbf{c}_{w_3}' \otimes \mathbf{d}_{w_4}' \end{bmatrix} \tag{6.11}$$

Equation (6.11) gives a clear idea on how the $\mathbf{M}^k$ matrices should be ordered to make equation (6.2) valid. The PARAFAC model can be found as the solution to equation (6.2) with the $\mathbf{M}^k$ matrices defined as described in table 6.2. It should be noted, that even though the successive matrices of $\mathbf{X}^k$ can be derived from each other by the use of algorithm 6.1, the kronecker product in the $\mathbf{M}^k$ matrices can not. The $\mathbf{M}^k$ matrices have to be calculated each time a set of factors has been updated. However, it is not difficult to understand the systematics of the successive $\mathbf{M}^k$ matrices shown in table 6.2.

$$\mathbf{M}^1 = \begin{bmatrix} \mathbf{b}_1' \otimes \mathbf{c}_1' \otimes \mathbf{d}_1' \\ \mathbf{b}_2' \otimes \mathbf{c}_2' \otimes \mathbf{d}_2' \end{bmatrix} \quad \mathbf{M}^2 = \begin{bmatrix} \mathbf{c}_1' \otimes \mathbf{d}_1' \otimes \mathbf{a}_1' \\ \mathbf{c}_2' \otimes \mathbf{d}_2' \otimes \mathbf{a}_2' \end{bmatrix}$$

$$\mathbf{M}^3 = \begin{bmatrix} \mathbf{d}_1' \otimes \mathbf{a}_1' \otimes \mathbf{b}_1' \\ \mathbf{d}_2' \otimes \mathbf{a}_2' \otimes \mathbf{b}_2' \end{bmatrix} \quad \mathbf{M}^4 = \begin{bmatrix} \mathbf{a}_1' \otimes \mathbf{b}_1' \otimes \mathbf{c}_1' \\ \mathbf{a}_2' \otimes \mathbf{b}_2' \otimes \mathbf{c}_2' \end{bmatrix}$$

Table 6.2: Example of a 2 factor 4-way PARAFAC. If these four matrices are used in conjunction with the **X** matrices in table 6.1 then, by using the regression in equation 6.4, one will obtain the unconstrained least squares solution to the PARAFAC model.

From the systematics of the **M** matrices, it is possible to establish a loop that can be used in a general *n*-way algorithm. Such an *n*-way PARAFAC algorithm is described in procedure 6.2. The complete Matlab program is listed in appendix C.

It is assumed that $\underline{\mathbf{X}}$ is an n-way array, initially arranged as the unfolded matrix $\mathbf{X}^1$. Vectors **r** and **w** must be defined. All elements of **w** must be equal. The factors are initially set to random values. The s-function is defined in equation 6.10a and 6.10b.

1. Calculate $j=\prod_{l=1}^{n} r_l$

2. for k=1...n
   o=[(k+1) k+2 ··· n 1 ··· (k-1)]

3. for u=1..w_k
   l=o_1
   p=reshape(**fac**(s(1,u,l):s(r_l,u,l),1,r_1)
   for v=2..(n-1)
   l=o_v
   q=reshape(**fac**(s(1,u,l):s(r_l,u,l),1,r_1)
   p= **p**⊗ **q**
   next v
   $\mathbf{M}^k(u,:)$= **p**'
   next u

4. if mode k is unconstrained then
   $\mathbf{F}^k=(\mathbf{X}^k\,\mathbf{M}^{k\prime})(\mathbf{M}^k\,\mathbf{M}^{k\prime})^{-1}$,
   end
   if mode k should have non-negativity constraints then
   $\mathbf{F}^k$=CNNLS($\mathbf{X}^k$,$\mathbf{M}^k$)
   end
   if mode k should have orthogonality constraints
   $\mathbf{F}^k=\mathbf{X}^k\mathbf{M}^{k\prime}(\mathbf{M}^k\mathbf{X}^{k\prime}\mathbf{X}^k\mathbf{M}^{k\prime})^{-½}$
   end
   fac(s(1,u,k):s(r_v,u,k)))=reshape($\mathbf{F}^k$,1,$w_v r_v$)

5. if k<n then
   $\mathbf{X}^{k+1}$=reshape($\mathbf{X}^k$,r_{k+1},j/r_{k+1})
   else
   $\mathbf{X}^1$=reshape($\mathbf{X}^k$,r_1,j/r_1)
   end

6. next k

7. Repeat from step 2 until convergence of fit is reached.

Procedure 6.2: An algorithm for estimating the solution to the general n-way PARAFAC model. Temporary vectors **p** and **q** are helping to simplify the algorithm. The factors (calculated as $\mathbf{F}^k$) are stored and returned in the vector **fac**.

In step 1 the total number of elements in the data array is calculated. Step 2 makes the routine update the w factors in each of the n modes by using k as a counter. For each mode, step 3 calculates a matrix corresponding to the $\mathbf{M}^k$ matrices in table 6.3. $\mathbf{M}^k$ is used to calculate $\mathbf{F}^k$ in step 4 depending on the chosen constraints. Also, in this step the $\mathbf{F}^k$ matrix is unfolded to a vector so that it can be stored in the **fac** vector. In step 5 the $\mathbf{X}^k$ matrix is rearranged so that it

is ready to be used for updating the factor in the next way. In step 6 the updating is shifted to the next way. Step 7 evaluates the convergence criteria and if necessary the steps are repeated from step 2.

6.4 General *n*-way TUCKER modelling

The theory of how to estimate the factors for the TUCKER model is very similar to that of the PARAFAC model. The discussions of how the factors are to be computed and how constraints are put on the solutions are exactly the same. We will continue to use the **X**, **M** and **F** matrix notation, but now we change how the $\mathbf{M}^k$ matrix is constructed. With a new construction of $\mathbf{M}^k$, equations 6.1 and 6.4 and the CNNLS algorithm can be used in the exact same manner as for the PARAFAC modelling. For an introduction to the TUCKER model, see Andersson (1995). Besides the factors in $\mathbf{F}^k$ a set of interaction parameters for the factors are estimated, these are stored in the core $\underline{\mathbf{G}}$. $\underline{\mathbf{G}}$ is an *n*-way array like $\underline{\mathbf{X}}$, thus $\underline{\mathbf{G}}$ needs unfolding into $\mathbf{G}^k$. The function of $\underline{\mathbf{G}}$ is illustrated in the preliminary report p. 20. $\underline{\mathbf{G}}$ has the same number of ways as $\underline{\mathbf{X}}$. However, the number of parameters in the *k*th way is equal to the number of factors to be derived in that way, that is w_k. $\mathbf{G}^k$ is having dimensions according to equation (6.12)

$$\left(w_k \times \frac{1}{w_k} \prod_{l=1}^{n} w_l \right) \tag{6.12}$$

The introduction of the characteristic core, $\mathbf{G}^k$, in the TUCKER model is no real obstacle since it is easily built into the $\mathbf{M}^k$ matrices as shown in table 6.3.

$$\mathbf{P}^1 = \left[\mathbf{B}' \otimes \mathbf{C}' \otimes \mathbf{D}'\right] \qquad \mathbf{P}^2 = \left[\mathbf{C}' \otimes \mathbf{D}' \otimes \mathbf{A}'\right]$$

$$\mathbf{P}^3 = \left[\mathbf{D}' \otimes \mathbf{A}' \otimes \mathbf{B}'\right] \qquad \mathbf{P}^4 = \left[\mathbf{A}' \otimes \mathbf{B}' \otimes \mathbf{C}'\right]$$

$$\mathbf{M}^k = \mathbf{G}^k \mathbf{P}^k$$

Table 6.3: Example of a 4-way TUCKER model. The four $\mathbf{P}^k$ matrices contain the kronecker products. By multiplying $\mathbf{P}^k$ by the *k*th unfolding of $\underline{\mathbf{G}}$ the $\mathbf{M}^k$ matrices to be used in eq. 6.2 are obtained.

However, it is now necessary to unfold both the $\underline{\mathbf{X}}$ array and the $\underline{\mathbf{G}}$ array during calculations. The rearrangement between the successive unfoldings can be computed according to procedure 6.1 for $\underline{\mathbf{X}}$ as well as for $\underline{\mathbf{G}}$. It can easily be shown that the arrangement of the elements in $\mathbf{G}^k$ complies to the SUM principle. Observe the systematics of the kronecker terms in the $\mathbf{M}^k$ matrices in table 6.3. It is noted that the *order* of the factors of the kronecker multiplication is the same as in the PARAFAC case, but when fitting the TUCKER model the

full factor matrices **A**, **B**, **C** and **D** are used, they are not used columnwise.

$\underline{\mathbf{G}}$ is included in the $\mathbf{M}^k$ matrices in order to keep a general base of discussion. The $\mathbf{G}^k$ matrices have to be calculated using the kronecker terms, $\mathbf{P}^k$, from table (6.3). The stepwise solution for $\mathbf{G}^k$ in the *k*th unfolding is given in equation (6.13).

$$\mathbf{G}^k = (\mathbf{F}^{k\prime}\mathbf{F}^k)^{-1}\mathbf{F}^{k\prime}\mathbf{X}^k\mathbf{P}^{k\prime}(\mathbf{P}^k\mathbf{P}^{k\prime})^{-1} \tag{6.13}$$

The systematic approach for constructing the $\mathbf{M}^k$ matrices is outlined in procedure 6.3. Step 1 calculates the total number of elements in the array. Step 2 will make the iterations step through the *n* different ways for updating the factors in each way. In step 3 the kronecker product, $\mathbf{P}^k$, and the multiplicator matrices, $\mathbf{M}^k$, are formed. The appropriate least squares solution is chosen for the current way in step 4. Step 5 updates the core. Note, that the core is updated every time a factor has been adjusted. In step 6 the unfolding of both the data array $\underline{\mathbf{X}}$ and the core array $\underline{\mathbf{G}}$ are changed so they can be used to update the factors in the next way.

It is assumed that $\underline{\mathbf{X}}$ is an n-way array, initially arranged as the unfolded matrix $\mathbf{X}^1$. Vectors $\mathbf{r}$ and $\mathbf{w}$ must be defined. The factors are initially set to random values. The s-function if defined in equation 6.10a and 6.10b.

1. Calculate $i=\prod_{l=1}^{n} w_l \quad , \quad j=\prod_{l=1}^{n} r_l$

2. for k=1...n
   $\mathbf{o}$=[(k+1) (k+2) ··· n 1 ··· (k-1)]

3. $l=o_1$
   $\mathbf{P}$=reshape(**fac**(s(1,1,l):s(r_1,w_1,l)),r_1,w_1)
   for v=2...n
   $l=o_v$
   $\mathbf{Q}$=reshape(**fac**(s(1,1,l):s(r_v,w_1,l)),r_1,w_1)
   $\mathbf{P}=\mathbf{P}\otimes\mathbf{Q}$
   next v
   $\mathbf{M}^k=\mathbf{G}^k\,\mathbf{P}$

4. if mode k is unconstrained then
   $\mathbf{F}^k=(\mathbf{X}^k\,\mathbf{M}^{k\prime})(\mathbf{M}^k\,\mathbf{M}^{k\prime})^{-1}$,
   end
   if mode k should have non-negativity constraints then
   $\mathbf{F}^k$=CNNLS($\mathbf{X}^k$,$\mathbf{M}^k$)
   end
   if mode k should have orthogonality constraints
   $\mathbf{F}^k=\mathbf{X}^k\mathbf{M}^{k\prime}(\mathbf{M}^k\mathbf{X}^{k\prime}\mathbf{X}^k\mathbf{M}^{k\prime})^{-½}$
   end
   fac(s(1,1,k):s(r_v,w_k,k)))=reshape($\mathbf{F}^k$,1,$w_k r_k$)

5. $\mathbf{G}^k=(\mathbf{F}^{k\prime}\mathbf{F}^k)^{-1}\mathbf{F}^{k\prime}\mathbf{X}^k\mathbf{P}^{k\prime}(\mathbf{P}^k\mathbf{P}^{k\prime})$

6. if k<n then
   $\mathbf{X}^{k+1}$=reshape($\mathbf{X}^k$,r_{k+1},j/r_{k+1})
   $\mathbf{G}^{k+1}$=reshape($\mathbf{G}^k$,w_{k+1},i/w_{k+1})
   else
   $\mathbf{X}^1$=reshape($\mathbf{X}^k$,r_1,j/r_1)
   $\mathbf{G}^1$=reshape($\mathbf{G}^k$,w_1,i/w_1)
   end

7. next k

8. Repeat from step 2 until convergence of the factors is reached.

Procedure 6.3: An algorithm for estimating the solutions to the general n-way TUCKER model. The factors (calculated as $\mathbf{F}^k$) are stored in the vector **fac**. The core elements (calculated as $\underline{\mathbf{G}}$) are stored in $\mathbf{G}^1$. **Q** is used to simplify the algorithm.

6.5 Rotation of *n*-way TUCKER cores

As mentioned above, the rotation problem of the TUCKER model is a severe drawback when it comes to interpretation of the derived factors. Several authors have tried to circumvent this problem by different approaches. Henrion (1993) optimizes a two factor solution from a 3-way TUCKER model by determining the angle of rotation for each set of factors that gives the maximum diagonality of the transformed core. Also Kiers (1992) aims at reaching a hyperdiagonal core. The hyperdiagonality of the core facilitates easy interpretation due to the PARAFAC likeness of non-interacting factors.

However, here an alternative and flexible solution is presented for rotation of the TUCKER solutions. It aims at rotating the factors to obtain the closest possible fit to a *structure* given by the analyst. Optaining likeness in structure, not in fit, is the key point since this will allow the analyst to obtain more interpretable factors. Therefore an algorithm have been developed that can rotate the factors so that the new and rotated core has structural resemblence with one specified by the user. This will allow the user to search for hidden PARAFACN-like behaviour in the data. Insignificant off-diagonal elements in a TUCKER solution rotated to hyper diagonality could indicate that the choice of model should rather be a PARAFAC model than a TUCKER model.

The rotation of a set of components stored columnwise in a matrix, $\mathbf{A}$ $(r_1 \times w_1)$, is done by multiplying by a mapping matrix, $\mathbf{A}_M$ $(w_1 \times w_1)$ as shown in equation (6.14) .

$$\mathbf{A}_N = \mathbf{A}\mathbf{A}_M \tag{6.14}$$

$\mathbf{A}_M$ contains the coordinates of the new rotated basis expressed in terms of the old basis. The rotated factors are contained in $\mathbf{A}_N$. If $\mathbf{A}$ represents components which are mutually columnwise orthogonal and $\mathbf{A}_M$ expresses an orthogonal basis, the rotated components will retain the orthogonality. In the following I will use the 3-way case to illustrate the methodology for obtaining the mapping matrix $\mathbf{A}_M$. Since $\mathbf{A}$ contains the factors in the first way it is understood that k=1. When the operations on the $\mathbf{A}$ factors are outlined the algorithm is easily extended to include $\mathbf{B}$ and $\mathbf{C}$ as well. Finally, method is generalized to n ways.

Rotation of the initial TUCKER components, e.g. $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, to new factors, $\mathbf{A}_N$, $\mathbf{B}_N$ and $\mathbf{C}_N$, will result in changes of the interaction coefficients in the unfolded hypercore $\mathbf{G}^k$ (dimensions as in equation (6.12)). Even though not strictly correct the changed core will be referred to as have been rotated. The rotated core, in unfolded form designated by $\mathbf{G}^k{}_N$ (same dimensions as $\mathbf{G}^k$) can be obtained by using the orthonormal mapping matrices $\mathbf{A}_M$, $\mathbf{B}_M$ $(w_2 \times w_2)$ and $\mathbf{C}_M$ $(w_3 \times w_3)$ for each mode, as described in (6.15) for the 3-way case, Henrion(1994). In the following the core $\underline{\mathbf{G}}_N$ have been specified by the analayst. The structure of the rotated core can be described by setting the desired elements in $\underline{\mathbf{G}}_N$ to ones. However, it is also possible for the analyst to weigh the new core by setting the hyper diagonal elements to 10, and some off-diagonal elements to another value, e.g. 0.1. Any values can be used for the weighing since it will be the relative levels that will be used.

$$\mathbf{G}_N^1 = \mathbf{A}_M^{/}\mathbf{G}^1(\mathbf{B}_M \otimes \mathbf{C}_M) \tag{6.15}$$

Note that it is not the factors themselves, but only the mapping matrices, that are used to calculate the new core in equation (6.15). The problem is now to determine $\mathbf{A}_M$, $\mathbf{B}_M$ and $\mathbf{C}_M$ in such a way that $\mathbf{G}^1{}_N$ obtains the desired structure. From equation (6.15) the correlation matrix for $\mathbf{G}^1{}_N$ is obtained in equation (6.16).

$$\mathbf{G}_N^1\mathbf{G}_N^{1'} = \mathbf{A}_M^{/}\mathbf{G}^1(\mathbf{B}_M \otimes \mathbf{C}_M)(\mathbf{A}_M^{/}\mathbf{G}^1(\mathbf{B}_M \otimes \mathbf{C}_M))^{/} \tag{6.16}$$

This is reduced to the form in (6.17).

$$\mathbf{G}_N^1\mathbf{G}_N^{1'} = \mathbf{A}_M^{/}\mathbf{G}^1(\mathbf{B}_M \otimes \mathbf{C}_M)(\mathbf{B}_M \otimes \mathbf{C}_M)^{/}\mathbf{G}^{1/}\mathbf{A}_M \tag{6.17}$$

This equation express the problem that is under investigation. We want to determine the mapping matrix $\mathbf{A}_M$ so that $\mathbf{G}^1{}_N\mathbf{G}^1{}_N{}'$ is obtained by using the initial core and the other mapping matrices. However, in order to obtain the other mapping matrices, $\mathbf{B}_M$, $\mathbf{C}_M$ and so on, an iterative scheme has to be used. At present we will focus on how to calculate the correct $\mathbf{A}_M$ and use these equations to find the other mapping matrices. Equation (6.17) is rewritten to the form of (6.18) introducing a simplifying matrix $\mathbf{T}^1$ ($w_1 \times w_1$). $\mathbf{T}^k$ is symmetric and is required to be non-singular.

$$\mathbf{G}_N^1\mathbf{G}_N^{1'} = \mathbf{A}_M^{/}\mathbf{T}^1\mathbf{A}_M \quad , \qquad \mathbf{T}^1 = \mathbf{G}^1(\mathbf{B}_M \otimes \mathbf{C}_M)(\mathbf{B}_M \otimes \mathbf{C}_M)^{/}\mathbf{G}^{1/} \tag{6.18}$$

Equation (6.18) is recognized as being a similarity problem, see Fabricius-Bjerre (1987), page 175. The solution to such a problem requires that the singular values of the two involved matrices, $\mathbf{G}_N{}^1\mathbf{G}_N{}^{1'}$ and $\mathbf{T}^1$, are equal. This is done by decomposing $\mathbf{G}^1{}_N\mathbf{G}^1{}_N{}'$ and $\mathbf{T}^1$ using SVD as shown in equations (6.19) and (6.20). Using the normalized eigenvectors from the decomposition, $\mathbf{U}_G$ and $\mathbf{U}_T$, as done in equation (6.21) will give the mapping matrix $\mathbf{A}_M$ that will give the specified core $\underline{\mathbf{G}}_N$ in equation (6.17).

$$\mathbf{U}_G\mathbf{S}_G\mathbf{V}_G{}^{/} = \text{svd}(\mathbf{G}_N^1\mathbf{G}_N^{1'}) \tag{6.19}$$

$$\mathbf{U}_T\mathbf{S}_T\mathbf{V}_T{}^{/} = \text{svd}(\mathbf{T}^1) \tag{6.20}$$

Note that $\mathbf{U}_G$=$\mathbf{V}_G$ and $\mathbf{U}_T$=$\mathbf{V}_T$. $\mathbf{A}_M$ is found as shown in equation (6.21).

$$\mathbf{A}_M = \mathbf{U}_G\mathbf{U}_T^{-1} \tag{6.21}$$

The method just presented derives only matrix $\mathbf{A}_M$. In order to derive mappings for rotating the other factors the procedure has to be repeated with $\mathbf{G}^1$ rearranged to an appropriate unfolding. Also $\mathbf{G}^1{}_N$ has to be unfolded into a new folding so that equations (6.15)-(6.21) still applies. Since the mappings are dependent of how many factors there are in each way, and first and foremost by the value of n, the found mappings have to be stored in a long vector, **trm**. This is parallel to storing the factors in the long vector **fac** as described above. The number of elements in **trm** is given in equation (6.22).

$$\sum_{l=1}^{n}(w_l)^2 \tag{6.22}$$

The algorithm can be summerized to the following: to find the mapping matrices for rotating the factors for the kth way, one arranges the kth unfolding matrices $\mathbf{G}^k$ and $\mathbf{G}^k{}_N$, Then the proper kronecker products are calculated and $\mathbf{T}^k$ is derived. The mapping matrix for the kth way is then found by dividing the normalized score matrices from the decomposition of these matrices as done in equation (6.21). The scheme is repeated until convergende of the rotated factors are derived.

From the method outlined above, procedure 6.4 will calculate the orthonormal mapping matrices for rotating the found factors. The rotated estimates will not necessarily be more interpretable, but with the core having fewer significant elements, the interpretation is made easier.

An important note should be made concerning the dependency of the derived core. Since the estimates, factors as well as the core, are solutions to the TUCKER model they can be rotated. Due to the non-uniqueness of the factors, the estimated core need not be the same when more solutions are compared. Since the structure of the *rotated* core is dependent on the structure of the core *being* rotated, the rotation algorithm should be applied to several solutions to the same data giving an idea of how much the cores can be made to resemble each other. Currently the core-rotation, presented here as a stand-alone procedure for postprocessing the core, is being implemented into the TUCKER calculation kernel to obtain TUCKER estimates where the most significant elements are concentrated around positions specified by the analyst. If this modified algorithm can be obtained it will greatly improve on the interpretability of the TUCKER estimates without discarding the interesting feature of the TUCKER model, namely interaction of factors.

