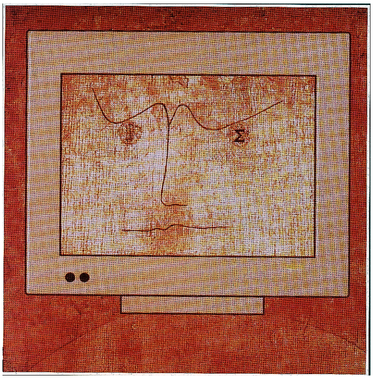
GWDG-Bericht Nr. 63

Kurt Kremer, Volker Macho (Hrsg.)

Forschung und wissenschaftliches Rechnen

Beiträge zum Heinz-Billing-Preis 2003



Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Forschung und
wissenschaftliches Rechnen

Titelbild:
Logo nach Paul Klee „Der Gelehrte“, Modifizierung durch I. Tarim,
Max-Planck-Institut für Psycholinguistik, Nijmegen.

GWDG-Bericht Nr. 63

Kurt Kremer, Volker Macho (Hrsg.)

Forschung und wissenschaftliches Rechnen

Beiträge zum Heinz-Billing-Preis 2003

Gesellschaft für wissenschaftliche Datenverarbeitung
Göttingen

2004

Die Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen ist eine gemeinsame Einrichtung der Max-Planck-Gesellschaft zur Förderung der Wissenschaften e. V. und des Landes Niedersachsen. Die Max-Planck-Gesellschaft hat diesen Bericht finanziell gefördert.

Redaktionelle Betreuung:
Volker Macho (Max-Planck-Institut fuer Polymerforschung, Mainz)

Satz:
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen,
Am Faßberg, D-37077 Göttingen

Druck: klartext GmbH, D-37073 Göttingen, www.kopie.de

ISSN: 0176-2516

Inhalt

Weitere Beiträge für den Heinz-Billing-Preis 2003

Vorwort

Der vorliegende Band ist der elfte Band der Reihe „Forschung und wissenschaftliches Rechnen“. Er enthält sieben der neun für den Heinz-Billing-Preis des Jahres 2003 eingereichten Beiträge. In diesem Jahr ging der Preis erstmalig an eine Forschergruppe außerhalb der Max-Planck-Gesellschaft, und zwar an Roland Chrobok, Sigurdur F. Hafstein und Andreas Pottmeier vom Fachbereich Physik der Universität Duisburg-Essen. Sie wurden ausgezeichnet für ihre Arbeit „OLSIM: A New Generation of Traffic Information Systems“. OLSIM errechnet aus Verkehrsdaten verschiedenster Quellen den aktuellen Verkehrszustand des Autobahnnetzes von Nordrhein-Westfalen und simuliert die zukünftige Verkehrssituation. Über die Internetseite www.autobahn.nrw.de kann sich jeder über den aktuellen Verkehrszustand und über Prognosen für die nächsten 30 und 60 Minuten informieren.

Weitere Auszeichnungen erhielten eine Arbeit zur Berechnung von Feynman Diagrammen sowie ein Programmpaket zur Durchführung von MD/MC Simulationen an Soft Matter. Auch die übrigen eingereichten und hier veröffentlichten Beiträge decken ein breites Spektrum ab, angefangen von Problemen der empirischen Sozialforschung über Visualisierung von 3-dimensionalen Datensätzen in der Tomografie und Generierung und Visualisierung von komplexen NMR Pulssequenzen bis hin zur Berechung von komplexen Transportphänomenen.

Viele der eingereichten Arbeiten benutzen im Internet frei verfügbare und standardisierte Module. Darüber hinaus hat sich die Art und Weise geändert, wie die Autoren über die Nutzung und Weiterverbreitung der eingereichten Programme denken. Während es in der Vergangenheit in der Regel darum ging, mit Hilfe der Datenverarbeitung Lösungen für ein spezielles im Rahmen der eigenen Forschungsarbeit aufgetretenes Problem zu finden, gehen inzwischen immer mehr der Wissenschaftler, die ihre Arbeiten für den Heinz-Billing-Preis einreichen, dazu über, ihre Ansätze der wissenschaftlichen Community über das Internet zur Verfügung zu stellen. Das findet seinen Ausdruck in verschiedenen Beiträgen, die hier vorgestellt werden.

Ein herzlicher Glückwunsch geht an dieser Stelle an den Stifter des Preises, Herrn Prof. H. Billing, der im Februar dieses Jahres 90 Jahre alt geworden ist. Wir möchten uns im Namen der Heinz-Billing-Vereinigung und im Namen aller Preisträger der letzten Jahre sehr für sein großes Interesse am wissenschaftlichen Rechnen in seiner ganzen Vielfalt sowie seine fortwährende Unterstützung bedanken.

Bedanken wollen wir uns auch bei Herrn Günter Koch, GWDG, für die Umsetzung der eingesandten Manuskripte in eine für das Offsetdruckverfahren kompatiblen Druckvorlage.

Die Vergabe des Preises wäre ohne Sponsoren nicht möglich. Wir danken der Firma IBM Deutschland, welche für 2003 als Hauptsponsor aufgetreten ist.

Die hier abgedruckten Arbeiten sind ebenfalls im Internet unter der Adresse

www. billingpreis.mpg.de

zu finden.

Kurt Kremer, Volker Macho

Der Heinz-Billing-Preis 2003

Ausschreibung des Heinz-Billing-Preises 2003 zur Förderung des wissenschaftlichen Rechnens

Im Jahre 1993 wurde zum ersten Mal der Heinz-Billing-Preis zur Förderung des wissenschaftlichen Rechnens vergeben. Mit dem Preis sollen die Leistungen derjenigen anerkannt werden, die in zeitintensiver und kreativer Arbeit die notwendige Hard- und Software entwickeln, die heute für neue Vorstöße in der Wissenschaft unverzichtbar sind.

Der Preis ist benannt nach Professor Heinz Billing, emeritiertes wissenschaftliches Mitglied des Max-Planck-Institutes für Astrophysik und langjähriger Vorsitzender des Beratenden Ausschusses für Rechenanlagen in der Max-Planck-Gesellschaft. Professor Billing stand mit der Erfindung des Trommelspeichers und dem Bau der Rechner G1, G2, G3 als Pionier der elektronischen Datenverarbeitung am Beginn des wissenschaftlichen Rechnens.

Der Heinz-Billing-Preis zur Förderung des wissenschaftlichen Rechnens steht unter dem Leitmotiv

„EDV als Werkzeug der Wissenschaft“.

Es können Arbeiten eingereicht werden, die beispielhaft dafür sind, wie die EDV als methodisches Werkzeug Forschungsgebiete unterstützt oder einen neuen Forschungsansatz ermöglicht hat.

Der folgende Stichwortkatalog mag den möglichen Themenbereich beispielhaft erläutern:

– Implementation von Algorithmen und Softwarebibliotheken
– Modellbildung und Computersimulation
– Gestaltung des Benutzerinterfaces
– EDV gestützte Meßverfahren
– Datenanalyse und Auswertungsverfahren
– Visualisierung von Daten und Prozessen

Die eingereichten Arbeiten werden referiert und in der Buchreihe "Forschung und wissenschaftliches Rechnen" veröffentlicht.

Die Jury wählt einen Beitrag für den mit € 3000,- dotierten Heinz-Billing-Preis 2003 zur Förderung des wissenschaftlichen Rechnens aus. Die Beiträge, in deutscher oder englischer Sprache abgefasst, müssen keine Originalarbeiten sein und sollten möglichst nicht mehr als fünfzehn Seiten umfassen.

Da zur Bewertung eines Beitrages im Sinne des Heinz-Billing-Preises neben der technischen EDV-Lösung insbesondere der Nutzen für das jeweilige Forschungsgebiet herangezogen wird, sollte einer bereits publizierten Arbeit eine kurze Ausführung zu diesem Aspekt beigefügt werden.

Der Heinz-Billing-Preis wird jährlich vergeben. Die Preisverleihung findet anlässlich des 20. DV–Treffens der Max-Planck-Institute am 20. November 2003 in Göttingen statt.

Beiträge für den Heinz-Billing-Preis 2002 sind bis zum 15. Juli 2003 einzureichen.

Heinz-Billing-Preisträger

1993: Dr. Hans Thomas Janka, Dr. Ewald Müller, Dr. Maximilian Ruffert
Max-Planck-Institut für Astrophysik, Garching

Simulation turbulenter Konvektion in Supernova-Explosionen in massereichen Sternen

1994: Dr. Rainer Goebel
Max-Planck-Institut für Hirnforschung, Frankfurt

- Neurolator - Ein Programm zur Simulation neuronaler Netzwerke

1995: Dr. Ralf Giering
Max-Planck-Institut für Meteorologie, Hamburg

AMC: Ein Werkzeug zum automatischen Differenzieren von Fortran Programmen

1996: Dr. Klaus Heumann
Max-Planck-Institut für Biochemie, Martinsried

Systematische Analyse und Visualisierung kompletter Genome am Beispiel von S. cerevisiae

1997: Dr. Florian Mueller
Max-Planck-Institut für molekulare Genetik, Berlin

ERNA-3D (Editor für RNA-Dreidimensional)

1998: Prof. Dr. Edward Seidel
Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, Potsdam

Technologies for Collaborative, Large Scale Simulation in Astrophysics and a General Toolkit for solving PDEs in Science and Engineering

1999: Alexander Pukhov
Max-Planck-Institut für Quantenoptik, Garching

High Performance 3D PIC Code VLPL:
Virtual Laser Plasma Lab

2000: Dr. Oliver Kohlbacher
Max-Planck-Institut für Informatik, Saarbrücken
BALL – A Framework for Rapid Application Development in Molecular Modeling

2001: Dr. Jörg Haber
Max-Planck-Institut für Informatik, Saarbrücken
MEDUSA, ein Software-System zur Modellierung und Animation von Gesichtern

2002: Daan Broeder, Hennie Brugman und Reiner Dirksmeyer
Max-Planck-Institut für Psycholinguistik, Nijmegen
NILE: Nijmegen Language Resource Environment

2003: Roland Chrobok, Sigurður F. Hafstein und Andreas Pottmeier
Universität Duisburg-Essen
OLSIM: A New Generation of Traffic Information Systems

Das Kuratorium des Heinz-Billing-Preises

Prof. Dr. Heinz Billing
Emeritiertes Wissenschaftliches Mitglied des Max-Planck-Institut für Astrophysik, Garching

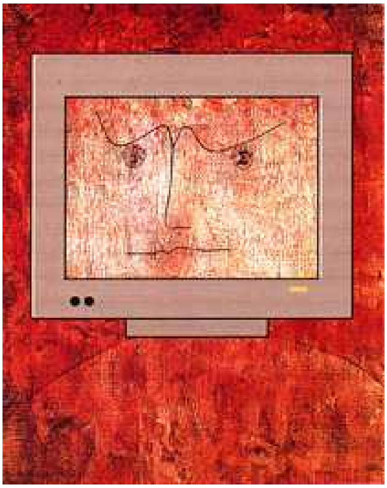
Prof. Dr. Friedel Hossfeld
Forschungszentrum Jülich GmbH, Jülich

Prof. Dr. K. Ulrich Mayer
Max-Planck-Institut für Bildungsforschung, Berlin

Prof. Dr. Stefan Müller
Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig

Prof. Dr. Jürgen Renn
Max-Planck-Institut für Wissenschaftsgeschichte, Berlin

Prof. Dr. H. Wolfgang Spiess
Max-Planck-Institut für Polymerforschung, Mainz



Roland Chrobok, Sigurður F. Hafstein und Andreas Pottmeier,
Universität Duisburg-Essen,

erhalten den

Heinz-Billing-Preis 2003

zur Förderung

des wissenschaftlichen Rechnens

als Anerkennung für ihr Projekt

OLSIM: A New Generation of Traffic Information Systems

Laudatio

Der Heinz-Billing-Preis 2003 wird für das Programmpaket *OLSIM: A New Generation of Traffic Information Systems* verliehen. Durch die Online Vernetzung von aktuellen Verkehrsdaten aus einem flächendeckenden Netz von Messpunkten mit modernen Verfahren der Computersimulation von Verkehrsflüssen lassen sich mit bisher nicht gekannter Zuverlässigkeit gegenwärtige Verkehrszustände analysieren und zukünftige vorhersagen. Das Verkehrsinformationssystem OLSIM ist über Internet öffentlich zugänglich und gibt jedem Benutzer die Möglichkeit (für das Autobahnnetz von Nordrhein-Westfalen) diese Informationen zu nutzen.

Bei den Prognosen werden neben den aktuellen Verkehrsdaten zeitbezogene typische „Fingerabdrücke" der einzelnen Strecken berücksichtigt. Das Programmpaket stellt damit einen wichtigen Schritt für eine optimale Information des einzelnen Verkehrsteilnehmers sowie ein wesentlich verbessertes Verkehrsleitsystem dar.

Verleihung des Heinz-Billing-Preises 2003 durch Prof. Kurt Kremer (mitte) an Roland Chrobock (links) und Sigurður F. Hafstein (rechts)

OLSIM: A New Generation of Traffic Information Systems

Roland Chrobok, Sigurður F. Hafstein, Andreas Pottmeier

University Duisburg-Essen

Summary

Detailed and reliable information about the current traffic state is hardly obtainable by the road user. Therefore, we propose a web based visualization of the current and future traffic load of the autobahn network of North Rhine-Westphalia, Germany. This novel traffic information system called OLSIM is based on an efficient and highly realistic traffic flow model, which is fed by traffic data of 4,000 detecting devices across the road network every minute, and a graphical user interface which can be accessed at *www.autobahn.nrw.de*.

1. Introduction

Since the construction of the first autobahn in Germany in the early 30th between Bonn and Cologne, the vehicular traffic has risen in a dramatic manner, especially in North Rhine-Westphalia. Whereas at first the autobahns could handle the traffic demand easily, nowadays, particularly in densely populated regions, the existing autobahn network has reached its capacity limit. The daily occurring traffic jams cause significant economic damage. Moreover, in these areas, it is usually hardly possible and socially untenable to enlarge the existing network. This is in particular true for the German state of North Rhine-Westphalia. The network is not able to cope with the demand in the daily rush hours and the drivers have to deal with the traffic jams in the Rhine-Ruhr region (Dortmund, Duisburg, Essen, Krefeld, Düsseldorf, etc.) and around Cologne (Leverkusen, Neuss, etc.). The prognosis for the future paints an even worse picture as the demand will increase further. New information systems and traffic management concepts are thus truly needed.

Therefore, we established the advanced traffic information system OLSIM which gives the internet user the opportunity to get information about the current traffic state, a 30, and a 60 minute prognosis of the autobahn network of North Rhine-Westphalia. Our approach to generate the traffic state in the whole autobahn network is to use locally measured traffic data, mainly provided by about 4,000 loop detectors as the input into an advanced cellular automaton traffic simulator. These measured data, which are delivered minute by minute, include especially the number of vehicles and trucks passed, the average speed of the passenger cars and trucks, and the occupancy, i.e., the sum of the times a vehicle covers the loop detector. The simulator does not only deliver information about the traffic states in regions not covered by measurement, but also gives reasonable estimates for other valuable quantities like travel times for routes, a quantity that is not directly accessible from the measurements of the detectors. As a further improvement we combine the current traffic data and heuristics of aggregated and classified traffic data to forecast the future traffic state. In the first step we gave a short-term forecast for 30 minutes, which was extended in the next step by a 60 minute prognosis. This information is completed by the temporal and spatial road work and actual road closures. All these valuable traffic information is integrated in a Java applet that can be accessed by every internet user at *www.autobahn.nrw.de.*

2. General Concept of the Traffic Information System OLSIM

The intention in developing the traffic information system OLSIM is to offer the opportunity to inform the road user fast and efficient about the current and the predictive traffic state. Therefore, the information mentioned above has to be collected and prepared in a manner that is useful for the user. The general setup of the traffic information system OLSIM is depicted in Fig. 1.

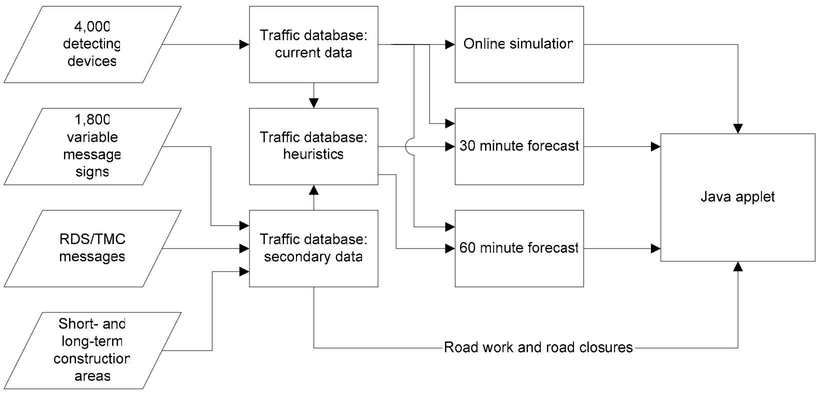

4,000 detecting devices
1,800 variable message signs
RDS/TMC messages
Short- and long-term construction areas
Traffic database: current data
Traffic database: heuristics
Traffic database: secondary data
Online simulation
30 minute forecast
60 minute forecast
Java applet
Road work and road closures

Fig. 1: The architecture of the traffic information system OLSIM.

First of all the different kinds of data have to be collected. Especially, the traffic data is stored in a database. These are sent from 4,000 loop detectors to the central OLSIM server every minute. The same holds for the data of the control states of about 1,800 variable message signs (VMS) that are located across the network. Furthermore, the data of road works are sent from the traffic centrals to OLSIM. The messages of short term construction areas are sent daily, those of permanent construction areas every two weeks. The data include the location and the duration of the construction area and an estimate whether the construction area will cause congestion or not.

Another data source are the so called RDS/TMC-messages. These messages are information provided by the traffic warning service and include all kind of warnings concerning the current traffic like traffic jams, accidents, road closures, and reroutings. These data are sent to the OLSIM server immediately when they are generated.

To generate a valid picture of the traffic state many kinds of data fusion techniques are needed. First, the actual traffic data are integrated into the microscopic traffic simulator. Using it, every vehicle that is measured at any time at one of the 4,000 loop detectors is directly fed into the simulation and virtually moves on. In this way the point information of the loop detectors is merged into a network wide traffic state. Such simulations are running for the current traffic state, for the 30 minutes, and for the 60 minutes forecast. In contrast to the online simulation, the forecasts are based on a combination of the actual traffic data and heuristics that are frequently generated and stored in a second database.

These heuristic traffic patterns are aggregated data which are classified in different days (work days, holidays, etc.) and secondary data like road constructions, variable message signs, and special events.

The second level of data fusion is done in the java applet at the website *www.autobahn.nrw.de*. Traffic state, construction areas, and road closures are integrated in one graphical user interface. Here each section is colored according to its calculated traffic state. Moreover, the construction areas and the road closures are marked in the map at their location. Their temporal parameters are shown in the status bar. The user can easily choose between the current traffic situation, the 30, and the 60 minute prognosis.

The microscopic traffic simulation, on which the core of the information system is based, is focused on in the next sections. So, in the following the traffic simulator, the topology, and some special problems which arise when such a complex network is mapped in the computer are explained in detail.

3. Simulation Model

The kernel of the online simulation is an advanced and highly realistic traffic simulation model. Because the data is fed into the simulator and processed by it every minute it has to be at least real-time. Due to their design cellular automata models are very efficient in large-scale network simulations (5, 11, 20, 22, 24). The first cellular automaton model for traffic flow that was able to reproduce some characteristics of real traffic like jam formation was suggested by Nagel and Schreckenberg (18) in 1992. Their model has been continuously refined in the last 10 years. The model we implemented in our simulator uses smaller cells in comparison with the original Nagel-Schreckenberg model, a slow-to-start rule, anticipation, and brake lights. With these extensions the cellular automaton traffic model is able to reproduce all empirically observed traffic states. Further, we use two classes of different vehicles, passenger cars and trucks, where the trucks have a lower maximum velocity and different lane changing rules.

Smaller cells allow for a more realistic acceleration and more speed bins. Currently an elementary cell size of 1.5 m is used, in contrast to the 7.5 m in the original Nagel-Schreckenberg model. A vehicle occupies 2-5 consequent cells. This corresponds to speed bins of 5.4 km/h and an acceleration of 1.5 m/s^2 (0-100 km/h in 19 s), which is of the same order as the "comfortable" acceleration of about 1 m/s^2. By using velocity dependent randomization (1), realized through the introduction of 'slow-to-start rules', meta stable traffic flows can be reproduced in the simulation, a phenomenon observed in empirical studies of real traffic data (7, 12, 25). The inclusion of anticipation and brake lights (2, 15) in the modeling leads to a more realistic driving, i.e., the cars no longer determine their velocity solely in dependency of the distance to the next car in front, but also take regard to its speed and whether it is reducing its speed or not.

In the Nagel-Schreckenberg model there is only one global parameter, the probability constant (or dawdling parameter) *p*, and every vehicle, say vehicle *n*, is completely determined by two parameters: its position $x_n(t)$ and its velocity $v_n(t) \in \{0, 1, \dots, v_{max}\}$ at time *t*. When the vehicle *n* decides in the time-step t • $t+1$ how fast it should drive, it does this by considering the distance $d_{n,m}(t)$, i.e., the number of empty cells, to the next vehicle *m* in front. The modifications mentioned above of the Nagel-Schreckenberg model imply that we have to add some new parameters to the model. When the simulation algorithm decides whether a vehicle *n* should brake or not it does not only consider the distance to the next vehicle *m* in front, but estimates how far the vehicle *m* will move during this time-step (anticipation). Note, that the moves are done in parallel, so the model remains free of collision. This leads to the effective gap

$$d_{n,m}^{eff}(t) := d_{n,m}(t) + \max(v_m^{\min}(t) - d_s, 0)$$

seen by vehicle *n* at time *t*. In this formula d_S is a safety distance and

$$v_m^{\min}(t) := \min(d_{m,l}(t), v_m(t)) - 1$$

is a lower bound of how far the vehicle *m* will move during this time-step. $d_{m,l}(t)$ is the number of free cells between car *m* and car *l* in front. Brake lights are further components of the anticipating driving. They allow drivers to react to disturbances in front earlier by adjusting their speed. The variable $b_n(t) = on$ if car *n* has its brake lights on and $b_n(t) = off$ if they are off.

Several empirical observations suggest that drivers react in a temporal- rather than a spatial-horizon (6, 17). For this reason the velocity-dependent temporal interaction horizon

$$t_n^s(t) := \min(v_n(t), h)$$

is introduced in the model. The constant *h* determines the temporal range of interaction with the brake light $b_m(t)$ of the car *m* ahead. Car *n* does only react to $b_m(t)$ if the time to reach the back of car *m*, assuming constant velocity ($v_n = const.$) and car *m* standing still, is less than $t_n^s(t)$, i.e.,

$$t_{n,m}^h(t) := \frac{d_{n,m}(t)}{v_n(t)} < t_n^s(t).$$

The estimations for *h* vary from 6 s (6), 8 s (17), 9 s (10) to 11 s (4). Another estimation can be obtained from the analysis of the perception sight distance. In (21) velocity-dependent perception sight distances are presented that, for velocities up to 128 km/h, are larger than 9 s. Therefore *h* is set to 6 s as a lower bound for the time headway (16).

The third modification of the Nagel-Schreckenberg model implemented in the simulator is a velocity dependent randomization, which means that the probability constant *p* is replaced with a probability function dependent on the velocity of the vehicle. Further, the probability is also a function of the brake light of the next vehicle in front. In every time-step for every vehicle *n* with vehicle *m* next in front, the probability that the vehicle *n* brakes is

$$p = p(v_n(t), b_m(t)) := \begin{cases} p_b, & \text{if } b_m(t) = on \text{ and } t^h_{n,m}(t) < t^s_n(t), \\ p_0, & \text{if } v_n(t) = 0, \\ p_d, & \text{default.} \end{cases}$$

The parameter p_0 tunes the upstream velocity of a wide moving jam and p_d controls the strength of the fluctuations.

With this parameter set the model is calibrated to the empirical data. The best agreement can be achieved for $d_s = 7$ cells, $h = 6$, $p_b = 0.96$, $p_0 = 0.5$, and $p_d = 0.1$. For a detailed analysis of the parameter set see (16).

To sum up, to move the vehicles forward in the network the algorithm executes the following steps in parallel for all vehicles *n*:

3.1. *Move forward (drive):*

– Step 0: Initialization:

For car *n* find the next car *m* in front. Set $p_n(t) := p(v_n(t), b_m(t))$ and $b_n(t+1) :=$ *off*.

– Step 1: Acceleration:

$$v_n(t + \frac{1}{3}) := \begin{cases} v_n(t), & \text{if } b_n(t) = on \text{ or } (b_m(t) = on \text{ and } t^h_n(t) < t^s_n(t)), \\ \min(v_n(t) + 1, v_{\max}), & \text{default.} \end{cases}$$

– Step 2: Braking:

$$v_n(t+\frac{2}{3}) := \min(v_n(t+\frac{1}{3}), d_{n.m}^{eff}(t)).$$

Turn brake light on if appropriate:

$$\text{if } v_n(t+\frac{2}{3}) < v_n(t), \quad \text{then } b_n(t+1) := on.$$

– Step 3: Randomization with probability *$p_n(t)$*:

$$v_n(t+1) := \begin{cases} \max(v_n(t+\frac{2}{3})-1,0), & \text{with probability } p_n(t), \\ v_n(t+\frac{2}{3}), & \text{default.} \end{cases}$$

Turn brake light on if appropriate:

$$\text{if } p = p_b \text{ and } v_n(t+1) < v_n(t+\frac{2}{3}), \quad \text{then } b_n(t+1) := on.$$

– Step 4: Move (drive):

$$x_n(t+1) := x_n(t) + v_n(t+1).$$

Free lane changes are needed so that vehicles can overtake slower driving passenger cars and trucks. When designing rules for the free lane changes, one should take care of that overtaking vehicles do not disturb the traffic on the lane they use to overtake to much, and one has to take account of German laws, which prohibit overtaking a vehicle to the left. Further, it is advantageous to prohibit trucks to drive on the leftmost lane in the simulation, because a truck overtaking another truck forces all vehicles on the left lane to reduce their velocity and produces a deadlock that may not resolve for a long time (14).

One more variable is needed for the free lane changes, $l_n \in \{$*left*, *right*, *straight*$\}$ notes if the vehicle *n* should change the lane during the actual time-step or not. This variable is not needed if the lane changes are executed sequentially, but we prefer a parallel update. For the left free lane changes the simulator executes the following steps parallel for all vehicles *n*:

3.2. Overtake on the lane to the left:

– Step 0: Initialization:

For car n find the next car m in front on the same lane, the next car *s* in front on the lane left to car *n*, and the next car *r* behind car *s*. Set l_n := *straight*.

– Step 1: Check lane change:

if $b_n(t) = off$ and $v_n(t) > d_{n,m}(t)$ and $d^{eff}_{n,s}(t) \geq v_n(t)$ and $d_{r,n}(t) \geq v_r(t)$,
then set $l_n := left$.

– Step 2: Do lane change:

if l_n = left, then let car *n* change lane to the left.

The definition of the gaps $d^{eff}_{n,s}(t)$ and $d^{eff}_{r,n}(t)$ in the lane-change-blocks is an obvious extension of the above definition; one simply inserts a copy of the car *n* on its left or right side. These overtake rules used by the simulator can verbally be summed up as follows: first, a vehicle checks if it is hindered by the predecessor on its own lane. Then it has to take into account the gap to the successor and to the predecessor on the lane to the left. If the gaps allow a safe change the vehicle moves to the left lane. For the right free lane changes the simulator executes the following steps parallel for all vehicles n:

3.3. Return to a lane on the right:

– Step 0: Initialization:

For car *n* find the next car *m* in front on the same lane, the next car *s* in front on the lane right to car *n*, and the next car *r* behind car *s*. Set l_n := straight.

– Step 1: Check lane change:

if $b_n(t) = off$ and $t^h_{n,s}(t) > 3$ and $(t^h_{n,m}(t) > 6$ or $v_n(t) > d_{n,m}(t))$ and $d_{r,n}(t) > v_r(t)$,
then set l_n := right.

– Step 2: Change lane:

if l_n = right, then let car n change lane to the right.

Thus, a vehicle always returns to the right lane if there is no disadvantage in regard to its velocity and it does not hinder any other vehicle by doing so.

It should be noted, that it is not possible to first check for all lane changes to the left and to the right and then perform them all in parallel without doing collision detection and resolution. This would be necessary because there are autobahns with three lanes and more. To overcome this difficulty, the lane changes to the left, i.e., overtake, are given a higher priority than the lane changes to the right. For a systematic approach to multi-lane traffic, i.e., lane-changing rules, see, for example, (19). For a detailed discussion of the different models see (3, 9, 23) and the references therein.

4. Validation of the Model

A core requirement in the discussion of the simulation model is the detailed comparison with empirical data. Only if the model maps the real world sufficiently, it is capable to deal as the kernel of the online-simulation. Furthermore, one of the most puzzling points for any model is to reproduce significant empirical data on a macroscopic and microscopic level as well as the empirical observed coexistence of stable traffic states and, especially, the upstream propagation of wide moving jams through both free flow and synchronized traffic with constant velocity and without disturbing these states (13).

In analogy to the empirical setup, the simulation data are evaluated by a virtual loop detector, i.e., the number of cars passing a given link is measured as well as their velocity. This allows for the calculation of aggregated minute data of flow, speed and occupancy like for the empirical data.

The simulation run emulates a few hours of highway traffic, including the realistic variation of the number of cars that are fed into the system. Thereby, a large input rate leads to the emergence of synchronized flow, whereas at small rates small short-living jams evolve, as expected, in the vicinity of on-ramps, because of the local perturbations (8).

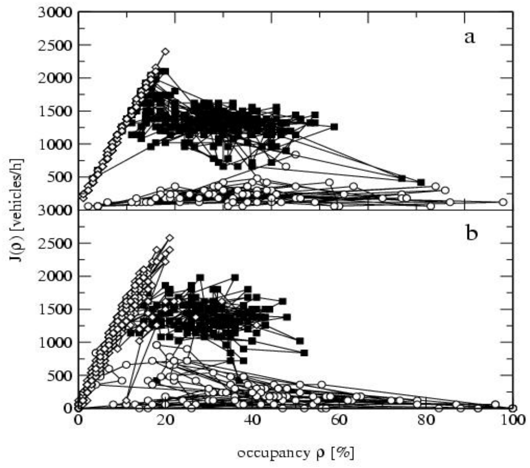


Fig. 2: Comparison of the simulation results (a) with real traffic data (b). Diamonds correspond to free flow, squares to synchronized traffic, and circles to wide jams. Each point represents the average over an one-minute interval. The empirical data are from a detector on the A40 near Moers junction (synchronized state) and near Bochum-Werne.

The simulation shows that the empirical results can quantitatively be recovered (see Fig. 2). A detailed analysis of the two dimensional region of synchronized traffic in the fundamental diagram reveals a high correlation between the two lanes with respect to the velocity time-series of both lanes.

5. Implementation of the Topology

An important point in the design of a simulator is the representation of the road network. Therefore, the network is divided into links. The main links connect the junctions and highway intersections representing the carriageway. Each junction and intersection consists of another link, like on/off-ramps or right/left-turn lanes. The attributes of each link are the length, the number of lanes, a possible speed limit, and the connecting links. In case of more than one connecting link, like at off-ramps or highway intersections, there is also a turning probability for each direction. The turning probability is calculated by taking into account the measured traffic data. All these spatial and functional data was collected to build a digital image of the topology of the whole network.

Area	34,000 km²
Inhabitants	18,000,000
On- and off-ramps	862
Intersections	72
Online loop detectors	4,000
Offline loop detectors	200
Number of links	3,698
Overall length	2,250 km

Tab. 1: Design parameters of the North Rhine-Westphalian autobahn network.

Another crucial information concerns the positions of the installed loop detectors. They also have to be included in the digital map of the network. The positions in the simulation are called ‘checkpoints’, and at these checkpoints the simulation is adapted to the measured traffic flow of the loop detectors. Tab. 1 shows some design parameters of the network. North Rhine-Westphalia is approximately one fifth of whole of Germany with respect to many numbers, e.g., number of cars, inhabitants, length of the autobahn network, et cetera.

6. Additional Rules for Complex Real Networks

The cellular automaton model for traffic flow used by the simulator was designed to be able to reproduce the main aspects of the fundamental diagram for real traffic flows (vehicle flow as a function of vehicles per km) and the fundamental microscopic properties, like the time headway distribution. This ability was verified by testing it on topologically simple networks. When simulating the traffic on a large and topologically complex network, like the autobahn network in North Rhine-Westphalia, some extensions to the cellular automaton model have to be considered. One is the guidance of vehicles and another is a strategy to integrate the measured flow from the loop detectors into the simulation.

A real driver usually has the intention to reach some goal with his driving. This makes it necessary to incorporate routes in the modeling. In principle, there are two different strategies to solve this problem. One can assign an origin and a destination to the road user and then guide him through the network according to this route (20, 22). For our network origin-destination information with a sufficient temporal and spatial resolution is not available. Therefore, the vehicles are guided in the network according to the probabilities calculated on the basis of the measured data. This means that a vehicle is not guided through the whole network, but every time it reaches a new link it will decide in accordance with the measured probabilities how it leaves the link.

To implement this we use forced lane changes. Forced lane changes are necessary so that the vehicles can drive from on-ramps on the autobahn, from the autobahn on off-ramps, when the autobahn narrows, and when vehicles drive from one particular section of the autobahn on another over an intersection. Forced lane changes differ from free lane changes in a fundamental way. While free lane changes give vehicles the opportunity to overtake vehicles driving slower and thus reduce disturbances, forced lane changes stem from the need to reach a node and are obviously an additional source for disturbances.

The simulator uses gradually increasing harsh measures to force lane changes. At the beginning of an area where a vehicle could change to the target lane, it does so, if the gap is sufficiently large and no vehicle is severely hindered. At the end of the area it will bully into any gap regardless of velocity differences. Further, a vehicle driving on its target lane should not leave the lane to overtake. An efficient implementation of this strategy is to store the lane change information in the cells. This gives a fast access through the coordinates of a vehicle. Of course this information depends on the node chosen and whether the vehicle is a truck or a passenger car. Because of this, every link has several versions of the lane change information.

To incorporate the real world measurements from the loop detectors into the simulation vehicle-moving, inserting, and removing algorithms have to be applied. This is done at the so-called checkpoints, which are located at those places in the network where a complete cross-section is available, i.e., all lanes are covered by a loop detector. Every time, when checkpoint-data is provided, the simulator uses the measured values to adjust the traffic state in the simulation. The first step is to try to move vehicles behind the checkpoint in front of it and vice versa. If this is not sufficient, vehicles are inserted or removed. This should be preferred to pure insert/removal strategies, as these can completely fail due to positive feedback if a non-existing traffic jam is produced by the simulation. In this case the simulation measures a low flow in comparison with the real data, so vehicles are added periodically to the ever growing traffic jam leading to a total breakdown.

7. The Website *www.autobahn.nrw.de*

The design of the simulator was financially supported by the Ministry of Transport, Energy and Spatial Planning of North Rhine-Westphalia, the reason being, that it wanted a novel web-based traffic information system for the public. This information system is provided by a Java applet at the URL *www.autobahn.nrw.de* (Fig. 3). The Java applet draws a map of North Rhine-Westphalia, where the autobahns are colored according to the level of service of the simulated traffic state, from light green for free flow, over dark green and yellow for dense and very dense synchronized flow, to red for a traffic jam. Additionally, after numerous requests, we integrated a color-blind mode, where dark green is replaced by dark grey and yellow by blue. Further, construction areas are drawn at the appropriate positions on the map and their estimated influence on the traffic is shown through red construction signs for a high risk of a traffic jam and green construction signs for a low risk. Road closures, which have a deep impact not only on the specific track the closure happens, but also on the traffic in a wide part of the network, are shown as well.

To make orientation easier the number and name of each junction is also written in the status bar when the mouse moves over the pictogram of the junction. All this valuable information assists the road user to choose the best route and the best time for his trip.

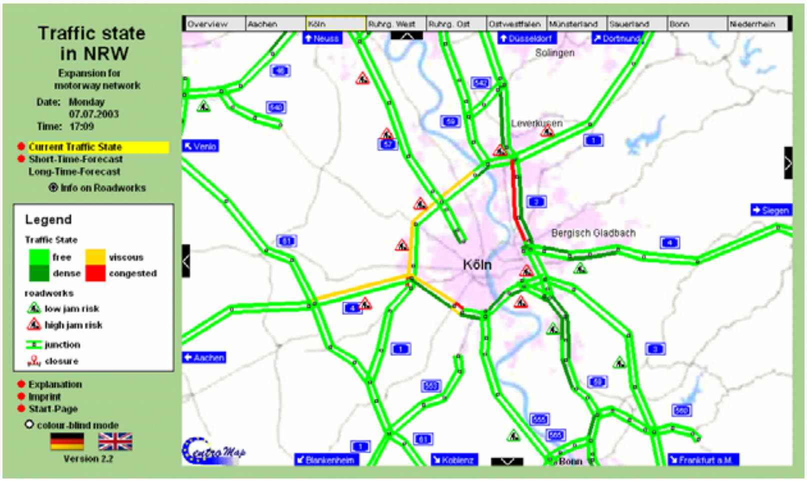


Fig. 3: *The current traffic state is visualized at www.autobahn.nrw.de using a Java applet.*

The rising accesses to OLSIM and the nearly throughout positive feedback show, that this information system is accepted by many people and used regularly. The daily requests increased from about 20,000 on work days at the beginning in September 2002 up to 200,000 regular accesses after the implementations of the 30 minute forecast in March 2003 and the 60 minute forecast in December 2003. The positive results are underlined by TV-stations, newspapers, and magazines which have made positive tests where they compared the actual traffic state to the traffic state presented by our simulation.

8. Summary

In this paper we present a new advanced traffic information system OLSIM which gives the internet user the opportunity to get the information about the current traffic state and a 30 and 60 minute prognosis of the autobahn network of North Rhine-Westphalia. The system rests upon a microscopic traffic simulator of the autobahn network in North Rhine-Westphalia. The simulator uses an advanced cellular automaton model of traffic flow and adjusts the traffic state in accordance with measurements of the real traffic flow provided by 4,000 loop detectors installed locally on the autobahn. The cellular automaton model, the abstraction of the network, the guidance of the vehicles, and the data integration strategies to periodically adjust the traffic flow in the simulation in accordance with the measured flow on the autobahn were discussed, as well as some details on the efficient implementation of the dynamics and the presentation of the simulated traffic state to the public. A graphical user interface implemented by a Java applet can be accessed by every internet user. In a simple to navigate window the user can choose between the current traffic state, the 30, and the 60 minute prognosis. Additional information like road works can be chosen with a simple click.

References

(1) R. Barlovic, L. Santen, A. Schadschneider, M. Schreckenberg. “Metastable states in cellular automata for traffic flow.” Eur. Phys. J. B 5, 1998, pp. 793-800.

(2) C. Barrett, M. Wolinsky, M. Olesen. “Emergent local control properties in particle hopping traffic simulations.” In: (9), 2000, pp. 169-173.

(3) D. Chowdhury, L. Santen, A. Schadschneider “Statistical Physics of Vehicular Traffic and Some Related Systems”. Physics Reports 329, 2000, pp. 199-329.

(4) L.C. Edie, R.S. Foot. “Traffic flow in tunnels.” Proc. HRB 37, 1958, pp. 334-344.

(5) J. Esser, M. Schreckenberg. “Microscopic simulation of urban traffic based on cellular automata.” Int. J. of Mod. Phys. C 8, 1997, pp. 1025-1036.

(6) H. George. “Measurement and Evaluation of Traffic Congestion.” Bureau of Highway Traffic, Yale University, 1961, pp. 43-68.

(7) D. Helbing. “Empirical traffic data and their implications for traffic modelling”. Phys. Rev. E 55, 1996, pp. R25-R28.

(8) D. Helbing, A. Henneke, M. Treiber. “Phase diagram of traffic states in the presence of inhomogenities.” Phys. Rev. Lett. 8, 1999, pp. 4360–4363.

(9) D. Helbing, H. Herrmann, M. Schreckenberg, D.E. Wolf (Eds.) *Traffic and Granular Flow '99.* Springer, Heidelberg. 2000.

(10) Highway Capacity Manual. HRB Spec. Rep. 87. U.S. Department of Commerce, Bureau of Public Road, Washington, D.C., 1965.

(11) O. Kaumann, K. Froese, R. Chrobok, J. Wahle, L. Neubert, M. Schreckenberg. “On-line simulation of the freeway network of North Rhine-Westphalia.” In: (9), pp. 351-356.

(12) B. Kerner, H. Rehborn. “Experimental properties of phase transitions in traffic flow.” Phys. Rev. Lett. 79. 1997. pp. 4030-4033.

(13) B.S. Kerner. “Complexity of synchronized flow and related problems for basic assumptions of traffic flow theories.” Network and Spatial Economics 1, 2001, pp. 35-76.

(14) W. Knospe, L. Santen, A. Schadschneider, M. Schreckenberg. “Disorder effects in cellular automata for two-lane traffic.” Physica A 265, 1999, pp. 614-633.

(15) W. Knospe, L. Santen, A. Schadschneider, M. Schreckenberg. “Towards a realistic microscopic description of highway traffic.” J. Phys. A 33, 2000, pp. L1-L6.

(16) W. Knospe. “Synchronized traffic: Microscopic modeling and empirical observations.” Ph.D. Thesis, Gerhard-Mercator-University Duisburg, Germany, 2002.

(17) A. Miller. “A queuing model for road traffic flow.” J. of the Royal Stat. Soc. B1, 23, University Tech. Rep. PB 246, Columbus, USA, 1961, pp. 69-75.

(18) K. Nagel, M. Schreckenberg. “A cellular automaton model for freeway traffic.” J. Physique I 2, 1992, pp. 2221-2229.

(19) K. Nagel, D.E. Wolf, P. Wagner, P. Simon. “Two-lane traffic rules for cellular automata: A systematic approach.” Phys. Rev. E 58, 1998, pp. 1425-1437.

(20) K. Nagel, J. Esser, M. Rickert. “Large-scale traffic simulations for transport planning.” In: D. Stauffer (Ed.), Ann. Rev. of Comp. Phys. VII, World Scientific, Singapore, 2000, pp. 151-202.

(21) R.C. Pfefer. “New safety and service guides for sight distances.” Transportation Engineering Journal of American Society of Civil Engineers 102, 1976, pp. 683-697.

(22) M. Rickert, P. Wagner. “Parallel real-time implementation of large-scale, route-plan-driven traffic simulation.” Int. J. of Mod. Phys. C 7, 1996, pp. 133-153.

(23) M. Schreckenberg, D.E. Wolf (Eds.) *Traffic and Granular Flow '97.* Springer, Singapore, 1998.

(24) M. Schreckenberg, L. Neubert, J. Wahle. “Simulation of traffic in large road networks.” Future Generation Computer Systems, 17, 2001, pp. 649-657.

(25) J. Treiterer. “Investigation of traffic dynamics by aerial photogrammatic techniques.” Tech. report, Ohio State University Tech. Rep. PB 246, Columbus, USA, 1975.

(26) Q. Yang, H.N. Koutsopoulos. “A microscopic traffic simulator for evaluation of dynamic traffic management systems.” Transp. Res. C 4, 1996, pp. 113-129.

Nominiert für den Heinz-Billing-Preis 2003

Berechnung von Feynman-Diagrammen mit *FeynArts*, *FormCalc* und *LoopTools*

Thomas Hahn
Max-Planck-Institut für Physik, München

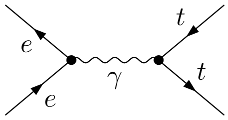
Abstract

In diesem Beitrag werden die drei Programmpakete *FeynArts*, *FormCalc* und *LoopTools* vorgestellt, mit denen die Berechnung von Feynman-Diagrammen mit bis zu einer Schleife sehr weitgehend automatisiert werden kann. Solche Berechnungen sind für die Überprüfung der gegenwärtigen Theorie der Elementarteilchen, d.h. der fundamentalen Naturgesetze unabdingbar, ohne automatisierte Schritte jedoch sehr aufwendig und fehleranfällig. Durch die Automatisierung können binnen Minuten Ergebnisse ausgerechnet werden, für die früher Mannjahre nötig waren.

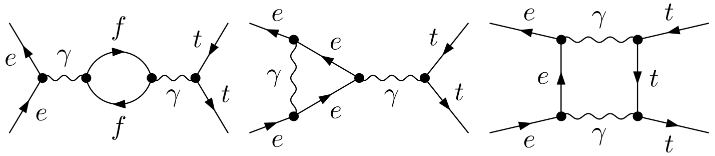
1 Einführung

In der Teilchenphysik werden quantenfeldtheoretische Modelle benutzt, um die Elementarteilchen und ihre Wechselwirkungen zu beschreiben. In Streuexperimenten wird hingegen z.B. der Wirkungsquerschnitt gemessen, das ist vereinfacht ausgedrückt die (geeignet normierte) Wahrscheinlichkeit, bestimmte Teilchen im Detektor zu sehen. Um nun die theoretische Vorhersage mit dem Experiment vergleichen und damit die Theorie testen zu können, steht man vor dem Problem, aus dem theoretischen Modell zunächst den Streuoperator und dann daraus den Wirkungsquerschnitt zu berechnen. Dies wird in der Regel störungstheoretisch mit Hilfe von Feynman-Diagrammen gemacht, d.h. man betrachtet die Wechselwirkung der Teilchen als eine kleine Störung ihrer andernfalls freien Ausbreitung und entwickelt den Streuoperator mathematisch in eine Reihe in der Kopplungsstärke.

Beispiel: Das folgende Feynman-Diagramm trägt zum Wirkungsquerschnitt des Prozesses $e^+e^- \to \bar{t}t$ (Top–Antitop-Paarproduktion an einem Elektron–Positron-Beschleuniger) bei:



Dieses Diagramm gibt nicht nur ein intuitives Bild von dem Streuprozeß, es läßt sich auch nach Regeln, die durch das Modell festgelegt sind, eindeutig in Formeln, die sog. Feynman-Amplituden, übersetzen. Insbesondere symbolisiert jeder der Punkte (•) eine Kopplung der Stärke $\sqrt{\alpha}$ zwischen den Fermionen und dem zwischen ihnen ausgetauschten Photon (γ), wobei $\alpha \simeq 1/137$ die Feinstrukturkonstante ist. Obiges Diagramm ist somit insgesamt von der Ordnung α. In der nächsten Ordnung gibt es schon wesentlich mehr Diagramme, von denen hier nur drei Repräsentanten gezeigt sind:



Hier hat jedes Diagramm vier Punkte, ist also von Ordnung α^2, gleichzeitig erkennt man aber auch, daß jetzt jedes Diagramm eine geschlossene „Schleife“ besitzt. Das ist kein Zufall, denn die Störungsreihe ist gleichzeitig auch eine Entwicklung in der Anzahl der Schleifen. Man spricht von „Baumdiagrammen“ (keine Schleife), „Ein-Schleifen-Diagrammen,“ „Zwei-Schleifen-Diagrammen“ usw. Physikalisch lassen sich die Schleifen als Quantenfluktuationen interpretieren, im linken Diagramm z.B. spaltet das intermediäre Photon in ein virtuelles Fermion–Antifermion-Paar ($f\bar{f}$) auf.

Je mehr Schleifen man mitnimmt, desto höher ist die Ordnung in α und desto genauer das Ergebnis. Dies „bezahlt“ man jedoch mit der Anzahl der zu berechnenden Diagramme, die mit der Anzahl der Schleifen rasch anwächst.

Im folgenden werden die drei Programmpakete *FeynArts*, *FormCalc* und *LoopTools* vorgestellt, mit denen derartige Rechnungen mit bis zu einer Schleife sehr weitgehend automatisiert werden können. Dazu werden zunächst in den Abschnitten 2 und 3 die mathematischen Probleme und das algorithmische Vorgehen bei der Berechnung von Feynman-Diagrammen beschrieben. In Abschnitt 4 wird dann die Benutzung von *FeynArts*, *FormCalc* und *LoopTools* beschrieben. Abschnitt 5 gibt einen Abriß über die Entwicklung und Abschnitt 6 einen Überblick über Anwendungen der Programme.

2 Feynman-diagrammatische Rechnungen

Aus der mathematischen Perspektive sind nur die Integrale, die den Schleifen in den Feynman-Diagrammen entsprechen, „schwierig.“ Bereits auf dem Zwei-Schleifen-Niveau ist es nicht mehr allgemein möglich, die Integrale auszurechnen, d.h. sie durch elementare Funktionen auszudrücken.

Die Ein-Schleifen-Integrale sind jedoch bekannt, sie lassen sich durch Logarithmen und Dilogarithmen ausdrücken. Es ist daher möglich, eine Rechenvorschrift für ein beliebiges Diagramm mit bis zu einer Schleife anzugeben, d.h. die Berechnung ist streng algorithmisch und läßt sich im Prinzip vollständig automatisieren. Dies ist zu keinem geringen Teil das Verdienst der Herren ’t Hooft und Veltman, die u.a. dafür 1999 den Nobelpreis erhielten.

Abgesehen von den Integralen läßt sich die verbleibende Rechnung mit algebraischen Methoden bestreiten. Was das Leben dennoch schwer macht, ist die extrem schnell anwachsende Anzahl der Feynman-Diagramme, wenn man Prozesse mit mehr äußeren Linien oder mehr Schleifen, oder Modelle mit mehr Teilchen und Kopplungen betrachtet. Um eine Vorstellung davon zu geben, ist in der folgenden Tabelle die Anzahl der Ein-Schleifen-Topologien für einen $2 \to 2$-, $2 \to 3$- und $2 \to 4$-Prozeß[*] aufgelistet, das ist die Anzahl der Möglichkeiten, die vorgegebenen äußeren Linien miteinander zu verbinden, so daß die resultierenden Diagramme genau eine Schleife besitzen:

Prozeß	Anzahl Schleifen	Anzahl der Topologien
$2 \to 2$	1	99
$2 \to 3$	1	947
$2 \to 4$	1	11460

Noch dramatischer wächst die Zahl der Topologien mit der Anzahl der Schleifen:

Prozeß	Anzahl Schleifen	Anzahl der Topologien
$2 \to 2$	0	4
$2 \to 2$	1	99
$2 \to 2$	2	2214
$2 \to 2$	3	50051

Es ist eindeutig die Kombinatorik, die auch auf moderner Hardware die Machbarkeit auf relativ kleine Werte des Produkts (Anzahl der Schleifen)$\cdot$ (Anzahl der äußeren Linien) beschränkt.

[*] Die Notation „$m \to n$“ bezeichnet die Zahl der einlaufenden bzw. auslaufenden Teilchen.

Die in Abschnitt 4 vorgestellten Programme sind bisher auf Ein-Schleifen-Rechnungen beschränkt, da wie erwähnt die höheren Schleifenintegrale derzeit nur teilweise bekannt sind. Auf dem Ein-Schleifen-Niveau ist momentan die Berechnung von $1 \to 2$- und $2 \to 2$-Prozessen ohne jede Einschränkung möglich. Für $2 \to 3$-Prozesse fehlt noch eine letzte Komponente, das Fünf-Punkt-Integral, dessen Einbau in das bestehende Programm bereits in Arbeit ist. Einige physikalisch relevante $2 \to 3$-Rechnungen benötigen diese Funktion allerdings nicht und sind schon jetzt möglich, z.B. [1]. Mit einer erst kürzlich implementierten neuen Methode zur Vereinfachung von Fermionketten scheint es möglich, auch die sehr rechenaufwendigen $2 \to 4$-Prozesse mit einer Schleife ins Auge zu fassen – hier wäre aus physikalischer Sicht besonders die Berechnung von $e^+e^- \to 4$ Fermionen wünschenswert – jedoch wird dies angesichts der enormen Zahl von Feynman-Diagrammen wohl noch einige Zeit und Ideen erfordern.

3 Schritte zur Berechnung eines Feynman-Diagramms

Im folgenden werden die Schritte zur Erzeugung und Berechnung der Feynman-Diagramme aufgelistet, wie man sie „mit Hand“ anwenden würde. Die Implementierung in *FeynArts*, *FormCalc* und *LoopTools*, die im nächsten Abschnitt vorgestellt wird, ist eng an dieses Schema angelehnt.

1. Stelle eine Liste aller Diagramme auf, die zu dem betrachteten Streuprozeß beitragen:
   a) Zeichne alle Möglichkeiten, die einlaufenden mit den auslaufenden Linien so zu verbinden, daß die gewünschte Anzahl von Schleifen entsteht.
   b) Bestimme anhand des Modells, welche Teilchen auf jeder Linie „laufen“ können, wobei die äußeren Linien mit den Teilchen im Anfangs- und Endzustand des betrachteten Streuprozesses identifiziert werden.
2. Übersetze die erhaltenen Diagramme mittels der Feynman-Regeln, die aus dem Modell folgen, in Formeln.
3. Vereinfache die Formeln analytisch. Dies geschieht vor allem in Hinblick auf die folgende numerische Auswertung, so müssen z.B. offene Indizes kontrahiert werden, tensorielle Objekte zerlegt werden usw.
4. Schreibe ein Programm, das die Formeln numerisch auswertet.

Offensichtlich sind hierbei Probleme sehr verschiedener Natur zu lösen, z.B. ist die Diagrammerzeugung eine topologisch-kombinatorische Aufgabe oder die Anwendung der Feynman-Regeln ein Datenbankzugriff. Hinzu kommt, daß die Amplitude algebraische Objekte enthält, die für die direkte numerische Auswertung ungeeignet sind, wie Tensoren oder Generatoren von Symmetriegruppen, man aber andererseits eine schnelle numerische Auswertung des Endergebnisses braucht, z.B. für Monte-Carlo-Generatoren, wo u.U. mehrere Millionen Events gesampled werden müssen.

Ein wichtiges Hilfsmittel bei der Umsetzung obigen Schemas ist daher die Computeralgebra, mit der die strukturellen und algebraischen Operationen bewältigt werden, in Kombination mit schneller und präziser numerischer Auswertung („Number Crunching“) in einer Hochsprache.

4 *FeynArts*, *FormCalc* und *LoopTools*

FeynArts, *FormCalc* und *LoopTools* sind drei Programmpakete, mit denen sich Feynman-Diagramme erzeugen, analytisch vereinfachen und numerisch auswerten lassen. Mit Hilfe dieser Programme ist es möglich, Streuprozesse mit bis zu einer Schleife sehr weitgehend zu automatisieren und damit eine Arbeit, die noch vor kurzem in Mannjahren bemessen wurde, in Minuten zu erledigen.

Die modulare Unterteilung in drei verschiedene Programmpakete ist nicht nur von der Art der Aufgaben her sinnvoll, vielmehr werden von vielen Benutzern nur Teile des Programms benutzt, so wird *FeynArts* etwa auch für die Erzeugung von Zwei-Schleifen-Diagrammen eingesetzt [2], selbst wenn diese derzeit nicht von *FormCalc* vereinfacht werden können.

FeynArts und *FormCalc* sind *Mathematica*-Programme und auch *LoopTools* besitzt ein *Mathematica*-Interface. Dieser Umstand ist sehr hilfreich, da er dem Benutzer erlaubt, die erhaltenen Ausdrücke an praktisch jeder beliebigen Stelle mit Hilfe des *Mathematica*-Befehlssatzes zu modifizieren. Ein Beispiel: Um dem Higgs-Propagator eine endliche Breite zu geben, muß man lediglich das Endergebnis mit der folgenden Substitutionsregel in *Mathematica* transformieren: `Den[p_, MH2] -> Den[p, MH2 - I MH GammaH]`.

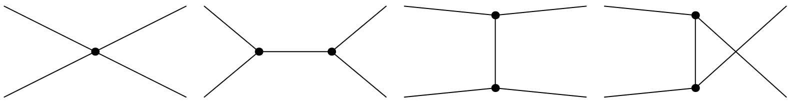
4.1 FeynArts

FeynArts erzeugt Feynman-Diagramme und -Amplituden mit derzeit bis zu drei Schleifen. Die Information über das betrachtete Modell wird aus einer speziellen Datei, dem „Model-File,“ gelesen. Derzeit existieren Model-Files für das elektroschwache Standardmodell mit und ohne QCD (einschließlich Counter-Termen), das Minimale Supersymmetrische Standardmodell (MSSM) und das Zwei-Higgs-Dublett-Modell. Der Benutzer kann aber auch eigene Model-Files erstellen oder vorgegebene modifizieren. Außerdem steht ein Hilfsprogramm zur Verfügung, mit dem das Model-File aus der Lagrangedichte der zugrundeliegenden Theorie erzeugt werden kann.

Die Erzeugung der Feynman-Amplituden verläuft im wesentlichen wie in Abschnitt 3 skizziert: Zunächst werden mit der *FeynArts*-Funktion `CreateTopologies` die Topologien mit der gewünschten Anzahl äußerer Beine und Schleifen erzeugt, z.B. die Baum-Topologien (null Schleifen) für einen $2 \to 2$-Prozeß:

```
top = CreateTopologies[0, 2 -> 2]
```

Die so erhaltenen Topologien lassen sich mit der `Paint`-Funktion zeichnen:



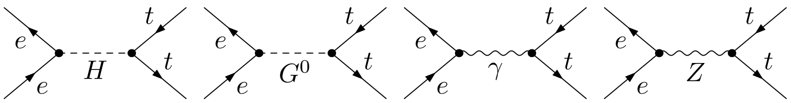
Die Diagramme können entweder am Bildschirm, als PostScript- oder als LaTeX-Datei ausgegeben werden, wobei letzteres Format problemlos in Publikationen eingebunden sowie auf einfache Weise nachbearbeitet werden kann.

In diese Topologien werden nun Felder eingesetzt, d.h. es werden zu einem vorgegebenen Streuprozeß alle im Modell möglichen Kombinationen gesucht, die Linien der Topologie mit Feldern des Modells zu bestücken. Dazu wird die Funktion `InsertFields` auf das Ergebnis von `CreateTopologies` angewendet:

```
ins = InsertFields[top,
  {-F[2,{1}], F[2,{1}]} -> {-F[3,{3}], F[3,{3}]}]
```

Hier werden die Diagramme für den Prozeß $e^+e^- \to \bar{t}t$ aus den in `top` gespeicherten $2 \to 2$-Topologien erzeugt. Als Model-File wird die Voreinstellung `SM.mod`, das elektroschwache Standardmodell benutzt. Dieses gibt auch die Bezeichnung der Felder vor: in `SM.mod` heißt das Elektron `F[2,{1}]`, es ist also das erste Mitglied der Fermionklasse Nr. 2, die aus Elektron, Myon und Tauon besteht, und analog heißt das Top-Quark `F[3,{3}]`, wobei die dritte Fermionklasse das Up-, Charm- und Top-Quark umfaßt. `-F[2,{1}]` und `-F[3,{3}]` sind die jeweiligen Antiteilchen.

Auch die Ergebnisse von `InsertFields` lassen sich mit `Paint` zeichnen. Man sieht, daß einige Topologien nicht realisiert werden können, z.B. weil die Erhaltung der elektrischen Ladung verletzt wäre, andere dagegen mehrfach vorkommen:



Schließlich müssen die Feynman-Regeln angewandt werden, um die Amplituden zu erhalten. Das geht mit

```
amp = CreateFeynAmp[ins]
```

TECHNISCHER EXKURS Ein für den Benutzer zwar weitgehend unsichtbares, für die weitere Vereinfachung aber enorm wichtiges Konzept ist die Unterscheidung von drei Ebenen („Levels“) von Feldern. Auf dem „Generic Level“ werden nur die Typen der Felder spezifiziert, z.B. `F` = Fermion oder `S` = Skalarfeld. Auf dem „Classes Level“ werden dann Klassen solcher Felder betrachtet, z.B. `F[3]` = die Klasse der Quarks mit Isospin $+\frac{1}{2}$. Schließlich können auf dem „Particles Level“ einzelne Repräsentanten der Klassen ausgewählt werden, so z.B. `F[3,{3}]` = das Top-Quark.

Da die kinematische Struktur eines Diagramms schon auf dem Generic Level festliegt, muß die aufwendige Vereinfachung der kinematischen Objekte wie z.B. die Tensorreduktion (s.u.) nur für die generischen Diagramme durchgeführt werden, von denen es in der Regel erheblich weniger gibt. Von den vier zuvor gezeigten Diagrammen etwa müssen nur zwei wirklich ausgerechnet werden: anstelle der linken zwei Diagramme rechnet man nur ein Diagramm mit Austausch eines generischen Skalarfelds aus und analog für die rechten zwei Diagramme. Durch Einsetzen der tatsächlichen Kopplungskonstanten in die generische Amplitude erhält man den vollständigen Ausdruck für alle vier Diagramme.

4.2 FormCalc

FormCalc vereinfacht die von *FeynArts* ausgegebenen Amplituden analytisch. Das Ergebnis kann entweder direkt als *Mathematica*-Formel weiterverwendet werden (z.B. für bestimmte Konsistenzchecks) oder als Fortran-Programm zur Berechnung des Wirkungsquerschnitts ausgegeben werden. In der analytischen Vereinfachung werden konkret folgende Umformungen vorgenommen (für mathematische Details siehe z.B. [3]):

– Kontraktion aller Indizes,
– Berechnung der fermionischen Spuren,
– Vereinfachung der äußeren Spinor- und Gruppenstrukturen,
– Reduktion der Tensorintegrale auf skalare Koeffizienten,
– Einführen von Abkürzungen.

Das Einführen von Abkürzungen ist ein sehr wesentlicher Punkt, mit dem die Größe des Ergebnisses drastisch reduziert werden kann.

Alle diese Operationen sind in einer Funktion für den Benutzer zusammengefaßt, die auf das Ergebnis von `CreateFeynAmp` angewendet wird:

```
result = CalcFeynAmp[amp]
```

Intern delegiert `CalcFeynAmp` viele Aufgaben an das Computeralgebra-Programm *FORM* [4] (daher der Name *FormCalc*), das zwar nur einen begrenzten, speziell auf die Anwendungen in der Teilchenphysik zugeschnittenen Befehlssatz hat, dafür aber sehr schnell ist und auch mit sehr großen

Ausdrücken mühelos fertig wird. *FORM* ist jedoch nicht unbedingt leicht zu programmieren, daher bleibt der Austausch von Programmcode und Daten zwischen *Mathematica* und *FORM* dem Benutzer erspart.

Zur weiteren numerischen Auswertung wird das Ergebnis von `CalcFeynAmp` als Fortran-Programm ausgegeben:

```
SetupCodeDir["fortrandir"]
WriteSquaredME[result, {}, Abbr[], "fortrandir"]
```

`SetupCodeDir` legt ein Unterverzeichnis namens `fortrandir` an und kopiert die notwendigen Treiberprogramme dort hinein. Danach schreibt `WriteSquaredME` das Ergebnis der obigen Rechnung zusammen mit den von `CalcFeynAmp` eingeführten Abkürzungen, die mit `Abbr[]` abgerufen werden, als Fortran-Code in dieses Verzeichnis. Dazu wird ein entsprechendes `makefile` angelegt, wodurch auch die Kompilierung automatisiert wird. Ein wesentlicher Punkt ist, daß die von `WriteSquaredME` ausgegebenen Dateien in sich völlig abgeschlossen sind und nicht mehr von Hand nachbearbeitet werden müssen, was viele „menschliche“ Fehlerquellen ausschließt. Hingegen werden die Treiberprogramme vom Benutzer angepaßt, dort müssen z.B. die numerischen Werte der Modellparameter angegeben werden.

Von den Anpassungen der Treiber abgesehen genügt ein

```
./configure
make
```

im neu angelegten Verzeichnis `fortrandir`, um das erzeugte Programm zu kompilieren. Ausgeführt wird es z.B. mit

```
./run uuuu 350 1000
```

was den Wirkungsquerschnitt für unpolarisierte äußere Teilchen im Energiebereich von 350 bis 1000 GeV berechnet. Man erhält ein Datenfile `run-tot.pol=UUUU.E=00350-01000`, das in Abb. 1 geplottet ist.

Der gesamte Ablauf, von `CreateTopologies` bis zum Plotten des Wirkungsquerschnitts, dauert nur wenige Minuten.

Der generierte Fortran-Code wird von *FormCalc* in verschiedener Weise optimiert, so werden z.B. mehrfach vorkommende Unterausdrücke nur einmal berechnet, ebenso wird die Liste der Abkürzungen so gruppiert, daß beim Durchlaufen der internen Schleifen nur die sich wirklich ändernden Teile neu berechnet werden müssen. Bei den Treiberprogrammen wurde großer Wert auf modularen Aufbau in einer übersichtlichen und gut dokumentierten Programmierweise gelegt. Alles in allem wird damit das Ziel verfolgt, ein effizientes und gleichzeitig für den Benutzer möglichst durchschaubares Programm zu erstellen, denn es gibt viele Fälle, in denen der Benutzer das Programm nicht in seiner eigentlichen Funktion benutzen will, sondern

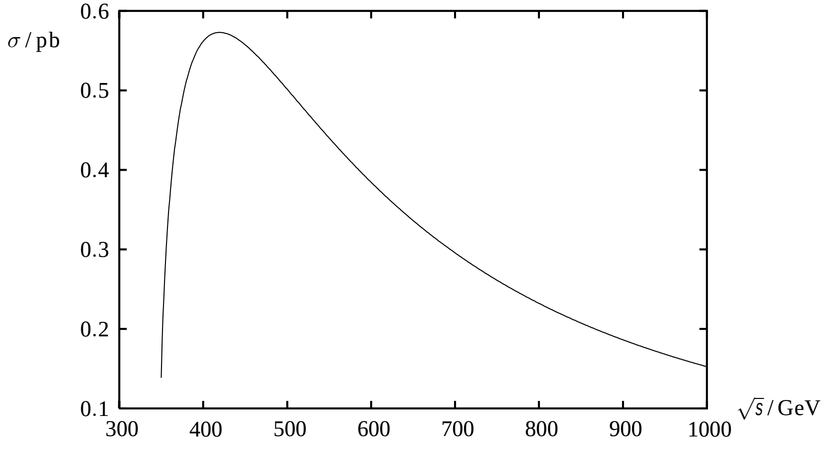


Abb. 1: Der Wirkungsquerschnitt des Streuprozesses $e^+e^- \to \bar{t}t$

es entweder als Modul in existierende Programme einbinden oder aber zusätzliche Funktionalität einbauen möchte. Die Implementierung des Fortran-Generators einschließlich der zugehörigen Treiberprogramme zählt daher zu den Programmteilen von *FormCalc*, die im Laufe der Versionen die größten Änderungen durchgemacht haben.

Schließlich ist noch ein Wort über die Wahl der Sprache angebracht: Fortran 77 wird von vielen Programmierern als „Dinosaurier“ unter den Programmiersprachen verschmäht, dabei ist es im großen und ganzen für genau die Aufgabe optimiert [5], die hier gebraucht wird: „FORmula TRANslation,“ effizientes und präzises Auswerten langer Formeln. Beispielsweise sind in Fortran so wichtige Dinge wie komplexe Zahlen schon eingebaut. Dazu existieren hervorragende Compiler für fast alle Plattformen. Unter Physikern ist Fortran nach wie vor weit verbreitet, außerdem es ist relativ unkompliziert, Fortran-Routinen auch von anderen Sprachen aus aufzurufen.

4.3 LoopTools

Bislang wurde überhaupt noch nicht auf die in Abschnitt 2 erwähnten Ein-Schleifen-Integrale eingegangen. Diese werden von *FormCalc* durch spezielle Funktionen der mathematischen Physik, die Passarino–Veltman-Funktionen, ausgedrückt, ansonsten aber weitgehend als „Black Boxes“ behandelt. Die numerische Implementierung dieser Funktionen geschieht durch die *LoopTools*-Bibliothek, die alle Ein-Schleifen-Integrale bis zur Vier-Punkt-Funktion zur Verfügung stellt, und zwar einschließlich aller Koeffizienten-Funktionen, die bei der Zerlegung der Tensorintegrale bis zur vierten Stufe anfallen. Zusätzlich enthält es die Ableitungen der Zwei-Punkt-Funktionen, die zur Berechnung der Renormierungskonstanten gebraucht werden.

LoopTools ist eine Fortran-Bibliothek, stellt dem Benutzer aber neben dem Fortran- auch ein C++- und *Mathematica*-Interface zur Verfügung. Insbesondere das *Mathematica*-Interface ist sehr einfach zu bedienen: man muß nur mit `Install["LoopTools"]` das Package laden, danach stehen alle Ein-Schleifen-Integrale als *Mathematica*-Funktionen zur Verfügung.

Für die skalaren Integrale greift *LoopTools* auf die FF-Bibliothek [6] zurück, in der diese in numerisch stabiler Weise implementiert und für viele Fälle getestet sind. Es sollte hierbei erwähnt werden, daß die numerische Behandlung nicht einfach ist, da in verschiedenen Bereichen des Phasenraums unterschiedliche Parametrisierungen bzw. Näherungen benötigt werden, etwa nahe der Schwelle, oder wenn zwei Impulse fast kollinear sind. Um numerische Stabilität zu erreichen, müssen also viele Spezialfälle berücksichtigt werden. Aus diesen Gründen besitzt *LoopTools* auch eine „check“-Version, in der alle skalaren Integrale durch eine zweite, von FF unabhängige Implementierung kontrollgerechnet werden und alle Diskrepanzen oberhalb einer vom Benutzer wählbaren Schranke ausgegeben werden.

Mit Compilern, die den Datentyp `REAL*16` zur Verfügung stellen, ist *LoopTools* auch in einer vierfach-genauen Version verfügbar. Dies wird in seltenen Fällen gebraucht, wenn es sehr große Kompensationen innerhalb einer Amplitude gibt.