It is assumed that $\underline{\mathbf{G}}$ is an n-way array, initially arranged as the unfolded matrix $\mathbf{G}^1$. Also the core to aim at, $\underline{\mathbf{G}}_N$, must be defined as $\mathbf{G}^1{}_N$. Vector $\mathbf{w}$ must be defined. The mapping matrices are initially set to random values. The s-function if defined in equation 6.10a and 6.10b.

1. Calculate $j = \prod_{l=1}^{n} w_l$

2. for k=1...n

3. $\mathbf{o}$=[(k+1) (k+2) ··· n 1 ··· (k-1)]
   l=o_1
   P=reshape(**trm**(s(1,1,l):s(w_1,w_l,l)),w_l,w_l)
   for v=2...n
   l=o_v
   Q=reshape(**trm**(s(1,1,l):s(w_v,w_l,l)),w_l,w_l)
   P= **P**⊗ **Q**
   next v
   $\mathbf{T}^k$=$\mathbf{G}^k\mathbf{P}\mathbf{P}'\mathbf{G}^{k\prime}$

4. [$\mathbf{U}_G$ $\mathbf{S}_G$ $\mathbf{V}_G$]=svd($\mathbf{G}^k{}_N$ $\mathbf{G}^k{}_N$')
   [$\mathbf{U}_T$ $\mathbf{S}_T$ $\mathbf{V}_T$]=svd($\mathbf{T}^k$)
   Q=$\mathbf{U}_G$ $\mathbf{U}_T^{-1}$
   trm(s(1,1,k):s(w_v,w_k,k)))=reshape(**Q**,1,(w_k)^2)

5. if k<n then
   $\mathbf{G}^{k+1}$=reshape($\mathbf{G}^k$,w_{k+1},j/w_{k+1})
   $\mathbf{G}^{k+1}{}_N$=reshape($\mathbf{G}^k{}_N$,w_{k+1},j/w_{k+1})
   else
   $\mathbf{G}^1$=reshape($\mathbf{G}^k$,w_1,j/w_k)
   $\mathbf{G}^1{}_N$=reshape($\mathbf{G}^k{}_N$,w_1,j/w_k)
   end

7. next k

8. Repeat from step 2 until convergence of the rotated factors is reached.

Procedure 6.4: Rotation of factors from TUCKER models. Procedure for determining mapping matrices to obtain maximum structural similarity of the rotated core with $\underline{\mathbf{G}}_N$. The mapping matrices (calculated as **Q**) are stored in the vector **trm**.

6.6 General *n*-way PLS calibration

One of the most used algorithms for predictive purposes is the PLS algorithm, see Bro (1996) and Wold et al. (1987). Having made the PARAFAC and TUCKER algorithms for decomposing n-way arrays, the implementation of a PLS algorithm capable of handling n-way data in both $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ is easily done.

The algorithm is outlined in procedure 6.5. $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ are nx-way and ny-way structures. The dimensions of $\underline{\mathbf{X}}$ (rx_1×rx_2×···×rx_{nx}) and $\underline{\mathbf{Y}}$ (ry_1×ry_2×···×ry_{ny}) are contained in the vectors **rx** and **ry** The data are consider to be present as $\mathbf{X}^1$ and $\mathbf{Y}^1$ unfoldings with rx_1=ry_1. In order to

obtain simple expressions the number of columns in $\mathbf{X}^1$ are calculated once and for all according to equation (6.23)

$$jx = \prod_{l=2}^{nx} rx_l \tag{6.23}$$

and for $\mathbf{Y}^1$ according to equation (6.24)

$$jy = \prod_{l=2}^{ny} ry_l \tag{6.24}$$

Step 1 will make the program derive the number of desired factors, namely *f*. Step 2 initialises the score for $\mathbf{Y}^1$, **u** ($ry_1 \times 1$), to be equal to the first column in $\mathbf{Y}^1$. In step 3 the weights, **we** ($1 \times jx$), for the variables of $\mathbf{X}^1$ are calculated by projection on the score vector for $\mathbf{Y}^1$. To ensure true *n*-way linear composition of the weights, the array is decomposed using either singular value decomposition or PARAFAC decomposition. If $\underline{\mathbf{X}}$ is a matrix the weight should not be decomposed further. From the *n*-way linear solutions the weight array is reconstructed using the kronecker multiplication to give a **we** factor that can be used for scoring the objects of $\mathbf{X}^1$ directly. This is done in step 4. From the scores of $\mathbf{X}^1$ the weights of $\mathbf{Y}^1$ are determined as **q** ($jy \times 1$). The weights in **q** is used for calculating updated scores in **u**. The routine is repeated until convergence is reached, e.g. almost zero change of **t**. After convergence, the modelled part is removed and the residuals in $\mathbf{X}^1$ and $\mathbf{Y}^1$ are modelled.

It is assumed that $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ are n-way arrays, initially arranged as $\mathbf{X}^1$ and $\mathbf{Y}^1$. Their dimensions are stored in vectors **rx** and **ry**, and the number of ways are stored in nx and ny. Vector **w** must contain the number of factors to derive in each way. The s-function is defined in equation 6.10a and 6.10b. The number of PLS factors to derive is described by f.

1. For a=1..f

2. **u**=$\mathbf{Y}^1$(1,:)

3. **we**=($\mathbf{X}^1$)'**u**
   If $\underline{\mathbf{X}}$ is 2-way (nx=2) then
     normalize(**we**)
   end;
   If $\underline{\mathbf{X}}$ is 3-way (nx=3) then
     [**P S V**] = svd(reshape(**we**,rx_2,rx_1))
     we=**p**(:,1)'⊗**v**(:,1)'
     normalize(**we**)
   end;
   If $\underline{\mathbf{X}}$ is n-way (nx>3) then
     Derive a one-component solution using parafac on the unfolded matrix
       reshape(**we**,rx_2 ,$jx/(rx_1 \cdot rx_2)$), and store the solutions in **fac.**
     z=**fac**(1:rx_1)
     for l=2..nx
       v=**fac**(s(1,1,l):s(rx_l,1,l))
       normalize(**v**)
       z=**z**⊗**v**
     end
     we=**z**
     normalize(**we**)
   end;

4. **t**=($\mathbf{X}^1$)**we**

5. **q**=($\mathbf{Y}^1$)'**t**
   If $\underline{\mathbf{Y}}$ is 2-way (ny=2) then
     normalize(**q**)
   end;
   If $\underline{\mathbf{Y}}$ is 3-way (ny=3) then
     [**P S V**] = svd(reshape(**q**,ry_2,ry_1))
     q=**p**(:,1)'⊗**v**(:,1)'
     normalize(**q**)
   end

...continues on next page...

If $\underline{\mathbf{Y}}$ is n-way (ny>3) then
  Derive a one-component solution using parafac on the unfolded matrix
    reshape(**q**,ry_2 ,$jy/(ry_1 \cdot ry_2)$)), and store the solutions in **fac.**
  z=**fac**(1:ry_1)
  for l=2..ny
    v=**fac**(s(1,1,l):s(ry_1,1,l))
    normalize(**v**)
    z=**z**⊗**v**
  end
  q=**z**
  normalize(**q**)
end

6. **u**=(**Y**^{1})**q**

7. If not converged (e.g. change in **t**) repeat from step 2.

8. **T**(:,a)=**t**
   U(:,a)=**u**
   Q(:,a)=**q**
   WE(:,a)=**we**

9. **B**(1:a,a)=(**T**(:,1:a)'*****T**(:,1:a))$^{-1}$ **T**(:,1:a)**u**

10. **Y**^{1}=**Y**^{1}-**T**(:,1:a)**B**(1:a,1:a)**Q**(:,1:a)'
    X^1=**X**^{1}-**T***(**WE**)'

11. next a

Procedure 6.5: n-way PLS2. The procedure determines weights (loadings) for $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ and regression coefficients to be used for predictions.

In procedure 6.5 PARAFAC has been used to decompose $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$. When the weights from the n-way PLS algorithm have been derived, these can be used to predict the response of new samples. Prerequisit is of course, that the samples being predicted are comparable to the ones that were used during calibration. Or, put differently, that proper calibration samples have been selected.

6.7 Optimization

The n-way procedures requires much computing time in relation to conventional 2-way algorithms. Of course the computing time is very dependent of the size of data, number of factors to derive and the convergence criteria. In the Matlab procedures used here, some improvements have been done towards reducing the calculation time.

The PARAFAC algorithm have been extended by a linear extrapolation sub-procedure which uses the 2 previous values of each element in the factors to extrapolate new, and most often, better estimates. The efficiency of the extrapolation depends strongly on the data, and

the stage of solution. In the first iterations of the PARAFAC procedure, big changes occur in the factors. As the iterations proceed the factors come closer and closer to the correct solutions. This makes it possible to extrapolate, sometimes up to 50 iterations, saving lots of time. Each successful extrapolation will bring the factors nearer the correct solution. Hence, a little improvement in the start of the iterations will be a good basis for even more extrapolations later. In general this feature makes the extrapolation a very efficient optimization scheme. Besides, there is only very little overhead connected with calculating the extrapolated factors.

Using constraints may have heavy impact on the computation time for obtaining the factors. Up to 200 times longer iteration time has been experienced with data sets when using NNC in comparison to finding un-constrained estimates. To circumvent the problem, a data compression algorithm have been developed and used. The algorithm divides data into a number of sub-arrays. For each array an average spectrum is calculated. The average spectra are reshaped to form a much lesser array - although containing almost the same information as the uncompressed array. This method, which does not use SVD, will allow derivation of NNC factors since the level of the data will remain almost the same. After the solutions to the reduced array have been found, these factors are used as initial guess for estimating the model of the full array.

Another, and more direct optimization in a classical sense, is the compilation of Matlab code to run independently of the Matlab Command Window. This is done by translating the Matlab code into C-code. The C-code can then be compiled to give faster programs, however still accessible from within Matlab. For this purpose the Matlab Compiler Toolbox have been used. In some cases the Matlab-to-C compiler from MathWorks did not succeed in obtaining the fastest possible code. In these cases completely Matlab-independent C code was programmed with high gain in computational speed. The routine for kronecker multiplication and the routine used for reshaping matrices have been made this way. The routine for finding non-negativity least squares solutions was compiled by the use of the Toolbox to give a small increase in speed.

Not all optimization is concerned with increasing the speed of which the estimates are found. Also the stability of the algorithms must be optimized. For this purpose the PARAFAC algorithm normalizes the factors in all ways but the first. In order to model the levels of the data array being modelled, at least one set of factors must be able to vary freely. This is not the case for the implementation of the TUCKER solution, were all estimates in all ways are normalized since the core elements can contain the levels. Since a set of estimates in one way are updated using matrices comprised of estimates in all other ways, it is necessary to ensure that the matrices are non-singular. If the lengths of two factors in the same way differ by several magnitudes, the matrix may appear almost singular since the shortest column may appear to be 0 in the presence of the large column. By normalizing all factors this will not happen since the lengths of all columns will be one. Still, deriving more factors than actually present in the data may give factors that are linear dependent. This can cause the algorithm to fail.

Finally, it should be noted that the convergence criteria must be selected carefully. Too hard a criteria will require too long computation time and too soft convergence criteria may

result in finding local minima, see Mitchell and Burdick (1994).

7. Results from *n*-way analysis and calibration

In the following the developed algorithms will be used for calibration purposes using the 4-way data from the measurements described in chapter 4.4. To investigate on the possible stabilizing effects of using multi-way structures, the 4-way array was unfolded into 3-way and 2-way arrays. The predictions from 2-way analyses are compared to the predictions based on 3- and 4-way arrays. The models will briefly be used to decompose the 4-way arrays to explore the data and examplify the proposed algorithm for core rotation.

A calibration model establishes a relation between a set of *independent variables* and a set of *dependent variables*. In this context the independent variables are the wavelengths used to measured the fluorescence intensities and the dependent variables are the concentrations of the chemical species. We will refer to the array containing the independent variables as $\underline{\mathbf{X}}$ (fluorescence intensities) and the array with the dependent variables as $\underline{\mathbf{Y}}$ (concentrations). In order to establish the models, a data set with known independent and dependent variables must be used to calibrate the parameters of the model. Data used for calibrating the models are referred to as *calibration data*. In contrast, the samples being used to verify the accuracy of the model are termed *validation data* or the *test set*. The test set is excluded from calibration, so that it can provide objective estimates of the future predictive accuracies of the models.

In the following discussion 3 approaches have been taken for establishing predictive models; these are PARAFAC, TUCKER and PLS1 models. The factors in the way of the $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ arrays representing the samples will be referred to as *scores*. The factor estimates in the other ways will generally be referred to as *loadings* in line with the established terminology in chemometrics. Between the scores of $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ a set of regression coefficients are calculated. In the case of predicting a new sample, its scores are calculated by using the predetermined loadings and the chosen model. Finally, the values of the dependent variables can be predicted by using the score and the predetermined regression coefficients. The approach taken here is to obtain maximum predictive accuracy with the models used, therefore only PLS1 models are discussed, that is, making models for each $\underline{\mathbf{Y}}$-variable independently of the others. It should be noted, that since the PARAFAC and TUCKER algorithms decompose $\underline{\mathbf{X}}$ with no consideration to $\underline{\mathbf{Y}}$ there is no gain by modelling each variable in $\underline{\mathbf{Y}}$ individually. However the number of factors to be used may differ. Thus, all $\underline{\mathbf{Y}}$-variables are modelled individually with both PARAFAC, TUCKER and PLS1 models.

The $\underline{\mathbf{X}}$ array consists of 18 samples, each with a 28 by 64 excitation-emission landscape measured at 3 different pH levels. Note that the emission spectra have been reduced from 256 to 64 diodes by cutting off non-significant diode readouts. The remaining part of the emission spectra covers the range from approximately 306 nm - 512 nm. The result is a 4-way array $\underline{\mathbf{X}}$ with dimensions (18×3×28×64). The sequence of the latter 3 dimensions is an arbitrary result from the arrangement in Matlab. To enable a comparison between 2-way, 3-way and 4-way analysis, the 4-way array is by unfolding arranged into 3-way (18×84×64), (18×28×192), (18×3×1792) and 2-way (18×5376) arrays. Since the $\underline{\mathbf{Y}}$ array undertakes the form of a matrix, the concentrations will be held in the matrix $\mathbf{Y}$ (18×6), implying the presence of 18 samples with concentrations of 6 possible analytes.

The predictive accuracies of the models will be assessed by evaluating the error when model-independent test set objects are predicted. It should be noted that two chemical species, DOPA and TRYP, are excluded from calibration. This is done with the intention to examine the possible effects of having unknown analytes, or interferences, in the samples to be predicted. DOPA and TRYP are both very well known from previous investigations, hence, experience was sought with the other components in the model system. Near future chromatographic analyses will bring about a very similar situation, where concentrations of a few known analytes will be estimated by using calibration algorithms on thick juice.

The true concentrations of the 6 components are shown in table 7.1. For the sake of convenience, the values are repeated from table 4.4 to facilitate easier comparison between the model predictions and the true values. The test set membership of the samples are indicated by asterisks. Note that there are 8 calibration samples and 10 validation samples.

	Concentration [M · 10^6]					
#	CATE	DOPA	HQUI	INDO	PHEN	TRYP
1	300					
2*		180				
3			20.0			
4				17.5		
5					600	
6*						2.0
7*	300	60.0				
8	300			16.0		
9*		80.0			600	
10*		50.0				1.8
11	250		25.0		600	
12	100		30.0		400	
13			20.0	17.5	800	
14*	400	70.0		16.0		1.8
15*		50.0	20.0	18.0	900	2.0
16*		70.0	30.0	16.0	900	1.6
17*	300	60.0	30.0	14.0	800	1.4
18*	400	80.0	30.0	16.0	800	1.8

Table 7.1: True concentrations of the six species in the 18 samples. The 10 test set samples (*) are used to validate the results from the calibration models. DOPA and TRYP will not be predictable since they are excluded from calibration.

For each model it will be necessary to estimate the correct number of factors to use for modelling. The number of factors to be used in the calibration models will be found by using full cross validation (CV) on the calibration data. During full CV each sample will, in turn, be left out from calibration. After the model has been estimated the left-out sample will be predicted and the error calculated. The error of the models are found by evaluating the root mean square error of cross validation (RMSECV). RMSECV is a single parameter estimate of the predictive accuracy of the model. RMSECV is found according to equation (7.1), where I designates the number of objects to predict during cross validation, i.e. I=8 (8 calibration samples) and J designates the number of variables to predict, i.e. J=1.

$$\text{RMSECV} = \sqrt{\frac{1}{I \cdot J}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(y^{pred}_{i,j} - y^{true}_{i,j}\right)^2} \tag{7.1}$$

Full CV is performed for 1 to 8 factors and the lowest possible number of factors giving a sufficiently low RMSECV will be used. When the number of factors has been found, a new model including all the calibration objects is made. From this model the objects belonging to the test set are predicted and the errors are inspected sample-wise.

The issue of data preprocessing, e.g. centring and scaling, becomes rather complicated when turning from 2-way to n-way analysis in general, see Henrion (1994) and Kroonenberg (1983) chapter 6.5. Since the variables are measured on the same axis (apart from the different sensitivities of the diodes in the DA) no scaling is necessary. The applied calibration models are estimated for one variable at a time in $\mathbf{Y}$, hence scaling is obsolete for both $\underline{\mathbf{X}}$ and $\mathbf{Y}$. As can be seen from table 7.1 the ratio between the maximum concentrations of TRYP and PHEN is approximately 500, so weighing would be necessary if all variables were to be modelled simultaneously as in PLS2. During regression and prediction the arrays have been mean centred to avoid offsets in the regression coefficients. Some of the predicted concentrations are negative and they should be regarded as expressing zero concentration levels. However, in order to facilitate a discussion of interactions between the chemical species the predictions are listed as they are estimated from the models.

7.1 2-way analysis

The comparison of methods begins with 2-way principal component regression (PCR) and PLS1 regression. The $\underline{\mathbf{X}}$ array (8×3×28×64) is unfolded to a 2-way structure with dimensions (8×5376).

7.1.1 2-way PCR

PCR was applied to the unfolded data to investigate the predictive accuracy, see Andersson (1995). After CV using the 8 calibration samples the RMSECV parameters were calculated for 1 to 8 factors for all 4 **Y** variables, and for all 4 models the lowest values were obtained by using 6 factors. The predictions are listed in table 7.2.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	357	(-)	8.3	(-)	-6.6	(-)	-205	(-)
6	-100	(-)	3.1	(-)	4.9	(-)	15	(-)
7	354	(18.0)	4.9	(-)	-1.3	(-)	40	(-)
9	95	(-)	2.7	(-)	0.2	(-)	501	(16.4)
10	82	(-)	1.8	(-)	2.4	(-)	-124	(-)
14	511	(27.9)	6.7	(-)	14.9	(7.1)	-73	(-)
15	6	(-)	23.4	(17.0)	18.6	(3.4)	889	(1.2)
16	62	(-)	33.8	(12.6)	13.5	(15.4)	859	(4.5)
17	345	(15.0)	35.0	(16.6)	12.4	(11.1)	559	(30.1)
18	579	(44.7)	37.4	(24.8)	13.7	(14.4)	621	(22.4)
PC's	6		6		6		6	

Table 7.2: Predictions of the test set using 2-way PCR. 6 factors were used in all 4 models.

The largest errors are observed for the very composite samples, i. e. 14 to 18, where 4 to 6 fluorophores are present simultaneously.

7.1.2 2-way PLS1 calibration

In addition to PCR calibration PLS1 was applied to the unfolded data. For PLS the number of factors giving the lowest RMSECV was obtained by using 6 factors in the 4 models. The predicted concentrations are listed in table 7.3.

Little improvement is observed in predictive accuracy when using PLS in comparison to PCR.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	358	(-)	8.1	(-)	-6.7	(-)	-196	(-)
6	-102	(-)	3.5	(-)	5.2	(-)	-3	(-)
7	355	(18.2)	4.8	(-)	-1.3	(-)	45	(-)
9	96	(-)	2.6	(-)	0.1	(-)	506	(15.6)
10	80	(-)	2.2	(-)	2.6	(-)	-142	(-)
14	510	(27.5)	7.0	(-)	15.0	(6.4)	-84	(-)
15	7	(-)	23.2	(16.1)	18.4	(2.4)	902	(0.2)
16	63	(-)	33.7	(12.4)	13.4	(16.0)	865	(3.9)
17	344	(14.7)	35.1	(17.1)	12.5	(10.7)	553	(30.8)
18	580	(44.9)	37.4	(24.7)	13.6	(15.1)	628	(21.6)
PC's	6		6		6		6	

Table 7.3: Predictions of the test set using 2-way PLS1 calibration. As with PCR, 6 factors were used in all 4 models.

There is no significant difference between the accuracies of the predictions using PCR compared to using PLS regression. The relative errors on the predictions of the most composite samples are generally high, but taking into account that only 8 samples, each containing no more than 3 samples at a time, have been used for calibration, the relative errors are acceptable.

7.2 3-way analysis

Entering the analysis of 3-way structures a very powerful tool becomes available. This is the PARAFAC model, which, under appropriate conditions, is capable of providing estimates of the pure underlying spectra of the samples. However, since the key issue is calibration, the resolution potential of the PARAFAC model will not be discussed for the 3-way case. Instead, the application of this more explorative approach will be postponed to the 4-way analysis.

The 3-way structures are obtained by appending the EEM's along the excitation, emission and pH axes, thereby obtaining structures with dimensions (18×3·28×64), (18×28×3·64) and (18×3×28·64). Each of the unfoldings require different numbers of parameters to be calibrated. An interesting issue is then to investigate if the models with the most parameters also yields the best predictive accuracies. Since only loadings are used to predict future samples, the number of loading parameters will be referred to when discussing the number of calibration parameters. In order to keep the discussion focused on the calibration models, the effects of having different unfoldings will be limited to the PARAFAC case. Investigations using the PLS and TUCKER models have been conducted with all 3 types

of unfoldings, however, in order to maintain a clear view of the applied calibration models, the unfoldings resulting in the best predictions of the calibration data have been chosen with no further discussion.