Da die Berechnung der Ein-Schleifen-Integrale einen beträchtlichen Teil der CPU-Zeit ausmachen kann, wird intern ein Cache-Mechanismus verwendet, um Mehrfachberechnungen zu vermeiden, wo es insbesondere bei der Berechnung der Tensor-Koeffizientenfunktionen einen beträchtlichen Überlapp von Zwischenergebnissen gibt.

5 Historische Entwicklung

5.1 *FeynArts*

FeynArts hat die längste Geschichte der drei Programme und geht auf die Würzburger Arbeitsgruppe um M. Böhm zurück. H. Eck und J. Küblbeck entwickelten 1990 *FeynArts* 1.0 als einen Diagrammgenerator für das elektroschwache Standardmodell [7]. Wesentlich verallgemeinert wurde *FeynArts* 1995 durch neue Konzepte, die in die Version 2 einflossen [8]. So wurde z.B. speziell für *FeynArts* der „Fermion-Flip-Algorithmus“ entwickelt [9], der es erlaubt, Diagramme für Theorien zu erzeugen, die fermionzahlverletzende Kopplungen besitzen, dazu gehören insbesondere supersymmetrische Modelle. Danach verließen Eck und Küblbeck jedoch die Physik und die

Entwicklung von *FeynArts* wurde erst 1998 von T. Hahn weitergeführt. Die vorläufig letzten großen Änderungen waren die grundlegende Umgestaltung des Grafikteils [10], die 2000 von Hahn im Rahmen eines Forschungsaufenthaltes bei Wolfram Research vorgenommen wurde und mit der die fast immer anfallende Nachbearbeitung der Diagramme für die Publikation entscheidend vereinfacht wurde, sowie die Fertigstellung des MSSM-Model-Files [11].

FeynArts ist fast vollständig in *Mathematica* geschrieben, nur der Topologieeditor ist in Java kodiert. Im Vergleich zu anderen Diagrammgeneratoren besitzt *FeynArts* ausgezeichnete Grafikfähigkeiten sowie für die Behandlung grundlegender Fragen der Quantenfeldtheorie nützliche Features wie z.B. Hintergrundfelder und Mischungspropagatoren.

FeynArts ist ein Open-Source-Programm und auf http://www.feynarts.de erhältlich. Derzeit wird das Programm 150 bis 200 mal im Monat heruntergeladen (Anzahl der erfolgreichen Downloads des .tar.gz-Files).

5.2 *FormCalc*

Im Rahmen seiner Diplomarbeit 1994 [12] war T. Hahn damit konfrontiert, Box-Diagramme mit vier internen Fermionlinien auszurechnen. Das damals für die Berechnung der mit *FeynArts* erzeugten Feynman-Diagramme benutzte Programm *FeynCalc* [13] brauchte für ein einziges solches Diagramm ca. eine Woche auf der schnellsten Workstation im Würzburger Rechenzentrum. Als sich hinterher auch noch herausstellte, daß aufgrund eines ungültig gesetzten Flags in *FeynCalc* das Ergebnis Makulatur war, entwickelte Hahn in zwei Wochen einen rohen Prototypen des Programms, das heute *FormCalc* heißt. Wichtigste Neuerung war, daß für die langwierigen Vereinfachungen (besonders die fermionischen Spuren) die Amplitude an *FORM* geschickt und das Ergebnis hinterher wieder in *Mathematica* eingelesen wurde. Dieses Programm brauchte einige Minuten für die besagten Box-Diagramme und schlug somit selbst einschließlich seiner Entwicklungszeit *FeynCalc* um Längen. Weil es *FORM* benutzte, aber im Prinzip dasselbe tat wie *FeynCalc*, erhielt es den Namen *FormCalc*.

Nach einiger Weiterentwicklung im Zuge der Doktorarbeit wurde das Programm 1996 im Internet zur Verfügung gestellt. 1998 wurde in Kollaboration mit der Universität Granada das Verfahren der differentiellen Renormierung in *FormCalc* eingebaut [14]. Damit war *FormCalc* in der Lage, auch supersymmetrische Amplituden zu berechnen, für die das übliche Verfahren der dimensionalen Regularisierung nicht anwendbar ist. Weitere Verbesserungen betrafen vor allem die Code-Generierung, wodurch die Rechnung erheblich stärker als zuvor automatisiert werden konnte.

Die jüngste, erst kürzlich fertiggestellte Neuerung ist der Einbau des Weyl–van-der-Waerden-Formalismus [15]. Mit diesem kann die Berechnung von

Diagrammen mit äußeren Fermionen drastisch vereinfacht werden. Es ist unschwer, Fälle zu finden, in denen der herkömmliche Formalismus mit Helizitätsamplituden schon für einen $2 \to 3$-Prozeß in die Knie geht, weil z.B. 20000 Helizitäts-Matrixelemente berechnet werden müßten. Außerdem erhält man mit dem Weyl–van-der-Waerden-Formalismus polarisierte Wirkungsquerschnitte, also die ganze Spin-Physik, ohne zusätzlichen Rechenaufwand.

FormCalc ist in *Mathematica*, *FORM*, C und Fortran geschrieben und benutzt das standardisierte *MathLink*-Protokoll, um Daten zwischen *Mathematica* und *FORM* auszutauschen. Wie *FeynArts* ist auch *FormCalc* ein Open-Source-Programm und steht auf der Webseite http://www.feynarts.de/formcalc zur Verfügung.

5.3 LoopTools

Die von G.J. van Oldenborgh geschriebene FF-Bibliothek war lange Zeit die einzige öffentlich verfügbare Implementierung der Ein-Schleifen-Integrale. Allerdings enthielt die Bibliothek neben den skalaren Integralen nur wenige Tensorkoeffizienten. Die Routinen erforderten außerdem eine umfangreiche Deklaration der Arrays, in denen die Parameter übergeben wurden, man konnte also nicht „mal schnell" eine Funktion aufrufen.

Für die automatische Fortran-Code-Generierung durch *FormCalc* war dies ein Hindernis, so daß T. Hahn 1995 zunächst die Tensorkoeffizienten hinzufügte und sukzessive auch das Interface verbesserte. Später folgte ein autoconf-artiges `configure`-Skript, mit dem die zuvor zahlreichen Probleme bei der Kompilierung auf unterschiedlichen Plattformen weitgehend eliminiert wurden. (So kostete die Installation von *LoopTools* auf dem AIX-System des MPI für Physik 1997 noch fast einen ganzen Tag.)

Eine besonders wichtige Erweiterung, die derzeit in Arbeit ist, ist der Einbau der Fünf-Punkt-Funktion, die für $2 \to 3$-Prozesse benötigt wird. Eine numerisch stabile Version dieser Funktion existiert bereits [16] und muß nur noch implementiert und getestet werden.

Die *LoopTools*-Bibliothek ist in Fortran geschrieben. Lediglich die für den internen Cache-Mechanismus nötige dynamische Speicherallozierung ist (mit einigen Tricks) durch eine C-Funktion gelöst, da Fortran 77 nur statische Arrays kennt. Das C++-Interface besteht aus einer einzigen Header-Datei, die nur ein paar Inline-Funktionen als „Wrapper" für die Bibliotheksroutinen enthält. Das *Mathematica*-Interface ist ein C-Programm, das das *MathLink*-Protokoll zur Interaktion mit *Mathematica* benutzt. *LoopTools* ist ebenfalls ein Open-Source-Programm und steht auf http://www.feynarts.de/looptools zur Verfügung.

6 Anwendungen

6.1 Rechnungen im Standardmodell

Mit der jüngsten Generation von Beschleunigern, besonders seit LEP, liegen für viele Observable sehr genaue experimentelle Daten vor. Das hat natürlich auch den Bedarf an theoretischer Genauigkeit gesteigert, so daß Ein-Schleifen-Rechnungen im Standardmodell heute in vielen Fällen eine Minimalanforderung darstellen.

Ein-Schleifen-Rechnungen im Standardmodell umfassen typischerweise einige 10 bis 100 Diagramme, was bei Rechnung mit Hand einem Aufwand von einem bis wenigen Mannjahren entspricht, also ein typisches Diplom- oder Doktorarbeitsthema. Es gibt aber auch Beispiele, die ohne den Einsatz von automatisierten Programmen kaum denkbar sind, z.B. die elastische W–W-Streuung, wo auf Ein-Schleifen-Niveau ca. 1000 Feynman-Diagramme beitragen [17].

Für das Standardmodell existiert eine große Zahl von Originalarbeiten, die ganz oder teilweise mit *FeynArts*, *FormCalc* und *LoopTools* berechnet wurden [18]. Natürlich wäre es vermessen, bei der Komplexität dieser Programme davon auszugehen, daß selbige „fehlerfrei“ sind, daher wurden und werden zur Kontrolle auch viele bekannte Ergebnisse nachgerechnet (siehe z.B. [19]) und die Programme ggf. nachgebessert. Generell ist die Zuverlässigkeit der Ergebnisse aber sehr hoch, so wurden auch schon des öfteren Fehler in bereits publizierten Rechnungen gefunden, etwa in [20].

6.2 Rechnungen im Minimalen Supersymmetrischen Standardmodell

Das Minimale Supersymmetrische Standardmodell (MSSM) hat ein mehr als doppelt so großes Teilchenspektrum wie das Standardmodell, da abgesehen von einem größeren Higgs-Sektor die Supersymmetrie zu jedem Teilchen einen sog. Superpartner postuliert.

Als Folge des großen Teilchenspektrums besitzt das MSSM über 400 Kopplungen, was Rechnungen mit Hand im allgemeinen Fall, d.h. ohne daß man nur bestimmte Sektoren des MSSM betrachtet (oder andere Näherungen macht), sehr mühsam macht. Daher war die Veröffentlichung des *FeynArts*-Model-Files für das MSSM 2001 [11] ein wichtiger Schritt. Der aufwendigste Teil dabei war der Test möglichst aller Sektoren des Modells durch Reproduktion diverser Ergebnisse aus der Literatur, um auch im MSSM die gleiche Zuverlässigkeit wie im Standardmodell zu gewährleisten.

Experimentell wurden noch keine Superpartner-Teilchen nachgewiesen. Gerade aus diesem Grund ist aber die Betrachtung der MSSM-Schleifenkorrekturen zu bekannten Prozessen besonders wichtig, da über diese die Effekte der neuen Teilchen in Präsizionsobservablen eingehen und so Einschränkungen an die Parameter des MSSM abgeleitet oder sogar indirekte Hinweise auf Superpartner gefunden werden können.

Danksagung

An dieser Stelle möchte ich besonders Manfred Böhm und Wolfgang Hollik danken, die die Entwicklung von Programmen wie *FeynArts*, *FormCalc* und *LoopTools* stets als integralen Bestandteil der Forschung in der Teilchenphysik betrachtet und dementsprechend unterstützt haben.

Literatur

[1] T. Hahn, S. Heinemeyer, G. Weiglein, *Nucl. Phys.* **B652** (2003) 229 [hep-ph/0211204].
[2] S. Heinemeyer, hep-ph/0102318.
[3] http://maria-laach.physik.uni-karlsruhe.de/1999/Folien/uebungenII.ps.gz.
[4] J. Vermaseren, math-ph/0010025, http://www.nikhef.nl/$\sim$form.
[5] http://sunsite.informatik.rwth-aachen.de/fortran/ch1-2.html.
[6] G.J. van Oldenborgh, J.A.M. Vermaseren, *Z. Phys.* **C46** (1990) 425.
[7] J. Küblbeck, M. Böhm, A. Denner, *Comp. Phys. Comm.* **60** (1990) 165.
[8] H. Eck, Doktorarbeit, Universität Würzburg (1995), die PostScript-Version kann von der *FeynArts*-Webseite www.feynarts.de heruntergeladen werden.
[9] A. Denner, H. Eck, O. Hahn, J. Küblbeck, *Nucl. Phys.* **B387** (1992) 467;
A. Denner, H. Eck, O. Hahn, J. Küblbeck, *Phys. Lett.* **B291** (1992) 278.
[10] T. Hahn, *Comp. Phys. Comm.* **140** (2001) 418 [hep-ph/0012260].
[11] T. Hahn, C. Schappacher, *Comp. Phys. Comm.* **143** (2002) 54 [hep-ph/0105349].
[12] A. Denner, S. Dittmaier, T. Hahn, *Phys. Rev.* **D56** (1997) 117 [hep-ph/9612390].
[13] R. Mertig, M. Böhm, A. Denner, *Comp. Phys. Comm.* **64** (1991) 345.
[14] T. Hahn, M. Pérez-Victoria, *Comp. Phys. Comm.* **118** (1999) 153 [hep-ph/9807565].
[15] T. Hahn, *Nucl. Phys. Proc. Suppl.* **B116** (2003) 363 [hep-ph/0210220].
[16] A. Denner, S. Dittmaier, *Nucl. Phys.* **B658** (2003) 175 [hep-ph/0212259].
[17] A. Denner, T. Hahn, *Nucl. Phys.* **B525** (1998) 27 [hep-ph/9711302].
[18] Der Citation-Search der SPIRES-Literaturdatenbank liefert z.B. für *FeynArts* 1 [7] 155 Zitate, für *FeynArts* 3 [10] 55 Zitate und für *FormCalc*/*LoopTools* [14] 80 Zitate. Man darf davon ausgehen, daß die zitierten Programme dabei in einer integralen Weise benutzt wurden, da in der Regel nicht zitiert wird, wenn z.B. nur die Feynman-Diagramme mit *FeynArts* gezeichnet wurden.
[19] T. Hahn, W. Hollik, A. Lorca, T. Riemann, A. Werthenbach, hep-ph/0307132.
[20] W. Hollik, J. Illana, S. Rigolin, C. Schappacher, D. Stöckinger, *Nucl. Phys.* **B551** (1999) 3, Erratum: *Nucl. Phys.* **B557** (1999) 407 [hep-ph/9812298].

ESPResSo – An *E*xtensible *S*imulation *P*ackage for *Res*earch on *So*ft Matter Systems

A. Arnold, B.A. Mann, H.J. Limbach, C. Holm
Max-Planck-Institut für Polymerforschung, Mainz

Abstract

We describe a newly written program package, ESPResSo, that was designed to perform numerical MD/MC simulations for a broad class of soft matter systems in a parallel computing environment, and we present a few examples of ongoing research projects using ESPResSo. Our main concept in developing ESPResSo was to provide an easy to use simulation tool which serves at the same time as a research platform capable of rapidly incorporating the latest algorithmic developments in the field of soft matter sciences. The strength of the present version lies in its efficient treatment of long range interactions in various geometries in a parallel computing environment. The source code relies on simple ANSI-C, is Tcl-script driven, and possesses easily modifiable interfaces, for example for real-time visualizations, or a graphical interface. The distribution of the source code adheres to the open source standards. In this way we hope to make our own scientific achievements more rapidly available to a broader research community and, vice versa, also stimulate in this way researchers all over the world to contribute to our project.

1 Introduction

Soft condensed matter (or soft matter, as it is often called) is a term for materials in states of matter that are neither simple liquids nor hard solids of the type studied, for example, in solid state physics. Many such materials are familiar from everyday life - glues, paints, soaps, baby diapers - while others are important in industrial processes, such as polymer melts that are molded and extruded to form plastics [1]. Biological materials are mainly made out of soft matter as well - membranes, actin filaments, DNA, RNA, and proteins belong to this class. Furthermore, most of the food we digest is soft matter. All these materials share the importance of length scales intermediate between atomic and macroscopic scales: The relevant range for soft matter lies between nanometers and micrometers. Examples are polymers, colloids, liquid crystals, glasses, and dipolar fluids. Typical energies between different structures are similar to thermal energies. Hence, Brownian motion or thermal fluctuations play a prominent role. Another key feature of soft matter systems is their propensity to self-assemble. Again the energy differences during this process are small such that many neighboring states are normally accessible through fluctuations. This often results in complex phase behaviors yielding a rich variety of accessible structures. Order does not necessarily arise on the single molecule level, but quite commonly exhibits a multitude of hierarchically ordered structures of sometimes tremendous intricacy and complexity. Most of the biological systems are usually not even in equilibrium but evolve among switchable steady states.

Given this wide field, research on soft material substances often acquires knowledge from different areas of research, such as physics, chemistry, and biology, such that a high level of interdisciplinarity may be required for certain scientific questions.

In the past, our research has mainly focused on the study of charged polymers (polyelectrolytes) and charged colloids which serve as important substances for many technical applications. Charged systems also occur in biological environments (since most biological matter is charged), and modelling explicit water molecules requires partial charges as well. The simulation of these systems is not straightforward and very time consuming [2], thus the production of single data points could take weeks or even months for complex biomolecular problems. We have therefore developed a number of algorithms which yield fast expressions for the energy and forces of fully or partially periodic systems [3, 4, 5]. As these algorithms are normally quite complex, studying new problems in soft matter electrostatics commonly meant having to adapt that code into new programs. This required considerable amounts of valuable research time on coding issues with the final result of a highly specialized research tool. In this way, human resources were kept away from algorithmic improvements and scientific applications using them.

Looking at other available simulation packages, e.g BALL [6], GISMOS [7], GROMOS [8], LAMMPS [9], NAMD [10], polyMD [11], and OCTA [12], we did not find a single package which met all our needs. It should be easy to use, but scientificly sound; it should grant experts access to state-of-the-

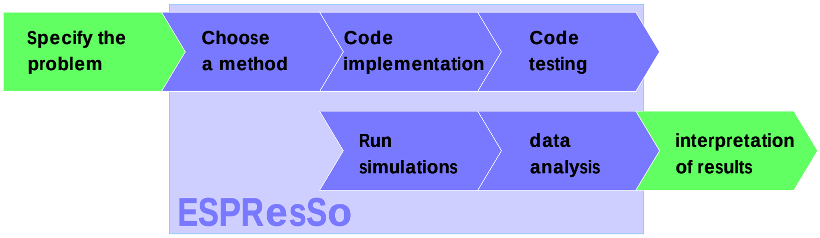


Fig. 1: A typical research problem and the part where ESPResSo *comes into play.*

art techniques, but enable beginners to become experts as well, not being limited to using a "black box", therefore it should be well-documented providing exhaustive informations. All these are as indispensable for successful scientific work as knowledge of how to specify a simulation problem and how to interpret its results, knowledge no program can ever compensate for. This led us to design a newly structured program for research on soft matter, which we called an **E**xtensible **S**imulation **P**ackage for **Res**earch on **So**ft Matter Systems, ESPResSo for short. The program enables us to study soft matter model systems via Molecular dynamics (MD) and Monte Carlo (MC) algorithms, with particular emphasis on extensibility for new, highly complex force/energy algorithms. Since the problems under investigation are located along scientific frontiers, meaning they are complex and computationally time-consuming, the program is parallizable, fast, accurate, and easily modifiable. Here, we present the first version of this ESPResSo-package. Updates and more documentation can be found on the web page http://www.espresso.mpg.de/. The distribution of the source code adheres to the open source standards and can be requested from the authors. By this we hope to ignite the further development of our code into a valuable research tool for the soft matter community.

2 Design

The ESPResSo design was developed to specifically serve the demands of a computational research group whose typical scientific projects can usually be structured into several stages such as those depicted in Fig. 1.

While most simulation programs focus only on single aspects of such a project, ESPResSo is suited to help researchers in the *whole* process between the specification of a scientific problem and the interpretation of the results. Since ESPResSo offers a variety of methods and combines the knowledge of tens of man-years of research expertise on soft matter it helps newcomers to get into simulation techniques and to choose the right method for a certain problem. On the other hand experts can easily implement their own special routines into the framework of ESPResSo, enabling them to explore paths outside the scope of their own programs. Often new problems require new algorithmic solutions. ESPResSo helps the user implement new features due to its hierarchical structure, its modularity, its general data structures and its well-defined interfaces. A test-suite which is part of ESPResSo helps in checking if new features reproduce well-known physical properties of model systems. The hierarchical structure is well-suited for running simulations without required knowledge of the whole program package. The whole system setup is contained within a Tcl [13] script. A large number of sample scripts for various simulation problems help in developing new applications. Inside a simulation script, one can handle the entire simulation process from the specification of a system, the actual simulation, its analysis and the graphical output of the results. We want to emphasize our goal to design a practical research tool which is easy to learn, use and extend. This is especially supported by ESPResSo being a team project, since this ensures that every part of the code has to pass through a discussion process provoking a simple, effective and understandable implementation.

A difficult task in the design process arises from conflicts between different requirements regarding the simultaneous optimization of several aspects. In order to ensure that new researchers do not need too much time to learn how ESPResSo works, the code has to be kept simple, which is sometimes in contradiction to code optimization for computational speed. Aiming at being able to handle a wide variety of topics instead of solving only specific problems leads to the same challenge of countering the code's tendency towards more complexity, hence less understandability.

HIERARCHICAL PROGRAM STRUCTURE: ESPResSo is built up using three hierarchical program levels. In Fig. 2, we show a sketch of this hierarchical program structure together with the program modules belonging to each level and their scope. The steering of the program is done on a script language level. All tasks are implemented as extensions to the script language dealing with input and output of data, setting of particle properties, interactions and parameters, and performing the integration and analysis of a given system. The basic simulation level is implemented in C. It contains the integrator as well as the calculation of fundamental observables like forces, torques, energies, pressure and temperature. These first two levels build up the part that is common to all investigated problems. Consequently, it is the part which should be known by a researcher using ESPResSo. Therefore, special emphasis was placed on simplicity and readability. The third level, also implemented in C, ensures the speed, efficiency and great generality of ESPResSo. This includes algorithms to accelerate the force and energy calculations used by the integrator as well as special algorithms to treat long-range interactions (see Section 3.3). Parallelization of all time-critical parts of the program enable efficient large scale simulations (see the benchmarks in Section 3.4). On this level, one also finds all implemented potentials for the particle interactions, and interfaces to other programs like VMD [14] for on-the-fly visualization of simulations.

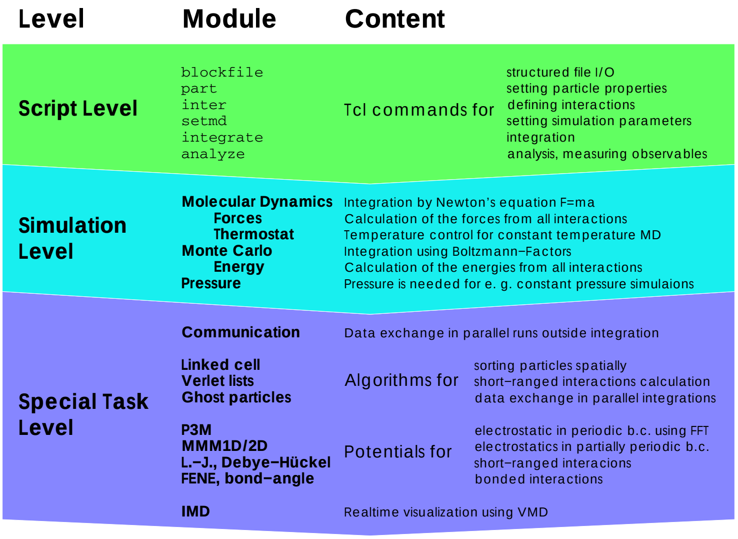

Level
Module
Content
Script Level
blockfile
part
inter
setmd
integrate
analyze
Tcl commands for
structured file I/O
setting particle properties
defining interactions
setting simulation parameters
integration
analysis, measuring observables
Simulation Level
Molecular Dynamics
Forces
Thermostat
Monte Carlo
Energy
Pressure
Integration by Newton's equation F=ma
Calculation of the forces from all interactions
Temperature control for constant temperature MD
Integration using Boltzmann−Factors
Calculation of the energies from all interactions
Pressure is needed for e. g. constant pressure simulaions
Special Task Level
Communication
Data exchange in parallel runs outside integration
Linked cell
Verlet lists
Ghost particles
Algorithms for
sorting particles spatially
short−ranged interactions calculation
data exchange in parallel integrations
P3M
MMM1D/2D
L.−J., Debye−Hückel
FENE, bond−angle
Potentials for
electrostatic in periodic b.c. using FFT
electrostatics in partially periodic b.c.
short−ranged interacions
bonded interactions
IMD
Realtime visualization using VMD

Fig. 2: Program hierarchy and modular structure of ESPResSo.

MODULARITY: The hierarchical structure is accomplished by splitting the levels into different modules, which subdivides the otherwise large program package into manageable pieces. This is particularly important for the aspect of extensibility because it ensures that an extension does not affect the entire package but rather one or few modules. It also allows the user to concentrate on understanding those modules and their scientific background that are actually used for the particular problem under investigation.

GENERALITY: To serve as a general research tool, ESPResSo needs to be able to handle a wide variety of problems. This includes different topologies, short- and long-range interactions, external fields, constraints, different boundary conditions, and various methods like MD or MC. In order to treat large scale simulations, it is also necessary to have an efficient parallelized code which runs on multiple CPU architectures. Since one of the main focuses of ESPResSo is long range interactions, we describe the implemented methods in more detail in Section 3, while abilities of the parallelization is demonstrated in 3.4.

The connectivity between the particles, which in other programs is often stored as a global topology, is incorporated locally at every particle. In this way the user is free specifying the needed topologies on the script level (see Section 4 for examples), and the connectivity information can easily be parallelized. Since this allows for any kind of topology, additional concepts such as molecules, polymer chains or proteins are unnecessary. There are however a number of predefined sample scripts and auxiliary routines provided for the user's convenience which set up polymer chains, simple model networks, or more complex structures such as fullerenes (see Fig. 3), representing tutorial-like shortcuts that facilitate writing new task scripts. Since both single molecule experiments as well as investigations of confined systems have gained increasing importance over the last years, ESPResSo also contains features to handle corresponding simulations. The program is able to deal with periodic boundary conditions in any combination of up to three spatial directions.

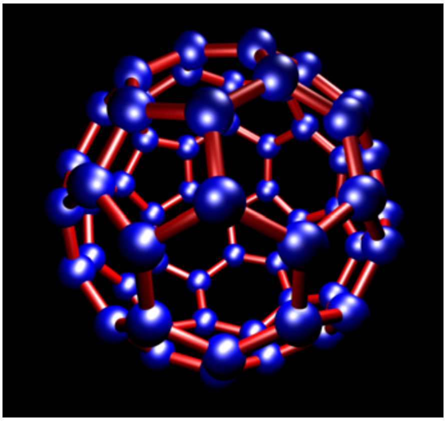


Fig. 3: Snapshot of a fullerene molecule (C60) built by ESPResSo.

When simulating bulk systems, e.g. a small representative portion of a solution, normally periodic boundary conditions are applied to avoid boundary effects. For simulations of thin films or surface effects, periodic boundary conditions in all three spatial dimensions do no make sense. The proper boundary conditions are periodic only in two out of the three spatial dimensions, while the remaining coordinate has a non priodic boundary condition. If rods are the object of interest (see e.g. Section 4.3), only one coordinate is left to have periodic boundary conditions. The complexity of an electrostatic simulation dramatically changes with different boundary conditions (see Section 3.3).

In the case of a thin film, the particles have to be confined to a fixed layer. For this ESPResSo supports constraints like walls, cylinders or spheres. It is also possible to simulate particles subject to external forces or fields. In order to cover a wide range of thermodynamic environments one can switch between simulating different thermodynamic ensembles like the NVE-, NVT- or NPT-ensemble.

DOCUMENTATION: ESPResSo is not intended to be a black-box-like package. Users are encouraged to try to understand its algorithms and routines, developers are strongly advised to do so before extending it. In order to preserve the knowledge about algorithms and their physical/chemical background, it is important to provide and maintain a well structured documentation. This is mainly done inside the code itself and then extracted and processed by doc++ [15] into a user-friendly html-manual mainly adressing specific code-/function-/procedure-related issues. It is supplemented by a stand-alone documentation on general topics such as the usage of the script commands, the general organization of the data structures, communication schemes, and analysis options. The code development itself is done in a CVS-environment (concurrent version system [16]). This helps to keep track of all changes, and provides information on what, when and by whom something has changed in the program.

PROGRAMMING ENVIRONMENT: We decided to use C as the only programming language in order to keep the code as simple to read as possible. Compared to C++ we think that this is still the language of choice in a research environment since it is easier to learn for people having a natural science rather than a computer science background. At the same time it provides all necessary features to create a modular and concise program package.

We use Tcl [13] as the script language since it contains a simple and effective interface to build C programs as extensions to the script language itself. Syntax and programming style are similar to C which makes it easy to learn. Another advantage is that there exist a large variety of extensions for Tcl. For example, with the Tk-extension it is straight forward to build a graphical user interface. This has already been done for presentation purposes. Tcl also gives us the possibility to easily create interfaces to other programs. Examples are gnuplot or xmgr for graphical processing of analysis results.

Another important choice was the type of communication for the parallelization. We decided on using MPI, as it is available for virtually all architectures; unlike e.g. OpenMP which requires shared memory, MPI also works on distributed memory computers such as Linux clusters. ESPResSo relies on the fact that MPI–implementations are normally well optimized for the underlying architecture.

EASE OF USE: For the ESPResSo-package to live up to its full potential, a straightforward and simple access is mandatory. The layered hierarchical program structure allows the user to focus on any aspect of his scientific simulation. This can be for example modeling complex physical systems with particle insertion/deletion, pressure-dependent volume changes and/or varying constraints can be done by simply creating a corresponding script file which specifies the basic rules of such a setup. Or it can be tweaking computational routines to utmost performance can be achieved by simply adding/modifying/replacing one single module of ESPResSo, immediately granting other users access to that improvement.

To demonstrate this simplicity, we give the complete script file required for the real simulation of a full 3D liquid Lennard-Jones system near the triple point with ESPResSo – just the following few lines:

```
# Sample Script:
# Lennard-Jones system near the triple point
# Create 32000 random particles at a density
# of 0.8442 in a cubic box.
setmd box_l  33.5919 33.5919 33.5919
for {set i 0} { $i < 32000 } {incr i} {
    set position_x [expr 33.5919*[t_random]]
    set position_y [expr 33.5919*[t_random]]
    set position_z [expr 33.5919*[t_random]]
    part $i position $position_x $position_y \
                     $position_z type 0
}

# Create Lennard-Jones-interactions
inter 0 0 lennard-jones 1.0 1.0 2.5 0 0

# Initializing a Langevin thermostat
setmd temperature 0.72
setmd gamma       1.0

# Integrate 1000 steps in a NVT-ensemble
setmd time_step   0.01
integrate 1000
```

This script, complemented by an initial warm-up period to prevent two of the LJ-particles to have their random starting positions too close to one another, is essentially the one which was used in Section 3.4 for creating the equilibrated starting configuration of the first benchmark scenario.

3 Included Algorithms

In the following we want to give a brief overview on the algorithms and data organization used in ESPResSo. The handling of the electrostatic interaction, one of the special features of our program, is described in more detail.

3.1 Data structure and link cells

The calculation of pairwise forces requires a loop over all possible particle pairs, which is computationally inefficient (scaling as $O(N^2)$, where $N =$

amount of particles). In the case of short-ranged forces, i.e. forces which only have a non-neglegible contribution within a range of less than 10% of the simulation box, a standard way to overcome this is the link cell algorithm. The particles are sorted into cells which are about as large as the largest range of a short–ranged interaction. Then short–ranged interactions only occur between particles in adjacent cells. For systems of equal density the number of particles in these cells is constant, therefore reducing the computational order to $O(N)$. Distributing the particles according to their spatial position, known as domain decomposition, is also a standard method for parallelization in multiprocessor environments. Therefore, our way of storing the data also supports the parallelization of the code.

Standard MD/MC programs store the particle information consecutively and the cell information as a pointer array into the particle data. This results in less readable code since many indirect accesses to the particles occur. Therefore we decided to store the particle information split up into the cells, resulting in a much more elegant integrator code. A nice side effect is that the particle information is now also stored in the same order as it will be used during the simulation which linearizes memory accesses. Since memory bandwidth is the bottle neck on modern computer systems, this also results in a major speedup.

3.2 MD integrator

For MD simulations ESPResSo has a velocity Verlet integrator with a Langevin thermostat. The calculation of the short–ranged forces is done using the link cell structure and Verlet lists. For each pair of neighboring cells, ESPResSo maintains the Verlet list, that is a list of all the particle pairs which currently interact and pairs that will interact if they approach “just a bit” further. ESPResSo has to calculate only the interactions for these pairs as long as none of them have moved too far, which may take more than ten time steps. Bonded interactions such as spring forces are stored with one of the particles involved and calculated in a simple loop. Long–ranged interactions like the electrostatic potential need a much more sophisticated treatment, which is explained below. Particles that are not rotationally invariant are treated using a quaternion representation.

3.3 Electrostatics – P3M, MMM2D, ELC and MMM1D

The most time consuming part of a simulation incorporating charged particles is the calculation of the electrostatic interaction. This is due to the fact that the electrostatic interaction is long–ranged, so that interactions are negligible only in a range larger than several box lengths in general. In periodic boundary conditions on reasonably sized systems this problem is therefore only tractable by applying non–trivial mathematics. The first algorithm to treat this problem efficiently goes back to Ewald in 1921, whose algorithm had an order of $O(N^{3/2})$ and is still in use today. The potential is artificially split up into a short–ranged part and a long–ranged, smooth part, which is handled in Fourier space. ESPResSo uses an extension of this method called P3M[1] [17]. This method uses a grid approximation for the particle distribution in the Fourier space part, allowing the use of fast fourier transformations (FFT) which facilitates the overall computational time to drop to $O(N \log N)$. ESPResSo uses FFTW which is a fast and portable public domain FFT [18]. The overall performance of all Ewald-type methods critically depends on the choice for the splitting point of the potential. As P3M features additional parameters such as the grid size, choosing the optimal ones is even more demanding. ESPResSo helps the user by providing an automatic tuning tool which determines the optimal P3M parameter set [3] for a given system and a certain desired accuracy, e.g. 10^{-4}: `inter coulomb <bjerrum_length> p3m tune accuracy 1e-4`

If the system is replicated periodically in fewer than three dimensions, the situation is even worse due to the broken symmetry. Here, the classical Ewald approach leads to an $O(N^2)$ algorithm. ESPResSo here uses variants of the MMM [19] approach, MMM2D [4] and MMM1D [20], to tackle the 2D and 1D periodic cases for small numbers of particles. For large numbers of particles in two–dimensionally replicated systems, another method, called ELC[2] [5], is implemented that allows a computational effort similar to P3M. These methods are current developments and ESPResSo is the first simulation tool to use them. In the following we shortly describe the algorithms and their range of application.

MMM2D, obtains high accuracy and is very fast for small number of particles, but has a computational order of $O(N^{5/3})$ and therefore is for larger numbers of particles much slower than P3M. It allows for very simple error estimates and is easy to tune for optimal speed. MMM2D uses two different formulas to calculate the electrostatic interaction. The first one converges very fast, but only if the particles are sufficiently far away. The second formula also works for particles close together, but is much more time consuming. The only tuneable parameter is the distance at which the use of the two formulas is switched.

ELC is a completely different approach. The interaction is first calculated using P3M with full periodic boundary conditions, but in a second step the

[1]P3M: Particle-particle particle-mesh method

[2]ELC: Electrostatic layer correction

contribution of the additional image layers is subtracted again. To calculate this contribution parts of the MMM2D theory can be used, leading to a very fast linear algorithm for this calculation. The overall computational time is dominated by the P3M algorithm.

For the one–dimensional case MMM2D is easily modified to the MMM1D algorithm. It has a very unfavorable computational time scaling of $O(N^2)$, but still is as accurate as MMM2D, and for small numbers of particles very fast. An application of this algorithm is described in Section 4.3.

3.4 Benchmarks

Even though the primary goals of the ESPResSo-package are accessibility, modularity, flexibility, and extensibility, its secondary -and equally important- objective accounts for the scientific realities of tight schedules and time constraints: Being as optimized for speed as possible without sacrificing its primary benefits. Considering the timings we measured, the state-of-the-art algorithms adapted (see previous sections) meet both demands extremely well. The direct comparison of benchmark timings of some test scenarios to those of the corresponding highly specialized codes show that despite the unique nature of ESPResSo representing a very general multi-purpose tool, it still performs similarly (e.g. compared to LAMMPS, but around 1.5 times slower than polyMD) with a firm robustness among different architectures (e.g. AMD, Intel, IBM, Alpha), compilers (e.g. mpicci, mpich, mpicc), and operating systems (e.g. AIX, OSF1, Linux). Since to our knowledge there is no available program package similar in scope and design of ESPResSo, it is therefore safe to conclude that balancing both aforementioned goals has succeeded. Any potential difference in performance is negligible compared to either the time scales needed for algorithmical implementation of even faster code (particularly when recalling ESPResSo's mission to remain understandable, prohibiting most low-level trickery), or to the advancements in hardware technology, or simply overcompensated for by ESPResSo's design-inherent advantages since e.g. the modularity allows to account for any algorithmical improvements which might arise in the future.

Besides absolute speed, another fundamental feature of our scientific simulation system is its intrinsic parallelizability, which also distinguishes it from other projects: In ESPResSo the choices of data structures and algorithm implementations were optimized in this respect, so that the program is now able to use any reasonable number of processors on any computer system supporting one of the available MPI–environments. To demonstrate the parallel performance, Table 1 presents benchmarking results for three standard test scenarios. Note that the charged systems used the P3M routines, which is why for $N = 4$ the scaled dense ES-system seems more efficient compared to the scaled case at $N = 2$ or to the fixed system because the performance critically depends on the choice of P3M-parameters whose optimizability in turn depends on the number of particles and processors (see Section 3.3).

4 Applications

In this section, we will present a few ongoing research projects dealing with charged polymers and their counterions in different solvents and topologies; extensions to charged membranes and colloids are straightforward, as are those to neutral systems. Polyelectrolytes are polymers which have the ability to dissociate charges in polar solvents which results in charged polymer chains (macroions) and mobile counterions. They represent a broad and interesting class of soft matter [21, 22] that command increasing attention in the scientific community. In technical applications polyelectrolytes are used as viscosity modifiers, precipitating agents, and superabsorbers. A thorough understanding of charged soft matter has also become of great interest in biochemistry and molecular biology.

4.1 Polyelectrolyte bundles

Polyelectrolyte bundles can be used as templates to build up nanowires or model systems to study DNA agglomerates. Although applications are already in an advanced state, our understanding of the polyelectrolyte bundling processes is still quite limited. With the help of molecular dynamics simu-

# of Processors N	1	2	4	8	16	32
LJ-system (fixed)	1.00	0.99	0.97	0.96	0.92	0.84
LJ-system (scaled)	1.00	0.99	0.97	0.92	0.89	0.82
dense ES (fixed)	1.00	0.97	0.95	0.88	0.81	0.69
dense ES (scaled)	1.00	0.96	0.99	0.76	0.74	0.50
dilute ES (fixed)	1.00	0.87	0.83	0.73	0.60	0.46
dilute ES (scaled)	1.00	0.86	0.68	0.66	0.61	0.35

Tab. 1: Efficiency of the ESPResSo*-code on an IBM Regatta H Server (eServer 690 Modell 681 with 32 Power4 Processors at 1.3 GHz each) for three different systems: A neutral LJ fluid composed of either* $32000 \cdot N$ *(scaled) or 32000 particles (fixed) at a density of 0.8442 in a NVE-ensemble; a dense electrolyte system with either* $2000 \cdot N$ *(scaled) or 8000 particles (fixed) at a density of 0.07, friction and temperature of 1.0, Bjerrum length of 20.0 in a NVT-ensemble; a dilute electrolyte system which differs only in density (*$1 \cdot 10^{-4}$*), Bjerrum length (2.0), and fixed size (16000 particles). After warm-up and equilibration period the execution time* T_N *for integrating 1000 time steps (*`integrate 1000`*) was measured and compared between using one and N nodes: For a fixed-size system the efficiency reads* $\frac{T_1/N}{T_N}$*, for scaled-size* $\frac{T_1}{T_N}$.

lations, we investigate the stability of such bundles as a function of (i) the strength of the electrostatic interaction, (ii) the stiffness of the polyelectrolyte backbone, (iii) the solvent quality, and (iv) the chain length. Simulations are performed in a spherical simulation cell. This is possible in ESPResSo using non-periodic boundary conditions together with a spherical constraint. The concept to store the connectivity between the particles locally at each atom enables us to easily create the needed topology of the chains, namely a stiff backbone chain with a flexible hair attached at every third monomer. In Fig. 4, a snapshot of a polyelectrolyte bundle is shown. The simulation started as a bundle made of 8 polyelectrolyte chains. During the simulation, two of the chains split off the bundle and a bundle with 6 chains remains. This shows that we can in fact explain a thermodynamically limited finite bundle size with a relatively simple model.

4.2 Hydrogels - Polyelectrolyte Networks

Probably one of the most computationally demanding tasks on the coarse-grained level of bead-spring polymers is the simulation of a network of polyelectrolytes. Not only do the aforementioned long-range interactions between charged monomer units require sophisticated techniques, the cross-linked network bonds themselves heavily increase the amount of short-ranged excluded volume and bonding potentials to be considered while at the same time diminishing the effectiveness of cut-offs for saving computation time. Nevertheless, the wide applicability of such hydrogels for chemical, pharmaceutical, medical, biological, agricultural, environmental, and industrial settings more than justifies any efforts towards a deeper understanding of their fascinating and important properties.

With the ESPResSo package one can go easily from single chains to a network on the script level. A simple

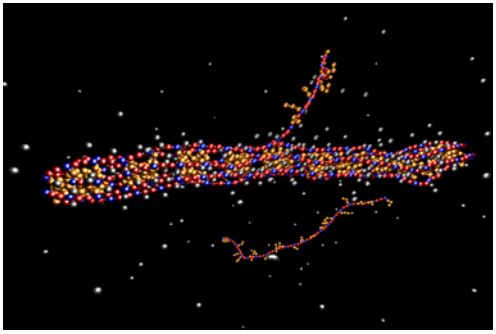


Fig. 4: Snapshot of a polyelectrolyte bundle: A bundle at the edge of stability. One chain has already fallen off and another chain is splitting from the bundle. Colors: neutral backbone - red, charged backbone - blue, hydrophobic side chains - orange, counterions - gray

```
for {set i 0} {$i < $number_of_chains} {
     part $start($i) bond $fene $partner1($i)
     part $end($i)   bond $fene $partner2($i)
}
```
in the Tcl-script is enough to crosslink a system of charged chains to become a hydrogel, while the remaining segments of script and code do not need to be changed at all.

Naturally, the behavior of such systems depends both on the environment, i.e. the strengths of electrostatic and bonding interactions as well as the presence and valency of eventual salt molecules, and on the topology of the network which may range from a well-ordered model network (diamond, star polymer) to randomly crosslinked polyelectrolytes.

Using ESPResSo, we find that the swelling behavior of polyelectrolyte gels does not only beat comparable neutral networks by at least an order of magnitude, showing unprecedented abilities of absorbing the surrounding solvent by growing several times its own volume, but it also turned out that a simple scaling argument seemed to be sufficient to describe these complex interplays between (partially screened) long-range interactions, short-range attraction and excluded volume effects by balancing the osmotic pressure of the counterions inside the gel due to electrostatic repulsion and the elastic contribution of the stretched chains [23, 24].

4.3 Stiff DNA–like polymers

Experiments have shown that DNA exhibits attractive interactions in the presence of multivalent counterions. This is believed to be the reason for the compactification of DNA, for example inside viral capsids. Computer simulations of polyelectrolytes in the presence of multivalent counterions also show an attractive force between the (like–charged) polymers.

To understand the effects in more detail, we model the DNA strands by two infinitely long charged rods. The system is neutralized by multivalent counterions. In such a system, mean field theories like Poisson–Boltzmann do not predict any attraction of the rods. Generally, one knows that correlations between the counterions, ignored in the mean–field treatment, are responsible for the observed rod–rod attractions. We use ESPResSo to study this system, using periodic boundary conditions only along the rods. The electrostatic interaction is calculated using MMM1D.

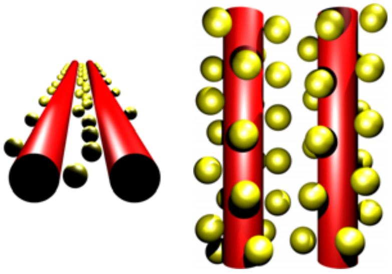
A first goal was to extend some results from previous simulations [25], which studied the effect of the electrostatic interaction by increasing the Bjerrum length[3]. The study showed that even for moderate Bjerrum lengths the behavior is dominated by the low temperature behavior, but depends heavily

[3]The Bjerrum length is the distance at which two charges interact with an energy of $1k_BT$

on other parameters like the rod radius and its line charge density. If one splits up the net force on the rods into the contributions from the electrostatic interaction and from the excluded volume interaction, it was found that both forces can be repulsive or attractive at large Bjerrum lengths, depending on the other parameters. We performed a number of simulations at $T = 0$ varying both rod radius and line charge density, showing an unexpectedly rich phase diagram (Fig. 5).

System 1:

$$r_{rod} = 1\sigma, \quad \lambda = 0.33/\sigma$$



System 2:

$$r_{rod} = 4\sigma, \quad \lambda = 6.5/\sigma$$

Fig. 5: At zero temperature, the ions form a crystal. For System 1 all counterions are trapped in the plane spanned by the rods, for System 2 a quasi hexagonal pattern is formed. The systems have different rod radii r_{rod} and line charge densities λ. Especially for System 1 the patterns show defects which are due to finite size effects and the kinetics of the freezing process.

Recent analytical predictions based on a strong coupling theory regarding the equilibrium distance between two charged rods were easily verified due to the flexibility of ESPResSo [26]. This distance was determined using a simple bisection algorithm combined with an interpolation, something which is easily implemented in Tcl and could therefore be done in the simulation script, while most other simulation packages would not have allowed such a simulation directly.

5 Further developments

As of this writing the ESPResSo-package continues to undergo significant enlargement. We are currently implementing a standard dipolar Ewald sum [27], which will enhance the capabilities to simulate ferrofluids or dipolar fluids like simple water models, and add an enhanced leap-frog algorithm for the rotational degrees of freedom. Also work has started to include an anisotropic short range potential, namely the Gay-Berne potential, which will allow the study of liquid crystals.

For the dynamics of soft matter systems it is often necessary to include hydrodynamic interactions. Since in practice one cannot include all molecular details of the systems this can be achieved on a coarse grained level by coupling the solvent degrees of freedom to the simulated particles. This will be implemented via an advanced lattice Boltzmann algorithm that has already proven its usefulness in polymer dynamics simulations [28]. An alternative way of coarse graining hydrodynamics, called dissipative particle dynamics (DPD) [29], is based on a momentum conserving thermostat. We will implement a version according to Soddemann et al. [30]. And finally, a non-equilibrium molecular dynamics algorithm [31], especially useful for driven systems, will also be added.

On top of the present strengths of ESPResSo concerning the efficient treatment of electrostatics, we plan to implement two very new ideas which promise to be a significant improvement in investigating media with varying local dielectric constants: While the first is a purely local algorithm by T. Maggs [32] which seems to be very well suited for MC simulations and can also be very useful for dense systems by using a constrained MD algorithm[4], the second is a finite difference multigrid scheme for electro- and magnetostatics[5] which appears to be better for parallel applications, promising to be fast for MD algorithms as well due to its recursiveness.

There are many more improvements planned for the next year, and hopefully the capabilities of ESPResSo will grow even further once other researchers take up our idea and contribute to ESPResSo by using, customizing and extending it.

Acknowledgements

ESPResSo is entirely based on teamwork. It is therefore a great pleasure to acknowledge all the ideas, user feedback, bug-reports, and code contributions we received. Particularly the members of the PeP-group [33] at our institute, namely D. Antypov, V. Lobaskin, F. Mühlbacher, M. Sayar, and Z. Wang, provided invaluable support by using ESPResSo in a scientific environment. Most of the aforementioned extensions to the original framework based on the requirements of their respective scientific projects. We also thank C. Abrams, B. Dünweg, and K. Kremer for helpful comments.

References

[1] R. A. L. Jones. *Soft Condensed Matter*. Oxford University Press, 2002.
[2] M. J. Stevens and K. Kremer. *J. Chem. Phys.*, **103**(4):1669–1690, 1995.
[3] M. Deserno and C. Holm. *J. Chem. Phys.*, **109**:7678, 1998.
[4] A. Arnold and C. Holm. *Comp. Phys. Comm.*, **148**(3):327–348, 2002.
[5] A. Arnold, J. de Joannis, and C. Holm. *J. Chem. Phys.*, **117**:2496–2502, 2002.

[4]B. Dünweg, private communication
[5]I. Tsukerman – private communication

[6] N. Boghossian, O. Kohlbacher, and H.-P.Lenhof. Ball: Biochemical algorithm library. Research report, Max-Planck-Institut für Informatik, 1999.

[7] C. J. Lejdfors. Gismos home page, 1998. `http://www.teokem.lu.se/gismos/`.

[8] W. F. van Gunsteren. Gromos96 homepage, 1996. `http://www.igc.ethz.ch/gromos/`.

[9] S. J. Plimpton. *J. Comp. Phys.*, **117**:1–19, 1995.

[10] M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. Kale, R. Skeel, and K. Schulten. *Int. J. Supercomput. Ap.*, **10**(4):251–268, 1996.

[11] M. Pütz and A. Kolb. *Comp. Phys. Comm.*, **113**:145–167, 1998.

[12] M. Doi. Octa homepage, 2003. `http://octa.jp/OCTA/whatsOCTA.html`.

[13] Tcl/Tk. Homepage, 2003. `http://tcl.activestate.com/`.

[14] VMD. Homepage, 2003. `http://www.ks.uiuc.edu/Research/vmd/`.

[15] R. Wunderling and M. Zöckler. DOC++: A documentation system for C/C++ and Java, 2003. `http://www.zib.de/Visual/software/doc++/`.

[16] CVS. Concurrent versions system - homepage, 2003. `http://www.cvshome.org/`.

[17] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. IOP, 1988.

[18] FFTW. Homepage, 2003. `http://www.fftw.org/`.

[19] R. Strebel. Pieces of software for the Coulombic m body problem. Dissertation 13504, ETH Zuerich, 1999.

[20] A. Arnold and C. Holm. in preparation.

[21] M. Hara, editor. *Polyelectrolytes: Science and Technology*. Marcel Dekker, 1993.

[22] C. Holm, P. Kékicheff, and R. Podgornik, editors. *Electrostatic Effects in Soft Matter and Biophysics*, volume 46 of *NATO Science Series II - Mathematics, Physics and Chemistry*. Kluwer Academic Publishers, 2001.

[23] J.-L. Barrat, J.-F. Joanny, and P. Pincus. *J. Phys. II France*, **2**:1531–1544, 1992.

[24] T. A. Vilgis, A. Johner, and J.-F. Joanny. *Eur. Phys. J. E*, **3**(3):237–244, 2000.

[25] M. Deserno, A. Arnold, and C. Holm. *Macromolecules*, **36**(1):249–259, 2003.

[26] A. Naji, A. Arnold, C. Holm, and R. R. Netz. submitted to Europhys. Lett.

[27] Z. W. Wang and C. Holm. *J. Chem. Phys.*, **115**:6277–6798, 2001.

[28] P. Ahlrichs and B. Dünweg. *J. Chem. Phys.*, **111**(17):8225–8239, 1999.

[29] P. Español and P. Warren. *Europhys. Lett.*, **30**:191, 1995.

[30] T. Soddemann, B. Dünweg, and K. Kremer. submitted to Phys. Rev. E.

[31] F. Müller-Plathe. *Phys. Rev. E;*, **59**:4894–4899, 1999.

[32] A. Maggs and V. Rosseto. *Phys. Rev. Lett.*, **88**:196402, 2002.

[33] Homepage. `http://www.mpip-mainz.mpg.de/~pep/`.

Weitere Beiträge für den Heinz-Billing-Preis 2003

Detonation Simulation with the AMROC Framework

Ralf Deiterding
California Institute of Technology, Pasadena

Abstract

Numerical simulations can be the key to the thorough understanding of the multi-dimensional nature of transient detonation waves. But the accurate approximation of realistic detonations is extremely demanding, because a wide range of different scales need to be resolved. This paper describes an efficient simulation strategy based on a generic implementation of a blockstructured dynamically adaptive mesh refinement technique for distributed memory machines. Highly resolved detonation structure computations with detailed hydrogen-oxygen chemistry demonstrate the effectiveness of the approach in practice.

1 Introduction

Reacting flows have been a topic of on-going research since more than hundred years. The interaction between hydrodynamic flow and chemical kinetics can be extremely complex and even today many phenomena are not very well understood. One of these phenomena is the propagation of detonation waves in gaseous media. While detonations propagate at supersonic velocities between 1000 and $2000\,\mathrm{m/s}$, they inhibit non-neglectable instationary sub-structures in the millimeter range. Experimental observations can provide only limited insight and it is therefore not surprising that the understanding of the multi-dimensionality has improved only little since the first systematic investigations [9, 26]. An alternative to laboratory experiments are direct numerical simulations of the governing thermo- and hydrodynamic equations.

But the additional source terms modeling detailed non-equilibrium chemistry are often *stiff* and introduce new and extremely small scales into the flow field. Their accurate numerical representation requires finite volume meshes with extraordinarily high local resolution.

In this paper, we summarize our successful efforts in simulating multi-dimensional detonations with detailed and highly stiff chemical kinetics on recent parallel machines with distributed memory, especially on clusters of standard personal computers [7]. We explain the design of our public-domain framework AMROC (Adaptive Mesh Refinement in Object-oriented C++) [8] that implements the blockstructured mesh refinement approach after Berger and Collela [2]. Briefly, we sketch the employed numerical methods and the treatment of the reaction terms.

2 Detonation Theory

A detonation is a shock-induced combustion wave that internally consists of a discontinuous hydrodynamic shock wave followed by a smooth region of decaying combustion. The adiabatic compression due to the passage of the shock rises the temperature of the combustible mixture above the ignition limit. The reaction results in an energy release driving the shock wave forward. In a self-sustaining detonation, shock and reaction zone propagate essentially with an identical speed d_{CJ} that is approximated to good accuracy by the classical Chapman-Jouguet (CJ) theory, cf. [30]. But up to now, no theory exists that describes the internal flow structure satisfactory. The Zel'dovich-von Neumann-Döring (ZND) theory is widely believed to reproduce the one-dimensional detonation structure correctly, but already early experiments [9] uncovered that the reduction to one space dimension is not even justified in long combustion devices. It was found that detonation waves usually exhibit non-neglectable instationary multi-dimensional sub-structures and do not remain planar. The multi-dimensional instability manifests itself in instationary shock waves propagating perpendicular to the detonation front. A complex flow pattern is formed around each *triple point*, where the detonation front is intersected by a transverse shock. Pressure and temperature are increased remarkable in a triple point and the chemical reaction is enhanced drastically giving rise to an enormous local energy release. Hence, the accurate representation of triple points is essential for safety analysis, but also in technical applications, e.g. in the pulse detonation engine. Some particular mixtures, e.g. low-pressure hydrogen-oxygen with high argon diluent, are known to produce very regular triple point movements. The triple point trajectories form regular "fish-scale" patterns, so called detonation cells, with a characteristic length L and width λ (compare left sketch of Fig. 1).

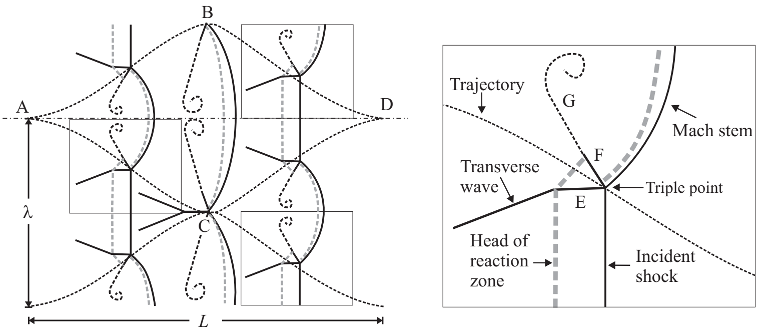


Fig. 1: Left: regular detonation structure at three different time steps on triple point trajectories, right: enlargement of a periodical triple point configuration. E: reflected shock, F: slip line, G: diffusive extension of slip line with flow vertex.

Fig. 1 displays the hydrodynamic flow pattern of a detonation with regular cellular structure as it is known since the early 1970s, cf. [26, 19]. The right sketch shows the periodic wave configuration around a triple point in detail. It consists of a Mach reflection, a flow pattern well-known from non-reactive supersonic hydrodynamics [4]. The undisturbed detonation front is called the incident shock, while the transverse wave takes the role of the reflected shock. The triple point is driven forward by a strong shock wave, called Mach stem. Mach stem and reflected shock enclose the slip line, the contact discontinuity.

The Mach stem is always much stronger than the incident shock, which results in a considerable reduction of the induction length l_{ig}, the distance between leading shock and measurable reaction. The shock front inside the detonation cell travels as two Mach stems from point A to the line BC. In the points B and C the triple point configuration is inverted nearly instantaneously and the front in the cell becomes the incident shock. Along the symmetry line AD the change is smooth and the shock strength decreases continuously. In D the two triple points merge exactly in a single point. The incident shock vanishes completely and the slip line, which was necessary for a stable triple point configuration between Mach stem and incident shock, is torn off and remains behind. Two new triple points with two new slip lines develop immediately after D.

3 Governing Equations

The appropriate model for detonation propagation in premixed gases with realistic chemistry are the inviscid Euler equations for multiple thermally perfect species with reactive source terms [12, 30]. These equations form a system of inhomogeneous hyperbolic conservation laws that reads

$$\begin{array}{rcrcl} \partial_t\rho_i & + & \nabla\cdot(\rho_i\mathbf{u}) & = & W_i\,\dot\omega_i\ , \quad i=1,\dots,K\ , \\ \partial_t(\rho\mathbf{u}) & + & \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u})+\nabla p & = & 0\ , \\ \partial_t(\rho E) & + & \nabla\cdot((\rho E+p)\mathbf{u}) & = & 0\ . \end{array} \tag{1}$$

Herein, ρ_i denotes the partial density of the ith species and $\rho=\sum_{i=1}^K\rho_i$ is the total density. The ratios $Y_i=\rho_i/\rho$ are called mass fractions. We denote the velocity vector by $\mathbf{u}$ and E is the specific total energy. We assume that all species are ideal gases in thermal equilibrium and the hydrostatic pressure p is given as the *sum* of the partial pressures $p_i=\mathcal{R}T\rho_i/W_i$ with $\mathcal{R}$ denoting the universal gas constant and W_i the molecular weight, respectively. The evaluation of the last equation requires the previous calculation of the temperature T. As detailed chemical kinetics typically require species with *temperature-dependent* material properties, each evaluation of T involves the approximative solution of an implicit equation by Newton iteration [7].

The chemical production rate for each species is derived from a reaction mechanism of J chemical reactions as

$$\dot\omega_i=\sum_{j=1}^{J}(\nu_{ji}^r-\nu_{ji}^f)\left[k_j^f\prod_{l=1}^{K}\left(\frac{\rho_l}{W_l}\right)^{\nu_{jl}^f}-k_j^r\prod_{l=1}^{K}\left(\frac{\rho_l}{W_l}\right)^{\nu_{jl}^r}\right],\ i=1,\dots,K\ , \tag{2}$$

with $\nu_{ji}^{f/r}$ denoting the forward and backward stoichiometric coefficients of the ith species in the jth reaction. The rate expressions $k_j^{f/r}(T)$ are calculated by an Arrhenius law, cf. [30].

4 Numerical Methods

We use the time-operator splitting approach or method of fractional steps [15] to decouple hydrodynamic transport and chemical reaction numerically. This technique is most frequently used for time-dependent reactive flow computations. The *homogeneous* Euler equations and the usually stiff system of ordinary differential equations

$$\partial_t\rho_i=W_i\,\dot\omega_i(\rho_1,\dots,\rho_K,T)\ ,\quad i=1,\dots,K \tag{3}$$

are integrated successively with the data from the preceding step as initial condition. The advantage of this approach is that a globally coupled implicit problem is avoided and a time-implicit discretization, which accounts for the stiffness of the reaction terms, needs to be applied only *local* in each finite volume cell. We use a semi-implicit Rosenbrock-Wanner method [16] to integrate Eq. (3) within each cell. Temperature-dependent material properties

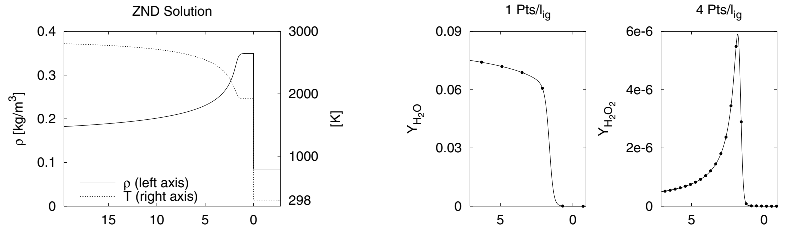


Fig. 2: A self-sustaining hydrogen-oxygen detonation ($d_{CJ} \approx 1627\,\mathrm{m/s}$, $l_{ig} \approx 1.404\,\mathrm{mm}$) calculated with the ZND theory and representation of two mass fraction distributions on grids with different mesh widths (right). The dots represent the values in the center of a finite volume. The abscissas display the distance behind the detonation front in mm.

are derived from look-up tables that are constructed during start-up of the computational code. The expensive reaction rate expressions (2) are evaluated by a mechanism-specific Fortran-77 function, which is produced by a source code generator on top of the Chemkin-II library [17] in advance. The code generator implements the reaction rate formulas without any loops and inserts constants like $\nu_{ji}^{f/r}$ directly into the code.

As detonations involve supersonic shock waves we use a finite volume discretization that achieves a proper upwinding in all characteristic fields. The scheme utilizes a quasi-one-dimensional approximate Riemann solver of Roe-type [14] and is extended to multiple space-dimensions via the method of fractional steps, cf. [27]. To circumvent the intrinsic problem of unphysical total densities and internal energies near vacuum due to the Roe linearization, cf. [11], the scheme has the possibility to switch to the simple, but extremely robust Harten-Lax-Van Leer (HLL) Riemann solver. Negative mass fraction values are avoided by a numerical flux modification proposed by Larrouturou [18]. Finally, the occurrence of the disastrous carbuncle phenomena, a multi-dimensional numerical crossflow instability that destroys every simulation of strong grid-aligned shocks or detonation waves completely [23], is prevented by introducing a small amount of additional numerical viscosity in a multi-dimensional way [25]. A detailed derivation of the entire Roe-HLL scheme including all necessary modifications can be found in [7]. This hybrid Riemann solver is extended to a second-order accurate method with the MUSCL-Hancock variable extrapolation technique by Van Leer [27].

4.1 Meshes for Detonation Simulation

Numerical simulations of detonation waves require computational meshes, which are able to represent the strong local flow changes due to the reaction correctly. In particular, the shock of a detonation wave with detailed kinetics can be very sensitive to changes of the reaction behind, and if the mesh is too coarse to resolve all reaction details correctly, the Riemann Problem at the detonation front is changed remarkably leading to a wrong speed of propagation. We make a simple discretization test in order to illustrate, how fine computational meshes for accurate detonation simulations in fact have to be. The two left graphs of Fig. 2 display the *exact* distributions of $Y_{\mathrm{H_2O}}$ and $Y_{\mathrm{H_2O_2}}$ according to the ZND detonation model for the frequently studied $\mathrm{H_2 : O_2 : Ar}$ Chapman-Jouguet detonation with molar ratios $2 : 1 : 7$ at $T_0 = 298\,\mathrm{K}$ and $p_0 = 6.67\,\mathrm{kPa}$ discretized with different grids.[1] Apparently, a resolution of 4 finite volumes per induction length ($4\,\mathrm{Pts}/l_{ig}$ with $l_{ig} = 1.404\,\mathrm{mm}$) is not sufficient to capture the maximum of the intermediate product $\mathrm{H_2O_2}$ correctly. This requires at least 5 to $6\,\mathrm{Pts}/l_{ig}$, but in triple points even finer resolutions can be expected. As discretizations of typical combustors with such fine uniform meshes typically would require up to 10^9 points in the two- and up to 10^{12} points in the three-dimensional case the application of a dynamically adaptive mesh refinement technique is indispensable.

5 An Adaptive Mesh Refinement Framework

In order to supply the required temporal and spatial resolution efficiently, we employ the blockstructured adaptive mesh refinement (AMR) method after Berger and Colella [2], which is tailored especially for hyperbolic conservation laws on logically rectangular finite volume grids. We have implemented the AMR method in a generic, dimension-independent object-oriented framework in C++. It is called AMROC (Adaptive Mesh Refinement in Object-oriented C++) and is free of charge for scientific use [8]. An efficient parallelization strategy for distributed memory machines has been found and the codes can be executed on all systems that provide the MPI library.

5.1 Berger-Collela AMR Method

Instead of replacing single cells by finer ones, as it is done in cell-oriented refinement techniques, the Berger-Collela AMR method follows a patch-oriented approach. Cells being flagged by various error indicators (shaded in Fig. 3) are clustered with a special algorithm [1] into non-overlapping rectangular grids. Refinement grids are derived recursively from coarser ones and a hierarchy of successively embedded levels is thereby constructed, cf. Fig. 3. All mesh widths on level l are r_l-times finer than on level $l-1$, i.e.

[1]Throughout this paper, only one hydrogen-oxygen reaction mechanism extracted from a larger hydrocarbon mechanism assembled by Westbrook has been employed [28]. The mechanism uses 34 elementary reactions for the 9 species H, O, OH, $\mathrm{H_2}$, $\mathrm{O_2}$, $\mathrm{H_2O}$, $\mathrm{HO_2}$, $\mathrm{H_2O_2}$ and Ar.

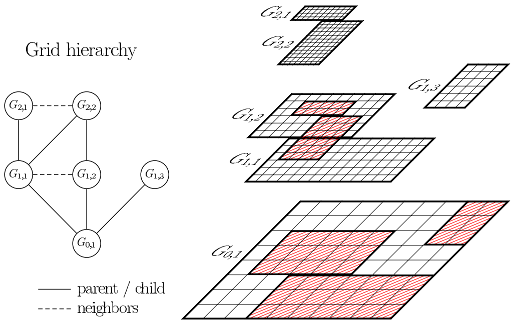


Fig. 3: The AMR method creates a hierarchy of rectangular subgrids.

$\Delta t_l := \Delta t_{l-1}/r_l$ and $\Delta x_{n,l} := \Delta x_{n,l-1}/r_l$ with $r_l \geq 2$ for $l > 0$ and $r_0 = 1$, and a time-explicit finite volume scheme (in principle) remains stable on all levels of the hierarchy. The recursive integration order visualized in the left sketch of Fig. 4 is an important difference to usual unstructured adaptive strategies and is one of the main reasons for the high efficiency of the approach.

The numerical scheme is applied on level l by calling a single-grid routine in a loop over all subgrids. The subgrids are computationally decoupled by employing ghost or halo cell values. Three types of different ghost cells have to be considered in the sequential case, see right sketch of Fig. 4. Cells outside of the root domain are used to implement physical boundary conditions. Ghost cells overlaid by a grid on level l have a unique interior cell analogue and are set by copying the data value from the grid, where the interior cell is contained (synchronization). On the root level no further boundary conditions need to be considered, but for $l > 0$ also internal boundaries can occur. They are set by a conservative time-space interpolation from two previously calculated time steps of level $l - 1$.

Beside a general data tree that stores the topology of the hierarchy (cf. Fig. 3), the AMR method requires at most two regular arrays assigned to each subgrid. They contain the discrete vector of state for the actual and updated time step. The regularity of the data allows high performance on vector and superscalar processors and cache optimizations. Small data arrays are effectively avoided by leaving coarse level data structures untouched, when higher level grids are created. Values of cells covered by finer subgrids are overwritten by averaged fine grid values subsequently. This operation leads to a modification of the numerical stencil on the coarse mesh and requires a special flux correction in cells abutting a fine grid. The correction replaces the coarse grid flux along the fine grid boundary by a *sum* of fine fluxes and ensures the discrete conservation property of the hierarchical method. See [2] or [7] for details.

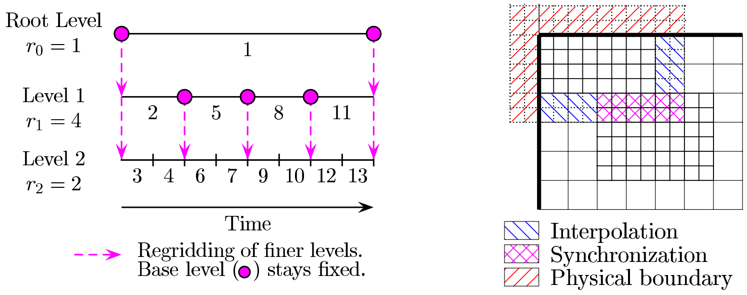


Fig. 4: Left: recursive integration order. Right: sources of ghost cell values.

5.2 Parallelization

Up to now, various reliable implementations of the AMR method for single processor computers have been developed [3, 5]. Even the usage of parallel computers with shared memory is straight-forward, because a time-explicit scheme allows the parallel calculation of the grid-wise numerical update [1]. But the question for an efficient parallelization strategy becomes more delicate for distributed memory architectures, because on such machines the costs for communication can not be neglected. Due to the technical difficulties in implementing dynamical adaptive methods in distributed memory environments only few parallelization strategies have been considered in practice yet, cf. [24, 22].

In the AMROC framework, we follow a rigorous domain decomposition approach and partition the AMR hierarchy from the root level on. The key idea is that all higher level domains are required to follow this “floor-plan”. A careful analysis of the AMR algorithm uncovers that the only parallel operations under this paradigma are ghost cell synchronization, redistribution of the AMR hierarchy and the application of the previously mentioned flux correction terms. Interpolation and averaging, but in particular the calculation of the flux corrections remain strictly local [6]. In AMROC we employ a generalization of Hilbert’s space-filling curve [21] to derive load-balanced root level distributions at runtime. The entire AMR hierarchy is considered by projecting the accumulated work from higher levels onto the root level cells.