The discussion of using pure spectra as loadings is closely connected to the application of the PARAFAC model. The pure-spectra approach is inherent in the PARAFAC-model since the estimated loadings resemble the pure spectra if the correct number of spectra are estimated.

7.2.1 3-Way PARAFAC calibration

The PARAFAC model is applied to the 3 different unfoldings as discussed earlier. With all 3 unfoldings it was found that 5 factors should be used in all the models to give the lowest RMSECV values.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	428	(-)	5.2	(-)	-6.0	(-)	-289	(-)
6	-59	(-)	2.2	(-)	5.2	(-)	-40	(-)
7	342	(13.9)	4.6	(-)	-1.3	(-)	63	(-)
9	97	(-)	2.6	(-)	0.3	(-)	503	(16.2)
10	78	(-)	1.6	(-)	2.4	(-)	-123	(-)
14	676	(68.9)	1.8	(-)	16.0	(0.2)	-284	(-)
15	216	(-)	18.5	(7.6)	20.3	(12.6)	619	(31.2)
16	240	(-)	29.3	(2.3)	15.0	(6.4)	631	(29.8)
17	423	(40.8)	32.7	(9.1)	13.0	(7.0)	462	(42.3)
18	793	(98.2)	31.5	(5.1)	15.3	(4.3)	347	(56.7)
PC's	5		5		5		5	

Table 7.4: Predictions of the test set using 3-way PARAFAC models calibrated on the (8×3·28×64) array. 5 factors were used in all 4 models.

In table 7.4 the predicted concentrations are listed when using a 5 factor PARAFAC on the unfolded 4-way data. The unfolding used in table 7.4 is obtained by appending the 3 pH landscapes for each sample along the excitation axis. This unfolding uses a total number of (5·(3·28)+5·64=) 740 parameters to predict the test set.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	404	(-)	8.3	(-)	-5.7	(-)	-288	(-)
6	-915	(-)	4.6	(-)	5.5	(-)	-15	(-)
7	327	(9.1)	5.9	(-)	-1.2	(-)	73	(-)
9	100	(-)	2.4	(-)	0.4	(-)	497	(17.2)
10	40	(-)	4.7	(-)	2.7	(-)	-93	(-)
14	645	(61.3)	4.8	(-)	16.3	(1.7)	-276	(-)
15	210	(-)	19.8	(1.2)	20.4	(13.4)	600	(33.3)
16	227	(-)	31.3	(4.5)	15.2	(5.2)	621	(31.0)
17	312	(3.9)	42.0	(40.0)	13.7	(2.4)	555	(30.6)
18	770	(92.5)	34.2	(14.1)	15.5	(3.1)	340	(57.5)
PC's	5		5		5		5	

Table 7.5: Predictions of the test set using 3-way PARAFAC models calibrated on the (8×28×3·64) array. 5 factors were used in all 4 models.

The unfolding used in table 7.5 is obtained by appending the 3 pH landscapes for each sample along the emission axis. This unfolding uses, in the 5 factor case, a total number of (5·28+5·(3·64)=) 1100 parameters to model the $\underline{\mathbf{X}}$ array.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	440	(-)	5.5	(-)	-6.1	(-)	-317	(-)
6	-67	(-)	2.2	(-)	5.3	(-)	-23	(-)
7	349	(16.4)	4.5	(-)	-1.4	(-)	47	(-)
9	105	(-)	2.4	(-)	0.3	(-)	488	(18.6)
10	69	(-)	1.9	(-)	2.4	(-)	-103	(-)
14	677	(69.1)	2.0	(-)	16.1	(0.4)	-288	(-)
15	220	(-)	18.6	(7.1)	20.3	(12.7)	607	(32.6)
16	247	(-)	29.4	(2.1)	15.0	(6.6)	613	(31.9)
17	416	(38.8)	33.1	(10.4)	13.1	(6.7)	469	(41.4)
18	801	(100.3)	31.8	(5.9)	15.3	(4.6)	323	(59.6)
PC's	5		5		5		5	

Table 7.6: Predictions of the test set using 3-way PARAFAC models calibrated on the (8×3×28·64) array. 5 factors were used in all 4 models.

The unfolding used in table 7.6 is obtained by appending the 28 emission spectra for each sample along the excitation axis. This unfolding uses, in the 5 factor case, a total number of (5·3+5(28·64)=) 8975 parameters to predict the test set.

Unfolding the 4-way array into 3 different 3-way arrays has some interesting effects on the resulting predictions. Primarily it is experienced, see tables 7.4 to 7.6, that the models making use of 8975 parameters does no result in more accurate predictions than models based on 740 and 1100 parameters. This must be regarded as a token of the redundancy of the fluorescent behaviour in the 3 ways since using almost 10 times as many parameters does not improve the predictions. It is also a confimation of the latent linearity of the ways spanning the data. The models making use of 1100 parameters gained no significant overall predictive accuracy over the models using 740 parameters. CATE was predicted significantly better with the (8×28×3·64) unfolding than with any of the 2 other unfoldings. The predictions of HQUI and INDO are seen to be dependent on the unfolding being most accurate when using the (8×3·28×64) unfolding. The predictions from the (8×3×28·64) are the most inaccurate for all 4 species. This may be due to the badly spanning of the array using only 3 observations (pH-levels) in the second way.

7.2.2 3-way TUCKER calibration

When using the PARAFAC model estimates of the pure spectra are obtained, thereby ensuring some covariance between the scores of $\underline{\mathbf{X}}$ and the values in $\mathbf{Y}$. In PLS the aim of the estimation of loadings is, in some sense, to maximize the covariance between the resulting scores of $\underline{\mathbf{X}}$ and $\mathbf{Y}$. But, with the TUCKER models, the estimated scores are not forced in any way to correlate to the columns of $\mathbf{Y}$. I will try to circumvent the problem by using the columns of $\mathbf{Y}$, added 10 % random noise to avoid singular matrices, as initial guesses for the scores and by fixing the scores during the first 5 iterations. Also, up-weighing factor combinations giving scores with high correlations to $\mathbf{Y}$ was implemented, the weights being proportional to the corresponding correlation factor. The weighing method that was finally implemented increased the weights from 1 to 5. Other implementations have been used, but the one mentioned gave the smoothest iterative runs. Furthermore, it has been experienced that constraining the factor estimates to be non-negative gives generally significantly higher correlation between the scores of $\underline{\mathbf{X}}$ and the column in $\mathbf{Y}$. Using NNC eliminates a little of the rotation problem to give more 'true' score vectors for use in the regression. Since all manipulation is done during the calibration, the test set will still offer an objective base for evaluation of the final model.

The unfolding of the 4-way array giving the most accurate predictions was the (18×3·28×64) unfolding for all 4 analytes. The calibration was initiated by extracting the same number of factor estimates in all ways using weighing of the core elements in accordance with the correlation with $\mathbf{Y}$ as mentioned above. The factor estimates and the full core were then used to predict the 4 analytes as shown in table 7.7.

#	Predicted conc. [M·10^6] (Rel. error [%])							
	CATE		HQUI		INDO		PHEN	
2	353	(-)	6.2	(-)	-5.9	(-)	-202	(-)
6	-93	(-)	2.3	(-)	5.3	(-)	1	(-)
7	357	(19.0)	4.8	(-)	-1.3	(-)	36	(-)
9	88	(-)	2.4	(-)	0.3	(-)	511	(14.9)
10	110	(-)	2.0	(-)	2.5	(-)	-169	(-)
14	482	(20.5)	2.5	(-)	16.1	(0.9)	-38	(-)
15	-24	(-)	18.6	(6.9)	20.3	(12.7)	928	(3.1)
16	34	(-)	29.6	(1.2)	15.0	(6.2)	895	(0.5)
17	357	(18.8)	33.3	(10.9)	13.1	(6.7)	545	(31.9)
18	552	(38.0)	32.3	(7.5)	15.4	(4.0)	656	(18.1)
PC's	6-6-6		5-5-5		5-5-5		6-6-6	

Table 7.7: Predictions of the test set using 3-way TUCKER models with the same number of factors in all 3 ways using the full core. The models were calibrated on the (8×3·28×64) array.

The results from the 3-way TUCKER model predictions are significantly better than the 3-way PARAFAC predictions for CATE and PHEN. For the 5-factor models ($5\cdot3\cdot28+5\cdot64+5^3=$) 865 parameters were used to predict the test set. In the 6-factor case ($6\cdot3\cdot28+6\cdot64+6^3=$) 1104 parameters were used. Appearantly, the introduction of the core elements, letting the factors interact in the modelling of the calibration data, improves the prediction accuracy.

In order to take advantage of the special features of the TUCKER model, i.e. different number of factors in each way and interaction of factors, the cores from the models used in table 7.7 were investigated for simplification. The motivation for simplifying the core is that the risk of overfitting the calibration data is more pronounced when using TUCKER models than with PARAFAC and PLS models due to the presence of the core. Thus, the 20 core elements with the highest correlation to **Y** were kept and all other elements were set to 0. The simplified core were used to recalibrate a set of regression coefficients using only the calibration data. The resulting predictions of the test set are shown in table 7.8.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	355	(-)	6.3	(-)	-0.7	(-)	-218	(-)
6	-93	(-)	2.3	(-)	5.8	(-)	1	(-)
7	359	(19.8)	4.8	(-)	-1.4	(-)	39	(-)
9	88	(-)	2.4	(-)	0.3	(-)	562	(6.4)
10	111	(-)	2.0	(-)	2.8	(-)	-184	(-)
14	486	(21.6)	2.5	(-)	17.8	(11.5)	-41	(-)
15	-24	(-)	18.5	(7.3)	22.0	(22.3)	1102	(22.4)
16	34	(-)	29.7	(0.9)	16.8	(4.7)	975	(8.3)
17	357	(19.1)	33.9	(12.9)	14.3	(2.3)	601	(24.9)
18	556	(39.0)	31.4	(4.7)	16.7	(4.6)	725	(9.4)
PC's	6-6-6		5-5-5		5-5-5		6-6-6	

Table 7.8: Predictions of the test set using 3-way TUCKER models with the same number of factors in all 3 ways and the simplified core with only 20 elements.

There is no significant loss of predictive accuracy when reducing the number of core elements from (6^3=) 216 and (5^3=) 125 to 20 using the core elements that represents the factor combinations with the highest correlation to **Y**. This experience shows that the TUCKER models discussed will have almost the same predictive accuracies by using a much lesser number of factors. Thus, the possibility of using fewer factors, i.e. smaller TUCKER models, must be examined.

A possible reduction of the number of factors were investigated by inspecting the elements of the simplified cores belonging to each of the 4 models. The 20 elements of the core from the CATE model made use of 4 factors in the first way, 2 factors in the second and 2 in the third way. The HQUI model made use of 2, 3 and 3 factors, whilst the INDO model used 3, 3 and 3 factors. The model to predict the concentrations of PHEN made use of 2, 3 and 3 factors. These number of factors indicates what the minimum number of factors should be to obtain the predictions listed in table 7.8. From these experiences 4 new models were calibrated using the number of factors just described. Still core simplification was used, allowing only 20 core elements in the HQUI model with the criteria that they should be the ones giving the most correlated scores to the column in **Y**. The predictions are listed in table 7.9.

#	Predicted conc. [M·10^6] (Rel. error [%])							
	CATE		HQUI		INDO		PHEN	
2	288	(-)	-20.4	(-)	-8.2	(-)	-173	(-)
6	36	(-)	-10.4	(-)	6.3	(-)	-29	(-)
7	189	(37.1)	9.0	(-)	-2.4	(-)	2	(-)
9	96	(-)	15.9	(-)	0.0	(-)	483	(19.6)
10	118	(-)	-46.3	(-)	3.3	(-)	-112	(-)
14	402	(0.5)	-4.2	(-)	14.7	(7.8)	57	(-)
15	251	(-)	23.3	(16.6)	18.7	(3.8)	868	(3.6)
16	315	(-)	27.0	(10.1)	14.0	(12.5)	738	(18.0)
17	369	(22.8)	31.9	(6.4)	13.8	(1.8)	819	(2.4)
18	565	(41.3)	27.1	(9.6)	14.0	(12.3)	738	(7.7)
PC's	4-2-2		2-3-3		3-3-3		2-3-3	

Table 7.9: Predictions of the test set using 3-way TUCKER models with different numbers of factors in the 3 ways using a simplified core with maximum 20 elements.

The reduction of the number of factors has only little effect of the accuracies of the models, see tables 7.7 and 7.8. In general it must be concluded that the models used for the predictions in tables 7.7 and 7.8 are making use of too many factors and core elements when almost the same accuracies can be obtained using much less numbers of parameters as done in table 7.9.

Also TUCKER models with 1 factor in the sample-way and 1 to 6 factors in the other ways were applied. However, the predictions were significantly poorer for the most composite samples, e.g. numbers 14 to 18. Hence, the predictions are discarded.

7.2.3 3-way PLS1 calibration

As in the case of the 3-way PARAFAC, the data used for PLS calibration can be arranged to 3 different structures. The unfolding giving the lowest RMSECV was found to be the (8×3·28×64) unfolding for all 4 species. The numbers of factors giving the lowest RMSECV values are listed in table 7.10 with the predictions.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	246	(-)	1.9	(-)	-5.4	(-)	-239	(-)
6	-46	(-)	1.6	(-)	5.0	(-)	-252	(-)
7	288	(4.0)	2.8	(-)	-1.1	(-)	75	(-)
9	63	(-)	1.2	(-)	0.6	(-)	530	(11.6)
10	43	(-)	-0.4	(-)	2.5	(-)	-482	(-)
14	587	(46.8)	1.0	(-)	16.0	(0.1)	-318	(-)
15	145	(-)	19.3	(3.4)	20.4	(13.1)	958	(6.4)
16	131	(-)	29.1	(3.0)	15.2	(4.9)	880	(2.2)
17	387	(28.9)	32.7	(9.1)	13.0	(7.3)	455	(43.2)
18	653	(63.2)	31.1	(3.6)	15.6	(2.5)	605	(24.4)
PC's	4		5		5		6	

Table 7.10: Predictions of the test set using 3-way PLS1 models calibrated on the (8×3·28×64) array.

The predictions from the 3-way PLS1 models are found to be significantly poorer than those of the 3-way TUCKER models. In fact, the predictions are also significantly poorer than the 2-way PLS1 predictions. This may be due to the fact that the PLS-model adapts to the almost pure spectra in the calibration data, resulting in poorer predictions of the more composite samples in the test set.

7.3 4-way analysis

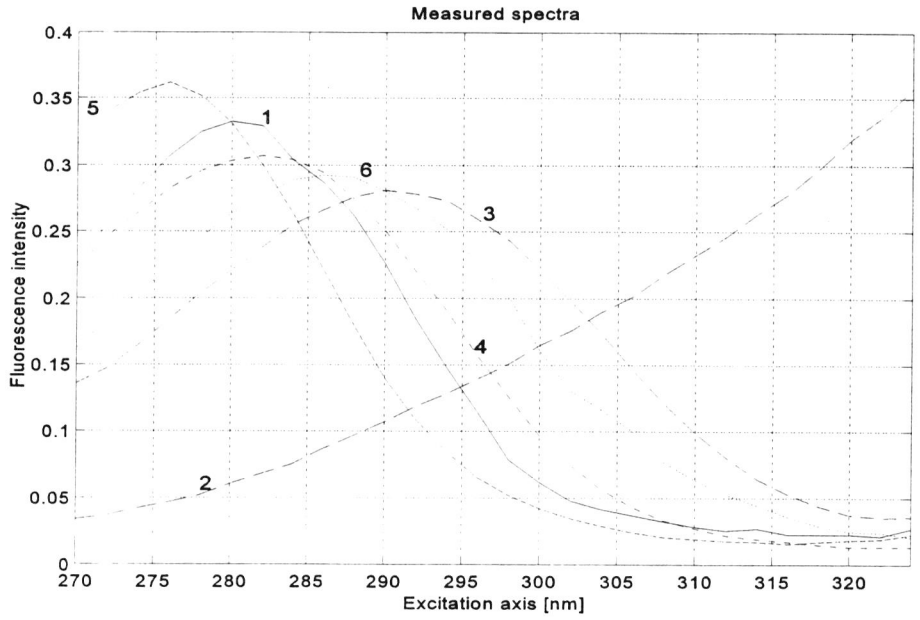
The application of 3-way, and higher, analysis algorithms facilitates exploration of data. The uniqueness of the PARAFAC estimates allows, ideally, to obtain the pure underlying spectra of composite samples, Kruskal (1989). This approach will be taken to allow for a discussion of the effects of resemblance among the pure spectra. In order to improve on the interpretability of the estimates NNC will be applied to the pure spectra, which *we know* in advance must be positive. Even though the TUCKER estimates pose problems due to rotational abilities, the developed core rotation algorithm will be applied to the estimates of a 4-way TUCKER model in a more exploratory approach.

7.3.1 4-way PARAFAC decomposition

As a supplement to the discussion of the possible reasons of the errorneous predictions, the true underlying spectra of samples 7 to 18 are resolved using the PARAFAC model with NNC. Hence, the investigated array has dimensions (12×3×28×6). This short discussion should be regarded as an explorative approach allowing us to experience *why* the predictions fail.

From the 12 composite samples in the $\underline{\mathbf{X}}$ array 6 factors were estimated using 4-way PARAFAC with NNC. The number of factors to extract was established by extracting a different number of factors and evaluating on the variance of the residual in $\underline{\mathbf{X}}$. When extracting from 3 to 7 factors the relative variance of the residual $\underline{\mathbf{X}}$ decreased as follows: 22.1%, 16.8 %, 8.7 %, 2.9 % and 1.6 %. This matches the fact that there actually is 6 components in the array. The estimates along the excitation-way, the emission-way and the pH-way are shown on figure 7.1 to 7.3 together with the *measured* pure spectra. Curve annotations are: 1. CATE, 2. DOPA, 3. HQUI, 4. INDO, 5. PHEN and 6. TRYP.

A



B

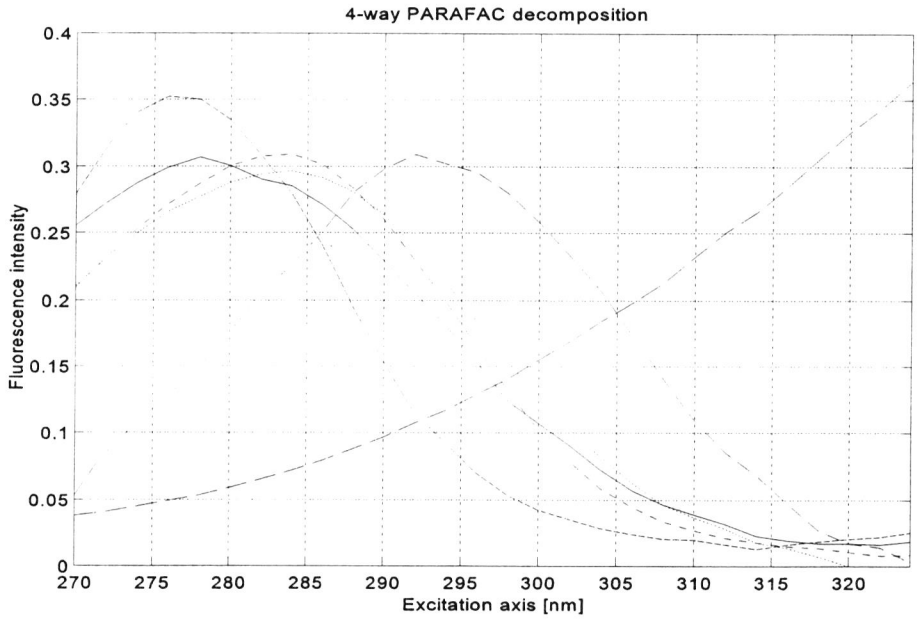
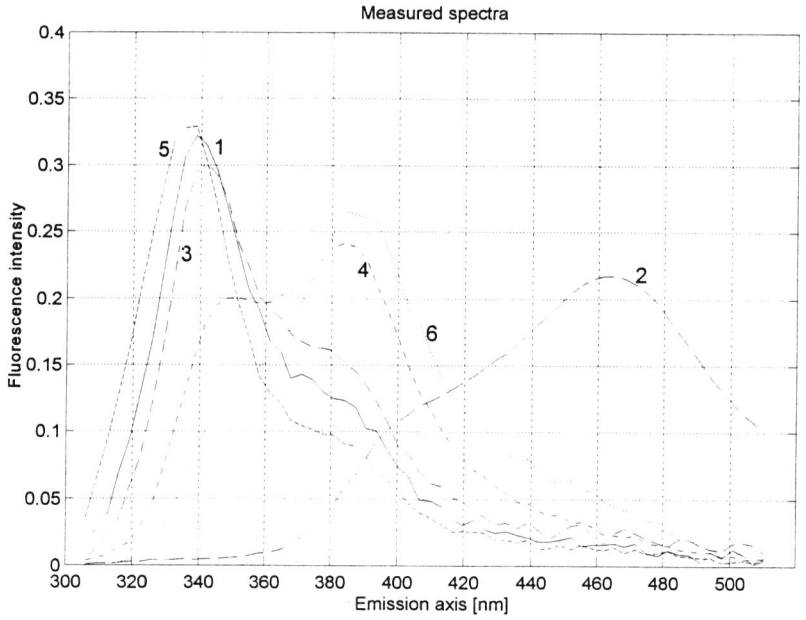


Figure 7.1 : Excitation spectra of the 6 species. A. Measured pure spectra. B. Resolved pure spectra using 4-way PARAFAC with NNC.