5.3 Object-oriented Implementation in AMROC

In principle, three main abstraction levels can be identified in AMR. At the top level, the specific application is formulated with single-grid routines. Mandatory are the numerical scheme and the setting of physical boundary and initial conditions. The results in Sec. 6 were produced with subroutines in Fortran-77. The parallel AMR algorithm and its components for error estimation, grid generation and flux correction make up the middle level, which is completely in C++ in AMROC. The middle level is independent of the spatial dimension or the specific numerical scheme at the top level. The base level stores the topology of the hierarchy and allocates all kind of grid-based data. Additionally, it provides standard operations that require topological information, like ghost cell synchronization, interpolation or averaging to the middle level. Furthermore, elementary topological operations on grid sets, like $\cap$, $\cup$ or $\setminus$ are supplied. The necessary calculations are done effectively in a global integer coordinate system, cf. [1].

AMROC's hierarchical data structures are derived from the DAGH (Distributive Adaptive Grid Hierarchies) package by Parashar and Browne [22] and are implemented completely in C++. A redesign of large parts of the DAGH package was necessary to allow the AMR algorithm as it was described in the previous sections. Additional new features in AMROC are level-dependent refinement factors r_l, periodic boundary conditions, a restart option from memory for automatic time step algorithms and a restart feature from checkpointing files for a variable number of computing nodes. Currently, AMROC consists of approximately $46,000$ lines of code in C++ and approximately $6,000$ lines for visualization and data conversion.

6 Numerical Results

The self-sustaining CJ detonation of Sec. 4.1 is an ideal candidate for fundamental detonation structure simulations, because it produces extremely regular detonation cell patterns [26]. The application of the numerical methods of Sec. 4 in the parallel AMROC framework allowed a two-dimensional cellular structure simulation, which is four-times higher resolved ($44.8\,\mathrm{Pts}/l_{ig}$) than the best reference result that has been presented so far [20, 10, 13]. This calculation was run on a small Beowulf-cluster of 7 Pentium III-850 MHz-CPUs connected with a 1 Gb-Myrinet network and required 2150 h CPU-time. On 24 Athlon-1.4 GHz double-processor nodes (2 Gb-Myrinet) of the HEidelberg LInux Cluster System (Helics) our approach allowed the first sufficiently resolved computation of the three-dimensional cellular structure of a hydrogen-oxygen detonation. The maximal effective resolution of this calculation is $16.8\,\mathrm{Pts}/l_{ig}$ and the run required 3800 h CPU-time. Further on, we present the first successful simulations of diffracting two-dimensional hydrogen-oxygen detonations that reproduce the experimentally measured critical tube diameter of 10 detonation cells. These computations demonstrate the advantages in employing a dynamically adaptive method impressively and used approximately 4600 h CPU-time on the Helics.

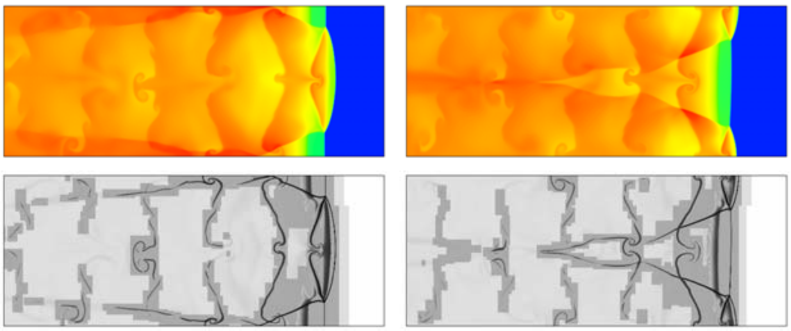


Fig. 5: Color plots of the temperature and schlieren plots of the density on refinement regions in the first (left) and second half (right) of a detonation cell.

6.1 Two-dimensional Cellular Structure

We extend the one-dimensional ZND detonation of Fig. 2 to two space dimensions and initiate transverse disturbances by placing a small rectangular unreacted pocket behind the detonation front, cf. [20] or [7]. After an initial period very regular detonation cells with oscillation period $\approx 32\,\mu$s show up. We exploit this regularity and simulate only a single cell. The calculation is done in a frame of reference attached to the detonation and requires just the computational domain $10\,\mathrm{cm} \times 3\,\mathrm{cm}$. The adaptive run uses a root level grid of 200×40 cells and two refinement levels with $r_{1,2} = 4$. A physically motivated combination of scaled gradients and heuristically estimated relative errors is applied as adaptation criteria. See [7] for details. Two typical snapshots with the corresponding refinement are displayed in Fig. 5.

The high resolution of the simulation now admits a remarkable refinement of the triple point pattern introduced in Sec. 2. As the two transverse waves form a perfectly regular flow, it suffices to zoom into a single triple point and to analyze the wave pattern between two triple point collisions in detail. Fig. 6 displays the flow situation around the primary triple point A that is mostly preserved during the last $7\,\mu$s before a collision. An analysis of the flow field uncovers the existence of two minor triple points B and C along the transverse wave downstream of A. While B can be clearly identified by a characteristic inflection, the triple point C is much weaker and very diffused. B is caused by the interaction of the strong shock wave BD with the transverse wave. The slip line emanating from B to K is clearly present. C seems to be caused by the reaction front and generates the very weak shock wave CI. Downstream of BD a weaker shock wave EF shows up. It is refracted in the point F as it hits the slip line BK. From F to G this minor shock is parallel and close to the transverse wave, which results in a higher pressure increase in the region FG

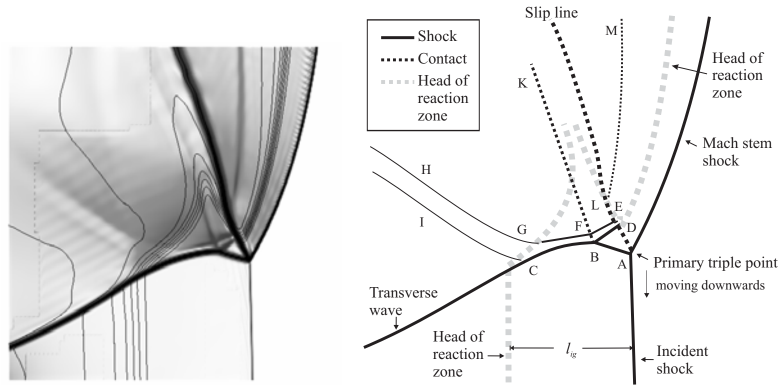


Fig. 6: Flow structure around a triple before the next collision. Left: isolines of Y_{OH} (black) on schlieren plot of u_2 (gray).

than in the region EF. Unreacted gas crossing the transverse wave between B and C therefore shows a shorter induction length than gas entering through AB. The minor shock is refracted and weakened by the reaction front at point G and forms the shock GH that is almost parallel to CI. The downstream line of separation between particles passing through incident or Mach Stem shock is the slip line AD. Along its extension DEL the movement of A results in a shear flow between the reaction zones behind the Mach stem and downstream of BD.

6.2 Three-dimensional Cellular Structure

We utilize the regular oscillating solution of the preceding section as initial condition for a three-dimensional simulation and disturb the oscillation in the x_2-direction with an unreacted pocket in the orthogonal direction. We use a computational domain of the size $7\,\mathrm{cm} \times 1.5\,\mathrm{cm} \times 3\,\mathrm{cm}$ that exploits the symmetry of the initial data, but allows the development of a full detonation cell in the x_3-direction. The AMROC computation uses a two-level refinement with $r_1 = 2$ and $r_2 = 3$ on a base grid of $140 \times 12 \times 24$ cells and utilizes between 1.3 M and 1.5 M cells, instead of 8.7 M cells like a uniformly refined grid.

After a simulation time of $\approx 600\,\mu\mathrm{s}$ a regular cellular oscillation with identical strength in x_2- and x_3-direction can be observed. In both transverse directions the strong two-dimensional oscillations is present and forces the creation of rectangular detonation cells of $3\,\mathrm{cm}$ width. The transverse waves form triple point lines in three space-dimensions. During a complete detonation cell the four lines remain mostly parallel to the boundary

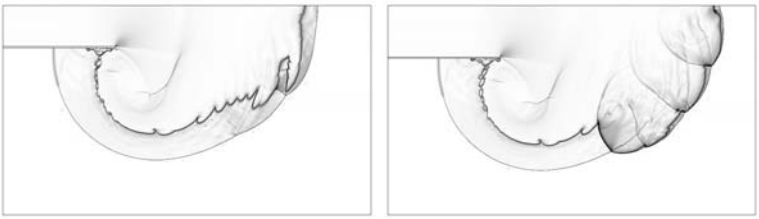


Fig. 7: Schlieren plots of ρ for a detonation diffracting out of the two different tubes. Left: detonation failure for the width $w = 8\lambda$, right: reinitiation for $w = 10\lambda$.

and hardly disturb each other. The characteristic triple point pattern can therefore be observed in Fig. 9 in all planes perpendicular to a triple point line. Unlike Williams et al. [29] who presented a similar calculation for an overdriven detonation with simplified one-step reaction model, we notice no phase-shift between both transverse directions. In all our computations for the hydrogen-oxygen CJ detonation only this regular three-dimensional mode, called "rectangular-mode-in-phase", or a purely two-dimensional mode with triple point lines just in x_2- or x_3-direction did occur.

6.3 Structure of Diffracting Detonations

Experiments have shown that the behavior of planar CJ detonations propagating out of tubes into unconfinement is determined mainly by the width of the tube. For square tubes the critical tube width has been found to be of the order of 10-times the cell width, i.e. 10λ [19]. For widths significantly below 10λ the process of shock wave diffraction causes a pressure decrease

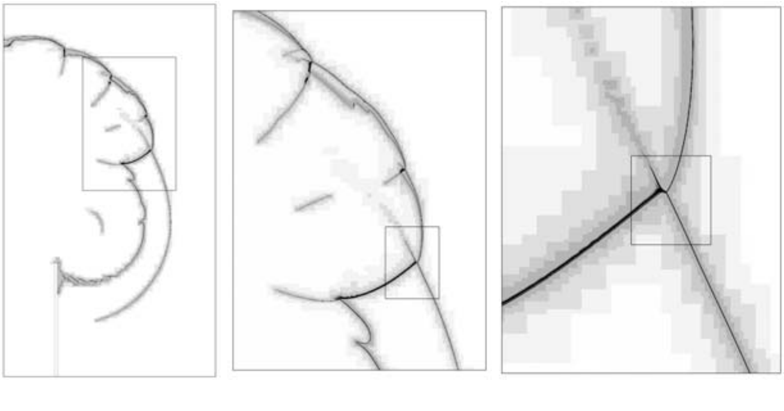


Fig. 8: Density distribution on four refinement levels at $t_{end} = 240\ \mu\text{s}$ for $w = 10\lambda$. Multiple enlargements are necessary to display the refinement levels (visualized by different gray tones).

at the head of the detonation wave below the limit of detonability across the entire tube width. Hydrodynamic shock and reaction front decouple and the detonation decays to a shock-induced flame. This observation is independent of a particular mixture. While the successful transmission of the detonation is hardly disturbed for tubes widths $\gg 10\lambda$, a backward-facing re-ignition wave reinitiates the detonation in the partially decoupled region for widths of $\approx 10\lambda$ and creates considerable vortices.

Adaptive simulations on a base grid of 508×288 cells and with four levels of refinement with $r_{1,2,3} = 2$, $r_4 = 4$ perfectly reproduce the experimental observations. The schlieren graphics of Fig. 7 clearly show the extinction for the tube width $w = 8\lambda$ and the re-ignition wave for $w = 10\lambda$. These computations correspond to a uniform grid with $\approx 150\,\mathrm{M}$ cells and have an

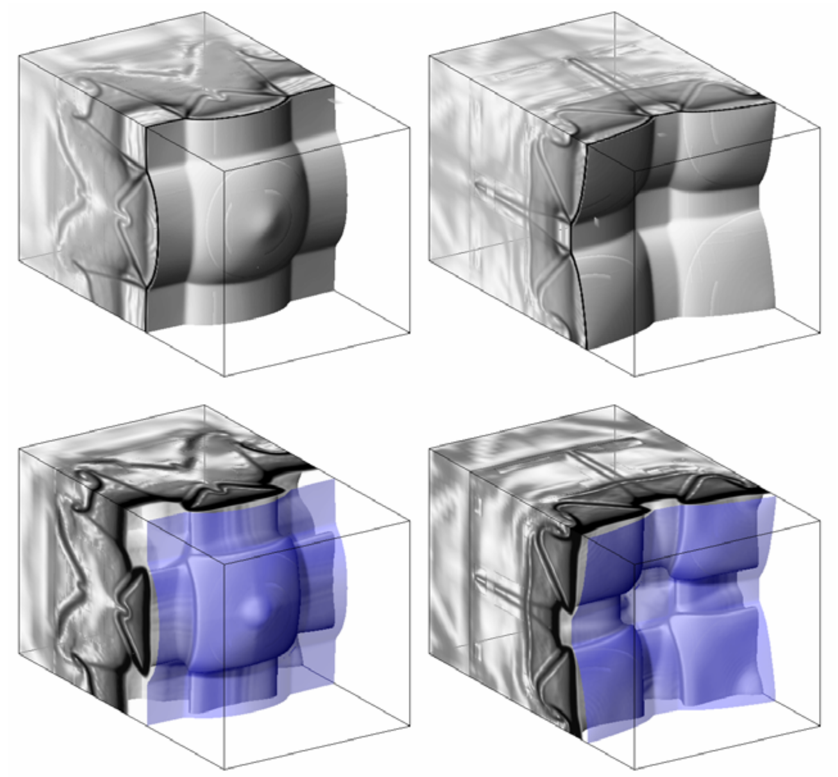


Fig. 9: Schlieren plots of ρ (upper row) and Y_{OH} (lower row) in the first (left) and second (right) half of detonation cell, mirrored at $x_2 = 0\,\mathrm{cm}$, $5.0\,\mathrm{cm} < x_1 < 7.0\,\mathrm{cm}$. The plots of Y_{OH} are overlaid by a blue isosurface of ρ that visualizes the induction length l_{ig}.

effective resolution of $25.5\,\mathrm{Pts}/\mathrm{l_{ig}}$ in the x_1-direction (with respect to the initial detonation). At the final time $t_{end} = 240\,\mu$s the larger run for $w = 10\lambda$ uses only $\approx 3.0\,\mathrm{M}$ cells on all levels. Fig. 8 visualizes the efficiency of the adaptive approach.

7 Conclusions

We have described an efficient solution strategy for the numerical simulation of gaseous detonations with detailed chemical reaction. All temporal and spatial scales relevant for the complex process of detonation propagation were successfully resolved. Beside the application of the time-operator splitting technique and the construction of a robust high-resolution shock capturing scheme, the key to the high efficiency of the presented simulations is the generic implementation of the blockstructured AMR method after Berger and Collela [2] in our AMROC framework [8]. AMROC provides the required high local resolution dynamically and follows a parallelization strategy tailored especially for the emerging generation of distributed memory architectures. All presented results have been achieved on Linux-Beowulf-clusters of moderate size in a few days real time, which demonstrates that advances in computational fluid dynamics do not necessarily require large-scale super-computers, but integrated approaches that combine fast and accurate discretizations with sophisticated techniques from computer science.

Acknowledgements

This work was supported by the DFG high priority research program “Analysis and Numerics of Conservation Laws”, grant Ba 840/3-3, while the author was at the Institute of Mathematics, Technical University Cottbus, Germany.

References

[1] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138, 1994.

[2] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1988.

[3] M. Berger and R. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35(6):2298–2316, 1998.

[4] R. Courant and K. O. Friedrichs. *Supersonic flow and shock waves.* Applied mathematical sciences, volume 21. Springer, New York, Berlin, 1976.

[5] W. Crutchfield and M. L. Welcome. Object-oriented implementation of adaptive mesh refinement algorithms. *J. Scientific Programming*, 2:145–156, 1993.

[6] R. Deiterding. Construction and application of an AMR algorithm for distributed memory computers. In *Proc. of Chicago Workshop on Adaptive Mesh Refinement Methods*, page to appear. ASCI/Flash Center, Sep 3-5 2003.

[7] R. Deiterding. *Parallel adaptive simulation of multi-dimensional detonation structures.* PhD thesis, Techn. Univ. Cottbus, Sep 2003.

[8] R. Deiterding. AMROC - Blockstructured Adaptive Mesh Refinement in Object-oriented C++. Available at http://amroc.sourceforge.net, Oct 2003.

[9] Y. N. Denisov and Y. K. Troshin. Structura gazovoi detonatsii v trubakh (Structure of gaseous detonations in tubes). *Zh. Eksp. Teor. Fiz.*, 30(4):450–459, 1960.

[10] C. A. Eckett. *Numerical and analytical studies of the dynamics of gaseous detonations*. PhD thesis, California Institute of Technology, Pasadena, California, Sep 2001.

[11] B. Einfeldt, C. D. Munz, P. L. Roe, and B. Sjögreen. On Godunov-type methods near low densities. *J. Comput. Phys.*, 92:273–295, 1991.

[12] W. Fickett and W. C. Davis. *Detonation*. University of California Press, Berkeley and Los Angeles, California, 1979.

[13] T. Geßner. *Dynamic mesh adaption for supersonic combustion waves modeled with detailed reaction mechanisms*. PhD thesis, Math. Fakultät, University Freiburg, 2001.

[14] B. Grossmann and P. Cinella. Flux-split algorithms for flows with non-equilibrium chemistry and vibrational relaxation. *J. Comput. Phys.*, 88:131–168, 1990.

[15] N. N. Janenko. *Die Zwischenschrittmethode zur Lösung mehrdimensionaler Probleme der mathematischen Physik*. Springer-Verlag, Berlin, 1969.

[16] P. Kaps and P. Rentrop. Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations. *Num. Math.*, 33:55–68, 1979.

[17] R. J. Kee, F. M. Rupley, and J. A. Miller. *Chemkin-II: A Fortran chemical kinetics package for the analysis of gas-phase chemical kinetics*. SAND89-8009, Sandia National Laboratories, Livermore, California, Sep 1989.

[18] B. Larrouturou. How to preserve the mass fractions positivity when computing compressible multi-component flows. *J. Comput. Phys.*, 95:59–84, 1991.

[19] J. H. S. Lee. Dynamic parameters of gaseous detonations. *Ann. Rev. Fluid Mech.*, 16:311–336, 1984.

[20] E. S. Oran, J. W. Weber, E. I. Stefaniw, M. H. Lefebvre, and J. D. Anderson. A numerical study of a two-dimensional H_2-O_2-Ar detonation using a detailed chemical reaction model. *J. Combustion Flame*, 113:147–163, 1998.

[21] M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proc. of the 29th Annual Hawaii Int. Conf. on System Sciences*, Jan 1996.

[22] M. Parashar and J. C. Browne. System engineering for high performance computing software: The HDDA/DAGH infrastructure for implementation of parallel structured adaptive mesh refinement. In *Structured Adaptive Mesh Refinement Grid Methods*, IMA Volumes in Mathematics and its Applications. Springer, 1997.

[23] J. J. Quirk. Godunov-type schemes applied to detonation flows. In J. Buckmaster, editor, *Combustion in high-speed flows: Proc. of a Workshop on Combustion, Oct 12-14, 1992, Hampton,*, pages 575–596, Dordrecht, 1994. Kluwer Acad. Publ.

[24] C. A. Rendleman, V. E. Beckner, M. Lijewski, W. Crutchfield, and J. B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3, 2000.

[25] R. Sanders, E. Morano, and M.-C. Druguett. Multidimensional dissipation for upwind schemes: Stability and applications to gas dynamics. *J. Comput. Phys.*, 145:511–537, 1998.

[26] R. A. Strehlow. Gas phase detonations: Recent developments. *J. Combustion Flame*, 12(2):81–101, 1968.

[27] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics*. Springer-Verlag, Berlin, Heidelberg, 2nd edition, 1999.

[28] C. K. Westbrook. Chemical kinetics of hydrocarbon oxidation in gaseous detonations. *J. Combustion Flame*, 46:191–210, 1982.

[29] D. N. Williams, L. Bauwens, and E. S. Oran. Detailed structure and propagation of three-dimensional detonations. In *26th International Symposium on Combustion*, Naples, Apr. 1996.

[30] F. A. Williams. *Combustion theory*. Addison-Wesley, Reading, Massachusetts, 1985.

An Online Recruitment System for Economic Experiments

Ben Greiner
Department of Economics, University of Cologne

Abstract

In this paper we introduce the Online Recruitment System for Economic Experiments (ORSEE). With this software experimenters have a free, convenient and very powerful tool to organize their experiments and sessions in a standardized way. Additionally, ORSEE provides subject pool statistics, a laboratory calendar and tools for scientific exchange. A test system has been installed in order to visually support the reader while reading the paper.[1]

1 Introduction

Laboratory experimentation has been a growing field in economics for the last decades.[2] But the more experiments have been conducted in economics, the more the issue of an appropriate methodology and organization has been raised.

At the moment, there are the following items which are commonly agreed to be symptomatic for economic experiments (compared to human subject experiments in psychology and other social sciences):

[1]See `http://www.orsee.org` for a test system, downloads and a complete manual.

[2]For an introduction into experimental methodology and an overview about history and topics of experimental economics see Davis and Holt (1993), Friedman and Sunder (1994) and Kagel and Roth (1995).

- Subjects are payed for their participation.
- Payment should reflect subjects' performance in the experiment, i.e. the strategy space should translate to the payoff space.[3]
- Subjects should be volunteers motivated by the experimenters' payment.
- Subjects should not be deceived.

However, there is a wide variety in the procedures of maintaining a subject pool and organizing experiments. In this paper we introduce the Online Recruitment System for Economic Experiments (ORSEE), which aims

- to simplify the organization of economic laboratory experiments,
- to standardize the procedures of experiment organization,
- to depersonalize the experimenter-subject interaction,
- to allow the conduction of simple internet experiments,
- to provide information and statistics about the subject pool.

ORSEE has been implemented and has been online in Jena, Germany since March 2003. Currently, it is used at four institutions.[4] The software is maintained at `sourceforge.net`.[5] There you find the orsee-announce mailing list, a bug report and a feature request tracker. In order to support the reader while going through this paper a test system has been installed.[6]

Section 2 lists the functions of ORSEE and some technical parameters. Next, we describe the two essential views of the system: the public and the administration area. Section 5 and 6 show how laboratory and online surveys can be conducted in ORSEE, respectively.

Before starting, some terms used throughout this paper should be defined: A 'session' is defined as processing an experiment at a certain time at a certain location. An 'experimenter' is a person who conducts and/or administrates an experiment. A 'subject'/'participant' is a person who is recruited to participate in an experiment. Using ORSEE, experimenters create experiments which may consist of several sessions and invite subjects. Invited subjects may register themselves at one of the experiments' sessions in order to participate.

[3]See Harrison (1989).

[4]Max Planck Institute for Research into Economic Systems in Jena, Humboldt University Berlin, University of Bonn and University of Cologne.

[5]`http://sourceforge.net/projects/orsee/`

[6]`http://www.orsee.org`

2 Features

2.1 General Features

- multiple laboratory/subject pool/experimenters/experiment types support
- attribute query selection (e.g., select female participants, select participants who have not participated in experiment X)
- random recruitment of subjects out of subject pool
- structured public and internal experiment calendar including lab reservation for maintenance etc.
- reputation system (number of no-shows, i.e. the number of times a participant registered for a session but did not show up)
- automated mailing for registration process
- subjects are informed by automated e-mails about the sessions they registered for
- rule based automated session reminder mailing
- subjects are able to manage their own account (without password, using an individualized URL)
- overview about registration state
- experimenters are informed about session's state by automated e-mails
- calendar and session's participants lists in printable format (pdf)
- upload / download section
- build-in module for designing and conducting complete online surveys
- regular e-mails with experiment calendar and subject pool statistics to experimenters
- multiple experimenter/subject language support
- easily configurable via the web interface
- customizable layout
- open source

2.2 Technical Features

- LAMP application
  - runs on *Linux*
  - *Apache* webserver recommended
  - uses *MySQL* database
  - implemented in *PHP*
- data is completely separated from the application
- recommended system: Linux on i386/i686 processors, other unixes and Windows Server should work as well
- further requirements: *PHP* on command line, *webalizer* for usage statistics, *cron daemon* for regular jobs

3 The Public Area

The public area is the part of ORSEE which visitors and (potential) subjects can access without having a password. All information provided at these pages can be edited in the administration section.

The *rules page* displays the general experiment rules set by the institution. These can include rules for laboratory experiments, internet experiments, video experiments, or online surveys. The rules should contain information about the reputation system used, and the experimenters' policy of handling no-show-ups and late-comers. They should also contain general information about the normal procedure of an experiment, if there are payments or not and so on. In order to deal with legal requirements and to provide trust to potential participants ORSEE provides a *privacy policy page* which should state the policy of the institution regarding the data in the recruitment system and the data collected in experiments.

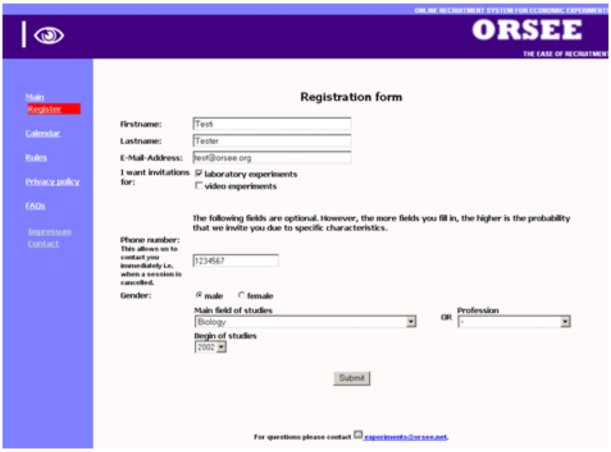
Subjects regularly have questions about details of the registration procedure, the use of the system and the practices you are using to organize and conduct experiments at your institution. The *FAQ page* answers these questions. They are ordered by an evaluation number, which is the sum of different participants who thought that this note answered their question. The answer page opens in a separate small window.

The *calendar* contains an overview of all experiments and their sessions. The information about a session contains the (public) name of the experiment, the time, date, duration and location of the session and the status of the session. The latter is denoted as *free places* if the registration time has not expired and there are free places left, and as *completed* otherwise. Thus, sessions which are not full already but whose registration period has expired are marked as complete at the *public* area.

To register in the system, potential subjects click on the appropriate link in the menu. First participants have to choose their own sub-subjectpool. The page will only show up if different subgroups are defined. After having selected their subgroup, people see the rules for experiments and the privacy policy. They have to accept both by clicking on the acceptance button before coming to the registration page.

On the *registration page* (see Figure 1) people can enter their data. Only the e-mail address, the last name, and the first name are required. A text shown above the form indicates that providing additional personal information can lead to more invitations. These details include gender, phone number, begin of study, field of study, and profession (depending on sub-subjectpool).

After submitting the registration form by clicking the button, the data is checked for doubletons with already registered participants and will be inserted only in a temporary table. The candidates are informed that they will



Registration form

Firstname:	Testi
Lastname:	Tester
E-Mail-Address:	test@orsee.org
I want invitations for:	[✓ laboratory experiments] video experiments

The following fields are optional. However, the more fields you fill in, the higher is the probability that we invite you due to specific characteristics.

Phone number: This allows us to contact you immediately i.e. when a session is cancelled.	1234567
Gender:	[✓ male] female
Main field of studies	Biology
OR Profession	-
Begin of studies	2002

For questions please contact experiments@orsee.net.

Fig. 1: *Participant Registration Form.*

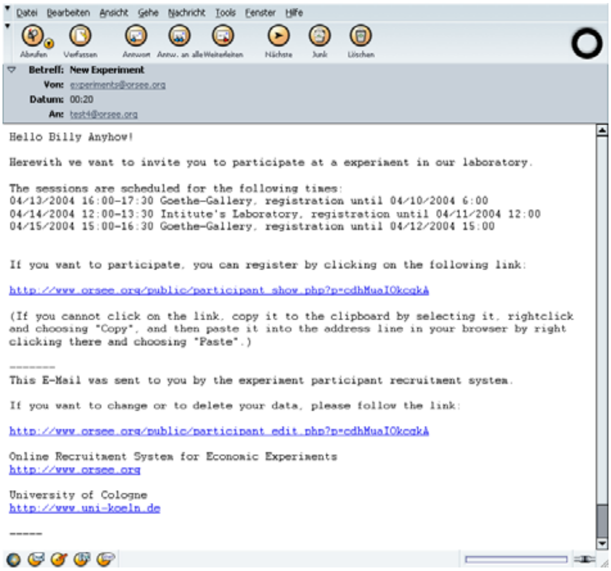
receive an e-mail to the account given to confirm their registration. This aims to avoid nonsense registrations. In their confirmation e-mail participants receive a link to click on. This brings them to a page confirming their successful registration. Now the data will be inserted in the regular participant table.

Every *e-mail* participants get from the system contains a link in the footer, which leads to the participant data editing page. Here a form similar to the registration form allows the user to change his data or to unsubscribe in order not to get further invitations. To keep database integrity the account is not deleted internally. If the subject tries later to register again with the system, the system recognizes him and an experimenter can reactivate his account by hand. On the privacy policy page we use to declare that we will delete personal data on written request (see next Section 4).

An experiment invitation e-mail (see Figure 2) includes another link which leads to the participant's *experiment registration page* (see Figure 3). This page consists of three parts:

– a list of future sessions of all experiments the participant has been invited for, yet has not participated or registered so far and for which the registration period has not expired
– a list of the future experiment sessions the participant has already registered for
– a list of former sessions a participant was registered for, including a summary of *finished* sessions

While the two latter parts only have informational character, the first list contains small *register* buttons on the right side. If a user clicks on one of



Betreff: New Experiment
Von: experiments@orsee.org
Datum: 00:20
An: test4@orsee.org

```
Hello Billy Anyhow!

Herewith we want to invite you to participate at a experiment in our laboratory.

The sessions are scheduled for the following times:
04/13/2004 16:00-17:30 Goethe-Gallery, registration until 04/10/2004 6:00
04/14/2004 12:00-13:30 Intitute's Laboratory, registration until 04/11/2004 12:00
04/15/2004 15:00-16:30 Goethe-Gallery, registration until 04/12/2004 15:00


If you want to participate, you can register by clicking on the following link:

http://www.orsee.org/public/participant_show.php?p=cdhMuaIOkcgkA

(If you cannot click on the link, copy it to the clipboard by selecting it, rightclick
and choosing "Copy", and then paste it into the address line in your browser by right
clicking there and choosing "Paste".)

--------
This E-Mail was sent to you by the experiment participant recruitment system.

If you want to change or to delete your data, please follow the link:

http://www.orsee.org/public/participant_edit.php?p=cdhMuaIOkcgkA

Online Recruitment System for Economic Experiments
http://www.orsee.org

University of Cologne
http://www.uni-koeln.de

-----
```

Fig. 2: Experiment invitation e-mail.

these buttons, he registers for the corresponding session. A text above the page informs participants about the binding character of the registration. At the same time an e-mail is sent out to the participant's e-mail address to inform him again about the successful registration, containing the date, time and location of the session.

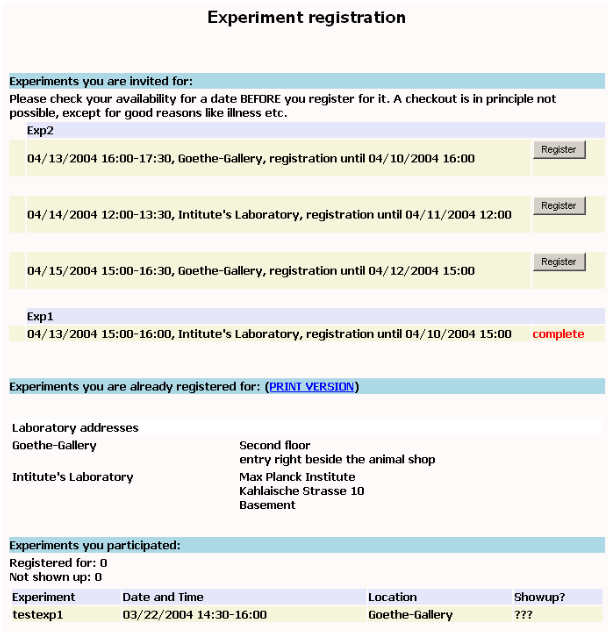
At a time specified at the administration page for the session a subject has registered for (e.g., 48 hours before the session starts), a session reminder e-mail is sent out to the participant.

ORSEE provides no mean for subjects to deregister from a session. We rather encourage subjects to check thoroughly if they are available for the time of the session at the moment of experiment registration. However, if there are reasons beyond his control, the participant can write an e-mail as a reply to his registration e-mail, and the experimenter will deregister him.

4 The Administration Area

To access the administration area, an experimenter has to log in first. Usernames, passwords and user rights are provided by a superuser administrator.

Beside the experiment organization logic, the administration area provides a bunch of useful functions. In the *options section* nearly all settings regarding


Experiment registration

Experiments you are invited for:

Please check your availability for a date BEFORE you register for it. A checkout is in principle not possible, except for good reasons like illness etc.

Exp2

04/13/2004 16:00-17:30, Goethe-Gallery, registration until 04/10/2004 16:00	Register
04/14/2004 12:00-13:30, Intitute's Laboratory, registration until 04/11/2004 12:00	Register
04/15/2004 15:00-16:30, Goethe-Gallery, registration until 04/12/2004 15:00	Register

Exp1

04/13/2004 15:00-16:00, Intitute's Laboratory, registration until 04/10/2004 15:00	complete

Experiments you are already registered for: (PRINT VERSION)

Laboratory addresses

Goethe-Gallery	Second floor entry right beside the animal shop
Intitute's Laboratory	Max Planck Institute Kahlaische Strasse 10 Basement

Experiments you participated:

Registered for: 0

Not shown up: 0

Experiment	Date and Time	Location	Showup?
testexp1	03/22/2004 14:30-16:00	Goethe-Gallery	???

Fig. 3: Experiment registration page.

the system can be done. These are general settings as system e-mail sender address, defaults for forms and output, colors used, professions and fields of study known by the system, the schedule of regular tasks to be done automatically by the system, default e-mail texts, the page content for the public pages, and the FAQs listed in the public area.

In this section, also the laboratories the system serves on can be registered. You may create different public experiment types to which participants may subscribe and match them with the internal experiment types implemented in ORSEE (laboratory and internet experiments, online surveys).

Different sub-subjectpools can be set up. At time of registration, subjects self-select to a subject pool by a provided self description. This allows you to administrate different populations, for example undergraduate students at your university, professional internet experiment participants, PhD students and so on.

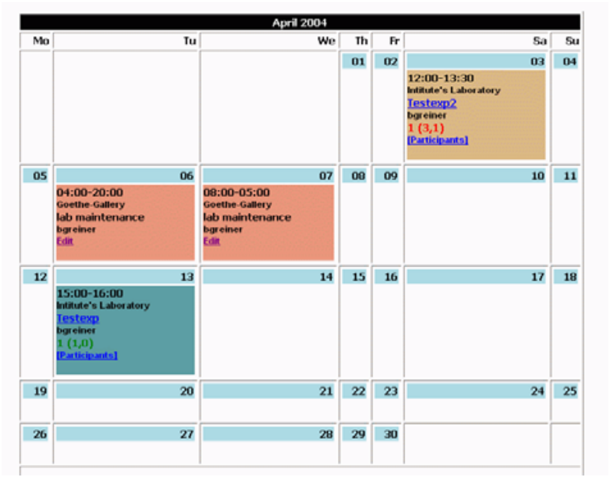
A special strength of ORSEE is it's multilingualism. Every output by the system (e-mails, pages, pdf files etc.) is configured in a huge language table. Thus, every experimenter and even every participant may select the language to communicate with the system given that the language is installed. You may edit all output via the web interface, and even create a new language. To do this, you have to translate the terms of an existing language to yours.

The *experiment calendar* (see Figure 4) provides the current state of laboratory booking, but also the timing and registration state of sessions etc. You may access the experiments, sessions and participant lists directly from the calendar. The calendar is sent to subscribed experimenters as a regular system task.

In the *downloads section* all general uploads like the system's manual and files for experiments (like instructions, programs, presentations) uploaded by experimenters can be found. This facilitates collaboration and learning between experimenters.

The *participants section* allows the experimenter to maintain the subject pool. You have the option to search through the current subject pool or 'deleted' participants (see below), to add new participants or to send out bulk mails to selected subjects. At each participant's page, you will see a complete history of his experiment registrations and participation (see Figure 5). To keep database integrity, in ORSEE you cannot really delete a participant from the database. There are three options you have:

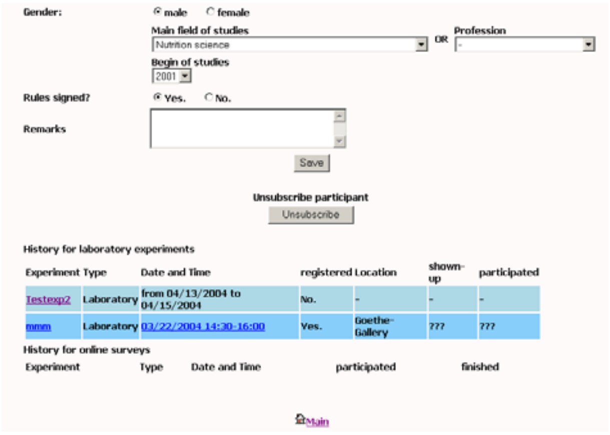
1. You can unsubscribe the subject. This is what the participant can also do himself at his personal data page. An unsubscribed participant will not receive any invitations to experiment sessions anymore.
2. You can exclude the subject. This is nothing else than the unsubscription with an additional flag set that it was not the participant's choice to get



April 2004

Mo	Tu	We	Th	Fr	Sa	Su
			01	02	03 12:00-13:30 Intitute's Laboratory Testexp2 bgreiner 1 (3,1) [Participants]	04
05	06 04:00-20:00 Goethe-Gallery lab maintenance bgreiner Edit	07 08:00-05:00 Goethe-Gallery lab maintenance bgreiner Edit	08	09	10	11
12	13 15:00-16:00 Intitute's Laboratory Testexp bgreiner 1 (1,0) [Participants]	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Fig. 4: Internal Experiment Calendar.


Gender: [✓ male] female

Main field of studies: Nutrition science OR Profession: -

Begin of studies: 2001

Rules signed? [✓ Yes.] No.

Remarks

History for laboratory experiments

Experiment	Type	Date and Time	registered	Location	shown-up	participated
Testexp2	Laboratory	from 04/13/2004 to 04/15/2004	No.	-	-	-
mmm	Laboratory	03/22/2004 14:30-16:00	Yes.	Goethe-Gallery	???	???

History for online surveys

Experiment	Type	Date and Time	participated	finished

Fig. 5: Participant Edit Page.

unsubscribed. A reason for exclusion might be that the subject has not shown up at his booked experiment sessions for a certain number of times.

3. You may empty the subject's personal data. This option is included due to privacy issues. Only the subject's ID will be kept in the system, but all personal data will be deleted. You may recover unsubscribed and excluded participant entries, but not emptied ones.

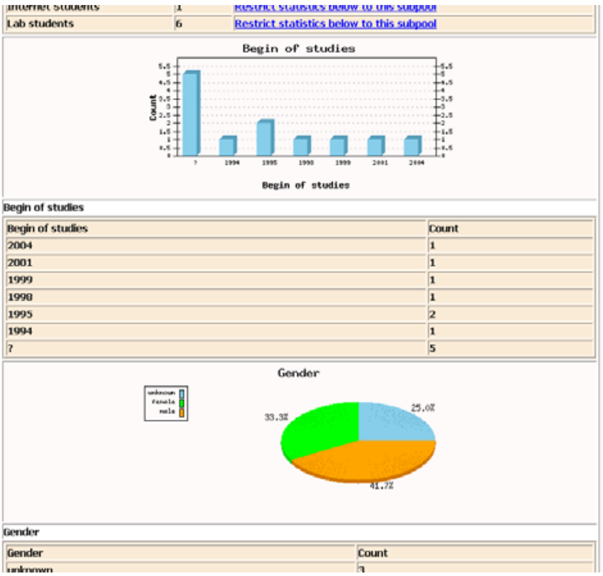
A comprehensive *statistics section* provides user with summarizing data. All actions of participants in the public area, experimenter actions in the administration area and all runs of regular tasks by the system are logged to the database. In this section you can surf these log files.

Moreover, you may have a look at the complete webserver statistics for the system's server directory (generated by *webalizer*), graphs and tables for experiment participation and user actions, and full subject pool statistics for gender, profession, field of studies, experience, no-shows per month and per count, which can be restricted to sub-subjectpools (see Figure 6).

5 Conducting a Laboratory Experiment

In this section we will describe the procedure of organizing a laboratory experiment with ORSEE. The *experiment overview page* lists the current experiments already registered in the system (see Figure 7). To list only the experimenter's or already finished experiments, use the links in the menu on the left side.

From here the experimenter may access the experiment main page of the



Internet students	1	Restrict statistics below to this subpool
Lab students	6	Restrict statistics below to this subpool

Begin of studies

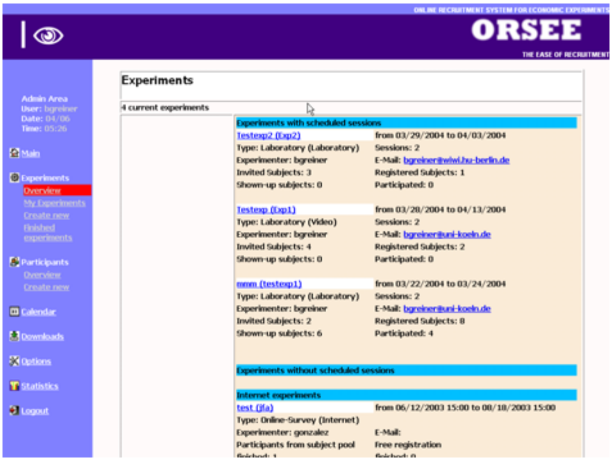
Begin of studies	Count
2004	1
2001	1
1999	1
1998	1
1995	2
1994	1
?	5

Gender

Gender	Count
unknown	3

Fig. 6: Subjectpool Statistics.

listed experiments by clicking on the appropriate name, or create a new experiment. On the *experiment creation page* (Figure 8) she fills in the internal



Experiments

4 current experiments

Experiments with scheduled sessions

Testexp2 (Exp2)	from 03/29/2004 to 04/03/2004
Type: Laboratory (Laboratory)	Sessions: 2
Experimenter: bgreiner	E-Mail: bgreiner@wiwi.hu-berlin.de
Invited Subjects: 3	Registered Subjects: 1
Shown-up subjects: 0	Participated: 0
Testexp (Exp1)	from 03/28/2004 to 04/13/2004
Type: Laboratory (Video)	Sessions: 2
Experimenter: bgreiner	E-Mail: bgreiner@uni-koeln.de
Invited Subjects: 4	Registered Subjects: 2
Shown-up subjects: 0	Participated: 0
mmm (testexp1)	from 03/22/2004 to 03/24/2004
Type: Laboratory (Laboratory)	Sessions: 2
Experimenter: bgreiner	E-Mail: bgreiner@uni-koeln.de
Invited Subjects: 2	Registered Subjects: 8
Shown-up subjects: 6	Participated: 4

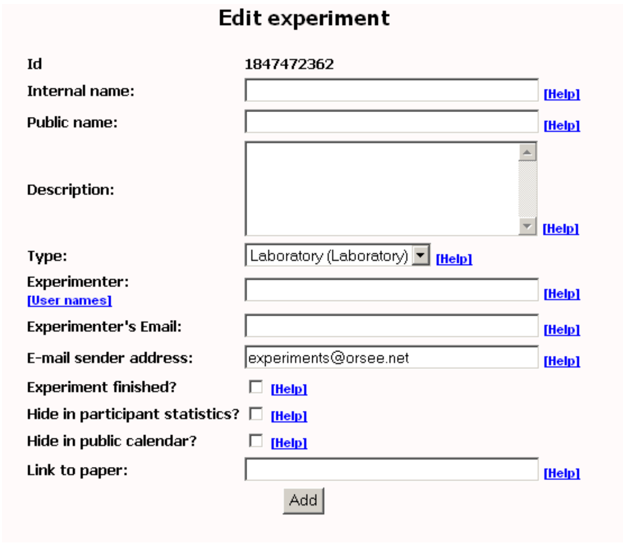
Experiments without scheduled sessions

Internet experiments

test (jfa)	from 06/12/2003 15:00 to 08/18/2003 15:00
Type: Online-Survey (Internet)	
Experimenter: gonzalez	E-Mail:
Participants from subject pool	Free registration
finished: 1	finished: 0

Fig. 7: Experiment Overview.

and public name of the experiment, a description, the type of experiment, her name and e-mail address. The public name is used to identify the experiment in the subject area of the system, the internal name is used at the administration pages and in e-mails to experimenters.



Edit experiment

Id	1847472362	
Internal name:		[Help]
Public name:		[Help]
Description:		[Help]
Type:	Laboratory (Laboratory)	[Help]
Experimenter: [User names]		[Help]
Experimenter's Email:		[Help]
E-mail sender address:	experiments@orsee.net	[Help]
Experiment finished?	☐	[Help]
Hide in participant statistics?	☐	[Help]
Hide in public calendar?	☐	[Help]
Link to paper:		[Help]

Fig. 8: Experiment Creation Page.

After adding an experiment, the experimenter uses the *session creation page* (which is accessible from the experiment's main page) to register each of the planned sessions with date and time, laboratory, experiment duration, number of participants needed and over-recruited, the time of registration and when the reminder should be sent to registered participants (see Figure 9). When creating or editing a session the system checks whether the session clashes with another laboratory booking and the experimenter gets a feedback.

Next, she *assigns subjects* registered in the database to her experiment. When doing so, she can use different queries including name, e-mail-address, number of no-shows and registrations, sub-subjectpool, gender, profession, field and start of studies, and participation/registration at certain old experiments.

ORSEE provides the feature to select a random subsample of a defined size from the registered participants matching the query. This should be used to prevent a bias regarding the fact that some subjects have immediate and more often internet access than others.

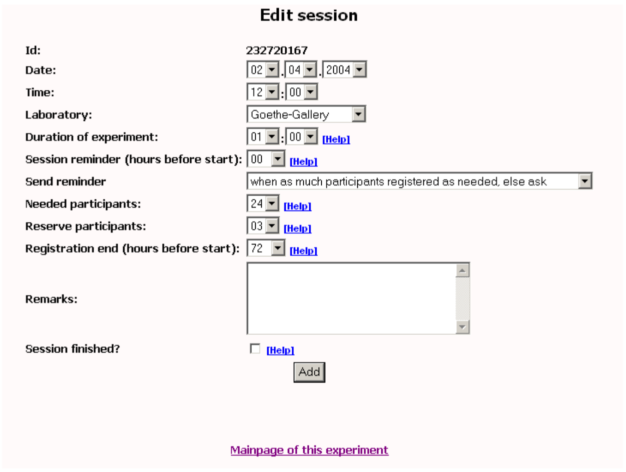
Once participants are assigned, the experimenter sends an *invitation e-mail* which lists the sessions' dates and times and includes a link to the subjects' individualized registration page. Following this link the subject can choose one date out of the sessions available. When registering for a session, a confirmation e-mail is sent to the subject.

When the registration period expires, a *regular job* of the system checks the state of registration for the experiment. For each session, the experimenter gets an e-mail informing her about the number of subjects registered, and having attached a pdf-file containing the list of the names of participants. In case of too few registrations the experimenter may now extend the registration period, or cancel the session at the very end. At the time specified at the sessions's edit page the session reminder e-mail will be sent out.

During the whole registration process, the experimenter can observe the current state of each session at the experiment main page. There are four states of a specific session:

1. *Not complete*: There are not enough participants.
2. *About complete*: There are as much participants as explicitly needed for the experiment, but not enough reserve participants.
3. *Complete*: The number of required participants plus the reserve is reached.
4. *Finished*: All data was filled in for the session. The participation data will be used for the reputation system.

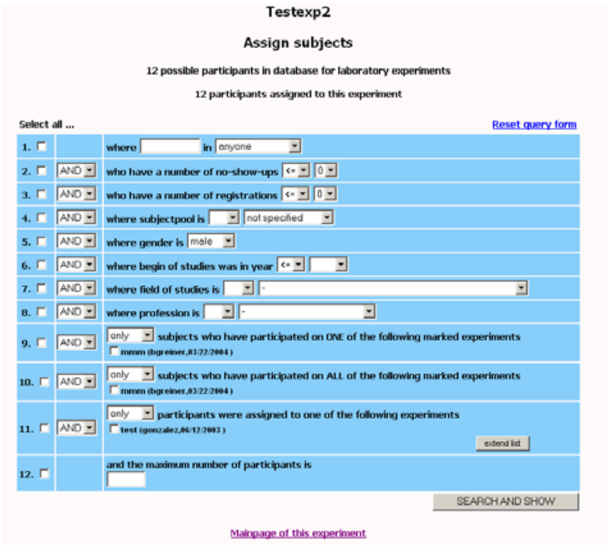
We distinguish between five different independent states (flags) of a participant with regards to a certain experiment:



Edit session

Id:	232720167
Date:	02 . 04 . 2004
Time:	12 : 00
Laboratory:	Goethe-Gallery
Duration of experiment:	01 : 00 [Help]
Session reminder (hours before start):	00 [Help]
Send reminder	when as much participants registered as needed, else ask
Needed participants:	24 [Help]
Reserve participants:	03 [Help]
Registration end (hours before start):	72 [Help]
Remarks:	
Session finished?	☐ [Help]

Fig. 9: Session Creation Page.



Testexp2

Assign subjects

12 possible participants in database for laboratory experiments

12 participants assigned to this experiment

Select all ...

Reset query form

1. ☐		where [ ] in [anyone]
2. ☐	AND	who have a number of no-show-ups [<=] [0]
3. ☐	AND	who have a number of registrations [<=] [0]
4. ☐	AND	where subjectpool is [ ] [not specified]
5. ☐	AND	where gender is [male]
6. ☐	AND	where begin of studies was in year [<=] [ ]
7. ☐	AND	where field of studies is [ ] [-]
8. ☐	AND	where profession is [ ] [-]
9. ☐	AND	[only] subjects who have participated on ONE of the following marked experiments ☐ mmm (bgreiner,03/22/2004)
10. ☐	AND	[only] subjects who have participated on ALL of the following marked experiments ☐ mmm (bgreiner,03/22/2004)
11. ☐	AND	[only] participants were assigned to one of the following experiments ☐ test (gonzalez,06/12/2003) extend list
12. ☐		and the maximum number of participants is [ ]

SEARCH AND SHOW

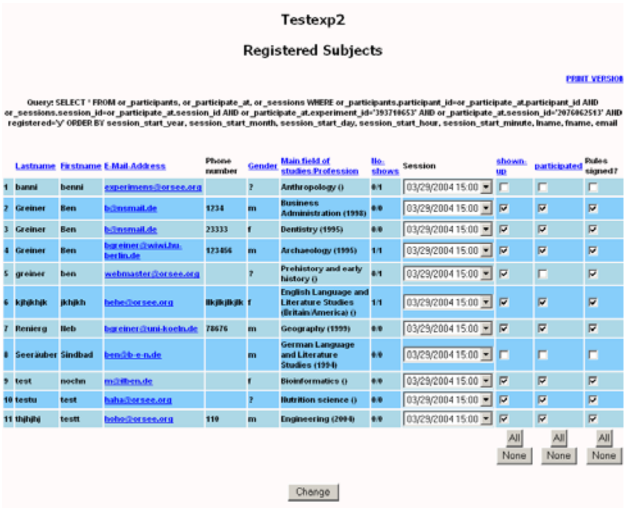
Mainpage of this experiment

Fig. 10: Participant Query.

1. *Assigned*: The participant is allowed to register for a session of this experiment.
2. *Invited*: The participant received an invitation e-mail from the system.
3. *Registered*: The participant is registered for a certain session of a laboratory experiment.
4. *Showed-up*: The participant was at the right location at the right time.
5. *Participated*: The participant really played the game.

The experimenter may also visit the actual *participant list* for each session (Figure 11).

When everything is o.k., the experimenter conducts her experimental session in the laboratory. She fills in the show-up and participation data for all participants. When all data is filled in, she marks the session as finished, and its data will be included in the calculation of reputation score for the participants. When all sessions are done, the experimenter marks the experiment as finished, and it will be listed in the "old experiments" section.


Testexp2

Registered Subjects

PRINT VERSION

Query: SELECT * FROM or_participants, or_participate_at, or_sessions WHERE or_participants.participant_id=or_participate_at.participant_id AND or_sessions.session_id=or_participate_at.session_id AND or_participate_at.experiment_id='393710653' AND or_participate_at.session_id='2076062513' AND registered='y' ORDER BY session_start_year, session_start_month, session_start_day, session_start_hour, session_start_minute, lname, fname, email

	Lastname	Firstname	E-Mail-Address	Phone number	Gender	Main field of studies/Profession	No. shows	Session	shown up	participated	Rules signed?
1	banni	benni	experimens@orsee.org		?	Anthropology ()	0/1	03/29/2004 15:00	☐	☐	☐
2	Greiner	Ben	b@nsmail.de	1234	m	Business Administration (1998)	0/0	03/29/2004 15:00	☑	☑	☑
3	Greiner	Ben	b@nsmail.de	23333	f	Dentistry (1995)	0/0	03/29/2004 15:00	☑	☑	☑
4	Greiner	Ben	bgreiner@wiwi.hu-berlin.de	123456	m	Archaeology (1995)	1/1	03/29/2004 15:00	☑	☑	☑
5	greiner	ben	webmaster@orsee.org		?	Prehistory and early history ()	0/1	03/29/2004 15:00	☑	☐	☑
6	kjhjkhjk	jkhjkh	hehe@orsee.org	lkjlkjlkjlk	f	English Language and Literature Studies (Britain/America) ()	1/1	03/29/2004 15:00	☑	☑	☑
7	Renierg	lleb	bgreiner@uni-koeln.de	78676	m	Geography (1999)	0/0	03/29/2004 15:00	☑	☑	☑
8	Seeräuber	Sindbad	ben@b-e-n.de		m	German Language and Literature Studies (1994)	0/0	03/29/2004 15:00	☐	☐	☐
9	test	nochn	m@illben.de		f	Bioinformatics ()	0/0	03/29/2004 15:00	☑	☑	☑
10	testu	test	haha@orsee.org		?	Nutrition science ()	0/0	03/29/2004 15:00	☑	☑	☑
11	thjhjhj	testt	hoho@orsee.org	110	m	Engineering (2004)	0/0	03/29/2004 15:00	☑	☑	☑

Fig. 11: Participant List for a Session.

6 Conducting an Online Survey

A special experiment type implemented in ORSEE is an 'Online Survey'.[7] After creation of the experiment, the experimenter fills in a *special properties page* stating the start and stop time of the survey, the browser window size, and a short description of the experiment (mentioning required technology for participation and so on). The experimenter can restrict the participation to invited subjects from the known subject pool or can allow for free registration, specifying whether unknown participants have to fill in a personal data form.

An online survey may consist of an introductory page, an instruction page, the personal data form, a number of questions, and an ending page. Each part is freely configurable in ORSEE. Questions are of a certain type, have certain predefined answering options and are organized in items. ORSEE supports 'text lines' and 'text fields', 'select fields' for numbers and text, 'checkboxes' and 'radio buttons'. The latter two can be presented as matrices. Questions and items can be given a numbered or random order.

After having created and configured all pages and questions, the survey is ready to start. Invited subjects can follow a link in their invitation e-mail to participate, and if free participation is enabled, the experiment is listed in the 'Internet Experiments' section of the public area. From the time specified as

[7]For the implementation of this module, we used methods described in Greiner, Jacobsen and Schmidt (2003).

the start time of the survey subjects are allowed to fill in the questionnaire. While the survey is running, the experimenter can observe the participation rate and some simple average statistics of answers.

When the survey time runs out, no subject can start the survey anymore. The experimenter may extend the time or end the survey by marking it as finished. Participant and decision data may be downloaded separately as excel spreadsheets.

7 License

The Online Recruitment System for Economic Experiments is available under a special open source license called 'Citeware'. Specifically, the source code may be copied, modified and distributed under terms complying with the Open Source Definition of the Free Software Foundation. However, the use of derivative products is restricted in a way that any academic report, publication, or other academic disclosure of results obtained with the use of this software will acknowledge the software's use by an appropriate citation of this paper.

Acknowledgements

I would like to thank Philipp Euskirchen, Jens Großer, Imke Jungjohann, Nikoletta Kiss, Carsten Schmidt and participants at the ESA European Meeting 2003 in Erfurt for valuable comments and suggestions on this paper, and the Strategic Interaction Group at the Max Planck Institute in Jena, Germany for testing first version of the system.

References

Davis, D. D. and Holt, C. A.: 1993, *Experimental Economics*, Princeton, NJ: Princeton University Press.

Friedman, D. and Sunder, S. (eds): 1994, *Experimental methods : a primer for economists*, Camebridge: Camebridge University Press.

Greiner, B., Jacobsen, H.-A. and Schmidt, C.: 2003, The Virtual Laboratory Infrastructure for Controlled Online Economic Experiments, *in* K. Kremer and V. Macho (eds), *Forschung und wissenschaftliches Rechnen 2002: Beiträge zum Heinz-Billing-Preis 2002*, GWDG Bericht 62, Göttingen : Ges. für Wiss. Datenverarbeitung, pp. 59–73.

Harrison, G.: 1989, Theory and misbehaviour of first-price auctions, *American Economic Review* **79**, 749–762.

Kagel, J. H. and Roth, A. E. (eds): 1995, *The Handbook of Experimental Economics*, Princeton, NJ: Princeton University Press.

ODIN – Object-Oriented Development Interface for NMR

Thies H. Jochimsen
Michael von Mengershausen
Max-Planck-Institute of Cognitive Neuroscience, Leipzig

Abstract

A cross-platform development environment for nuclear magnetic resonance (NMR) experiments is presented. It allows rapid prototyping of new pulse sequences and provides a common programming interface for different system types. With this object-oriented interface implemented in C++, the programmer is capable of writing applications to control an experiment that can be executed on different measurement devices, even from different manufacturers, without the need to modify the source code. Due to the clear design of the software, new pulse sequences can be created, tested and executed within a short time. To post-process the acquired data, an interface to well-known numerical libraries is part of the framework. This allows a transparent integration of the data processing instructions into the measurement module. The software focuses mainly on NMR imaging, but can also be used with limitations for spectroscopic experiments. To demonstrate the capabilities of the framework, results of the same experiment, carried out on two NMR imaging systems from different manufacturers are shown and compared with the results of a simulation.

1 Introduction

Nuclear magnetic resonance (NMR) is a versatile tool to investigate physical properties of materials and living tissue. The flexibility of the NMR technique can be attributed to the fact that a wide range of experiments is designed by solely altering the software that controls the hardware during the measurement. With a given set of hardware components, various parameters of the sample can be examined with different software-based experimental setups (i.e. pulse sequences). An important task of the NMR scientist who develops new NMR applications is therefore that of a software engineer. Provided a sophisticated programming interface for sequence design is available, advances in the field of computer science can accelerate the process of creating NMR applications.

Contemporary concepts like object-oriented design, polymorphism, and generic programming are used nowadays in software engineering to create modular, extensible, and easy-to-use software instead of procedural programming (an excellent overview of these programming paradigms and their implementation in C++ can be found in [1]). By contrast, NMR pulse sequences are usually programmed using the procedural approach. That is, the scientist provides a program that contains a list of sequential instructions to trigger hardware-events together with some calculations to achieve the required properties of the sequence (e.g. resolution, TE, TR). This results in a non-modular, monolithic implementation of the sequence which seriously limits the reuse of certain parts in another sequence, except for duplicating the source code. A modern approach would describe the sequence as a composition of reusable, self-consistent objects that can be combined freely to develop new experimental setups.

Recently, a software architecture has been presented [2] which makes use of this approach by a double-layered design whereby the user interacts with an application framework written in Java [3] which is mapped to corresponding C++ functionality on the hardware controller and signal processing computer. The programming interface is provided not only for sequence programming but also for developing work flows which incorporate different measurement techniques for clinical application. However, this framework is limited to the devices of one manufacturer and its double-layered design may impose a considerable overhead when adding new functionality, for example custom real-time feedback.

In contrast, ODIN, which is subsequently introduced, concentrates on platform-independent sequence design and data processing with a single open-source code basis in C++. The hardware-dependent components that drive the different scanners are encapsulated into low-level objects (pulses, gradients, data-acquisition) from which complex, platform-independent parts of the sequence are constructed. The same source code is used at all stages of sequence development, from simulation on a stand-alone platform to play-out on a real-time system. ODIN uses the native functionality of the graphical user interface on each platform, allowing a seamless integration of ODIN sequences.

In this paper, the first section gives an introduction into the ODIN sequence programming interface and its underlying concepts. The design of radio frequency (RF) pulses will be described in more detail as this is one of the major strengths of ODIN. The next two sections contain additional information about the internal representation of the sequence within the ODIN library and the mechanisms that are used to execute the experiment in different hardware environments. After that, strategies to visualize and simulate the sequence are presented, and the data processing framework of ODIN is discussed. Finally, experimental results obtained with ODIN on different platforms are shown and compared with the results of a simulation.

2 Platform-Independent Sequence Design

An NMR experiment is basically a sequence of periods where the sample is exposed to different magnetic field configurations, such as RF pulses and magnetic field gradients, or periods where data is acquired. From these basic sequence elements, complex experiments can be composed which measure spectroscopic properties, relaxation, and transportation processes of the spins within the sample. Magnetic field gradients extend these experiments to spatially resolved data sets, i.e. images of these parameters. In addition, repetitive measurements yield time series of physiological processes within living tissue, for example neuronal activity in the human brain.

The NMR sequence can be described in terms of the physical properties of their elements and the arrangement of these sequence elements as a function of time. A simple NMR sequence is shown in Fig.1. This level of description is independent of the measurement device. ODIN provides a programming interface in terms of a C++ class hierarchy which reflects the physical aspects of a sequence. A sequence program which is written using this framework can be executed on different NMR hardware. The system-specific actions are performed by a library that transfers the sequence-specific requests to the actual measurement hardware as depicted in Fig. 2. The benefit of separating the physical logic of the experiment from the peculiarities of the current hardware is the portability of the sequence program. It can be reused with other hardware, even from another manufacturer.

2.1 Sequence Programming Interface

In the following, the term *basic sequence objects* refers to elements of the sequence that cannot be divided into smaller elements from the physical point of view. Examples of such "sequence atoms" are periods of RF irradiation, the application of temporary field gradients or intervals of data acquisition. Each

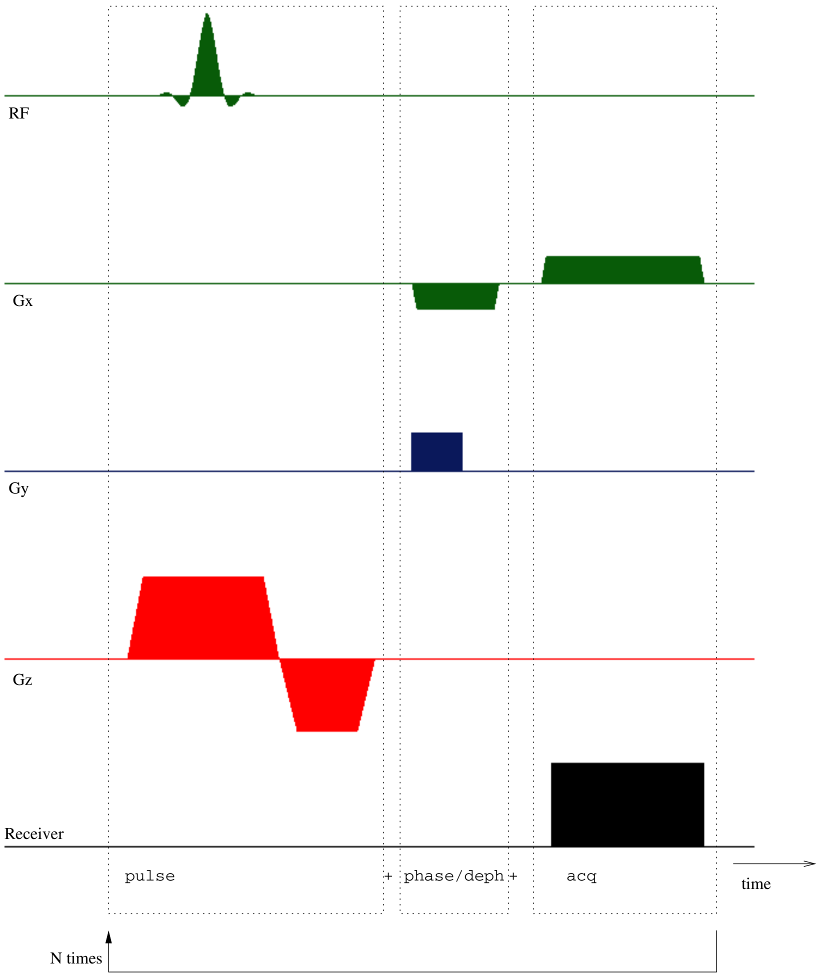


Fig. 1: A simple gradient-echo sequence. The inner part contains a slice-selective RF pulse, gradients Gx, Gy, Gz for spatial encoding, and a period during which the signal is received. This part is repeated N times for linear stepping of the gradient strength of Gy. The sequence objects of these elements are indicated below. The operators + and / between these objects combines them to form the sequence.

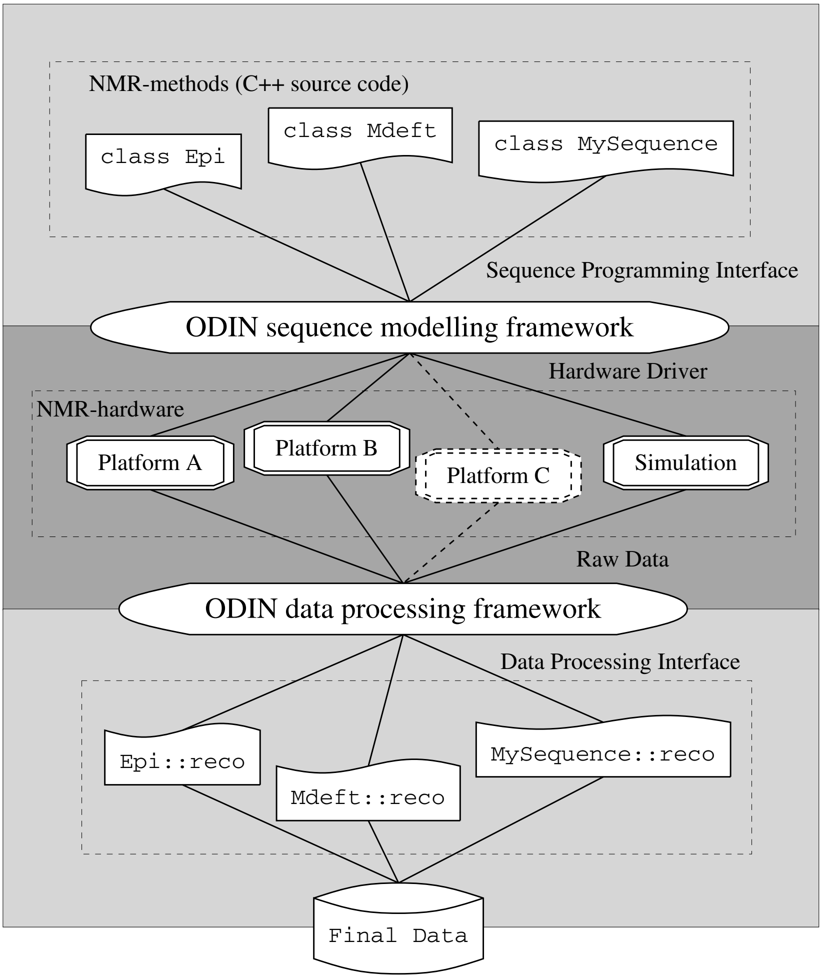


Fig. 2: Flowchart of an NMR experiment performed with the ODIN framework. The sequence programmer implements a C++ class that represents the experimental method and uses the platform-independent sequence programming interface. An object of this class is then used by the ODIN library to execute the sequence on the different platforms by means of hardware-specific instructions within the library. The acquired raw data is then post-processed by a member function `reco` of the same class that was used for the measurement. Finally, the processed data (images, spectra) are written to disk.

basic sequence object is instantiated from a C++ class which handles its physical properties, for example the duration. These objects are constructed during the initialization of the sequence according to the instructions given by the sequence programmer. From this collection, the sequence is constructed by grouping the sequence objects into container objects. To simplify the notion of composing new container objects, the operators `+` and `/` are overloaded and can be used to specify whether two sequence objects `a` and `b` should be played out sequentially `(a+b)` or in parallel `(a/b)`. As an example, the source code for the simple sequence visualized in Fig. 1 is printed in Fig. 3.

```
class SimpleSequence : public SeqMethod {

 private:
  // Sequence objects:
  SeqPulsarSinc pulse;      SeqGradPhaseEnc phase;
  SeqAcqRead    acq;        SeqAcqDeph      deph;
  SeqObjLoop    loop;       SeqDelay        delay;
  SeqObjList    oneline;

 public:
  SimpleSequence(const tjstring& label) : SeqMethod(label) {
    set_description("Simple Gradient Echo Sequence");
  }

  void method_pars_init() {
    // This is the place where sequence-specific parameters can be
    // initialized
  }

  void method_seq_init() {
    // This function builds the sequence, it is called every time
    // a parameter has been changed by the user

    // The global objects commonPars, geometryInfo and systemInfo
    // hold parameters that are common to most sequences,
    // the information about the selected geometry and system
    // specific properties, respectively. These parameters
    // can be accessed via the appropriate 'get' and 'set' functions.

    // Excitation pulse:
    pulse=SeqPulsarSinc("pulse",geometryInfo->get_sliceThickness());

    // Geometry:
    // calculate the resolution in the read direction and set the number of
    // phase encoding steps so that a uniform resolution will be obtained
    float resolution = geometryInfo->get_FOV(readChannel)
                / commonPars->get_MatrixSize(readChannel);
    commonPars->set_MatrixSize(phaseChannel,
       geometryInfo->get_FOV(phaseDirection) / resolution);

    // Phase encoding:
    phase = SeqGradPhaseEnc("phase",
       commonPars->get_MatrixSize(phaseChannel),
       geometryInfo->get_FOV(phaseChannel),
       phaseChannel,0.25*systemInfo->get_max_grad());

    // Frequency encoding:
    acq = SeqAcqRead("acq",commonPars->get_AcqSweepWidth(),
       commonPars->get_MatrixSize(readChannel),
       geometryInfo->get_FOV(readChannel),readChannel);

    // Dephasing for frequency encoding
    deph = SeqAcqDeph("deph",acq,FID);
    // One gradient echo to sample one line in k-space
    oneline = pulse + phase/deph + acq;

    // Sequence layout:
    set_sequence( loop ( oneline + delay ) [phase]  );
  }

  void method_rels() {
    // This is the place where sequence timing is performed

    // ensure correct repetition time by setting the duration of 'delay'
    double linedur = oneline.get_duration();
    if(linedur>commonPars->get_RepetitionTime())
      commonPars->set_RepetitionTime(linedur);
    delay.set_duration( (commonPars->get_RepetitionTime()-linedur));
  }

  void method_pars_set() {
    // This function is called once before the measurement is started
  }
};
```

Fig. 3: The source code of a simple gradient-echo sequence, implemented as a C++ class to be used within the ODIN framework.