Note the similarity of the pure spectra of CATE (1) and TRYP (6). Discussion will be postponed until after the other pure spectra have been estimated.

A



B

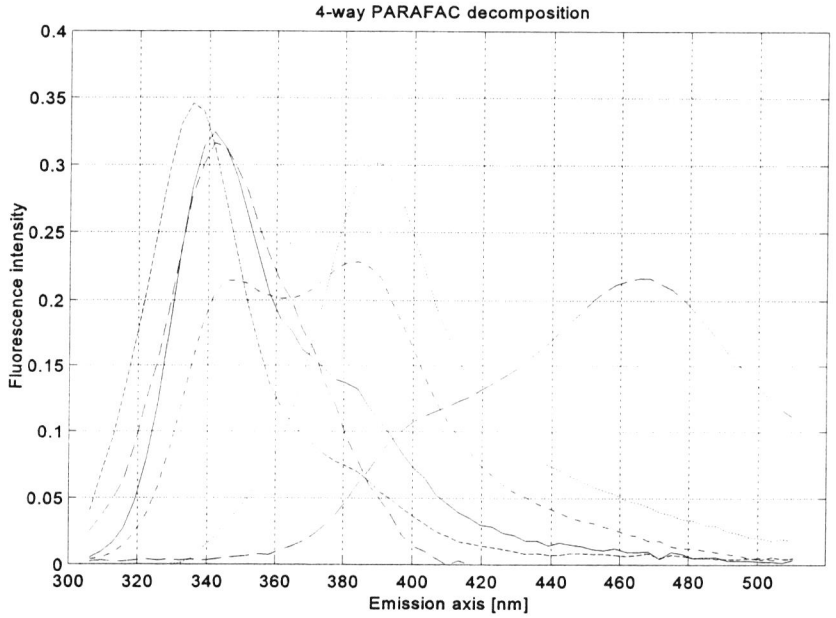
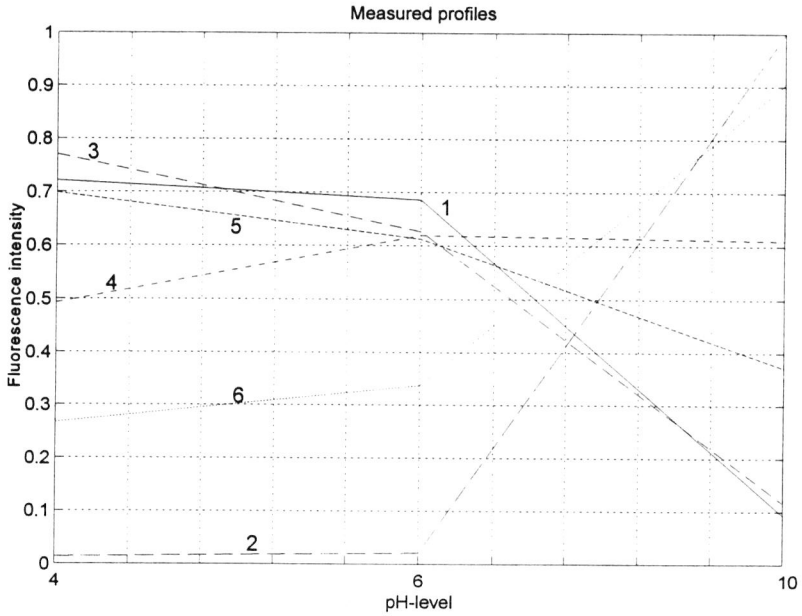


Figure 7.2 : Emission spectra of the 6 species. A. Measured pure spectra. B. Resolved pure spectra using 4-way PARAFAC with NNC.

There is some resemblance between the resolved emission spectra and the measured spectra. As with the resolved excitation spectra there is some ambiguety as to what species the reolved spectra represents. Note the resemblance between spectra number 1, 3 and 5 in fig. 7.2A.

A



B

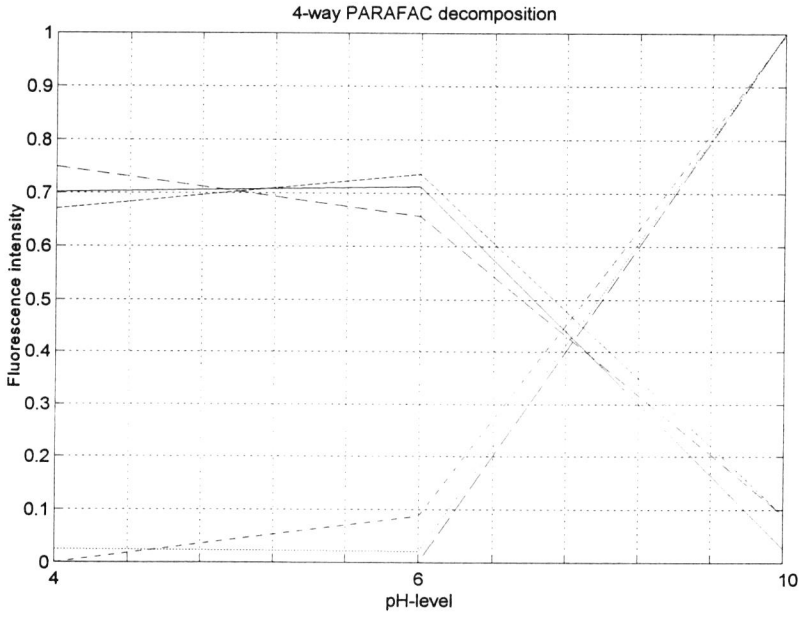


Figure 7.3 : pH-level profiles of the 6 species. A. Measured profiles. B. Resolved profiles using 4-way PARAFAC with NNC.

As can be seen from fig. 7.3 there is almost no change in intensity when comparing pH levels 4 and 6 due to the equal levels of these pH values. Species number 2 and 6, DOPA and CATE increase in fluorescence intensity in response to the increase in pH, whereas the opposite is observed for the other analytes. Future analyses should make use of more pH

levels with a broader span, say pH 2, 4, 8, 10 and 12.

Correlation factors between the estimated factors and the measured pure factors were calculated to evaluate on the degree of resemblance. All possible combinations of measured and estimated pure spectra were used to calculate correlation factors for the excitation and the emission spectra. For both of the ways 6 correlation factors were significantly higher than the 30 others. This indicates that it was possible, mathematically/automatically, to pair the *estimated* pure factor with *measured* pure factor. The lowest of the correlation coefficients was 0.963. The compared factor estimates are shown in figures 7.1A,B and 7.2A,B. Stabilizing effects on the calibration accuracies are observed when using multi-way data. As can be seen from the excitation and the emission spectra in figs. 7.1A and 7.2A there are spectral similarities among the analytes, but, since the spectral similarity between the same chemical species does not occur on both excitation and emission spectra, a stabilizing effect is most likely obtained by letting the excitation and emission ways supplement each other. The pH-way is also contributing to this stabilizing effect.

7.3.2 4-way PARAFAC calibration

The PARAFAC model has been used for predictive purposes using 4-way data. The predictions are listed in table 7.11. The RMSECV values suggested to use 5 factors for in all 4 models.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	409	(-)	7.1	(-)	-6.0	(-)	-280	(-)
6	-56	(-)	1.4	(-)	5.2	(-)	-34	(-)
7	339	(12.9)	5.0	(-)	-1.3	(-)	62	(-)
9	96	(-)	2.8	(-)	0.3	(-)	499	(16.9)
10	73	(-)	1.6	(-)	2.4	(-)	-112	(-)
14	667	(66.7)	2.4	(-)	16.0	(0.2)	-276	(-)
15	205	(-)	19.4	(3.2)	20.3	(12.9)	626	(30.5)
16	227	(-)	30.4	(1.5)	15.0	(6.2)	640	(28.9)
17	412	(37.2)	33.3	(11.1)	13.1	(6.7)	478	(40.3)
18	777	(94.3)	33.1	(10.2)	15.3	(4.2)	353	(55.8)
PC's	5		5		5		5	

Table 7.11: Predictions of the test set using 4-way PARAFAC models calibrated on the (8×3×28×64) array.

In comparison to the predictions from the 3-way PARAFAC models, there are no

significant gain in accuracy when using 4-way data. However, it should be noted that the 4-way model uses only (5·3+5·28+5·64=) 475 parameters to score the predicted samples.

7.3.3 4-way TUCKER decomposition - core rotation

The algorithm for core rotation proposed in chapter 6 will be used here for a more exploratory use of the TUCKER model. A TUCKER model using 6 factors in all 4 way was estimated from the (12×3×28×64) array of composite samples previously used for the PARAFAC decomposition. The 10 largest core elements are listed in table 7.12 together with the relative size of the corresponding factor combination. The relative size of a core element is calculated as the percentage of the squared values of the core element explained by the cited combination of factors in relation to the total sum of squares of the core.

#	Combination	Rel. size [%]
1	(1,3,2,6)	8.5
2	(1,3,2,3)	6.6
3	(1,6,5,4)	4.8
4	(2,6,1,4)	4.1
5	(1,6,1,4)	4.0
6	(1,6,5,6)	3.8
7	(1,6,1,6)	3.4
8	(1,6,4,4)	2.9
9	(1,3,2,4)	2.3
10	(1,6,2,6)	1.7

Table 7.12: The 10 largest core elements of the unrotated factor estimates. The degree of diagonality is 0.054 %

The degree of diagonality of the core was found to be 0.05%. The degree of diagonality (DGD) expresses the ratio between the sum of the squared diagonal elements and the total sum of squared elements, see Henrion (1993). Clearly, the DGD for the estimated model is very low. This means that the solutions derived may be very diffucult to interpret as table 7.12 indicates. Apart from way 1 and 2, the 10 most significant core elements are involving several different factor combinations. It will be investigated how much the core can be diagonalized by rotating the estimated factors. The algorithm for core rotation is applied giving as argument a (6×6×6×6) hyper diagonal as the *desired* structure. The 10 largest elements of the rotated core are listed in table 7.13. Due to extreme demands of computational time required to obtain the 4-way TUCKER estimates, only one core has been estimated and rotated. By estimating more solutions to the TUCKER model, also more different cores could be rotated to yield a better impression on how independent the TUCKER estimates can

maximum be.

#	Combination	Rel. size [%]
1	(1,1,1,1)	29.1
2	(1,2,2,1)	10.7
3	(2,1,1,2)	7.5
4	(2,1,2,1)	4.7
5	(1,2,1,2)	4.2
6	(1,2,3,1)	1.9
7	(2,1,3,1)	1.8
8	(1,3,1,1)	1.7
9	(2,1,1,1)	1.6
10	(1,1,3,1)	1.5

Table 7.13: The 10 largest core elements of the unrotated factor estimates. The degree of diagonality is 29.61 % after rotation.

From table 7.13 it is readily seen that the largest core elements, hence the most important factor combinations, have been concentrated around the diagonal of the core. The rotated factors have been inspected, but since they had absolutely no resemblance with the pure spectra found from the PARAFAC decomposition, hence, they are not shown. It should be noted that the factors did not resemble the pure spectra before rotation either. It is interesting that there is a clear jump in the size of the core elements when going from 5 core elements to 6 core elements in the rotated core. Whether this is important or not, future applications of the core rotation will have to reveal. Since the algorithm for rotating the factors is new, the discussed application is the first and only. When more experience has been gathered the usability of the algorithm can be evaluated in more detail. The proposed algorithm has shown to be able to derive with a hyperdiagonal core when exposed to synthesized 3-way data made from noise-free pure spectra and a hyperdiagonal core.

7.3.4 4-way TUCKER calibration

The application of the TUCKER model in 4-way calibration is very similar to that of the 3-way case. Of course, the matter of unfolding has no meaning using the data in its native 4-way form. Still, the TUCKER decompositions suffers from the drawback of having rotationable solutions. As in the 3-way analysis, the iterative schemes for obtaining the factor estimates were initiated with the concentration profiles of the calibration data, added some noise, as starting values for the scores. NNC was used to obtain scores that would more likely be correlated to the concentrations of the calibration samples. Following the same methodology as used during the 3-way TUCKER calibration, the following predictions of the

test set were obtained. The predictions from the best obtainable calibration models are descibed in table 7.11 along with the dimensions of the models found.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	252	(-)	14.5	(-)	3.4	(-)	-212	(-)
6	31	(-)	6.8	(-)	9.3	(-)	-36	(-)
7	191	(36.4)	14.9	(-)	-8.0	(-)	3	(-)
9	63	(-)	9.4	(-)	0.1	(-)	549	(8.6)
10	64	(-)	-1.7	(-)	0.9	(-)	-135	(-)
14	390	(2.6)	31.4	(-)	17.1	(6.9)	61	(-)
15	208	(-)	21.2	(5.9)	20.8	(15.4)	957	(6.3)
16	265	(-)	27.1	(9.6)	16.7	(4.6)	839	(6.8)
17	320	(6.6)	26.2	(12.7)	15.6	(11.1)	959	(19.9)
18	504	(26.0)	40.6	(35.5)	17.8	(11.0)	916	(14.5)
PC's	2-2-3-3		2-2-3-2		2-2-2-2		2-2-3-3	

Table 7.14: Predictions of the test set using 4-way TUCKER models calibrated on the (8×3×28×64) array with NNC and simplified cores.

The predictions in table 7.14 indicates that the 4-way calibration models were comparable with the 3-way models in predictive accuracy. As with PARAFAC, the lack of stabilization in the 4-way models must be due to the bad selections of pH levels. This is also marked by the dimensions of the models used to obtain the predictions in table 7.14, since only 2 factors are found to be significant in all 4 models in the pH-way, e. g. way number 2.

7.3.5 4-way PLS1 calibration

To complete the analysis, the 4-way data was used in PLS calibration. The dimensionalities of the models are listed in table 7.15.

	Predicted conc. [M·10^6] (Rel. error [%])							
#	CATE		HQUI		INDO		PHEN	
2	406	(-)	5.9	(-)	-5.7	(-)	-262	(-)
6	-279	(-)	2.1	(-)	4.9	(-)	-86	(-)
7	388	(29.3)	4.6	(-)	-1.3	(-)	97	(-)
9	99	(-)	2.4	(-)	0.4	(-)	532	(11.4)
10	-50	(-)	2.1	(-)	2.3	(-)	-161	(-)
14	390	(2.5)	1.8	(-)	15.9	(0.6)	-316	(-)
15	19	(-)	17.6	(12.1)	20.1	(11.9)	599	(33.5)
16	52	(-)	28.8	(4.1)	15.0	(6.5)	623	(30.8)
17	256	(14.8)	32.5	(8.5)	13.0	(7.4)	461	(42.4)
18	564	(41.0)	30.8	(2.6)	15.3	(4.2)	351	(56.1)
PC's	6		5		4		4	

Table 7.15: Predictions of the test set using 4-way PLS1 models calibrated on the (8×3×28×64) array.

7.4 A resume of the calibration results

Common for the predictive models is that they have been calibrated on 8 samples each containing no more than 3 analytes. The test set consists of 10 samples and is composed of samples containing up to 6 analytes simultaneously. This means that the models are not calibrated for the possible interactions between the analytes. Furthermore the 2 interferences (DOPA and TRYP) are present only in the test set.

In the analysis of 2-way structures the PCR and PLS models predicticted equally well with no significant differences. In the 3-way analyses the TUCKER models proved to be significantly better than the other 3-way models. However, the TUCKER models requires much more attention during calibration than any other of the models. From the PARAFAC models it was found that the unfolding could improve on the predictive accuracy of a model. However, this could not be explained by inspection of the pure spectra in figures 7.1A, 7.2A and 7.3A. For the 4-way models the significantly most accurate predictions were obtained from the PLS1 models, except from the prediction of PHEN. The 4-way TUCKER model offered the best general model for calibration, since all 4 analytes were predicted with high accuracy. The most accurate predictions of all the models was obtained using the 4-way PLS1,

however, being poor at prediciting the PHEN concentrations. The 4-way TUCKER model provded the most accurate predictions for all 4 analytes.

Additionally, it has been shown that the predictive models based on the TUCKER decomposition could be reduced dramatically with almost no effect on the prediction accuracies.

It should be obvious at this point, that there are many ways to get the best out of the models. There is not one path that will leed to the perfect calibration model, being simultaneously robust and accurate. How the data should be handled depends solely of the aim of the analysis. In the discussion above it was, by seeking the best calibration models, tried to propose some possible approaches that should be generally applicable.

8. Titration of thick juice samples

Previously it has been shown that titration curves from acidic titration of thick juice samples can be used to discriminate between some of the sugar factories, Andersson (1995). The acidic titration curves have been supplemented by alkalic titration curves in the present work. The earlier experiments are extended towards establishing a model for prediction of product parameters from the new data. The discussion starts with PCA and extends to PLS regression. The theory of the following applications of PCA and PLS is discussed in depth by Wold et al. (1987) and Martens and Næs (1989).

It should be noted that the titrated samples were collected during the 1993 campaign (October to December), and that they have been stored at -18°C since. During various investigations of the samples they have been thawed and refrozen several times. How this affects the samples has not been investigated, but it can not be excluded that the thawing and freezing homogenizes the samples to some extent. A likely consequence could be that the finer characteristics of each sample are levelled out by this.

8.1 Analysis of titration curves

The acidic titration curves have been collected by titration on solutions consisting of 15 ml thick juice and 15 ml distilled water. Two solutions were made from each thick juice sample of which one was titrated towards pH 1.5 with 0.1 M HCl and the other was titrated towards pH 12 with 0.1 M NaOH. 80 thick juice samples were titrated by both acidic and basic titrant, hence the measured data set consisted of 80 samples (rows) each with a total of 219 pH values (columns). The measured titration curves are designated **X**-data.

On fig. 8.1 such two append titration curves are shown for two different factories. The value of pH at zero volume titrant added is the initial pH of the solution. Since the consistence of non-diluted thick juice is like syrup due to the low water content, extra water had to be added. It is of course vital that pH is defined in the solution and this requires the presence of water. The reason for not diluting the sample more than 50 % was that the measured solution had to have the pH responding species in as high concentrations as possible to be able to monitor the pH-effects. If the dilution was too high the pH-effects of the acids and bases in the samples would be to small to be measured. Normally, the sensitivity of pH measurements in thin aqueous solutions is fairly high but the application of pH measurements to viscid thick juice solutions made the mentioned precautions necessary.

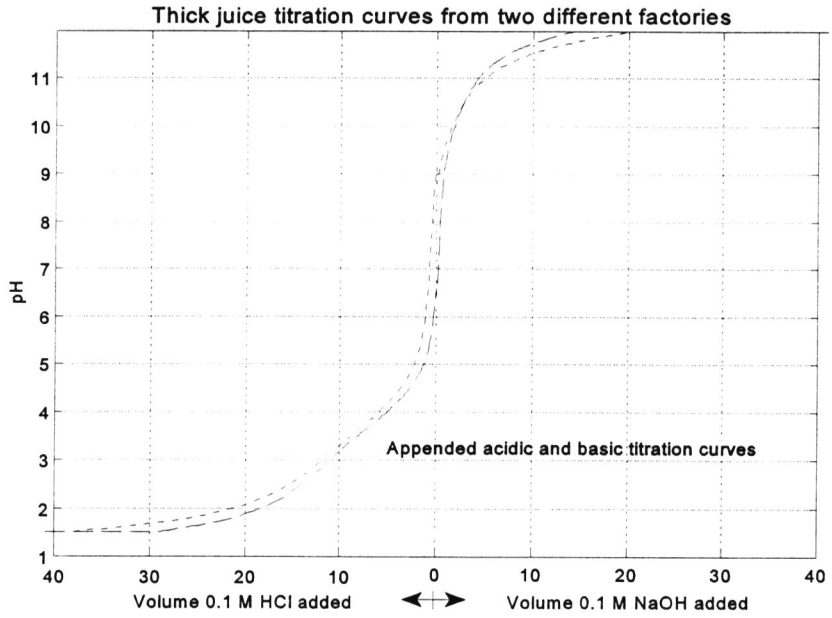


Figure 8.1: A complete titration curve is made up of two independent curves; One from the addition of 0.1 M HCl and one from the addition of 0.1 M NaOH.

The development of pH for each factory over the campaign is shown in fig. 8.2. The x-axis designates what week in the campaign the sample was collected. The overall relation between the week number and the measured pH can be depicted by overlaying regression lines from linear regression. Apart from factory E there is a trend towards a decrease in pH as the campaign runs. It is obvious the there is a relatively high level of noise on the pH measurements, however the trend of decreasing pH as the campaign runs is clear.