Besides this technique of building sequences from scratch by grouping basic sequence objects together, the ODIN library offers many predefined high-level sequence objects as C++ classes. For example, the object `acq` in Fig. 1 and 3 is an acquisition window with the simultaneous application of a gradient field that is used in many imaging sequences for spatial frequency encoding. These more complex objects are constructed from basic sequence objects within the library, using the same mechanism of building container objects as the sequence programmer would. In addition, the class of these composite objects provides an interface that is adjusted to its high-level concept. For instance, the object `acq` has a member function that returns the point in time of the center of the acquisition window with proper consideration of the ramp of the simultaneous gradient.

2.2 Pulse Design

A crucial part of the sequence is the application of RF pulses in order to generate a detectable signal from a limited spatial or spectral range of spins within the sample. The ODIN framework contains a flexible module for the generation and simulation of RF pulses. A wide range of pulses is supported by a plug-in style mechanism. The desired excitation profile, gradient shape and frequency filter can be selected and modified separately to match it optimally to the specific application. It can be easily extended by supplying the module with new plug-ins which generate k-space trajectories or calculate the RF waveform as a function of time or k-space coordinate. The following pulse types are already supported by existing plug-ins of the ODIN library:

- Slice-selective pulses (Sinc, Gauss), optionally with a VERSE [4] trajectory for reduced power excitation.
- Adiabatic pulses (Sech [5], WURST [6]).
- Spectrally and spatially selective pulses [7] for slice-selection with a predefined spectral profile (e.g. for fat suppression).
- Two dimensional (2D) pulses [8] with various excitation shapes and different spiral trajectories.
- Composite pulses [9] which are created by concatenating one of the above pulses with different transmitter phases and flip angles.

In addition, these pulses can be filtered either in k-space or in the time domain using a filter plug-in. The benefit from separating the pulse shape and the trajectory into different plug-ins can be illustrated by considering the generation of 2D pulses: Each of the excitation profiles (point, box, disk, user-defined list of points) can be used in combination with any of the 2D trajectories in order to generate a pulse profile that is well adjusted to the requirements. For example, an excitation profile that consists of a chain of adjacent points together with a slew-rate optimized trajectory is useful for curved slice imaging [10].

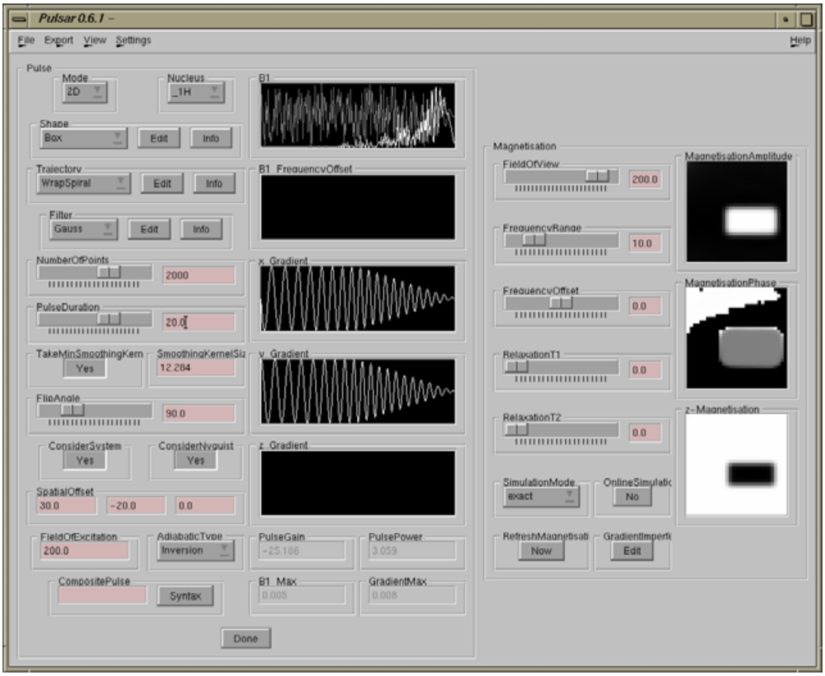
Because the pulse module is a regular sequence object, it can be integrated seamlessly into any NMR sequence. For example, the object `pulse` in Fig. 1 and 3 is a slice-selective specialization of this module using the `Sinc` plug-in for the pulse shape. In addition, a graphical user interface (Fig. 4) which acts as a front-end to the pulse module can be used for interactive pulse design and monitoring of the corresponding excitation profile.

2.3 Loops and Vectors

An essential aspect in most NMR experiments is to repeat certain parts of the sequence unchanged or with different settings. Examples are the repetition of a gradient-echo with different strength of the phase-encoding gradient in conventional Fourier imaging as used in the sequence of Fig.1, or the repetition with different pulse frequencies for multi-slice acquisition.

To use this technique in a uniform manner, ODIN introduces the concept of vector objects and loop objects. Vector objects are elements of the sequence that are used repeatedly with different settings. The following predefined vector classes, derived from a common base class `SeqVector`, are available to the sequence programmer:

- Gradient pulses with different gradient strengths for phase encoding or diffusion weighting.
- Sequence objects that drive the transmitter (RF pulses) or receiver (acquisition windows) contain two vector objects for frequency and phase switching to be used for multi-slice experiments or phase cycling.
- Delay objects with a variable duration, which is changed for each iteration.



Pulse
Mode 2D
Nucleus _1H
Shape Box
Trajectory WrapSpiral
Filter Gauss
NumberOfPoints 2000
PulseDuration 20.0
TakeMinSmoothingKern Yes
SmoothingKernelSiz 12.284
FlipAngle 90.0
ConsiderSystem Yes
ConsiderNyquist Yes
SpatialOffset 30.0 -20.0 0.0
FieldOfExcitation 200.0
AdiabaticType Inversion
CompositePulse
PulseGain -25.106
PulsePower 3.059
B1_Max 0.005
GradientMax 0.008

Magnetisation
FieldOfView 200.0
FrequencyRange 10.0
FrequencyOffset 0.0
RelaxationT1 0.0
RelaxationT2 0.0
SimulationMode exact
OnlineSimulatio No
RefreshMagnetisati Now
GradientImperfe Edit

Fig. 4: The Pulsar user interface for interactive pulse design and simulation. The panel to the left allows editing of the pulse parameters and shows the time courses of the RF and gradient fields. The current settings show a 2D selective pulse, i.e. a pulse that restricts the excited spins in two dimensions. The right-hand side displays the result of a simulation with this pulse.

– A list of user-defined rotation matrices that can be attached to gradient-related objects in order to alter their direction subsequently.
– A container object that holds a list of other sequence objects which are played out sequentially for each repetition.

Although this set of specialized vector classes is probably not exhaustive, the last class may be used to easily extend this list by storing sequence objects for each repetition into the container. This emulates the behavior of a built-in vector class.

To specify which parts of the sequence will be repeated and which vectors will be modified at each repetition, loop objects play a central role in sequence design with ODIN. They possess a function-like syntax (functors) when used within a sequence:

```
loop ( kernel ) [vector1][vector2]...
```

With this line of source code, the loop object `loop` is used to repeat the sequence part `kernel` while incrementing the properties of the vector objects `vector1,vector2,...` that are located within `kernel`. Instead of using a vector object, an integer number `N` can also be given as an argument to the loop, which will then repeat the sequence part `N` times unchanged. By using this common notation for all variable aspects of a sequence, new applications can be implemented rapidly without dealing with the specific aspects of the hardware.

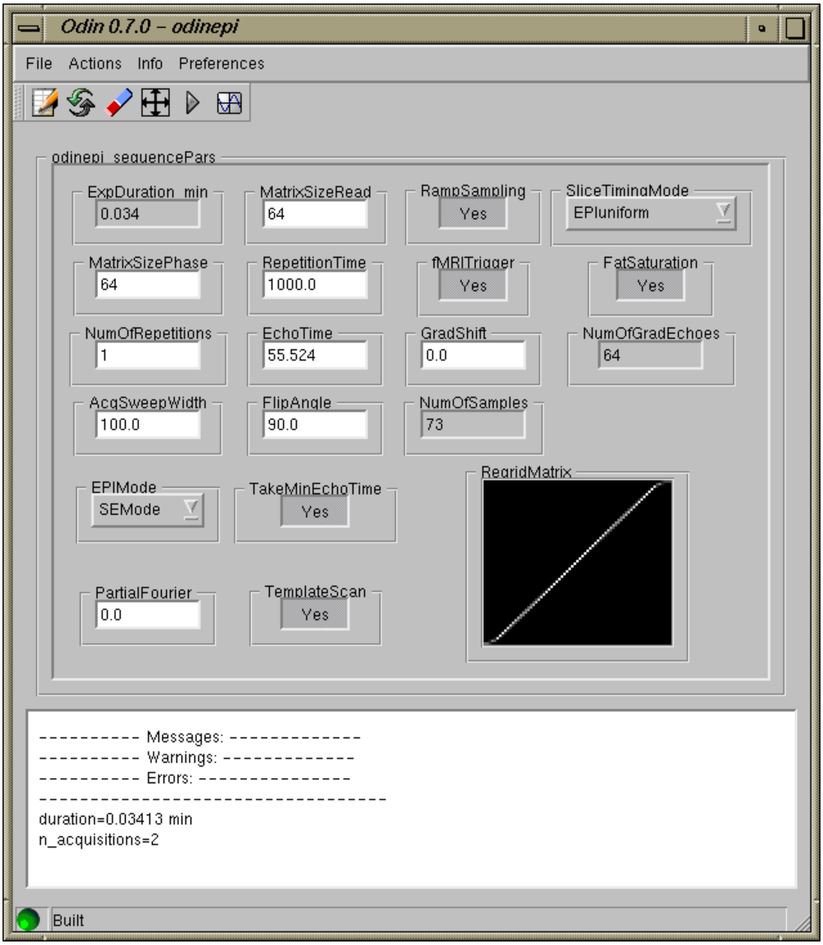
2.4 Sequence Parameters

Normally, each sequence has a set of parameters which specify the actual experiment, for example the sampling rate for data acquisition or the duration of the RF pulse. The sequence parameters are edited interactively within the user interface of the measurement device, and the sequence is recalculated according to the new settings. Within ODIN, these parameters are members of the C++ sequence class, allowing transparent access to their values in the member function that prepares the experiment. Well-known data types (integer numbers, floating point numbers, Boolean values) can be used as sequence parameters. They are designed to be used exactly like built-in types of the C++ language, resulting in understandable source code.

The ODIN library hooks the set of parameters specified by the sequence programmer into the native editing mechanism of the measurement device. After the measurement, the parameters are stored on disk in JCAMP-DX format [11] together with the raw data. In the post-processing step, the parameters and the raw data are then read from disk. If no native mechanism for parameter editing exists (e.g. on a stand-alone platform), ODIN provides its own set of widgets using the Qt library [12] to edit the parameters interactively (Fig. 5).

3 Internal Representation of the Sequence

Any NMR sequence has a nested structure, that is, basic sequence objects can be grouped together to form logical units, which in turn can be collected to build more complex units. This leads to an internal representation of the sequence as an ordered tree of sequence objects. The leaves of this sequence tree are the basic sequence objects (RF pulses, gradients, acquisition windows, evolution delays). The sequence containers are represented by nodes of the tree. They contain a list of references to their members in the same order as given by the sequence programmer. The nodes can contain additional information, e.g. a loop object contains the number of repetitions besides the elements of the sequence that are repeated.

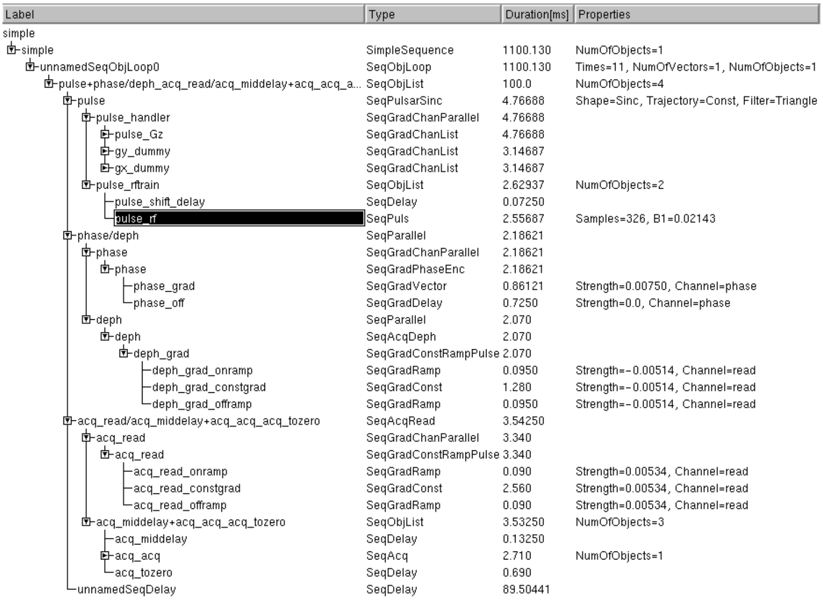


odinepi_sequencePars

ExpDuration_min: 0.034
MatrixSizeRead: 64
RampSampling: Yes
SliceTimingMode: EPIuniform
MatrixSizePhase: 64
RepetitionTime: 1000.0
fMRITrigger: Yes
FatSaturation: Yes
NumOfRepetitions: 1
EchoTime: 55.524
GradShift: 0.0
NumOfGradEchoes: 64
AcqSweepWidth: 100.0
FlipAngle: 90.0
NumOfSamples: 73
EPIMode: SEMode
TakeMinEchoTime: Yes
PartialFourier: 0.0
TemplateScan: Yes

```
---------- Messages: -------------
---------- Warnings: -------------
---------- Errors: ---------------
----------------------------------
duration=0.03413 min
n_acquisitions=2
```

Fig. 5: User interface for rapid sequence design. It combines frequently used functionality to edit, compile, visualize and simulate ODIN sequences. The set of widgets for the parameters is dynamically generated according to the specifications of the sequence module. The parameter set for an EPI sequence is shown here.



```
Label                                                      Type                      Duration[ms]  Properties
simple
 simple                                                    SimpleSequence            1100.130      NumOfObjects=1
   unnamedSeqObjLoop0                                      SeqObjLoop                1100.130      Times=11, NumOfVectors=1, NumOfObjects=1
     pulse+phase/deph_acq_read/acq_middelay+acq_acq_a...   SeqObjList                100.0         NumOfObjects=4
       pulse                                               SeqPulsarSinc             4.76688       Shape=Sinc, Trajectory=Const, Filter=Triangle
         pulse_handler                                     SeqGradChanParallel       4.76688
           pulse_Gz                                        SeqGradChanList           4.76688
           gy_dummy                                        SeqGradChanList           3.14687
           gx_dummy                                        SeqGradChanList           3.14687
         pulse_rftrain                                     SeqObjList                2.62937       NumOfObjects=2
           pulse_shift_delay                               SeqDelay                  0.07250
           pulse_rf                                        SeqPuls                   2.55687       Samples=326, B1=0.02143
       phase/deph                                          SeqParallel               2.18621
         phase                                             SeqGradChanParallel       2.18621
           phase                                           SeqGradPhaseEnc           2.18621
             phase_grad                                    SeqGradVector             0.86121       Strength=0.00750, Channel=phase
             phase_off                                     SeqGradDelay              0.7250        Strength=0.0, Channel=phase
         deph                                              SeqParallel               2.070
           deph                                            SeqAcqDeph                2.070
             deph_grad                                     SeqGradConstRampPulse     2.070
               deph_grad_onramp                            SeqGradRamp               0.0950        Strength=-0.00514, Channel=read
               deph_grad_constgrad                         SeqGradConst              1.280         Strength=-0.00514, Channel=read
               deph_grad_offramp                           SeqGradRamp               0.0950        Strength=-0.00514, Channel=read
       acq_read/acq_middelay+acq_acq_acq_tozero            SeqAcqRead                3.54250
         acq_read                                          SeqGradChanParallel       3.340
           acq_read                                        SeqGradConstRampPulse     3.340
             acq_read_onramp                               SeqGradRamp               0.090         Strength=0.00534, Channel=read
             acq_read_constgrad                            SeqGradConst              2.560         Strength=0.00534, Channel=read
             acq_read_offramp                              SeqGradRamp               0.090         Strength=0.00534, Channel=read
         acq_middelay+acq_acq_acq_tozero                   SeqObjList                3.53250       NumOfObjects=3
           acq_middelay                                    SeqDelay                  0.13250
           acq_acq                                         SeqAcq                    2.710         NumOfObjects=1
           acq_tozero                                      SeqDelay                  0.690
       unnamedSeqDelay                                     SeqDelay                  89.50441
```

Fig. 6: The sequence tree of the example sequence from Fig. 1 visualized within the ODIN framework. The first column depicts the structure of the tree whereby the basic sequence objects can be found at the end of each branch and the container objects at the nodes, indicated by small boxes to the left. The second and third column show the C++ type and the duration of each object. Properties that are specific to each object are shown in the last column, e.g. the selected RF object pulse_rf *has a waveform of 326 samples with the given amplitude B1.*

The tree is constructed during the preparation phase of the experiment according to the instructions of the sequence programmer. Each sequence has its special tree. As an example, Fig. 6 depicts the sequence tree structure for the sequence of Fig. 3. The created sequence tree is the central data structure that is used in further steps of the experiment. If a certain operation has to be performed for the sequence, e.g. calculating the total duration of the experiment, the sequence tree is traversed recursively, querying each object for a value (in this case its duration), or requesting a certain operation from the object. Thereby the starting point is the root of the sequence tree. At each node that contains an ordered list of other sequence objects, these sub-objects are in turn requested to perform the operation. This recursion in each branch terminates at the leaves, if a basic sequence object is reached. The two following sections describe how this technique of traversing the sequence tree is used to control the measurement device or to visualize and simulate the sequence.

The whole sequence (i.e. the root of the tree) is in itself a container object, instantiated from a C++ class, which is implemented by the sequence programmer. This class is derived from a base class that acts as an interface between the sequence and the ODIN library. By the mechanism of virtual functions in C++, a set of sequence-specific member functions must be provided by the sequence class that will be called during initialization, preparation, and data processing of the experiment. With this technique, all sequence modules share a common interface which can be used by the library in a uniform manner.

4 Hardware-Specific Implementation

In this section, two examples show how the ODIN sequence tree can be utilized to drive the hardware of two scanners from different manufacturers:

Platform A (Bruker Medspec, 3 Tesla) is driven by a pulse program which is an ASCII file that contains a list of sequential instructions for the hardware and controlling structures (loops, jumps) to repeat certain parts of the sequence. To perform an experiment, a set of parameters must be provided that contains the detailed settings for the measurement. The pulse program and the parameter set cover all characteristics of the experiment on this platform. ODIN maps its internal representation of the sequence to the device by traversing the sequence tree and generating an entry in the pulse program for each sequence object. In addition, each sequence object is asked to make an entry into the parameter set. After transferring the generated files to the system software, the sequence can be executed.

On platform B (Siemens Trio, 3 Tesla), the system components are driven directly by a C++ program in real time. The corresponding source code must be provided by the sequence programmer. It contains instructions to trigger hardware events (RF pulses, gradients) at specified points in time. The experiment is performed during run-time of this program. On this platform, ODIN executes a sequence by traversing the sequence tree at run-time, querying each sequence object for a corresponding event. An internal counter takes care of the correct starting time of each event.

The above procedures presume that at least the basic sequence objects contain code to map themselves to the current hardware. The hardware drivers for the different platforms are therefore located inside these objects. The container objects and the high-level sequence objects do not have to be aware of how to drive the current hardware. Because the number of basic sequence objects is limited, the hardware-specific code is located only at a few places within the library, allowing straightforward portability to new system types.

5 Sequence Visualization and Simulation

Even on computers where no NMR device is attached, the ODIN framework can be useful for developing sequences. On a stand-alone platform, the time courses of the different channels (RF, gradients and receiver) can be displayed, or a simulation of the sequence acting on a virtual sample can be performed. This is achieved by giving all basic sequence objects the capability to generate a digitized version of themselves, i.e. a function that returns the values of each channel for equally spaced points in time.

To generate a digitized version of the whole sequence for visualization, the container objects can combine them recursively, traversing the sequence tree until the whole sequence is processed. The result can then be displayed graphically. This is currently realized by generating a multi-channel audio file which is then displayed using conventional sound editors. In addition, predefined functions exist which calculate important aspects of the sequence numerically using the digitized sequence, for example gradient moments, the strength of diffusion weighting or the k-space encoding of different coherence pathways in a multi-pulse sequence.

For the simulation, a virtual sample that holds spatially resolved NMR-specific properties (spin density, relaxation rates T_1 and T_2, frequency offset) is required. It can be created by means of a graphic editor or a special ODIN sequence that measures these properties with a high resolution. The digitized version of each sequence object is then used to simulate its effect on the sample. By traversing the sequence tree, the simulation is performed in the same order as the sequence objects would be played out on a real NMR device. An exact solution of the Bloch equations for piecewise constant fields [13] is utilized for the calculation: It transforms the magnetization vector at each point of the sample recursively according to the set of values within the digitized arrays of the sequence object. During acquisition periods, a virtual NMR signal is generated by integrating over the transverse component of the magnetization vector for all points within the virtual sample. The result of the simulation is then a synthetic NMR signal that can be post-processed with the same algorithm as the real signal would be processed.

This simulation strategy is most useful for analyzing imaging sequences. Because it is limited to ensembles of isochromatic spins with single-quantum coherences and interactions simplified by T_1 and T_2, other tools [14–16] are more appropriate to generate virtual spectra of samples with different nuclei, to simulate higher-order quantum coherences or explicit interactions. Another limitation is given by the finite spatial size of the volume elements: The simulation does not account for static intra-voxel dephasing due to field inhomogeneities (T_2^*).

6 Data Processing

In a typical NMR experiment, the RF signal that is induced by the magnetization of the sample and received by the coil is post-processed to obtain interpretable data. This can be a frequency analysis for spectroscopic applications or the reconstruction of spatially resolved parameter maps for imaging. In general, the data processing algorithm is specific to the NMR sequence which was used to acquire the raw data. This step is supported by a software layer that integrates external numerical libraries consistently into ODIN.

After the measurement, the raw data is processed by a function of the same sequence module that was used for the experiment. Because this function is implemented as a C++ member function, all parameters of the measurement are directly accessible. The external numerical libraries can be used within this function. After the processing step, the final data is written back to disk.

6.1 Integration of External Libraries

As a basis for further integration of external libraries into ODIN, the expression-template based multidimensional array type provided by the Blitz++-library [17] is used to hold the NMR data during the different processing steps. Many useful functions that operate on multidimensional arrays are already made available by Blitz++. However, more complex numerical operations must be added separately as they are not part of Blitz++. Therefore, an interface to the following libraries has been implemented so that they always operate on the array type of Blitz++ and add the described functionality to it:

– NewMat [18]: Supports various matrix types and matrix calculations.
– GSL (GNU Scientific Library) [19]: Non-linear least-square fitting, interpolation.
– FFTW (Fastest Fourier Transform in the West) [20]: Fourier transform for multidimensional arrays.

For example, an FFT of arrays with arbitrary dimensionality can be programmed in one line of C++-code with this integration of external libraries:

```
blitz_fftw(data(all,0,all));
```

This instruction will perform a complex in-place FFT over the first and third dimension of the array `data` for all values with index `0` in the second dimension.

6.2 Processing of Large Data Files

When dealing with large datasets, e.g. for functional imaging, the problem arises that the whole record cannot be held in memory for analysis at once.

Splitting the file into blocks and processing them separately is therefore necessary. ODIN supports this technique by read and write functions that transparently iterate through the whole dataset. The programmer only needs to specify the operation for one block. When looping over this operation, ODIN will then read and write the appropriate block.

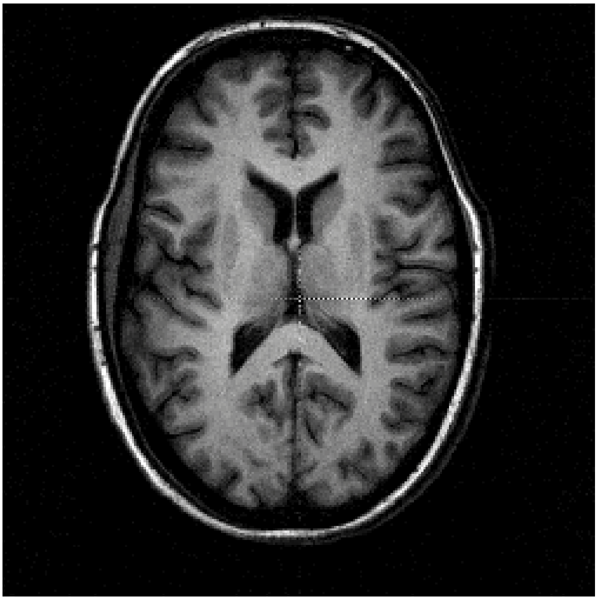
7 Experiments

Two sequences were executed with the same subject and the same settings on platform A and B. Figure 7 shows the reconstructed images from a power-reduced variant of the modified driven equilibrium Fourier transform (MDEFT) sequence [21]. Although the position of the brain within the slice differs due to different positioning of the subject within the magnet, both images show the same spatial pattern and comparable contrast with a signal-to-noise ratio of 30.5 (platform A) and 25.1 (platform B) in white matter.

In Fig. 8 the spin-echo EPI [22] experiments are compared with the result of a simulation which was performed using high-resolution maps of the NMR parameters (spin density, T_1, T_2 and frequency offset). These maps were acquired on platform A during the same session. The simulation was then carried out off-line on a Linux PC to generate a synthetic signal using the same sequence code that was used for the measurements. The images are similar in terms of contrast and image quality, but show slightly different field-of-views in phase encoding direction which is very sensitive to frequency offsets due to the small bandwidth. The mismatch may therefore be caused by non-optimal compensation of the field inhomogeneities (shimming) or eddy-currents modifying the phase encoding blips. This otherwise undesired discrepancy could be used here to study the effects of field variations and gradient imperfections. However, the general similarities between the result of the simulation and the actual experiments indicate that the simulation can be used to reproduce the measurement and that it is feasible to develop and test sequences on a stand-alone platform.

8 Availability and Licensing

The software package is published under the terms of the GNU General Public License. It can be obtained as source code and binary packages for different platforms (Linux, IRIX, Windows, VxWorks) from the web [23]. The online manual for the class hierarchy can also be found at this location.



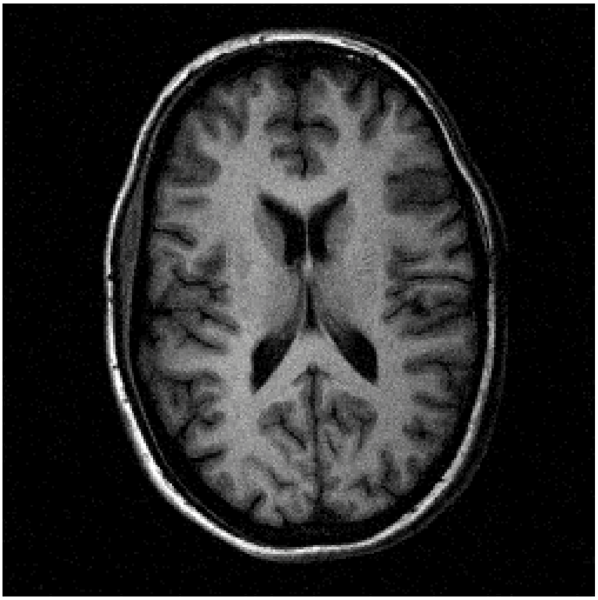


Fig. 7: MDEFT images from platform A (top) and B (bottom) with a matrix size of 252 $\times$ 252 pixels, $FOV = 220$ mm and a sweep-width of 25 kHz. This sequence type is highly sensitive to the T_1 relaxation time. Therefore it is well-suited to display anatomical structures.

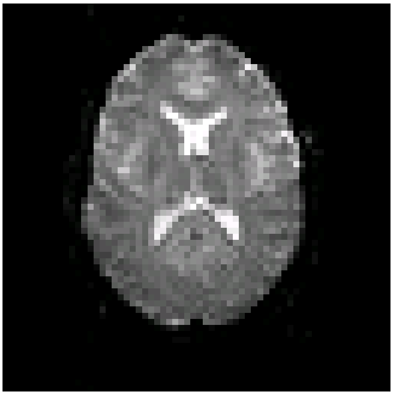
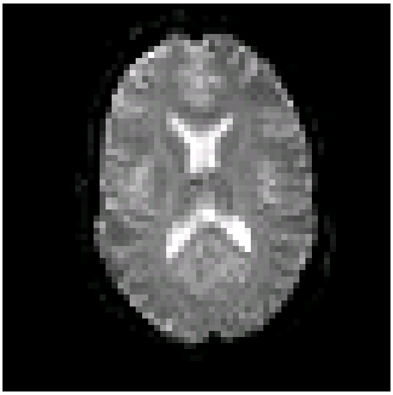
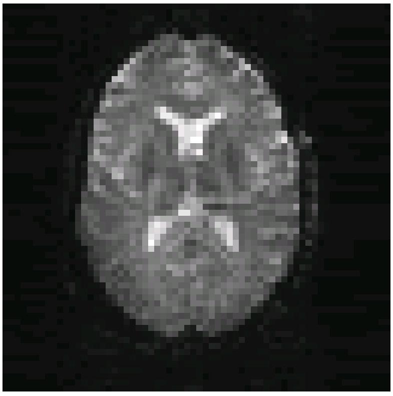




Fig. 8: Spin-echo EPI images from platform A (top), platform B (middle) and the simulation (bottom) with a matrix size of 64 × 64, 100 accumulations, 100 kHz sweep-width and the same slices as in Fig. 7. The phase encoding direction is aligned vertically.

9 Conclusion

It has been demonstrated that ODIN is a valuable tool when developing and testing NMR sequences. The sequence programming interface provides a concise C++ class hierarchy to set up an NMR experiment within a short time. Without changing the source code, the sequence can be visualized, simulated and executed on different NMR hardware. This is particularly useful in laboratories where more than one scanner exists, or to exchange sequences between research facilities with different hardware infrastructure. With the ODIN data processing framework, a consistent interface to reliable open-source libraries for calculating the final data is provided. The internal representation of the experiment by the sequence tree is adequately matched to the application domain and allows easy extensibility when porting the framework to new platforms.

Acknowledgements

The authors thank Robert Trampel, Markus Körber and Andreas Schäfer for improving the software and Harald E. Möller for helping with the manuscript.

References

[1] B. Stroustrup, The C++ Programming Language, Addison-Wesley, Boston, 2000.
[2] J. Debbins, K. Gould, V. Halleppanavar, J. Polzin, M. Radick, G. Sat, D. Thomas, S. Trevino, R. Haworth, Novel Software Architecture for Rapid Development of Magnetic Resonance Applications, Conc. Magn. Reson. (Magn. Reson. Engin.) 15 (3) (2002) 216–237.
[3] http://java.sun.com/java2/whatis.
[4] S. Conolly, D. Nishimura, A. Macovski, G. Glover, Variable-Rate Selective Excitation, J. Magn. Reson. 78 (1988) 440–458.
[5] M. S. Silver, R. I. Joseph, D. I. Hoult, Highly Selective $\pi/2$ and π Pulse Generation, J. Magn. Reson. 59 (1984) 347–351.
[6] E. Kupče, R. Freeman, Adiabatic Pulses for Wideband Inversion and Broadband Decoupling, J. Magn. Reson. A 115 (1995) 273–276.
[7] C. H. Meyer, J. M. Pauly, A. Macovski, D. G. Nishimura, Simultaneous Spatial and Spectral Selective Excitation, Magn. Reson. Med. 15 (1990) 287–304.
[8] J. Pauly, D. Nishimura, A. Macovski, A k-space Analysis of Small-Tip-Angle-Excitation, J. Magn. Reson. 81 (1989) 43–56.
[9] M. Levitt, Symmetrical Composite Pulse Sequence for NMR Population Inversion. I. Compensation of Radiofrequency Field Inhomogeneity , J. Magn. Reson. 48 (1982) 234–264.
[10] T. H. Jochimsen, D. G. Norris, Single-shot curved slice imaging, MAGMA 14 (2002) 50–55.
[11] A. N. Davies, P. Lampen, JCAMP-DX for NMR, Appl. Spectr. 47 (8) (1993) 1093–1099.
[12] http://www.trolltech.com.
[13] H. C. Torrey, Transient Nutation in Nuclear Magnetic Resonance, Phys. Rev. 76 (8) (1949) 1059–1068.
[14] S. A. Smith, T. O. Levante, B. H. Meier, R. R. Ernst, Computer simulations in magnetic resonance. An object-oriented programming approach, J. Magn. Reson. A 106 (1994) 75.

[15] P. Nicholas, D. Fushman, V. Ruchinsky, D. Cowburn, The Virtual NMR Spectrometer: A Computer Program for Efficient Simulation of NMR Experiments Involving Pulsed Field Gradients, J. Magn. Reson. 145 (2000) 262–275.

[16] W. B. Blanton, BlochLib: a fast NMR C++ tool kit, J. Magn. Reson. 162 (2003) 269–283.

[17] T. L. Veldhuizen, Arrays in Blitz++, in: Proceedings of the 2nd International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'98), Lecture Notes in Computer Science, Springer-Verlag, 1998.

[18] R.Davies, Writing a matrix package in C++, in: The second annual object-oriented numerics conference, 1994, pp. 207–213.

[19] http://www.gnu.org/software/gsl.

[20] M. Frigo, S. G. Johnson, FFTW: An Adaptive Software Architecture for the FFT, Vol. 3 of Lecture Notes in Computer Science, 1998, pp. 1381–1384.