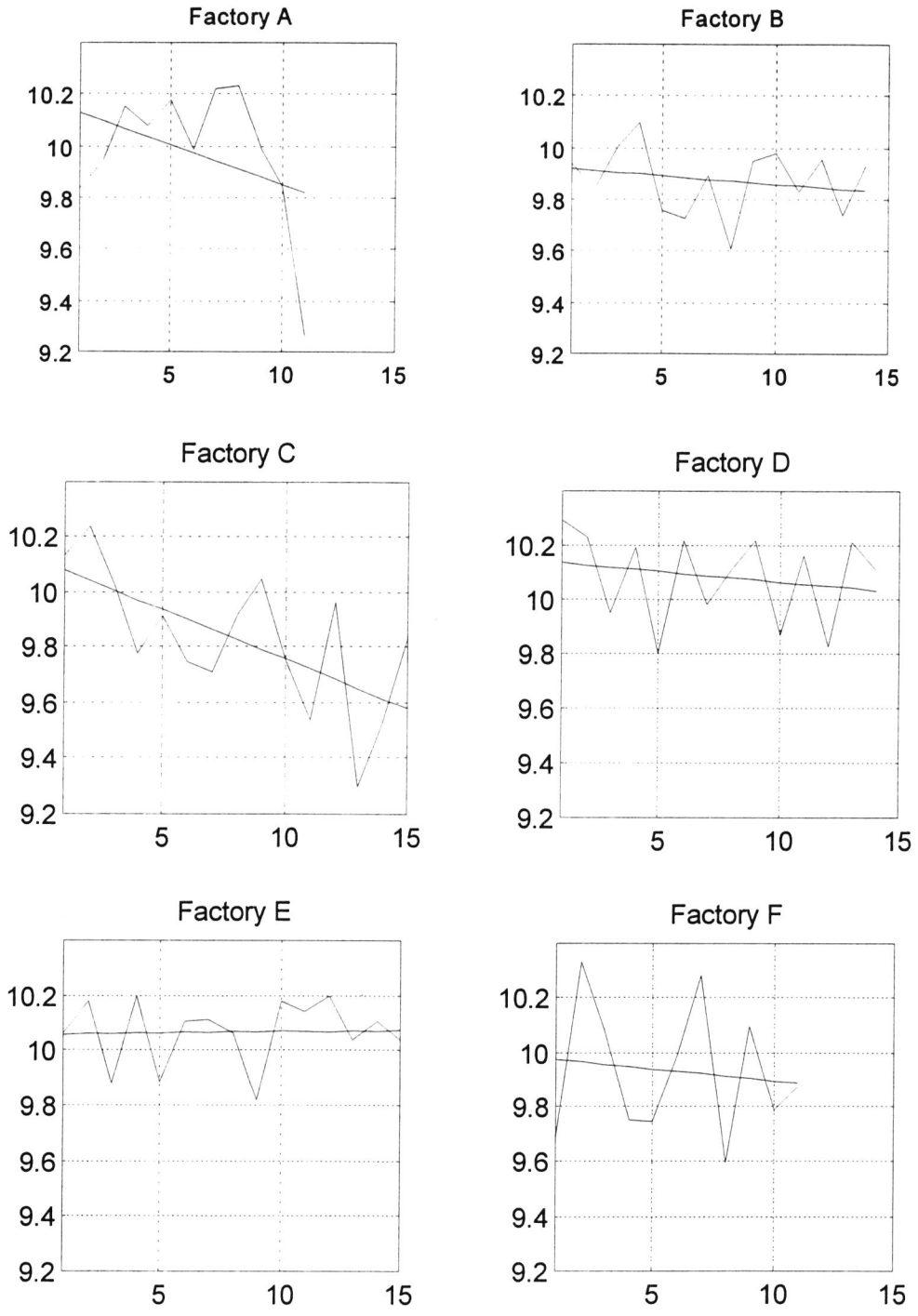


Figure 8.2: Measured pH in thick juice samples for each factory as a function of the week of sampling. Each plot have been added a linear regression line to visualize the trend of decreasing pH with time. The number of observations differs from factory to factory since different numbers of samples have been investigated.

In order to get acquainted with the data, a PCA was applied to the mean centred data. A PCA model using 1, 2 and 3 factors explained 45.6 %, 76.9 % and 92.6 % of the total variance. When the variables were autoscaled, i.e. scaled to zero mean with unit variance, a new PCA model explained 46.2 %, 84.4 % and 93.4 % with 1, 2 and 3 principal components.

For two reasons autoscaling was chosen; the explained variance of the autoscaled data was higher and the loadings derived from the autoscaled data were directly interpretable as shown on fig. 8.3.

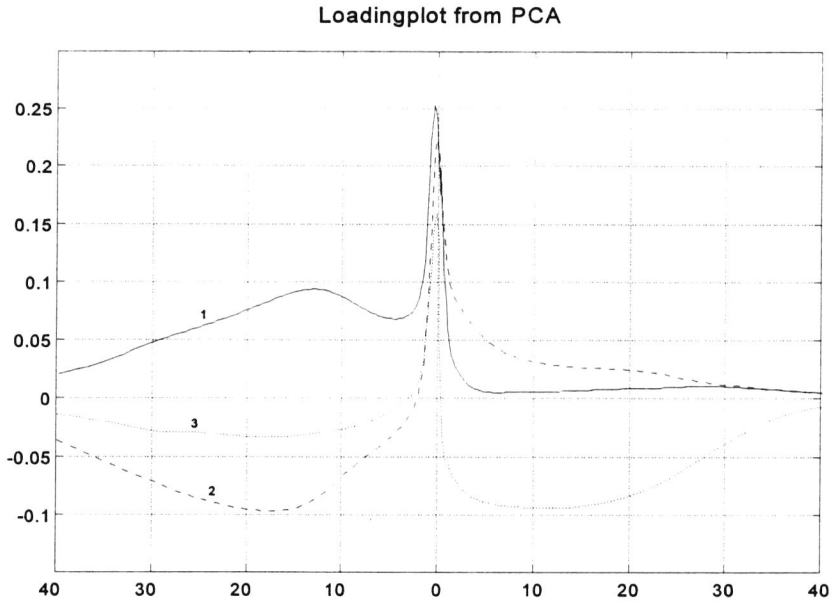


Figure 8.3: 3 loading vectors from PCA on 80 autoscaled pH curves. Loading #1 can be identified as explaining the variance caused by the pH measured before any titrant has been added. Loading #2 explains the acidic part of the titration curve whereas loading #3 represents the alkaline part of the curve. Compare the loadings to fig. 8.1.

The score for each object is depicted in fig. 8.4. Apparently the samples from the factories encoded as A and F forms two relatively independent clusters. This had previously been found from using only the acidic part of the spectra, Andersson (1995).

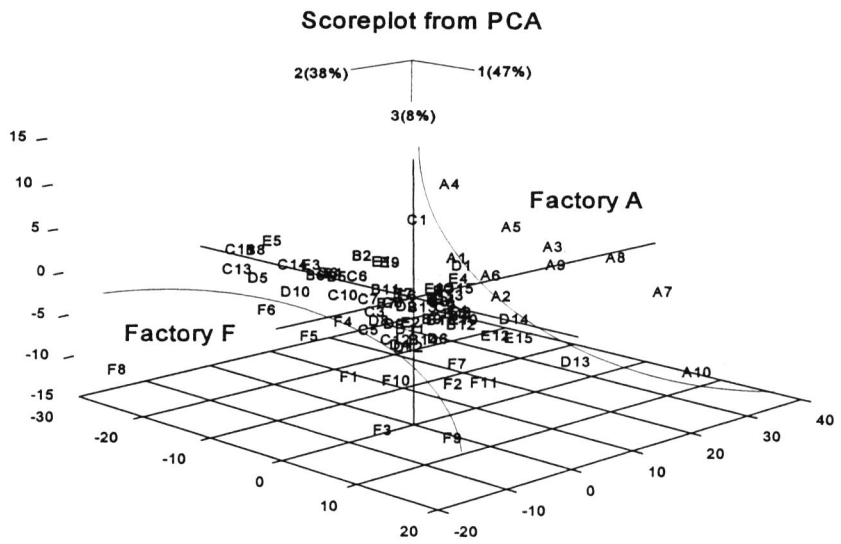


Figure 8.4: A 3-D plot of the scores of each of the 80 samples. Note that factories encoded as A and F constitutes two fairly distinct classes.

A PLS model was used to examine if the sampling week number could be predicted from the titration curves. This would indicate changes of the chemical behaviour in the thick juice with time, e.g. time of storage and so on. The same **X**-data as above was used again for seeking correspondence with a column of number representing the number of the week which the sample was collected. This column of **Y**-data had a week number for all samples represented as a column with 80 entries. However, it was not possible to establish a model relating the titration curves with the week of the sampling. This was also to be expected from the plots in figure 8.2. Since, the analysis gave no feasible results none is reported.

The method of soft independent modelling of class analogies (SIMCA) was applied to examine the potential of classifying the titration curves. However, no significant advantage over the total PCA approach was discovered, hence no results are to be reported on this approach.

8.2 Analysis of the chemical parameters

From measurements performed by DDS Nakskov, 12 chemical parameters had been determined for each titrated sample. The chemical parameters are RT% (w/w-% dry matter in relation to pure sugar), ash (w/w-% ash content), invert (w/w-%), amino-N (ppm), SO_2 (ppm) , SO_4^-(ppm), oxalic acid (ppm), Na (ppm), K (ppm), Ca (ppm), absorption (measured at 420 nm at pH 7) and turbidity. Refer to DDS (1985) for details on the applied analytical procedures. The measurements are made according to the procedures incorporated as standard methods in the factory laboratories.

The chemical parameters have been measured on all 80 samples, hence a matrix of 80

samples (rows) and 12 variables (columns) can be formed. These data are called **Y**-data.. A PCA on the **Y**-data was performed in order to reveal if some of the quality parameters were correlated. The explained variances of the autoscaled data were respectively 31.2 %, 47.4 %, 61.9 %, 71.6 %, 79.3 %, 83.3 %, 86.5 %, 88.6 %, 90.2 % and 92.2 % using 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10 components. From inspection of scoreplots and residual-leverage plots no outliers were found. From the low explained variances it was concluded that no significant correlation existed between the chemical parameters.

8.3 Predictive models based on titration curves

In order to evaluate the potential of the pH measurement as a mean of screening thick juice in-line, it was investigated if the titration curves could be used as a fast method for predicting the chemical parameters which requires time consuming wet chemical analysis.

The 80 titration curves were used to build PLS1 models for predicting each of the chemical parameters. However, the titration curves could in no way provide a basis for significant predictability of the chemical parameters. Since the analysis of the chemical parameters had shown that no correlation existed between the parameters, PLS2 modelling was not expected to provide better results. And this was also found by investigation.

From the 1995 campaign new thick juice samples have been collected. It will be investigated if the new samples, having not been frozen and thawn, are capable of providing a basis for predicting some of the quality parameters.

9. Conclusion

A review of the literature on the process of sugar production and the formation of coloured species during production is reported. An overview on the process of sugar production is presented. Four classes of fluorophoric products formed during sugar production are described; melanoidines, flavonoides, caramels and melanines. The reaction paths leading to the formation of melanoidines and melanines are discussed.

Until now, thick juice has been used as the intermediary product from which the quality parameters of the final sugar are predicted. However, from the analysis of the process of sugar production, it seems more feasible to use standard liquor for this purpose. The reason for preferring standard liquor, is that the sugar is made directly from this stream, rather than thick juice. The difference between thick juice and standard liquor is that the standard liquor includes recycled streams from the last production steps. The coloured components in the recycle stream may have significant influence on colour formation and other quality parameters of the final sugar. The next step, after predictive models have been obtained using standard liquor, would naturally be to use the thick juice to determine how early in the process the quality parameters could be satisfactorily predicted.

The instrumental part of the work has been successful with respect to achieving an instrumental setup from which the presented data have been collected. The problem with the lack of sensitivity of the diode array was circumvented by selecting modelling components that could be measured within the given limits of the apparatus. At the time of printing, the system has been optimized as suggested in the discussion of future activities. However, the given instruments could not be brought up to a sensitivity that allowed for collection of pH gradients as proposed in the preliminary report.

From the findings of the literature search on coloured species formed during sugar production, a simple 6-component model system has been made that roughly resembles the fluorescence behaviour of thick juice. Fluorescence intensities were collected from this model system using the spectrofluorometric instruments. The collected fluorescence intensities constituted a 4-way array which was investigated using 2-way, 3-way and 4-way chemometric models. For the purpose of investigating the potential increase in predictive accuracy when using multi-way data, the 4-way array was unfolded in several ways. The calibration set was comprised of 8 samples containing a maximum of 3 analytes simultaneously. The validation set, consisting of 10 composite samples, included the presence of 2 simulated interferences. There were no significant differences between the accuracies of the predictions using the 2-way calibration methods, PCR and PLS. Both 2-way models predicted the samples with a maximum error of 45 % of CATE in the most composite samples. For the purpose of analysing the data in 3-way unfoldings, PARAFAC, TUCKER and PLS1 models were calibrated. Among these, the TUCKER model gave the most accurate predictions with a maximum relative error of 41 % for CATE in the composite samples. However, calibration of the TUCKER models is elaborate and requires intensive computational efforts and a high degree of supervision. Other models providing predictive accuracies comparable to those of the TUCKER model are the 3-way PLS1 models with a relative error of 63 % for CATE in the most composite samples. The PLS regression is preferable to the TUCKER and

PARAFAC models due to the ease and speed with which it is calibrated. Another important conclusion regards unfolding. From the calibrations of the 3 different 3-way unfoldings of the 4-way data, it was found that significant differences in the predictive accuracies can be caused by arranging the data differently. However, by inspection of the pure spectra no systematics could be found for the unfoldings to ensure the highest accuracy of the predictions. The conclusion must be, that if higher-way data are to be unfolded for calibration purposes, it is generally a good idea to investigate the data in all the possible unfoldings. In the 4-way analysis the highest accuracies were obtained by TUCKER calibrations. The highest relative errors of the predictions from the TUCKER models were observed for CATE and HQUI with 36 % relative error. However, calibrations using TUCKER models were time-consuming and furthermore, the way the models are calibrated requires such high degree of supervision that it is not possible to devise a general approach by which the models can be obtained. As with the 3-way models, the PLS1 regression provided predictions that were accurate and these were obtained within a few minutes. In contrast, the 3-way and 4-way TUCKER and PARAFAC models typically required from 5 to 70 hours to reach the most modest convergence criteria. Therefore, in 4-way analysis the PLS1 models were superior with regard to the combination of computational time required and the accuracies of the achieved predictions. The PLS1 model performed suboptimally with regard to predicting PHEN with an average relative error of 32 %. The most accurate and easiest obtainable predictions over all the unfoldings were obtained by the use of PLS regression. Overall, the 4-way PLS1 regression gave the most feasible combination of speed and accuracy with the data at hand.

Another aspect briefly dealt with is the explorative use of the models for decomposing the composite samples into the pure underlying spectra. For this purpose the 4-way PARAFAC model was chosen. The estimated spectra are comparable to the pure spectra, and the estimates can be used to identify the analytes when all 3 characteristic spectra are considered simultaneously. The correlation coefficients between the resolved spectra from the composite samples and the measured spectra of the pure sample were all above 0.963.

The algorithm for core rotation was applied to the core of a 4-way TUCKER decomposition. As a result of the rotation, the degree of diagonality increased from below 0.1 % to approximately 30 %. Thereby the factors were rotated towards resemblance with the PARAFAC solution without loss of fit. The relatively low degree of diagonality obtained (30 %) indicates that a PARAFAC model will not be able to give nearly the same fit as that derived from a TUCKER model. The rotation in no way provided interpretable factor estimates, hence these have not been discussed.

Investigations of titration curves from frozen thick juice samples were conducted in continuation of the preliminary work. As previously shown, 2 of the factories stand out on PCA score plots from the titration curves. However, no models were obtainable that could satisfactorily predict the quality parameters of the final product. It is believed that better models can be established if the titration curves are measured from newly collected samples that have not been frozen.

On the whole, the necessary predictive accuracies have proven to be obtainable by using the discussed models. In addition, the core rotation proved to increase the degree of diagonality of the core, as expected. It must be concluded that the developed *n*-way algorithms

have proven their validity. No errors have been found when the proposed methodologies were used for implementation of the procedures.

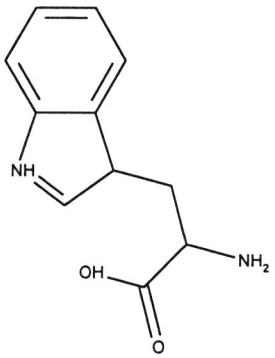
Future activities will include the use of more pH levels, ranging from pH 2 to pH 12 in order to ensure a better spectral separation between the fluorophores in the model system. The recent upgrade of the detector will be investigated with a view to using true FIA approaches. This will allow for the use of pH gradients instead of fixed pH levels. The new detector will also allow for the investigation of thick juice due to the wider detection range offered.

Appendix A

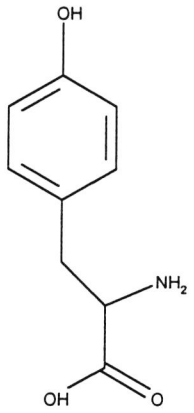
Structures of selected chemical components

This appendix contains structures of selected chemical compounds. The appendix implies the most significant structural properties of the discussed components. Since the present discussion has its offspring within fluorometry most of the selected compounds show fluorophoric activity. As discussed by Schulman (1979), the active fluorophoric sites contain electrons involved in π-bonds, e.g. in aromatic rings and generally conjugated bonds.

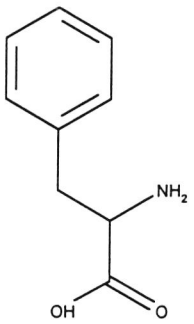
The structures should not be regarded as being qualitatively correct. In order to clarify the structural skeleton not all hydrogen atoms are shown. Substiuents are indicated by R's.



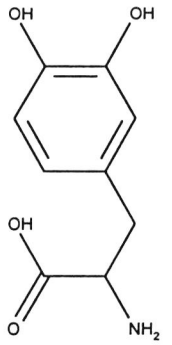
[1]
Tryptophane



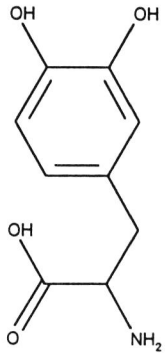
[2]
Tyrosine



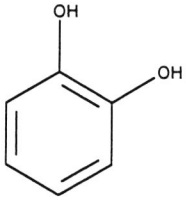
[3]
Phenylalanine



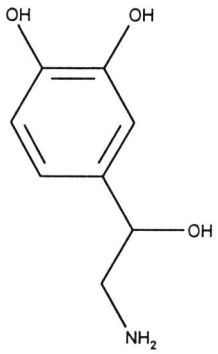
[4]
3,4-Dihydroxyphenylalanine (DOPA)



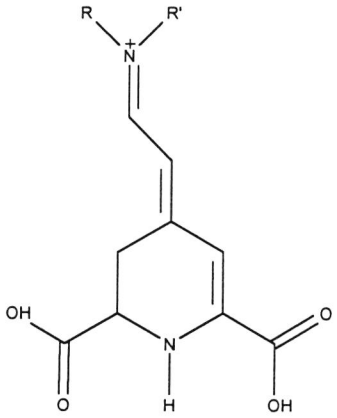
[5]
3,4-Dihydroxythyramine (DOPAmine)



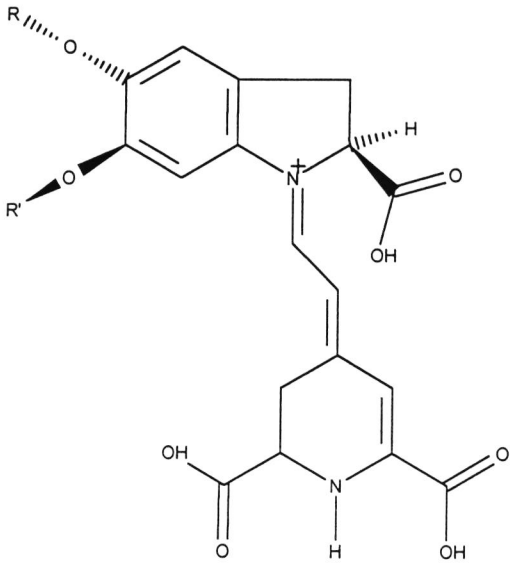
[6]
Catechol



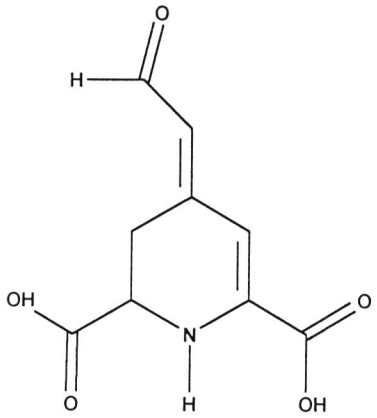
[7]
Noradrenaline



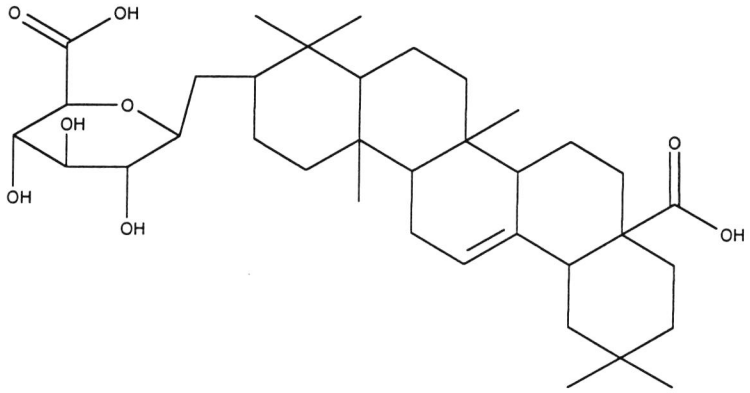
[8]
Betalaine base



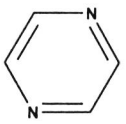
[9]
Betanine basis



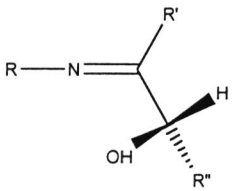
[10]
Betalamic acid



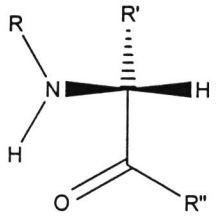
[11]
Olenanolic acid - glucoronoid



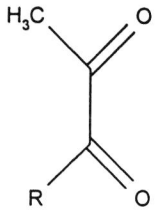
[12]
Pyrazine basis



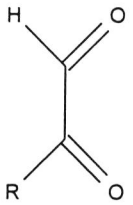
[13]
Schiff's base basis



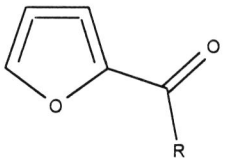
[14]
Amadori basis



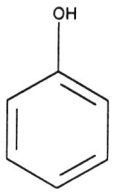
[15]
2,3-Dicarbonyl basis



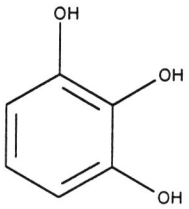
[16]
1,2-Dicarbonyl basis



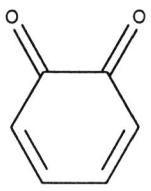
[17]
Furfural basis



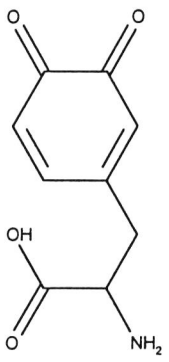
[18]
Phenol



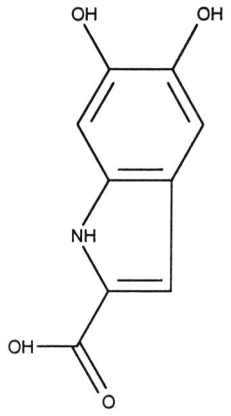
[19]
Pyrogallol



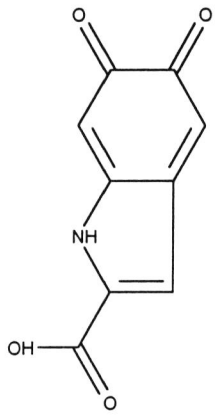
[20]
1,2-Benzodiquinone



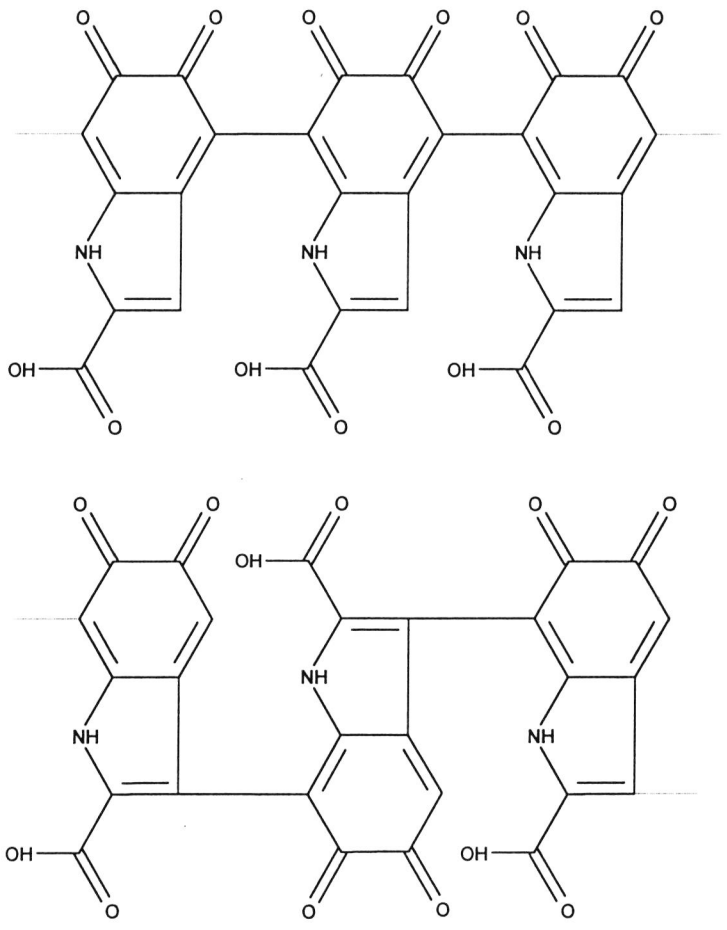
[21]
1,2-Diketophenylalanine



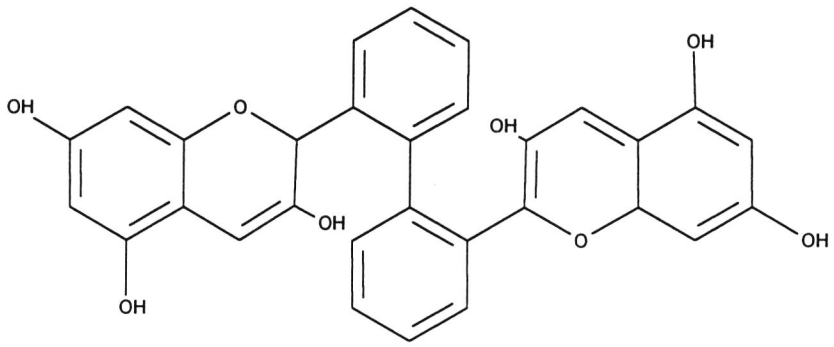
[22]
5,6-Dihydroxy 2-indolic acid



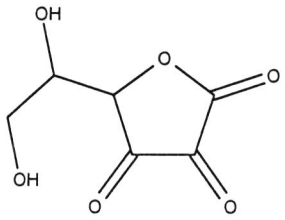
[23]
5,6-Diketo 2-indolic acid



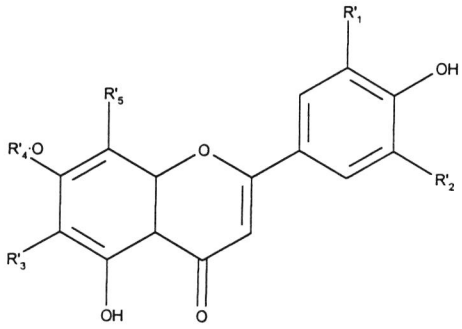
[24]
Example I of a melanin complex



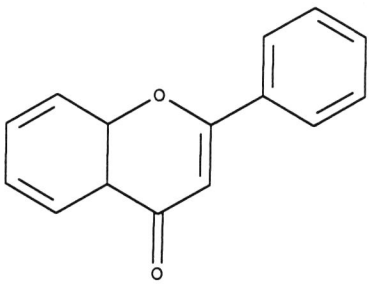
[25]
Example II of a melanine complex



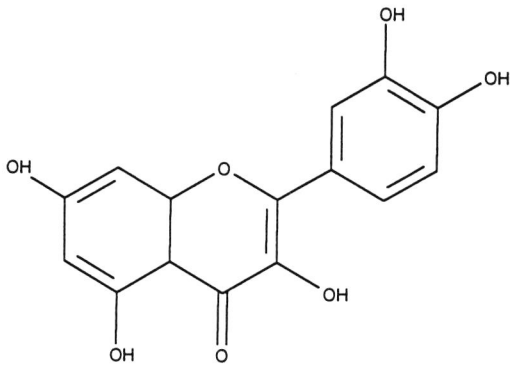
[26]
Dehydro ascorbic acid



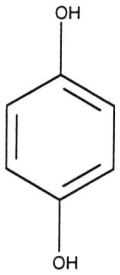
[27]
Flavonoidic basis



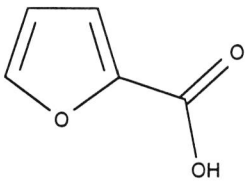
[28]
Flavone



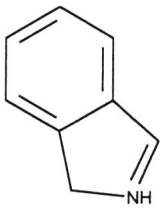
[29]
Quercetine



[30]
Hydroquinone



[31]
Furan 2-carboxylic acid



[32]
Indol

Appendix B

Chemical reagents and other specifications

Chemicals

Fluorometry, buffer chemicals

Citric acid 1-monohydrate, Merck, p.a.
Tri-sodium citrate, Merck, p.a.
Glycine, Merck, p.a.
Hydrochloric acid, Struers, Titrosol, p.a.
Sodium hydroxide, Struers Titrosol, p.a.

Fluorometry, buffer stock solutions

0.10 M Citric acid (21.01 $g \cdot l^{-1}$ $C_6H_8O_7 \cdot H_2O$)
0.10 M Sodium citrate (29.41 $g \cdot l^{-1}$ $C_6H_5O_7Na_3 \cdot 2H_2O$)
0.20 M Glycine (15.01 $g \cdot l^{-1}$ $C_2H_5O_2N$)
0.20 M Hydrochloric acid
0.20 M Sodium hydroxide

Fluorometry, modelling chemicals

DL-Tryptophane $C_{11}H_{12}N_2O_2$, Sigma, >98%
L-Tyrosine $C_9H_{11}NO_3$, Sigma, >99%
DL-Phenyl-alanine $C_9H_{11}NO_2$, Sigma, >98%
DL-DOPA $C_9H_{11}NO_4$, Sigma, p. a.
Phenol C_6H_6O, Merck, p. a.
Catechol $C_6H_6O_2$, Merck-Schuchardt, >99%
Pyrogallol $C_6H_6O_3$, Riedel-de-Haën, extra pure >99.5%
Hydroquinon $C_6H_6O_2$, Riedel-de-haën, extra pure >99.5%
Furan-2-carboxylic acid $C_5H_4O_3$, Merck-Schuchardt, p. a. >99%
Indol C_8H_7N, Riedel-de-Haën, p. a. >99%

Fluorometry, aqueous modelling stock solutions

0.0001 M Tryptophane (0.0204 $g \cdot l^{-1}$ $C_{11}H_{12}N_2O_2$)
0.01 M DOPA (0.0197 $g \cdot l^{-1}$ $C_9H_{11}NO_4$)
0.1 M Phenol (9.411 $g \cdot l^{-1}$ C_6H_6O)
0.1 M Catechol (11.01 $g \cdot l^{-1}$ $C_6H_6O_2$)
0.1 M Pyrogallol (12.61 $g \cdot l^{-1}$ $C_6H_6O_3$)
0.1 M Hydroquinone (11.01 $g \cdot l^{-1}$ $C_6H_6O_2$)
0.1 M Furan-2-carboxylic acid (11.21 $g \cdot l^{-1}$ $C_5H_4O_3$)
0.001 M Indol (0.1171 $g \cdot l^{-1}$ C_8H_7N)

pH-titration
0.1 M Hydrochloric acid, Struers, techn. quality.
0.1 M Sodium hydroxide, Struers, techn. quality.

All water was doubly ion-exchanged, Si-free and filtred through Milipore 0.22 μm filtres. All solutions were degassed using ultrasonics.

Computer

Intel, Pentium 90 Mhz , PCI
24 MB RAM, 0 KB Cache
Windows 95 operating system

Software

Matlab for Windows, Version 4.2c.1, 1994
User Interface Toolbox, Version 1.2, 1994
Matlab Compiler Toolbox, Version 1.0, 1995
MathWorks Inc., USA

MathCad, Version 5.0 PLUS, 1995
Symbolic Math Package Extension, Maple
MathSoft, Inc., USA

Watcom C/C++, Version 10.5, 1995
32 bit compiler for use with Matlab Compiler Toolbox
Watcom Corp., Ontario, Canada

Appendix C

Matlab source code

PARAFACN.M	This program finds the solutions to the PARAFAC model for *n*-way data structures. The size of *n* is only limited by computer memory.
PARASCOR.M	This procedure calculates the factors in *one* specified way/direction. The factors will be calculated directly (not iteratively) from the factors in the other ways. The set of factors given as argument must have been calculated previously (from PARAFACN.M). This procedure is used to calculate the score of an object for use as basis for prediction in multi-way PARAFAC calibration.
TUCKERN.M	This program finds the solutions to the TUCKER model for *n*-way data structures. The size of *n* is only restricted by memory demands.
TUCKSCOR.M	This procedure calculates the factors in *one* specified way/direction. The factors will be calculated directly (non iteratively) from the factors in the other ways and the core. The given arguments must have been calculated previously (from TUCKERN.M). The procedure calculates the score of an object to be used as basis for prediction in multi-way TUCKER calibration.
TUCKROTN.M	This procedure finds a least squares fit of a previously derived core to a new core. While doing this, mapping matrices are determined. The rotations of the original factors are done by simple multiplication of the original factors by the mapping matrices. If the desired structure of the core is given the routine runs without user intervention.
PLSN.M	PLSN.M makes a calibration model in theory comparable to conventional PLS2. The algorithm makes it possible to model the contents of an *n*-way structure on the Y-side on the basis of an *n*-way structure on the X side. Since the routine is built on PARAFACN.M the structures in X and Y can undertake any number of ways. E.g. it is possible to establish a PLS model between a 3-way array in X and a 4-way array in Y. The algorithm works for any *n*'s, $n \geq 2$.

PARAFACN.M

```
function
[Factors,G,XExpl,SSE,SSTot]=parafacn(Z,R,W,FacType,Tol,Factors,Options);
%[Factors,G,XExpl,SSE,SSTot]=parafacn(Z,R,W,FacType,Tol,Factors,Options)
%
% This Matlab program (parafacn.m) derives the factors of
% any number of variables in any number of ways
% according to the PARAFAC/CANDECOMP model.
% The PARAFAC model is a special case of the
% TUCKER interaction model.
% The math model is Z=A*(for i=1:w;[kron(B1,C1,D1);kron(B2,C2,D2);...])
%
% Z         : Data array to be decomposed.
%             Must be specified!
% R         : Number of observations along each mode.
%             E.g. [10 6 101 14 13] or [3 4 4 4 6]
%             There must be at least 3 elements in R
%             corresponding to 3 or more dimensions.
%             Must be specified!
% W         : Number of PC's to find along each mode
%             Ex. [3] or [3 3 3 3 3] for 5-way data
%             Must be specified!
% FacType   : A vector of options describing what types of factors
%             to derive. [2 1 3] means orthogonality constraints
%             applies to the first mode, no constraints in the second mode
%             and non-negativity in the third mode.
%             1=No constraints, 2=Orthogonality, 3=Non negativity.
%             Default: Factype = [1 1 ... 1]
%             Specify Factype=0 for default.
% Tol       : Tol designates the convergence critria. When the difference
%             between two succesive SSE's is equal to, or less than, the
%             value of Tol the algorithm stops.
%             Default: Tol = 10E-10*(sum(sum(Z.^2)))^(0.5);
%             Specify Tol=0 for default.
% Factors   : The calculated factors are returned in this VECTOR.
%             The set of factors for the n'th way is contained
%             is contained from places (n-1)*R(n) to n*R(n+1)-1.
%             If Factors are defined as input arguments then these
%             will be the initial values.
%             Default: Factors are initialized by equally distributed peaks.
% Options   : Mostly used by other programs calling this one.
%             0:All possible output is given.
%             1:Suppress all output to the screen.
% G       : G is the hyperdiagonal pseudo-core. This will represent
%             the explained variance by the actual factor combination.
% SSE       : Sum of squares of misfit between each element
%             in Z and the modelled value.
% SSTot     : Total sum of squares from the raw input
%
% Revised   : 3-11-95, Initial factors can be set to Gaussian peaks
%             8-1-96, Better convergence estimation
%             21-1-96, Making SSE to RMS
%             7-2-96, Options
%
% Author    : Claus Andersson, August 1995
%             KVL, Copenhagen, Chemometrics Group
% E-mail    : claus.andersson@pop.foodsci.kvl.dk
%
%For copying to the Matlab Command Window:
%[Factors,G,SSE,SSTot]=parafacn(Z,R,W,Factype,Tol,Factors,Options);
%[Factors,G,SSE,SSTot]=parafacn(Z,R,W);

 %Setting up the environment
  format compact

 %Number of dimensions of the data structure, C
  C=length(R(1,:));
```

```
%Check for consistency
 if C<3,
   disp('Cannot use fewer dimensions than three (3)')
   disp('R and W must contain more than three elements.')
   break;
 end;

%Ensures same number of components in each dimension, this is a PARAFAC
model
 w=W(1);
 W=w*ones(1,C);

%If no 'FacType's are defined - do it
 if exist('FacType'),
   if FacType==0,
     clear FacType;
   end;
 end;
 if ~exist('FacType'),
   FacType=ones(size(R));
 end;

%If no 'Factors' are defined - do it
 if exist('Factors'),
   if Factors==0,
     clear Factors;
   end;
 end;

%If no 'Options' are defined - do it
 if exist('Options'),
   if Options<=0,
     clear Options;
   end;
 end;
 if ~exist('Options'),
   Options=[0];
 end;

%Calculate the total variance (assuming that data are centred)
 SSTot=sum(sum(Z.^2));
 NumOfElem=size(Z,1)*size(Z,2);

%If no 'Tol' is defined - do it
 if exist('Tol'),
   if Tol<eps,
     clear Tol;
   end;
 end;
 if ~exist('Tol'),
   Tol=1E-10*sqrt(SSTot);
   if Tol<100*eps,
     Tol=100*eps;
   end;
 end;
 ConvLim=Tol;

%Create a table of starting indexes to the sets of factors in Factors
 FIdx0=cumsum([1 R(1:C-1).*W(1:C-1)]);
 FIdx1=cumsum([R.*W]);

%Have Factors been given as input?
InitForm=0;
if exist('Factors'),
  if Factors==0,
    clear Factors;
  end;
end;
if ~exist('Factors'),
  InitForm=2;
```

```
end;
%Initialization by rands if requested so
 if InitForm==1,
   Factors=rand(1,w*sum(R))+1;
 end;
%Initialisation by peaks if requested so
 if InitForm==2,
  Factors=zeros(1,w*sum(R));
  for iw=1:C,
   for i=1:w,
     I0=FIdx0(iw)+(i-1)*R(iw);
     I1=FIdx0(iw)+    i*R(iw)-1;
     temp=makepeak(ceil(R(iw)*0.08),R(iw),R(iw)*i/(w+1))';
     Factors(I0:I1)=temp/ss(temp);
   end;
  end;
 end;

 OldSSE=Inf;
 CorrelLim=0.995;
 converged=0;
 iter=0;
 DiagnoseEvery=50;
 SaveFactorsEvery=200;
 SSEEvery=10;
 if W(1)==1,
   SSEEvery=1;
 end;
 AccInProgress=0;
 NextUpdate=20;
 if W(1)==1,
   NextUpdate=5;
 end;
 AccFac=10;
 SpaceAcc=5;
 while converged==0,

   iter=iter+1;

  %Dimension loop starts
  %c holds the number of the current dimension being updated
   for c=1:C,

    %Create the order in which the serial kroneckers should be calculated.
     faclist=[c+1:C];
     if c-1>0,
       faclist=[faclist 1:c-1];
     end;

    %Create TmpKron to represent the seriel kronecker product in steps 1-2
    %1:Initiate with the three-way case
     TmpVec=R;
     TmpVec(c)=1;
     TmpVec=prod(TmpVec);
     KronProd=zeros(w,TmpVec);
     for iw=1:w;
       I0=FIdx0(faclist(1))+(iw-1)*R(faclist(1));
       I1=FIdx0(faclist(1))+    iw*R(faclist(1))-1;
       TmpMat1=Factors(I0:I1);
       I0=FIdx0(faclist(2))+(iw-1)*R(faclist(2));
       I1=FIdx0(faclist(2))+    iw*R(faclist(2))-1;
       TmpMat2=Factors(I0:I1);
       TmpKron=ckron(TmpMat1,TmpMat2);
      %2:continue with dimensions above three
       for i=3:C-1,
         I0=FIdx0(faclist(i))+(iw-1)*R(faclist(i));
         I1=FIdx0(faclist(i))+    iw*R(faclist(i))-1;
         TmpMat3=Factors(I0:I1);
         TmpKron=ckron(TmpKron,TmpMat3);
       end;
```

```
        KronProd(iw,1:TmpVec)=TmpKron(1,1:TmpVec);
       end;
      %3:Updating factor set c in TmpFac and then store to Factors
      %No constraints
       if (FacType(c)==1) | (FacType(c)==4),
         TmpMat4=Z*KronProd'/(KronProd*KronProd');
         if FacType==4,
           TmpMat4=csetbase(TmpMat4,0,0);
         end;
         if c>1,
           TmpMat4=normit(TmpMat4);
         end;
       end;
      %Orthogonality constraints
       if FacType(c)==2,
         TmpMat4=Z*KronProd'*(KronProd*Z'*Z*KronProd')^(-1/2);
         if c==1,

TmpC=(TmpMat4'*TmpMat4)\TmpMat4'*Z*KronProd'/(KronProd*KronProd');
           TmpMat4=TmpMat4*diag(diag(TmpC));
         end;
       end;
      %Non-negativity constraints
       if FacType(c)==3,
         TmpMat4=cnnls(eye(R(c)),Z*pinv(KronProd));
         if c>1,
           TmpMat4=normit(TmpMat4);
         end;
       end;
       Factors(FIdx0(c):FIdx1(c))=TmpMat4(:);

      %4:Rearranging the data in Z
       if c<C,
        %Rearranges to next unfolding
         TmpVec=R;
         TmpVec(c+1)=1;
         Z=cunfo(Z,prod(TmpVec),R(c+1))';
        else
        %Must rearrange to first unfolding
         TmpVec=R;
         TmpVec(1)=1;
         Z=cunfo(Z,prod(TmpVec),R(1))';
       end;

     %Dimension loop ends
     end;

   %5:Acceleration loop
   %If acceleration is in progress - then accelerate
    if (AccInProgress==1),
      BakFactors=Factors;
      Factors=OldFactors+AccFac*(Factors-OldFactors);
      AccInProgress=0;
      NextUpdate=iter+SpaceAcc;
     %Make sure that the accelerated fit is really better
      TmpVec=R;
      TmpVec(1)=1;
      TmpVec=prod(TmpVec);
      KronProd=zeros(w,TmpVec);
      TmpMat=cunfo(Factors(FIdx0(1):FIdx1(1)),R(1),W(1));
      for iw=1:w;
        I0=FIdx0(2)+(iw-1)*R(2);
        I1=FIdx0(2)+    iw*R(2)-1;
        TmpKron=Factors(I0:I1);
       %Continue with dimensions above three
        for i=3:C,
          I0=FIdx0(i)+(iw-1)*R(i);
          I1=FIdx0(i)+    iw*R(i)-1;
          TmpMat1=Factors(I0:I1);
          TmpKron=ckron(TmpKron,TmpMat1);
```

```
        end;
        KronProd(iw,1:TmpVec)=TmpKron(1,1:TmpVec);
      end;
      TmpMat4=TmpMat*KronProd;
      AccSSE=sum(sum( (Z-TmpMat4).^2 ));

     %If failed to accelerate then cancel accelerated factors
      if AccSSE>SSE,
        Factors=BakFactors;
        AccFac=ceil(AccFac/2);
        if AccFac<1,
          AccFac=1;
        end;
       else
        AccFac=AccFac+1;
      end;
    end;
   %If the improvement in SSE was approx less than 30% then accelerate
    if (OldSSE/SSE<1.3)&(iter>=NextUpdate),
      OldFactors=Factors;
      AccInProgress=1;
    end;

    if rem(iter,SSEEvery)==0,
     %6:Checks for convergence
     %Create TmpKron to represent the seriel kron-product.
     %Initiate with the three-way case
      TmpVec=R;
      TmpVec(1)=1;
      TmpVec=prod(TmpVec);
      KronProd=zeros(w,TmpVec);
      TmpMat=cunfo(Factors(FIdx0(1):FIdx1(1)),R(1),W(1));
      for iw=1:w;
        I0=FIdx0(2)+(iw-1)*R(2);
        I1=FIdx0(2)+    iw*R(2)-1;
        TmpMat1=Factors(I0:I1);
        I0=FIdx0(3)+(iw-1)*R(3);
        I1=FIdx0(3)+    iw*R(3)-1;
        TmpMat2=Factors(I0:I1);
        TmpKron=ckron(TmpMat1,TmpMat2);
       %Continue with dimensions above three
        for i=4:C,
          I0=FIdx0(i)+(iw-1)*R(i);
          I1=FIdx0(i)+    iw*R(i)-1;
          TmpMat3=Factors(I0:I1);
          TmpKron=ckron(TmpKron,TmpMat3);
        end;
        KronProd(iw,1:TmpVec)=TmpKron(1,1:TmpVec);
      end;
      TmpMat4=TmpMat*KronProd;
      SSE=sum(sum((Z-TmpMat4).^2));
      MSE=sqrt(SSE/NumOfElem);
      if abs(OldSSE-SSE)<ConvLim,
        converged=1;
      end,
      if Options==0,
        fprintf('Iteration %i, MSE %12.8f, Expl. %10.6f%%
\t\n',iter,MSE,(SSTot-SSE)*100/SSTot);
      end;
      OldSSE=SSE;
    end;

   %Are there degenerate solutions? Are factors strongly correlated?
   %All other modes than the first have already been normalized
    if (rem(iter,DiagnoseEvery)==0) & (Options==0),
      for i=1:C,
        I0=FIdx0(i);
        I1=FIdx1(i);
        TmpMat=cunfo(Factors(I0:I1),R(i),W(i));
        if i==1,
```

```
          TmpMat=normit(TmpMat);
        end;
        TmpMat1=TmpMat'*TmpMat;
        TmpMat2=TmpMat1(1:length(TmpMat1(:,1)),1:length(TmpMat1(1,:))-1);
       %Notify the user of the correlated factors and interprete them for
him
        [NoteI NoteJ]=find(abs(TmpMat2)>CorrelLim);
        for i1=1:length(NoteI),
          ai=NoteI(i1);
          aj=NoteJ(i1);
           cor=TmpMat1(ai,aj);
          if ai>aj,
            fprintf('High correlation (%5.3f) between factors %i and %i in
mode %i!\n',cor,aj,ai,i)
          end;
        end;
      end;
    end;

   %An exit to use for totally degenerated runs
    if isnan(SSE),
      converged=2;
    end;

   %Autosaves the temporary solutions for every 50th interation
   if rem(iter,SaveFactorsEvery)==0,
      save factors.mat Factors SSE SSTot
      if Options==0,
        disp('Saved current solution on harddisk as ''\factors.mat''.')
      end;
   end;

end;

%Tell which conv. criteria was used
if (converged==1) & (Options==0),
  fprintf('Convergence request was reached.\n')
  fprintf('Solution derived - execution stopped.\r\n')
end;
if (converged==2) & (Options==0),
  fprintf('Infeasible/Inconsistent solution!.\n')
  fprintf('No solution derived - execution aborted.\r\n');
end;
%Are the factors more negative than positive, can we change this?
%Checking the factors in mode 'inn'
innFeature=1;
if innFeature==1,
  nnMat=ones(w,C);
  for inn=1:C,
    I0=FIdx0(inn);
    I1=FIdx0(inn)+w*R(inn)-1;
    Art=Factors(I0:I1);
    Art=cunfo(Art,R(inn),w);
   %Check for each factor
    for cnn=1:w,
      if abs(min(Art(:,cnn)))>abs(max(Art(:,cnn))),
        nnMat(cnn,inn)=-1;
      end;
    end;
  end;
  %Reduce the negation matrix
  mnMat=nnMat;
  for cnn=1:w,
    pl=find(nnMat(cnn,:)==-1);
    if rem(length(pl),2)>0,
      mnMat(cnn,pl(length(pl)))=1;
    end;
  end;
  %Do the necessary change-of-sign where allowed
  for inn=1:C,
```

```
    I0=FIdx0(inn);
    I1=FIdx0(inn)+w*R(inn)-1;
    Art=Factors(I0:I1);
    Art=cunfo(Art,R(inn),w);
    Art=Art*diag(mnMat(:,inn));
    Factors(I0:I1)=Art(:);
  end;
end;

%Calculates the diagonal elements of a 'PARAFAC pseudo-core'
%because all the factors are normalized in the algorithm
I0=FIdx0(1);
I1=FIdx0(1)+w*R(1)-1;
Ar=Factors(I0:I1);
Ar=cunfo(Ar,R(1),w);
An=normit(Ar);
G=pinv((An'*An))*(An'*Ar);

%Calculate the variance explained
XExpl=100*(1 - sum(var(Z-TmpMat4))/sum(var(Z)))
```

PARASCOR.M

```
function [Scores,SSE,SSTot]=parascor(Z,R,W,Factors,Way);
%[Scores,SSE,SSTot]=parascor(Z,R,W,Factors,Way)
%
% This Matlab program (parascor.m) calculates the
% scores (factors) in way number 'Way' while
% using all other factors to do this.
%
% The math model is  Z=A*(for i=1:w;[kron(B1,C1,D1);kron(B2,C2,D2);...])
%
% Z         : Data array to be decomposed.
%             Must be specified!
% R         : Number of observations along each mode.
%             MUST REFER TO THE CALIBRATION SITUATION.
%             E.g. [10 6 101 14 13] or [3 4 4 4 6]
%             There must be at least 3 elements in R
%             corresponding to 3 or more dimensions.
%             Must be specified!
% W         : Number of PC's to find along each mode.
%             MUST REFER TO THE CALIBRATION SITUATION.
%             Ex. [3] or [3 3 3 3 3] for 5-way data
%             Must be specified!
% Factors   : Use to score the data
% Way       : The way(direction) in which the score vector is to be
%             determined must be specified in this scalar.
% Scores    : The returned values giving the ls fit to the data
%             through the given factors in all the other directions.
%             Of course also the core has to be used for this purpose.
% Levs      : The leverages for the score elements are calculated to
%             facilite a rough estimate of potential outliers.
%             High leverages means: Take in more more objects of this type
%             or exclude it (and the few other like it!).
%
% Revised   :
% Author    : Claus Andersson, December 1995
%             KVL, Copenhagen, Chemometrics Group
% E-mail    : claus.andersson@pop.foodsci.kvl.dk
%
%For copying to the Matlab Command Window:
%[Scores,SSE,SSTot]=parascor(Z,R,W,Factors,Way)
%[Scores]=parascor(Z,R,W,Factors,Way)

 %Setting up the environment
  format compact

 %Number of dimensions of the data structure, C
  C=length(R(1,:));

 %Check for consistency
  if C<3,
    disp('Cannot use fewer dimensions than three (3)')
    disp('R and W must contain more than three elements.')
    break;
  end;

 %Determine the length of the score to return
  RScor=R;
  [a b]=size(Z);
  RScor(Way)=1;
  RScor(Way)=(a*b)/prod(RScor);

 %Ensures same number of components in each dimension, this is a PARAFAC
model
  w=W(1);
  W=w*ones(1,C);

 %Calculate the total variance (assuming that data are centred)
  SSTot=sum(sum(Z.^2));
```

```
%Create a table of starting indexes to the sets of factors in Factors
 FIdx0=cumsum([1 R(1:C-1).*W(1:C-1)]);
 FIdx1=cumsum([R.*W]);

%Dimension loop starts
 for c=1:C,

 if c==Way,

  %Create the order in which the serial kroneckers should be calculated.
   faclist=[c+1:C];
   if c-1>0,
       faclist=[faclist 1:c-1];
   end;

   %Create TmpKron to represent the seriel kronecker product in steps 1-2
   %1:Initiate with the three-way case
    TmpVec=R;
    TmpVec(c)=1;
    TmpVec=prod(TmpVec);
    KronProd=zeros(w,TmpVec);
    for iw=1:w;
      I0=FIdx0(faclist(1))+(iw-1)*R(faclist(1));
      I1=FIdx0(faclist(1))+    iw*R(faclist(1))-1;
      TmpMat1=Factors(I0:I1);
      I0=FIdx0(faclist(2))+(iw-1)*R(faclist(2));
      I1=FIdx0(faclist(2))+    iw*R(faclist(2))-1;
      TmpMat2=Factors(I0:I1);
      TmpKron=ckron(TmpMat1,TmpMat2);
     %2:continue with dimensions above three
      for i=3:C-1,
        I0=FIdx0(faclist(i))+(iw-1)*R(faclist(i));
        I1=FIdx0(faclist(i))+    iw*R(faclist(i))-1;
        TmpMat3=Factors(I0:I1);
        TmpKron=ckron(TmpKron,TmpMat3);
      end;
      KronProd(iw,1:TmpVec)=TmpKron(1,1:TmpVec);
    end;
   %3:Updating factor set c in TmpFac and then store to Factors
    TmpMat4=Z*KronProd'/(KronProd*KronProd');
    if c>1,
      TmpMat4=normit(TmpMat4);
    end;
    Scores=TmpMat4;

 end;

%4:Rearranging the data in Z
 if c<C,
  %Rearranges to next unfolding
   TmpVec=RScor;
   TmpVec(c+1)=1;
   Z=cunfo(Z,prod(TmpVec),RScor(c+1))';
  else
  %Must rearrange to first unfolding
   TmpVec=RScor;
   TmpVec(1)=1;
   Z=cunfo(Z,prod(TmpVec),RScor(1))';
 end;

end;
```

TUCKERN.M

```
function
[Factors,G,XExpl,SSE,SSTot]=tuckern(Z,R,W,FacType,Tol,GAllow,s,Factors,G,Op
tions);
%[Factors,G,XExpl,SSE,SSTot]=tuckern(Z,R,W,FacType,Tol,GAllow,s,Factors,G,O
ptions);
%
% This Matlab program (tuckern.m) derives the factors of
% any number of variables in any number of ways
% according to the TUCKER3 model.
% The PARAFAC model is a special case of the
% TUCKER interaction model.
% The math model is Z=A*G*((B'(x)C')(x)D')...
%
% Z         : Data array to be decomposed, assumed to be
%             unfolded according to CHM 2 (see 'fixformn.m')
% R         : Number of observations along each mode.
%             E.g. [10 6 101 14 13] or [3 4 4 4 6]
%             There must be at least three elements in R
%             corresponding to three or more dimensions.
% W         : Number of PC's to find along each mode
%             Ex. [3 3 3 3 3] or [3 2 3 3 2]
%             The lengths of the W and R vectors must be the same.
% GAllow    : List of allowed elements in the hypercore.
%             E.g. GAllow=[1,1,1,1,1;2,2,2,2,2;2,1,1,1,1] allows only
%             interactions between the first factors and the second factors
%             and then the second factor of set 1 is allowed to interact
with
%             the first factors of the other sets.
%             Use GAllow=0 if void.
%             If GAllow is not specified, or zero, all elements in the core
will
%             be used (no restrictions).
% s         : The specified element-interactions can be reversed
%             to be forbidden by setting s=-1. In other words
%             the value of s reverses the specified factor interactions
%             in GAllow to be forbidden. Set s=0 if void. Default set to +1.
%
% Examples : [Factors,G,SSE,SSTot]=tuckern(Z,[9 7 6 181],[1 2 2
2],[1,1,1,1;1,2,2,2],1,Factors);
%             Allows the data in Z to be modelled using only interactions
between factors
%             number 1,1,1,1 and number 1,2,2,2. This is a restricted
Tucker model.
%             Also starting factors must be given in Factors.
%            [Factors,G,SSE,SSTot]=tuckern(Z,[9 7 6 181],[2 2 2
2],0,0,Factors);
%             Allows the data in Z to be modelled using a full Tucker
model.
%             Starting factors must be given in Factors.
%
% Revised   :
% Author    : Claus Andersson, August 1995
%             KVL, Copenhagen, Chemometrics Group
% E-mail    : claus.andersson@pop.foodsci.kvl.dk
%
%For copying to the Command Window:
%[Factors,G,SSE,SSTot]=tuckern(Z,R,W,FacType,Tol,GAllow,s,Factors,G,Options
);
%[Factors,G,SSE,SSTot]=tuckern(Z,R,W);

 %Setting up the environment
  format compact

 %Number of dimensions of the data structure, C
  C=length(R(1,:));

 %Checks for consistency
```

```
 if C<3,
   disp('Cannot use fewer dimensions than three (3)')
   disp('R and W must contain more than three elements.')
   %break;
 end;

%Calculate the total variance (assuming that data are centred)
 SSTot=sum(sum(Z.^2));
 NumOfElem=size(Z,1)*size(Z,2);

%If no 'FacType's are defined - do it
 if exist('FacType'),
   if FacType==0,
     clear FacType;
   end;
 end;
 if ~exist('FacType'),
   FacType=ones(size(R));
 end;

%Check 'Tol'
 if exist('Tol'),
   if Tol<eps,
     clear Tol;
   end;
 end;
 if ~exist('Tol'),
   Tol=1E-6*sqrt(SSTot);
 end;
 ConvLim=Tol;
 if ConvLim<100*eps,
   ConvLim=100*eps;
 end;

%Checks 'GAllow'
 if exist('GAllow'),
    if GAllow==0,
      clear GAllow;
    end;
 end;

%Checks 's'
 if exist('s'),
    if s==0,
      clear s;
    end;
 end;

%Check 'Factors'
 if exist('Factors'),
   if Factors==0,
     clear Factors;
   end;
 end;

%Check 'G'
 if exist('G'),
   if G==0,
     clear G;
   end;
 end;

%Check 'Options'
 if ~exist('Options'),
   Options=0;
 end;

%Checkup/correction of parameters
 if ~exist('GAllow'),
   GA=ones(W(1),prod(W(2:C)));
```

```
    s=0;
  else
    if ~exist('s'),
      s=1;
    end;
    if s==1,
      GA=zeros(W(1),prod(W(2:C)));
      for i=1:length(GAllow(:,1)),
        Indx=0;
        for c=2:C-1,
          Indx=Indx+(GAllow(i,c)-1)*prod(W(c+1:C));
        end;
        Indx=Indx+GAllow(i,C);
        GA(GAllow(i,1),Indx)=1;
      end;
    end;
    if s==-1,
      GA=ones(W(1),prod(W(2:C)));
      for i=1:length(GAllow(:,1)),
        Indx=0;
        for c=2:C-1,
          Indx=Indx+(GAllow(i,c)-1)*prod(W(c+1:C));
        end;
        Indx=Indx+GAllow(i,C)
        GA(GAllow(i,1),Indx)=0;
      end;
    end;
end;

 %Create a table of starting indexes to the sets of factors in Factors
  FIdx0=cumsum([1 R(1:C-1).*W(1:C-1)]);
  FIdx1=cumsum([R.*W]);

 %Initialization by rands
 %Have Factors been given as input?
  InitForm=0;
  if ~exist('Factors'),
    InitForm=2;
    if min(R)<=min(W),
      InitForm=1;
    end;
  end;
 %Initialization by rands if requested so
  if InitForm==1,
    Factors=rand(1,FIdx1(C))+1;
  end;
 %Initialisation by peaks if requested so
  if InitForm==2,
    Factors=zeros(1,FIdx1(C));
    for iw=1:C,
      w=W(iw);
      for i=1:w,
        I0=FIdx0(iw)+(i-1)*R(iw);
        I1=FIdx0(iw)+    i*R(iw)-1;
        temp=makepeak(ceil(R(iw)*0.08),R(iw),R(iw)*i/(w+1))';
        Factors(I0:I1)=temp/ss(temp);
      end;
    end;
  end;
 %Initialisation of the core if required
  if ~exist('G'),
    G=rand(size(GA));
    G=G.*GA;
  end;

  TotProdR=prod(R);
  TotProdW=prod(W);
  SSTot=sum(sum(Z.^2));
  MSETot=sqrt(SSTot);
  InterAct=1;
```

```
   OldSSE=Inf;
   converged=0;
   iter=0;
   while converged==0,

     iter=iter+1;

    %Dimension loop starts
    %Running through each C set of components.
     for c=1:C,

      %Create the order in which the serial kroneckers should be calculated.
       faclist=[c+1:C];
       if c-1>0,
         faclist=[faclist 1:c-1];
       end;

      %Create TmpKron to represent the seriel kronecker product in steps 1-2
      %1:Initiate with the three-way case

MatTmp1=cunfo(Factors(FIdx0(faclist(1)):FIdx1(faclist(1))),R(faclist(1)),W(
faclist(1)));

MatTmp2=cunfo(Factors(FIdx0(faclist(2)):FIdx1(faclist(2))),R(faclist(2)),W(
faclist(2)));
       TmpKron=ckron(MatTmp1',MatTmp2');
      %2:Continue with dimensions above three
       for i=3:C-1,
         MatTmp=Factors(FIdx0(faclist(i)):FIdx1(faclist(i)));
         MatTmp3=cunfo(MatTmp,R(faclist(i)),W(faclist(i)));
         TmpKron=ckron(TmpKron,MatTmp3');
       end;
      %3:Create the TmpMat4-matrix from which the factors will be updated
       TmpMat4=G*TmpKron;

      %3:Updating factor set c in TmpFac and then store to Factors
      %No constraints
       if (FacType(c)==1) | (FacType(c)==4),
         TmpFac=Z*TmpMat4'/(TmpMat4*TmpMat4');
              if FacType(c)==4,
               TmpFac=csetbase(TmpFac,0,0);
         end;
       end;
      %Orthogonality constraints
       if FacType(c)==2,
         TmpFac=Z*TmpMat4'*(TmpMat4*Z'*Z*TmpMat4')^(-1/2);
         if c==1,
           TmpC=(TmpFac'*TmpFac)\TmpFac'*Z*TmpMat4'/(TmpMat4*TmpMat4');
           TmpFac=TmpFac*diag(diag(TmpC));
         end;
       end;
      %Non-negativity constraints
       if FacType(c)==3,
         TmpFac=cnnls(eye(R(c)),Z*pinv(TmpMat4));
       end;
       TmpFac=normit(TmpFac);
       Factors(FIdx0(c):FIdx1(c))=TmpFac(:);
      %Non-negativity constraints (non-robust/non-leastsquares)
       TmpFac=normit(TmpFac);
       Factors(FIdx0(c):FIdx1(c))=TmpFac(:);


      %5:Updating the core by use of TmpFac and TmpKron
       G=(TmpFac'*TmpFac)\TmpFac'*Z*TmpKron'/(TmpKron*TmpKron');
      %Restricting G to resemble GA
       G=G.*GA;

      %6:Rearranging the data in Z
       if c<C,
        %Rearranges to next unfolding
```

```
        Z=cunfo(Z,TotProdR/R(c+1),R(c+1))';
       else
       %Must rearrange to first unfolding
        Z=cunfo(Z,TotProdR/R(1),R(1))';
      end;

     %7:Rearranging the core G and the allowed core GAllow
      if c<C,
        G=cunfo(G,TotProdW/W(c+1),W(c+1))';
        GA=cunfo(GA,TotProdW/W(c+1),W(c+1))';
       else
        G=cunfo(G,TotProdW/W(1),W(1))';
        GA=cunfo(GA,TotProdW/W(1),W(1))';
      end;

   %Dimension loop ends
    end;

   %Checks for convergence
   %Create TmpKron to represent the seriel kron-product in steps 8-9
   %8:Initiate with the three-way case
    MatTmp =cunfo(Factors(FIdx0(1):FIdx1(1)),R(1),W(1));
    MatTmp1=cunfo(Factors(FIdx0(2):FIdx1(2)),R(2),W(2));
    MatTmp2=cunfo(Factors(FIdx0(3):FIdx1(3)),R(3),W(3));
    TmpKron=ckron(MatTmp1',MatTmp2');
   %9:Continue with dimensions above three
    for i=4:C,
      MatTmp3=cunfo(Factors(FIdx0(i):FIdx1(i)),R(i),W(i));
      TmpKron=ckron(TmpKron,MatTmp3');
    end;
   %10:Calculate the model of Z at this stage of iteration
    TmpMat4=MatTmp*G*TmpKron;
    CurrSSE=sum(sum( (Z-TmpMat4).^2 ));
    MSE=sqrt(CurrSSE/NumOfElem);
    if abs(OldSSE-CurrSSE)<ConvLim,
      converged=1;
    end,
    if Options==0,
      fprintf('Iteration %g, MSE %12.8f, Expl. %10.6f %%
\n',iter,MSE,(1-(CurrSSE/SSTot))*100);
    end;
    OldSSE=CurrSSE;

   %Emergency break
    if isnan(CurrSSE),
      converged=2;
      fprintf('Inconsistency break - No solution ! \n')
    end;

%plotn(Factors,R,W,SSE,SSTot)

end;
SSE=OldSSE;

%Tell which conv. criteria was used
if (converged==1) & (Options==0),
  fprintf('Convergence request was reached.\n')
  fprintf('Solution derived - execution stopped.\r\n')
end;
if (converged==2) & (Options==0),
  fprintf('Infeasible/Inconsistent solution!.\n')
  fprintf('No solution derived - execution aborted.\r\n');
end;

%Calculate the variance explained
XExpl=100*(1 - sum(var(Z-TmpMat4))/sum(var(Z)))
```

TUCKSCOR.M

```
function [Scores]=tuckscor(Z,R,W,Factors,G,Way);
%[Scores,Levs,Factors]=tuckscor(Z,R,W,Factors,G,Way);
%
% This Matlab program (tuckscor.m) calculates the
% scores (factors) in way number 'Way' while
% using all other factors, and the core G, to do this.
%
% The math model is Z=A*G*((B'(x)C')(x)D')...
%
% Z         : Data array to be decomposed, assumed to be
%             unfolded according to CHM 2 (see 'fixformn.m')
% R         : Number of observations along each mode.
%             E.g. [10 6 101 14 13] or [3 4 4 4 6]
%             There must be at least three elements in R
%             corresponding to three or more dimensions.
% W         : Number of PC's to find along each mode
%             Ex. [3 3 3 3 3] or [3 2 3 3 2]
%             The lengths of the W and R vectors must be the same.
% Factors   : A complete set of factors must be given. The the program
%             selects the correct way to score using all the other and
%             the core to calculate the correct score values.
% G         : The core, used to relate the given factors that are used.
% Way       : The way(direction) in which the score vector is to be
%             determined must be specified in this scalar.
% Scores    : The returned values giving the ls fit to the data
%             through the given factors in all the other directions.
%             Of course also the core has to be used for this purpose.
% Levs      : The leverages for the score elements are calculated to
%             facilite a rough estimate of potential outliers.
%             High leverages means: Take in more more objects of this type
%             or exclude it (and the few other like it!).
%
% Author    : Claus Andersson, January 1995
%             Claus.Andersson@pop.foodsci.kvl.dk
%
% Revised   :
%
% Author    : Claus Andersson, December 1995
%             KVL, Copenhagen, Chemometrics Group
% E-mail    : claus.andersson@pop.foodsci.kvl.dk
%
%For copying to the Command Window:
%[Scores,Levs,Factors]=tuckscor(Z,R,W,Factors,G,Way);
%[Scores]=tuckscor(Z,R,W,Factors,G,Way);

 %Setting up the environment
  format compact

 %Number of dimensions of the data structure, C
  C=length(R(1,:));

 %Check for consistency
  if Way>C | Way<1,
    disp('Invalid ''Way''')
  end;

 %Determine the length of the score to return
  RScor=R;
  [a b]=size(Z);
  RScor(Way)=1;
  RScor(Way)=(a*b)/prod(RScor);

 %Create a table of starting indexes to the sets of factors in Factors
  FIdx0=cumsum([1 R(1:C-1).*W(1:C-1)]);
  FIdx1=cumsum([R.*W]);

  TotProdR=prod(R);
```

```
   TotProdRScor=prod(RScor);
   TotProdW=prod(W);
   SSTot=sum(sum(Z.^2));

    %Running through each C set of components.
     for c=1:C,

       if c==Way,

            %Create the order in which the serial kroneckers should be
calculated.
             faclist=[c+1:C];
             if c-1>0,
               faclist=[faclist 1:c-1];
             end;

            %Create TmpKron to represent the seriel kronecker product in
steps 1-2
            %1:Initiate with the three-way case

MatTmp1=cunfo(Factors(FIdx0(faclist(1)):FIdx1(faclist(1))),R(faclist(1)),W(
faclist(1)));

MatTmp2=cunfo(Factors(FIdx0(faclist(2)):FIdx1(faclist(2))),R(faclist(2)),W(
faclist(2)));
             TmpKron=ckron(MatTmp1',MatTmp2');
            %2:Continue with dimensions above three
             for i=3:C-1,
               MatTmp=Factors(FIdx0(faclist(i)):FIdx1(faclist(i)));
               MatTmp3=cunfo(MatTmp,R(faclist(i)),W(faclist(i)));
               TmpKron=ckron(TmpKron,MatTmp3');
             end;
            %3:Create the TmpMat4-matrix from which the factors will be
updated
             TmpMat4=G*TmpKron;
               TmpFac=Z*TmpMat4'/(TmpMat4*TmpMat4');
             Scores=TmpFac;

    end;
    %Scores have now been calculated

            %6:Rearranging the data in Z
             if c<C,
              %Rearranges to next unfolding
               Z=cunfo(Z,TotProdRScor/RScor(c+1),RScor(c+1))';
             else
              %Must rearrange to first unfolding
               Z=cunfo(Z,TotProdRScor/RScor(1),RScor(1))';
             end;

            %7:Rearranging the core G and the allowed core GAllow
             if c<C,
               G=cunfo(G,TotProdW/W(c+1),W(c+1))';
              else
               G=cunfo(G,TotProdW/W(1),W(1))';
             end;

   %Dimension loop ends
    end;
```

TUCKROTN.M

```
function [FactorRot,GRot,SSERot,GT]=tuckrotn(X,R,W,Factors,G,GAll);
%[FactorRot,GRot,SSERot,GT]=tuckrotn(X,R,W,Factors,G,GAll)
%
%Program for calculating transformation matrices for
%hyper cores of arbitrary dimensions.

C=length(R(1,:));

%Create a table of starting indexes to the sets of factors in Factors
FIdx0=cumsum([1 R(1:C-1).*W(1:C-1)]);
FIdx1=cumsum([R.*W]);
FIdxR0=cumsum([1 W(1:C-1).*W(1:C-1)]);
FIdxR1=cumsum([W.*W]);

%Initialization by rands
TransMat=rand(1,FIdxR1(C))+1;

%Initialize the core to 'aim at' GT
if ~exist('GAll'),
  for i=1:min(W),
    GAll(i,1:C)=i*ones(1,C);
  end;
end;

GT=zeros(size(G));
for i=1:length(GAll(:,1)),
 Indx=0;
  for c=2:C-1,
    Indx=Indx+(GAll(i,c)-1)*prod(W(c+1:C));
  end;
  Indx=Indx+GAll(i,C);
  GT(GAll(i,1),Indx)=1;
end;

%Calculate degree of structurel match before rotating
dsm=100*sum(sum((GT.*G).^2))/sum(sum(G.^2));
fprintf('Degree of structurel match before rotation : %.6g %%\r\n',dsm);

ConvLim=1E-8*sqrt(sum(sum(G.^2)));
converged=0;
CurrSSE=inf;
while converged==0,

  %Running through each C set of components.
  for c=1:C,

    %Create the order in which the serial kroneckers should be calculated.
    faclist=[c+1:C];
    if c-1>0,
      faclist=[faclist 1:c-1];
    end;

    %Create TmpKron to represent the seriel kronecker product in steps 1-2
    %1:Initiate with the three-way case

MatTmp1=reshape(TransMat(FIdxR0(faclist(1)):FIdxR1(faclist(1))),W(faclist(1
)),W(faclist(1)));

MatTmp2=reshape(TransMat(FIdxR0(faclist(2)):FIdxR1(faclist(2))),W(faclist(2
)),W(faclist(2)));
    TmpKron=kron(MatTmp1,MatTmp2);

    %2:Continue with dimensions above three
    for i=3:C-1,
      MatTmp=TransMat(FIdxR0(faclist(i)):FIdxR1(faclist(i)));
      MatTmp3=reshape(MatTmp,W(faclist(i)),W(faclist(i)));
      TmpKron=kron(TmpKron,MatTmp3);
```

```
      end;

      %3:Create the TmpMat4-matrix from which
      %  the transformations will be updated
      TmpMat4=G*TmpKron*TmpKron'*G';

      %4:Updating factor set c in TmpFac and then store to TransMat
      [UT ST VT]=svd(TmpMat4,0);
      [UGT SGT VGT]=svd(GT*GT',0);
      TmpFac=UGT\UT;
      TransMat(FIdxR0(c):FIdxR1(c))=TmpFac(:);

      %5:Determine the error in the first way
      if c==1,
        OldSSE=CurrSSE;
        CurrSSE=sum(sum((GT*GT'-TmpFac'*TmpMat4*TmpFac).^2));
      end;
      %6:Rearranging the core G and the allowed core GAll
      if c<C,
        TmpVec=W;
        TmpVec(c+1)=1;
        G=reshape(G,prod(TmpVec),W(c+1))';
        GT=reshape(GT,prod(TmpVec),W(c+1))';
      else
        TmpVec=W;
        TmpVec(1)=1;
        G=reshape(G,prod(TmpVec),W(1))';
        GT=reshape(GT,prod(TmpVec),W(1))';
      end;

    end;

    %Convergence check
    if abs(OldSSE-CurrSSE)<ConvLim,
      converged=1;
    end,

end;

%Rotate the factors from the mappings just determined.
FactorRot=zeros(size(Factors));

for c=1:C,
  Mat1=cunfo(Factors(FIdx0(c):FIdx1(c)),R(c),W(c));
  Mat2=cunfo(TransMat(FIdxR0(c):FIdxR1(c)),W(c),W(c));
  ResMat=Mat1*Mat2;
  FactorRot(FIdx0(c):FIdx1(c))=ResMat(:);
end;

%Create TmpKron to represent the seriel kronecker product
%Initiate with the three-way case
MatTmp1=reshape(TransMat(FIdxR0(1):FIdxR1(1)),W(1),W(1));
TmpKron=reshape(TransMat(FIdxR0(2):FIdxR1(2)),W(2),W(2));
%Continue with dimensions above three
for c=3:C,
  MatTmp=TransMat(FIdxR0(c):FIdxR1(c));
  MatTmp2=reshape(MatTmp,W(c),W(c));
  TmpKron=kron(TmpKron,MatTmp2);
end;
GRot=MatTmp1'*G*TmpKron;
%Calculate the new misfit
MatTmp =cunfo(FactorRot(FIdx0(1):FIdx1(1)),R(1),W(1));
MatTmp1=cunfo(FactorRot(FIdx0(2):FIdx1(2)),R(2),W(2));
MatTmp2=cunfo(FactorRot(FIdx0(3):FIdx1(3)),R(3),W(3));
TmpKron=ckron(MatTmp1',MatTmp2');
%Continue with dimensions above three
for i=4:C,
  MatTmp3=cunfo(FactorRot(FIdx0(i):FIdx1(i)),R(i),W(i));
  TmpKron=ckron(TmpKron,MatTmp3');
end;
```

```
%Calculate the model of X at this stage
TmpMat4=MatTmp*GRot*TmpKron;
SSERot=sum(sum( (X-TmpMat4).^2 ));

%Calculate degree of structurel match after rotation
dsm=100*sum(sum((GT.*GRot).^2))/sum(sum(GRot.^2));
fprintf('Degree of structurel match after rotation : %.6g %%\r\n',dsm);
```

PLSN.M

```
function
[T,P,U,Q,B,We,YPred,Xexpl,Yexpl,XFactors,XR,YFactors,YR]=plsn(X,Rx,Y,Ry,W,O
ptions);
%[T,P,U,Q,B,We,YPred,Xexpl,Yexpl,XFactors,XR,YFactors,YR]=plsn(X,Rx,Y,Ry,W,
Options);

  Cx=length(Rx(:));
  Cy=length(Ry(:));
  XOld=X;
  YOld=Y;
  TotVarX=sum(sum(X.^2));
  TotVarY=sum(sum(Y.^2));
  XTol=1E-8*sqrt(TotVarX);
  YTol=1E-8*sqrt(TotVarY);

  if Rx(1)~=Ry(1),
    disp('The arrays must have same number of oberservations'),
    disp('in the common score mode!'),
  end;

  if ~exist('Options'),
    Options=0;
  end;

  T=zeros(Rx(1),W);
  P=zeros(prod(Rx(2:length(Rx))),W);
  We=zeros(prod(Rx(2:length(Rx))),W);
  U=zeros(Ry(1),W);
  Q=zeros(prod(Ry(2:length(Ry))),W);
  B=eye(size(W));

  XR=Rx(2:length(Rx));
  lx=sum(XR);
  XFactors=zeros(W,lx);
  Rw=Rx(2:length(Rx));
  Cw=length(Rw);

  YR=Ry(2:length(Ry));
  ly=sum(YR);
  YFactors=zeros(W,ly);
  Rq=Ry(2:length(Ry(:)));
  Cq=length(Rq);

  conv1lim=1E-8;
  for i=1:W,
    u=Y(:,1);
    iter=0;
    oldt=zeros(Rx(1),1);

    if Options==0,
      disp(['Deriving PLS factor # ' int2str(i)])
    end;

    converged1=0;
    while converged1==0,
      iter=iter+1;

     %Create weights for columns in X, w
      we=(X'*u);

     %X is a matrix
      if Cw==1,
        XFactor=we';
        we=normit(we);
      end;
     %Decompose we using svd
      if Cw==2,
```

```
          [WArray1 S WArray2]=svd(cunfo(we,Rw(2),Rw(1))',0);
          we=ckron(WArray1(:,1)',WArray2(:,1)')';
         we=normit(we);
          XFactor=[WArray1(:,1)' WArray2(:,1)'];
        end;
       %Decompose W using PARAFAC
        if Cw>=3,
          In1=Rw(1);
          In2=prod(Rw(2:length(Rw)));

[XFactor]=parafacn(cunfo(we,In2,In1)',Rw,1,ones(size(Rw)),XTol,0,Options);
          XFactor(1:Rw(1))=normit(XFactor(1:Rw(1))')';
          we=XFactor;
         %Make the appropriate kronecker product of normalized we
          Indx0=1+[0 cumsum(Rw(1:(length(Rw)-1)))];
          Indx1=[cumsum(Rw)];
          TmpMat=we(Indx0(1):Indx1(1));
          TmpMat=normit(TmpMat');
          for in=2:Cw,
            TmpMat1=we(Indx0(in):Indx1(in));
            TmpMat1=normit(TmpMat1');
            TmpMat=ckron(TmpMat,TmpMat1);
          end;
          we=TmpMat;
          we=normit(we);
        end;

       %Get the scorevector, t
        t=X*we;

       %Calculate scores of Y
        q=Y'*t;

       %Decompose Y-weights as vector
        if Cq==1,
          YFactor=q';
          q=normit(q);
        end;
       %Decompose Y-weights using svd
        if Cq==2,
          [WyArray1 S WyArray2]=svd(cunfo(q,Rq(2),Rq(1))',0);
          q=ckron(WyArray1(:,1)',WyArray2(:,1)')';
          YFactor=[WyArray1(:,1)' WyArray2(:,1)'];
          q=normit(q);
        end;
       %Decompose Y-weights using PARAFAC
        if Cq>=3,

[YFactor]=parafacn(cunfo(q,Rq(1),prod(Rq(2:length(Rq)))),Rq,1,ones(size(Rq)
),YTol,0,Options);
          YFactor(1:Rq(1))=normit(YFactor(1:Rq(1))')';
          qwe=YFactor;
          YR=Rq;
         %Make the appropriate kronecker produck of normalized we
          Indx0=1+[0 cumsum(Rq(1:(length(Rq)-1)))];
          Indx1=[cumsum(Rq)];
          TmpMat=qwe(Indx0(1):Indx1(1));
          TmpMat=normit(TmpMat');
          for in=2:Cw,
            TmpMat1=qwe(Indx0(in):Indx1(in));
            TmpMat1=normit(TmpMat1');
            TmpMat=ckron(TmpMat,TmpMat1);
          end;
          q=TmpMat;
          q=normit(q);
        end;

       %Make score for Y
       u=Y*q;
```

```
      currdif=sum(sum((t-oldt).^2));
      if currdif<conv1lim,
        converged1=1;
      end;

      oldt=t;

      if Options==0,
        disp(['Iteration: ' int2str(iter) ',  difference: '
num2str(currdif)])
      end;

    end;

   %Store the weights
    XFactors(i,1:lx)=XFactor;
    YFactors(i,1:ly)=YFactor;

   %Store the T's
    T(:,i)=t;

   %Store the W's
    We(:,i)=we;

   %Store the U's
    U(:,i)=u;

   %Store the Q's
    Q(:,i)=q;

   %Calculate regression coefficients, B
    B(1:i,i)=(T(:,1:i)'*T(:,1:i))\T(:,1:i)'*u;

   %Calculate the model of Y
    Ymod=T(:,1:i)*B(1:i,1:i)*Q(:,1:i)';
    Y=YOld-Ymod;

   %Calculate the model of X
    Xmod=T*We';
    X=XOld-Xmod;

  end;

  Xexpl=1 - sum(sum(X.^2))/TotVarX;
  Yexpl=1 - sum(sum(Y.^2))/TotVarY;

  YPred=T*B*Q';
```

References

Atkins, P. W.
Physical Chemistry, 4th Ed.
Oxford University Press, Oxford, 1990

Bro, R.
Multiway calibration. Multilinear PLS
Journal of Chemometrics, 10, 47-61 (1996)

Kruskal, J.
Multiway data analysis
NHPC, North-Holland, 1989
Edited by Coppi, R., Bolasco, S.

DDS, De Danske Sukkerfabrikker
Om sukker og sukkerfabrikation hos De Danske Sukkerfabrikker
Aktieselskabet De Danske Sukkerfabrikker, Copenhagen, 1985 (in Danish)

Drewnowska, W., Waleriañczyk, E., Butwilowicz, A., Jarzebiñski, J., Fitak, B., Gajewska, M.
Contents of free amino acids and amides in sugar beet and sugar industry
Acta Alimentaria Polonica, 5 (4), 315-322 (1979)

Ewing, G. W.
Instrumental methods of chemical analysis
McGraw-Hill, Singapore, 1985

Fabricius-Bjerre, Fr.
Lærebog i Geometri
Polyteknisk Forlag, Lyngby, 1987 (in Danish)

Geladi, P.
Analysis of multi-way (multi-mode) data
Chemometrics and Intelligent Laboratory Systems, 7, 11-30 (1990)

Gross, D., Coombs, J.
Enzymic colour formation in beet and cane juices
International Sugar Journal, 78, 69-109 (1976)

Handbook of chemistry and physics, 67th ed.
Chemical Rubber Company, Ohio, 1989

Harshman, R. A., Lundy, M.E.
PARAFAC: Parallel factor analysis
Computational Statistics & Data Analysis, 18, 39-72 (1994)

Heimdahl, H.
Enzymatisk brunfarvning af snittet salat
Mejeri- og Levnedsmiddelinstituttet, KVL, 1995 (in Danish)

Henrion, R.
N-way principal component analysis
Theory, algorithms and applications
Chemometrics and Intelligent Laboratory Systems, 25, 1-23 (1994)

Henrion, R.
Body diagonalization of core matrices in three-way principal component analysis: Theoretical bounds and simulation
Journal of Chemometrics, 7, 477-494 (1993)

Kaipainen, A., Laitinen, M.
The effect of pH on volatile compunds in molasses
16th ISCC, Poster, 1994

Kiers, H. A. L.
TUCKALS core rotations and constrained TUCKALS modelling.
Statistica Applicata, 4, 659-667 (1992)

Kroonenberg, P. M.
Three-mode principal component analysis
DSWO Press, Leiden, 1983 (reprinted 1989)

Larsson, H.
Svenskt socker
Sockerbolaget, Vejbyställe, 1989 (in Sswedish)

Lawson, C. L., Hanson, R. J.
Solving least squares problems
Prentice-Hall Inc., New Jersey, 1994

Maag, G.W., Hecker, R. J., Whitaker, P. A.
Nitrogeneous compounds in in sugar beet juices
J. of the ASSBT, 17, 154-163 (1972)

Macrae, R., Robinson, R. K, Sadler, M. J.
Encyclopaedia of food science, food technology and nutrition
Academic Press, London (1993)

Madsen, R. F., Nielsen, W. K., Winstrøm-Olsen, B., Nielsen, T. E.
Formation of colour compunds in production of sugar from sugar beet
Sugar Technology Reviews, 6, 49-115 (1978)

Martens, H., Næs, T.
Multivariate Calibration
Wiley, England, 1989

McGhie, T. K.
Analysis of sugarcane flavonoids by capillary zone electrophoresis
Journal of Chromatography, 634, 107-112 (1993)

Mitchell, B. C., Burdick, D. S.
Slowly converging PARAFAC sequences: Swamps and two-factors degeneracies
Journal of Chemometrics, 8, 155-168 (1994)

Morrison, R. T., Boyd, R. N.
Organic Chemistry, 5. Ed.
Allyn and Bacon Inc., New York, 1987

Namiki, M. et al.
Weak chemilumiscence at an early stage of the Maillard reaction
Journal of agricultural food chemistry, 41, 1704-1709 (1993)

Perrion, D. D., Dempsey, B.
Buffers for ion and pH control
Chapman and Hall, London, 1974

R.S. Shallenberger, R.S., Birch, G. C.
Sugar Chemistry
The AVI Publishing Company, Westport, 1975

Schneider, F. et al.
Technologie des zuckers
M. & H. Schaper, Hannover, 1966

Schulman, S. G.
Fluorescence and phosphorescence spectroscopy:
Physicochemical principles and practice
Pergamon Press, Great Britain, 1979

Stryer, L.
Biochemistry, 3. Ed.
W. H. Freeman and Company, New York, 1988

Winstrøm-Olsen, B.
Enzymic colour formation in sugar beet, Part II
International Sugar Journal, 137-140 (1982)

Winstrøm-Olsen, B., Madsen, R. F., Kofod Nielsen, W.
Sugar beet phenols,
Investigation of phenolic compunds from sugar beet in relation to the formation of colour
International Sugar Journal, 81, 332-336 and 362-367 (1979)

Winstrøm-Olsen, B.
Enzymic colour formation in sugar beet, Part I
International Sugar Journal, 102-105 (1981)

Wold, S., Esbensen, K., Geladi, P.
Principal Component Analysis
Chemometrics and Intelligent Laboratory Systems, 2, 37-52 (1987)

Wold, S., P. Geladi, Esbensen, K., Öhman, J.
Multi-way principal components- and PLS-analysis
Journal of Chemometrics, 1, 41-56 (1987)

Index