[21] D. G. Norris, Reduced power multislice MDEFT imaging, J. Magn. Reson. Imag. 11 (4) (2000) 445–51.

[22] P. Mansfield, Multi-planar image formation using NMR spin echoes, Solid State Phys. 10 (1977) L55–L58.

[23] http://od1n.sourceforge.net.

VINCI – “Volume Imaging in Neurological Research, Co-Registration and ROIs included”

Stefan Vollmar, Jiří Čížek, Michael Sué, Johannes Klein, Andreas Jacobs, Karl Herholz
Max-Planck-Institut für neurologische Forschung, Köln

Abstract

VINCI (“Volume Imaging in Neurological Research, Co-Registration and ROIs included”) was designed for the visualization and analysis of volume data generated by medical tomographical systems with special emphasis on the needs for brain imaging with Positron Emission Tomography (PET). VINCI is highly modular, extensible, compact and runs well on current PCs, no installation is required. We achieve this with a plugin architecture; VINCI can be remotely controlled through several high-level language interfaces, at the basis of which is our own XML-based scripting language. VINCI is entirely true color based and allows online fusion and contour rendering of several images, more than 50 studies can be displayed simultaneously in orthogonal views on current machines. We also have a fully automatic registration tool which is suitable for routine usage and gives online feedback of a running registration.

1. Introduction

Modern brain imaging is multidimensional in many respects. Data sets are volume data sets rather than planar images, and they can be very voluminous. Thus, there are three basic spatial dimensions, and imaging software needs to provide convenient means to represent that in an intuitive and representative way as multiple planes (typically oriented in parallel or orthogonally). Additional dimensions are along the time axis (from dynamic data acquisition or multiple follow-up studies in the same subject), or representing multiple modalities (different PET or SPECT tracers, different MRI acquisition protocols), or data stemming from a sample of different subjects' brains (represented as individuals or as statistical parametric images derived from the sample).

These various additional dimensions imply special demands for efficient processing and display. Accurate spatial alignment of data sets that were recorded from the same brain, usually called co-registration, is a prerequisite and the software needs to provide tools to achieve that in a fast, robust and reproducible way. Some data sets will represent primarily structural information (where borders between anatomical structures or delineating lesions carry the essential information), whereas others will represent primarily quantitative information (functional images, where voxel values carry the essential information). Typically, these two types of images need to be displayed together as closely as possible to assign the quantitative data to anatomical structures or lesions. Biological analysis of quantitative data requires statistics based on the investigation of multiple subjects. Quantitative values can either be obtained by extraction from anatomically defined volumes of interest (VOIs, based on co-registered structural and quantitative images), or on a voxel-by-voxel basis as statistical parametric images after some spatial normalization of different brain shapes to a common template. Thus, the software also needs to provide means for VOI placement and evaluation as well as for interindividual registration and mathematical transformation of images.

A more recent field of research with rapidly growing impact is *Molecular Imaging*: using multi-tracer animal studies on high-resolution PET systems, it allows a non-invasive characterization of endogenous molecular markers. Co-registration of these PET images with high-resolution MRI data sets is even more demanding (more image artefacts, small movements of animals): it requires elaborate support for manual intervention, also the ability to compare, view and analyze several different studies simultaneously.

Another important usage for a general imaging package is display and analysis of raw and calibration data from PET scanners. This is essential to locate potential hardware failures and find problems that have lead to artefacts in the reconstructed image data.

2. Previous and Related Work

The MPI-Tool ("Multi Purpose Imaging Tool") is a visualization package previously developed at our institute ([2], [3]). It has pioneered several aspects of manual co-registration for multi-modality imaging and was primarily developed for older versions of Solaris. Since 1998 it is available commercially, also for the Linux platform, [4].

With the introduction of our new HRRT (High Resolution Research Tomograph) system in 1999, it was felt that an entirely new design for a general visualization and co-registration package was needed, addressing new challenges and shortcomings of previous work, e.g. a more flexible architecture with plugins, fully automatic co-registration, export of vector graphics and true color support, improved handling of increased image data sizes, scripting and remote control. VINCI was first presented as a visualization tool for HRRT data [6], as its true color engine was originally conceived for online display (separate thread on multi-processor systems) of a running reconstruction [7], [8].

VINCI can import protocol files of the older MPI-tool, we have also kept the reslicing engine compatible.

Another software package developed here for multi-modality imaging, especially for surface rendering of co-registered functional and structural images and interactive definition of VOIs, is the 3D-Tool, [5]: VINCI can read objects/regions defined in its volume definition file format and display them as fusion overlay.

3. Implementation and Design Goals

This section describes the VINCI framework we have conceived for scripting, communication between different components of VINCI and remote control. We also explain the circumstances that have motivated some design decisions.

3.1. General Concepts and Definitions

VINCI supports an arbitrary number of image buffers: a system's physical memory and the choice of data sets will limit the amount of buffers one can effectively use. Usually data from one image file will be assigned to one buffer. It can then be used by several modules for graphical display or analysis, e.g., one might have it displayed in two different OrthoView widgets and in a PlanesView at the same time.

A VinciProject comprises any number of image buffers with reslicing and color settings, OrthoViews, PlanesViews, PlotViews and the respective widget settings (options, window sizes and positions). Each Project has an unambiguous name and is associated with a file from which the Project can be restored in a later session. Within one instance of VINCI one can work on several Projects at the same time.

Each image buffer has reslicing and color properties that will be consistent for one entire Project: changes to either reslicing or color properties of one image buffer affect (in most cases immediately) all displays and tools.

3.2. Choice of Platform and Languages

We are actively involved in the development of our PET scanners, so it was mandatory that we follow the manufacturer's preference of platform: this explains a general shift from Solaris 2.6 (the platform favoured by previous generations of PET scanners) to MS Windows NT/2000/XP in the institute's IT infrastructure.

We try to use platform-independent standards throughout large portions of VINCI, a good example being the plugin for fully automatic co-registration: the registration engine is also available in a non-interactive version for Solaris and Linux systems using the same XML-based format to describe registration jobs. Furthermore, VINCI's framework is based on an entirely platform independent concept (VRegistry and XML-based VinciScript).

Similar to a software architecture employed by Wolfram Research's Mathematica package [9], we have a non-GUI backend which is mostly platform independent. For performance reasons and flexibility, we have chosen to use C++ as the principal language for development. When development started in 1998, for the same reasons, MS Window's native MFC with C++ was the obvious choice for the GUI: clinical routine and several research applications require VINCI to have a highly optimized and customized user frontend, something that is notoriously difficult to do with toolkits for platform independent GUIs (with the notable exception of Qt, which was lacking much of its current attractiveness when we first evaluated it in 1999).

We find Linux an excellent platform for our online update service and we use HTML templates without browser or platform specific additions for displaying header and status information.

3.3. Build Process

We are approaching 200,000 lines of source code (including comments) for VINCI and the co-registration plugin. In order to manage this amount of code at a level required for routine usage, especially with a view to team work and international collaboration, it was instrumental to adopt several standard practices for software development and testing, an excellent summary of which we have found in [11].

As VINCI is shipped with the HRRT PET system which is used in a clinical setting, the manufacturer requires a certain level of certification related to the quality assurance of software now that the HRRT is no longer a prototype.

- Source Code Repository: all code, documentation and test data is checked into MS SourceSafe which is integrated into MS Visual Studio 7.1. Accessing the source code repository from outside the institute's network is done frequently using VPN connections through our Checkpoint firewall.
- All changes to code, documentation and test data can be retrieved from the repository for any stage of the development process.
- Fully automated build process: all binaries are built using the latest versions of sources from the repository and then subjected to a number of automated tests, the result of which is summarized and sent by Email to the developers.
- C++ specific documentation (class hierarchies, dependencies and our comments) is automatically created by Doxygen, [12], in the format of an HTML tree which is subsequently packaged into one compressed HTML file.
- A PERL script extracts documentation for VinciScript commands from the sources and assembles the output into another compressed HTML file.
- A successully tested new release is automatically packaged and versioned. The result is ready for manual or automated installation.
- We keep a detailed version history which is included with every installation.

3.4. Installation

VINCI will run immediately on all current Windows versions (NT/2000/XP), no installation is required. However, we also provide an installer which mostly contains an engine for decompression of zip-files and creates shortcuts on the desktop and in the programs menu. As VINCI does not rely on external toolkits, frameworks or libraries other than MFC (this is not necessarily true for third party plugins) a full distribution is contained in a 3.3 MB size installer.

We have derived our installer from the NSIS installer engine, [13], which is extremely fast and compact: it carries almost no overhead compared to the compressed size of the distribution's directory.

A popular way to install software for several systems on a network is to use a shared directory on a file server. Provided the server's availability is satisfactory, this has proved to be a good solution for software that is static for longer periods of time. Compared to the size of current PC disks,VINCI's small footprint is negligible which has allowed us to pursue a more flexible strategy:

- VINCI is *installed locally* on each system. This policy also eliminates problems for laptops and, more general, installations outside the institute's network.
- We have added functionality for *remote network installation*, so several machines can be updated in an automated fashion. We support *concurrent use of multiple versions* of VINCI: each distribution is self-contained and independent.
- As installation is safe, easy and fast we can and have published releases frequently (about once per week), reacting quickly to bug reports and feature requests.

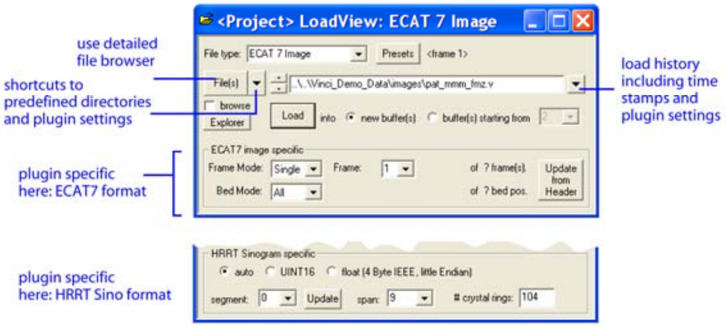
3.5. Online Update

We have developed a one-click online update mechanism: a separate update process is spawned that kills running instances of VINCI and contacts the update server in the institute's DMZ. After sending authentication and site specific license information, it transfers the current installer binary and launches a silent (non-GUI) default installation. Within the institute a full update takes less than 4 seconds, with a DSL-type connection it typically takes around half a minute.

The update service has been implemented on a Linux system using http (cgi) for communication and ftp for the transfer. Authentication and logging are handled by PERL scripts, command and control is XML-based.

3.6. Plugins

An important part of VINCI's architecture is the concept of *plugins* to easily add new functionality in certain, well-defined contexts. By design, plugins only require knowledge of a small (public) subset of VINCI's framework. They are easy-to-use and maintain and thus are very suitable for third- party developers. At startup, VINCI scans its `bin` directory for files containing



use detailed file browser
shortcuts to predefined directories and plugin settings
load history including time stamps and plugin settings
plugin specific here: ECAT7 format
plugin specific here: HRRT Sino format

<Project> LoadView: ECAT 7 Image

File type: ECAT 7 Image Presets <frame 1>

File(s) ..\..\Vinci_Demo_Data\images\pat_mmm_fmz.v

browse Explorer Load into (•) new buffer(s) ( ) buffer(s) starting from 2

ECAT7 image specific

Frame Mode: Single Frame: 1 of ? frame(s) Update from Header

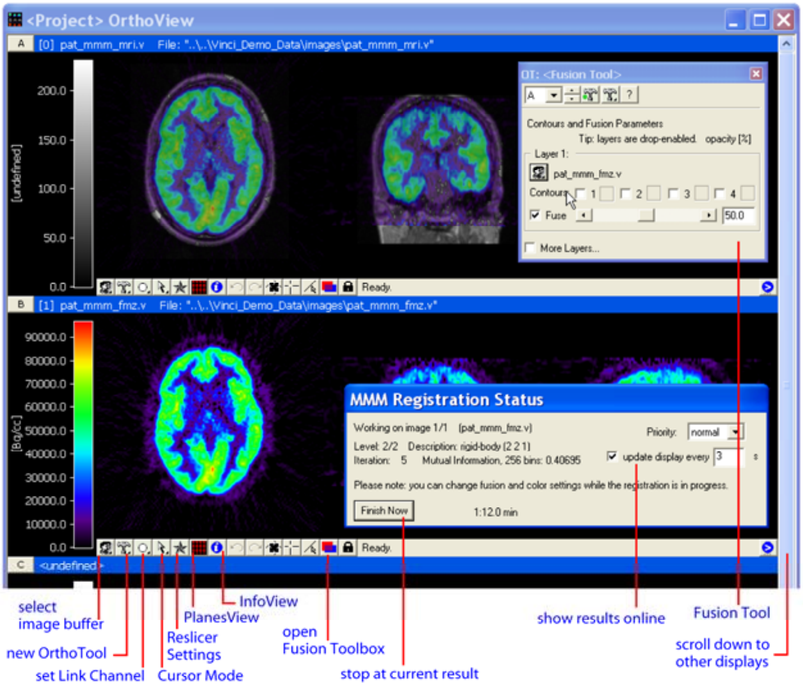
Bed Mode: All of ? bed pos.

HRRT Sinogram specific

(•) auto ( ) UINT16 ( ) float (4 Byte IEEE, little Endian)

segment: 0 Update span: 9 # crystal rings: 104

Fig. 1: LoadView. Depending on the image format you have chosen, the widget will adjust to the size of the plugin that controls the lower part of the widget. Usually, just dragging a file from the Explorer to a suitable drop target in the main window will work: VINCI makes educated guesses about the required plugin and useful default settings. The *Load History* will show full file names with arbitrary length, VINCI has a custom *file browser.*



<Project> OrthoView

A [0] pat_mmm_mri.v File: "..\..\Vinci_Demo_Data\images\pat_mmm_mri.v"

OT: <Fusion Tool>

Contours and Fusion Parameters

Tip: layers are drop-enabled. opacity [%]

Layer 1: pat_mmm_fmz.v

Contours 1 2 3 4

[✓] Fuse 50.0

More Layers...

Ready.

B [1] pat_mmm_fmz.v File: "..\..\Vinci_Demo_Data\images\pat_mmm_fmz.v"

MMM Registration Status

Working on image 1/1 (pat_mmm_fmz.v) Priority: normal

Level: 2/2 Description: rigid-body (2 2 1)

Iteration: 5 Mutual Information, 256 bins: 0.40695 [✓] update display every 3 s

Please note: you can change fusion and color settings while the registration is in progress.

Finish Now 1:12.0 min

Ready.

C <undefined>

Fig. 2: *OrthoView* showing a registration in progress (MMM plugin not shown). The lower *OrthoDisplay* (B) shows a PET image using a rainbow type color scale. The upper OrthoDisplay (A) features a fusion overlay of the MRI image (grayscale) and the PET image. During registration, current registration results will be displayed periodically, fusion and color settings can be adjusted online.

plugins. A plugin is basically a DLL that defines an arbitrary number of classes derived from `CVinci_Plugin`. VINCI will try to call the mandatory function `RegisterLocalPlugins()` which creates instances of each plugin that should be active and adds them to a global list.

We have defined several types of plugins, the most important being the plugins for adding *import filters*. VINCI supports several very different file formats, among them ECAT7, Analyze, MINC, Howmedica Leibinger (used by neurosurgery guidance systems) and the native file types of the HRRT and microPET scanner families.

Another type of plugin works on the image buffers of an OrthoView (OrthoTool Plugins, e.g. MMM, the registration plugin or the 3D-Gauss-filter).

The global list of plugins is then used to assemble the file format or toolbox menu at runtime. We have found [14] and [15] helpful in deciding how to minimize dependencies between VINCI's components.

Loading a particular kind of image data requires a suitable *Load Plugin*. Starting from a generalized sample plugin we provide, it is basically sufficient to implement the functions `IsThisMyFileFormat()` so VINCI can cycle through the list of all available load plugins to automatically select a suitable handler if it is supposed to make an educated guess, e.g. when a file has been dropped on it from the Explorer. For adapting the other mandatory function, `LoadSpecificFileFormat()`, to a particular type of image data, little knowledge about VINCI's internals is needed. Adding an optional user interface only requires implementing a standard CDialog of arbitrary size. Load Plugins are embedded in the LoadView widget, see Fig. 1 for an example.

The *OrthoTool Plugins* are activated through an *OrthoToolBox*: each toolbox has a target display which is the default candidate for manipulations. Creating a plugin of this kind is very similar to writing a load plugin; the toolbox automatically adjusts to the size of the plugin's user interface, s. Figs. 2, 3 and 5 for examples. Each OrthoView can have an arbitrary number of toolboxes associated with it, several tools can be associated with each OrthoDisplay.

3.7. VinciScript

We realized at an early development stage that scripting, or, in general, a (type) safe and easy way to exchange messages between VINCI'S objects was desirable. The heart of which is a global entity, VRegistry, that maintains a flat namespace (using a hash for efficiency) where each object of class `CVinci_XMLTarget` is automatically registered with a unique name on instantiation that also reflects heritage between objects, e.g.

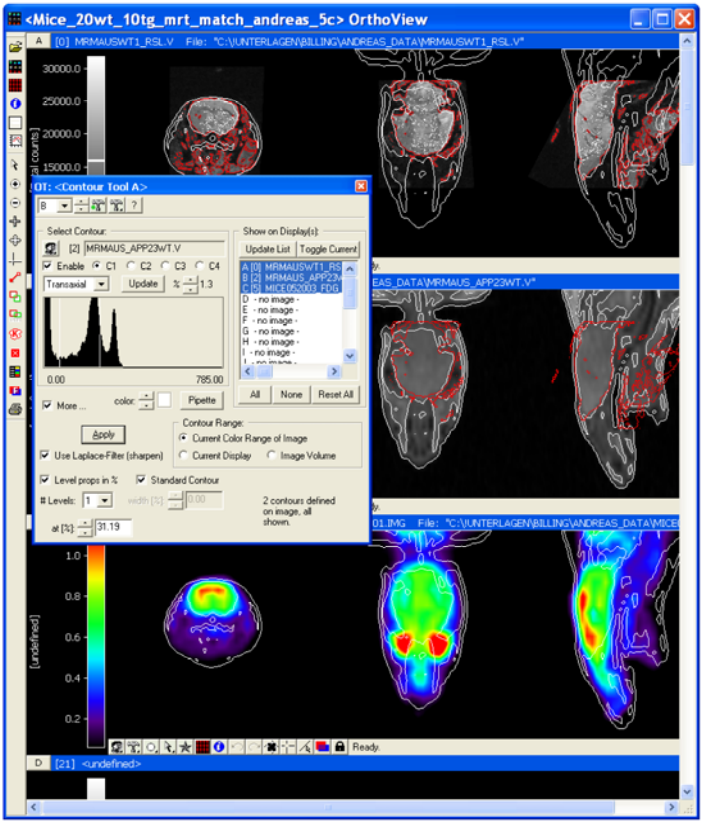


Fig. 3: Phenotyping mouse models of neurological disease by Molecular Imaging technology. 18-FDG microPET study of APP23 Alzheimer's disease mouse model (lower panel; wholebody microPET) matched to a wholebody low-resolution MRI (1.5 T; middle panel). For co-registration of brain metabolic activity to detailed anatomical information, a high-resolution MRI (7 T) was obtained (top panel) and co-registered to the wholebody MRI using the Contours Tool (shown here in advanced mode) in conjunction with rotation, move and zoom cursor modes. See [18] for more information.

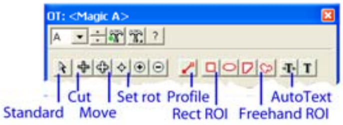


Fig. 4: Upper part of *Magic* plugin showing shortcuts to cursor modes for reslicing and generation of profiles, 2D ROIs and annotation. *AutoText* replaces keywords ($Patientname) by the appropriate information from the current image file.

`::Project_0::LoadView`
to identify the LoadView of the first VinciProject (s.b.), or
```
::Project_1::OrthoView_1::ToolBox_3
 ::Image Volume Filter_0
```
to refer to an image filter tool owned by a certain toolbox that belongs to the second OrthoView. Each `CVinci_XMLTarget` has a handler to process XML-based commands, VinciScript, addressed to it. We provide a large number of methods to navigate the VRegistry's namespace and to send messages safely: by checking the address range and a magic function first, we make sure objects are valid before they are accessed.

Processing of VinciScript has been optimized: top-level XML-commands are delegated to a hash-based handler (which also automatically processes several special commands, see next section) and we can mostly use our own XML parser, an iterator, which is easy to use, compact and very fast because it only needs to understand a small subset of XML (one level only, no attributes).

3.8. VinciProject and Validation

A *VinciProject* in many ways resembles the document part of a Document-View framework. To stay with that terminology, we have implemented a multi-document interface: several VinciProjects can be opened in one session, the context is defined by the heritage of the top-most window.

Any *VinciProject* file consists of `ParameterSection` blocks, which contain *VinciScript* that is addressed to a particular target. The project parser treats the sections as black boxes and dispatches the commands to the specified targets. The meaning of some special tags (`New`, `Del`, `Create`, `Action`, `Current`) can be deduced from several example scripts.

Each VinciProject is signed with a MD5 checksum: if it has been edited outside of VINCI, a warning message will be issued. We also save MD5 fingerprints of image data referred to in the project file. Thus we can ensure the fidelity of VinciProjects which have been relocated, e.g. from an institute's network with central file servers to a laptop (USB stick) for external presentations using VINCI'S "Pack and Go" manager.

3.9. Remote Control, High-Level Interface

The project parser in conjunction with VinciScript is a good basis for scripting VINCI. Remotely controlling one or more instances of VINCI only requires one more ingredient: a means for inter-process communication. We have chosen to use *named pipes*: they have a sound UNIX heritage, are robust and efficient.

Several examples in the `external`-directory of a VINCI distribution show how VINCI can be controlled remotely by programs written in C/C++, IDL and PERL. In a standard installation of a VINCI distribution, we run a post-install program (see `PostInstall.xml` in VINCI's `bin`-directory) to adjust the one statement that links an external program to one particular VINCI distribution.

We provide a number of higher-level routines that conveniently summarize several VinciScript commands and also allow fast, diskless transfer of binary data from an instance of Vinci to the external C/C++/IDL program, and back.

We find IDL particularly suitable to use with VINCI because we were able to put a lot of consistency checks into our library, the language is easy to learn and very popular with researchers involved with tomography.

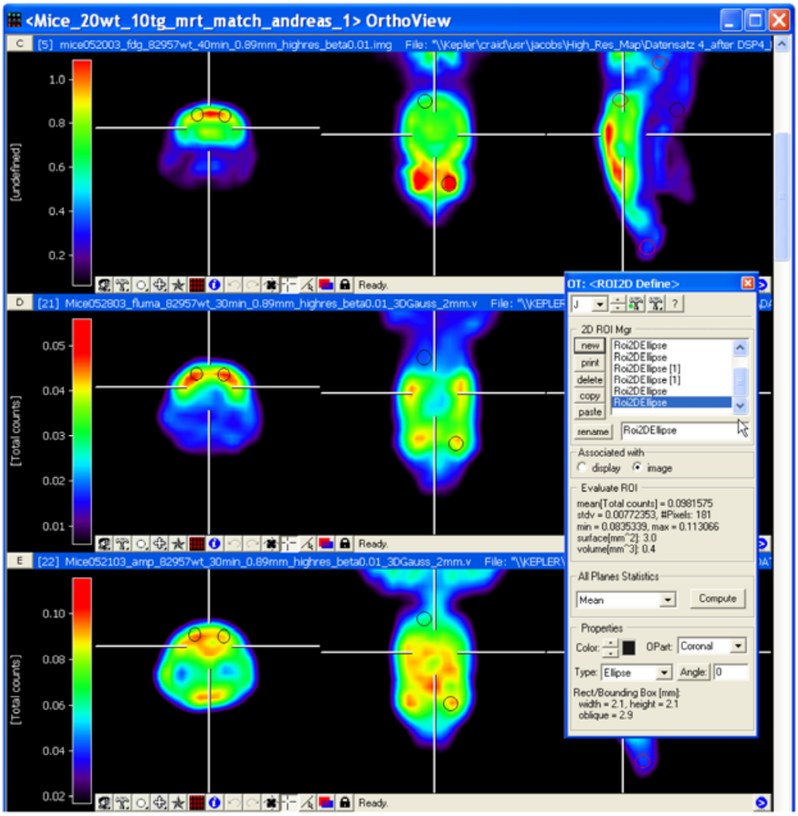
3.10. Automated Tests, Online Tutorial

We have learned to appreciate automated tests as an integral part of the build process. Testing non-GUI backends is mostly a straight-forward task in our case: we have conceived literally hundreds of tests that compare test answers to "manually" validated reference data and report differences.

However, testing complex user interfaces is generally a much more demanding issue. We have started to enhance our framework with functionality found in macro-recorders: VINCI keeps track of (high-level) user actions (changing color settings, opening a toolbox, changing reslicing settings, manipulating ROIs, etc.) and can play them back. This is very different from hooking into mouse and keyboard drivers and playing them back as low-level events (an approach favoured by many APIs for automated tests).

When recording a test sequence, VINCI creates a number of files with state information about recent operations. Replaying the sequence during an automated test, creates a new set of files which are compared (using MD5 checksums) to data generated during a supervised reference run.

We have compiled a CD-ROM that contains a ready-to-run distribution of VINCI, some demonstration data and an interactive tutorial (demo). The latter is largely a by-product of our efforts with automated tests and does not involve Macromind Director or any such product.



<Mice_20wt_10tg_mrt_match_andreas_1> OrthoView

[5] mice052003_fdg_82957wt_40min_0.89mm_highres_beta0.01.img File: "\\Kepler\craid\usr\jacobs\High_Res_Map\Datensatz 4_after DSP4_

[21] Mice052803_fluma_82957wt_30min_0.89mm_highres_beta0.01_3DGauss_2mm.v File: "\\KEPLER

[22] Mice052103_amp_82957wt_30min_0.89mm_highres_beta0.01_3DGauss_2mm.v File: "\\KEPLER\

OT: <ROI2D Define>

2D ROI Mgr

Roi2DEllipse
Roi2DEllipse
Roi2DEllipse [1]
Roi2DEllipse [1]
Roi2DEllipse
Roi2DEllipse

rename: Roi2DEllipse

Associated with: display [✓ image]

Evaluate ROI

mean[Total counts] = 0.0981575
stdv = 0.00772353, #Pixels: 181
min = 0.0835339, max = 0.113066
surface[mm^2]: 3.0
volume[mm^3]: 0.4

All Planes Statistics: Mean

Properties

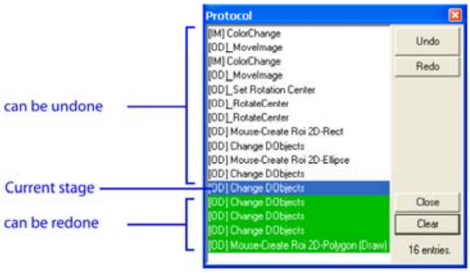
Color: ■ OPart: Coronal
Type: Ellipse Angle: 0

Rect/Bounding Box [mm]:
width = 2.1, height = 2.1
oblique = 2.9

Fig. 5: Phenotyping mouse models of neurological disease by Molecular Imaging technology. Co-registration of 18-FDG- (upper panel) with 11C flumazenil- (middle panel) and 11C MP4A-microPET of a APP23 mouse brain. FDG-PET is used as direct measure of endogenous hexokinase gene expression, flumazenil-PET serves a surrogate marker for neuronal integrity, and MP4A-PET localizes involvement of the nicotinergic system in the disease process.

ROI analysis is being obtained in cortical regions (left column) and reference regions within cerebellum and haderian glands (middle column) as well as further background regions in mediastinum, salivary glands and nose (right column). See [18] for more information.

Also shown: the ROI2D Tool and the Protocol widget (right).



Protocol

[IM] ColorChange
[OD]_MoveImage
[IM] ColorChange
[OD]_MoveImage
[OD]_Set Rotation Center
[OD]_RotateCenter
[OD]_RotateCenter
[OD] Mouse-Create Roi 2D-Rect
[OD] Change DObjects
[OD] Mouse-Create Roi 2D-Ellipse
[OD] Change DObjects
[OD] Change DObjects
[OD] Change DObjects
[OD] Change DObjects
[OD] Change DObjects
[OD] Mouse-Create Roi 2D-Polygon (Draw)

16 entries.

3.11. Multi-stage Undo/Redo

Another by-product of our efforts for automated tests is VINCI's *Protocol* widget. Similar to Adobe Photoshop's History feature, it lists recorded actions eligible for Undo and Redo operations. The screenshot in Fig. 5 shows a typical listing, green entries can be redone, clicking any entry of the list will undo or redo all steps back or forward to that stage—which is helpful for manual co-registration.

3.12. Graphics

VINCI is entirely true color based which eliminates a number of complications and limitations compared to systems relying on 8-bit indexed colors. The true color engine is built around a small number of comparatively low-level blitting functions which are supported by all graphics boards for the MS Windows platform. VINCI offers live (immediate feedback) fusion overlay for up to four images, see Figs. 2 and 6 for a fusion of two images. A *Palette Editor* is included which can be accessed through the color plugin and allows to create new color palettes. All builtin palettes are encoded (XML) using interpolation markers and can be customized.

VINCI can export graphics through the clipboard (a particular OrthoDisplay, a PlanesDisplay, Scatter Plots, a color bar) *retaining vector properties* (text can still be edited) or as bitmaps rendered at a higher resolution to minimize aliasing. The former is especially useful if working with MS Powerpoint, OpenOffice Presentation or Adobe Illustrator to create posters.

Our reslicer supports several rendering modes: depending on the dimensions of the original image and the display size, a 3D-interpolation (trilinear, next neighbour) can be followed by a 2D-interpolation (linear, bicubic and high resolution cubic spline, [10]). For viewing and analysis of scanner raw data, we also support a "pixel native" mode (dubbed *TruePixel* mode): it guarantees to reslice parallel to the image volume's native axes only (no interpolation), 2D-rendering, if necessary, is limited to pixel reduplicaton.

3.13. Printing/pdf, MS Excel/OpenOffice support, Online help

You can print OrthoViews and PlanesViews directly to any printer supported by MS Windows. The layout can be customized with some XML-based configuration files: `PR*.xml` in VINCI's `template`- directory. By

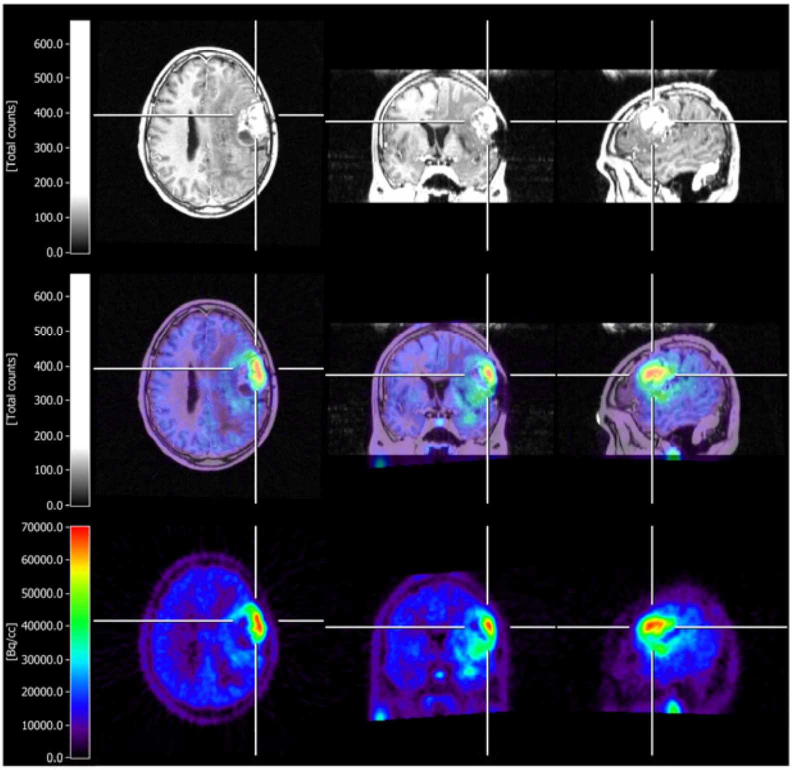
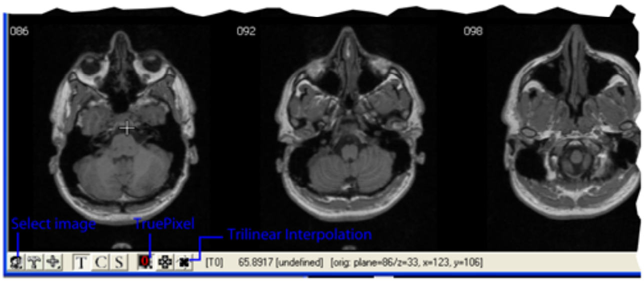


Fig. 6: Co-registration and image fusion of functional scan (C-11-Methionin, depicting metabolically active tumor parts) and structural brain scan (T1-weighted MRI with contrast agent) for planning of tumor surgery. Please note: images have been inserted by "Copy & Paste" from VINCI.



[T0] 65.8917 [undefined] [orig: plane=86/z=33, x=123, y=106]

Fig. 7: Lower Part of a PlanesView. You can also use the Cut tool (cross cursor) on a plane: this will affect the reslicing parameters of the image buffer currently displayed. All displays subscribing to this image buffer will be updated immediately.

default, the layout includes information automatically inserted from the image file and the institute's logo (vector graphic).

If Acrobat Distiller (or any similar printer driver) is installed, VINCI can generate high-quality electronic documentation (pdf) by simply printing to the Distiller printer driver.

Several tools generate table-type output, e.g. the ROIReport which evaluates ROIs in several scopes (project, OrthoView, OrthoDisplay) over several frames or files, or the Time Activity Curve tool. VINCI can write native file formats for MS Excel and OpenOffice Spreadsheet using an XML template-driven approach which allows predefined layouts with style and type attributes (thus numbers will be interpreted correctly regardless of a system's locale).

VINCI has a "one-source" online help system: the online manual is written with Adobe FrameMaker and then converted to compressed HTML using Webworks Publisher Pro with a customized rule base, or saved as pdf.

3.14. Optimization

We have been using the Intel compiler plugin, [17], since version 5.0 for validation and debugging. Several optimization options are now also available with MS Visual Studio 7.1 and have given us a significant speed gain for co-registration, using Pentium IV specific optimization for vectorization.

3.15. UNIX support

Our institute has a heterogeneous IT-infrastructure (Solaris 2.6/7/8, Linux Suse 7/9/Red Hat, Windows NT/2000/XP). Several central file systems are (traditionally) served by Solaris machines running Samba which is a popular and mostly efficient solution to share data between UNIX workstations and PCs. We use NIS+ to manage network wide (virtual) directories.

VINCI uses a rule based approach to map UNIX file names to Samba paths. Text in clipboards can optionally be converted to UNIX line endings.

Clipboard export also works for *OpenOffice Presentation* and *Text*, as does writing of native OpenOffice spreadsheet files (s.a.). OpenOffice files can then be processed on several UNIX platforms.

4. Fully Automatic Co-Registration (MMM-Plugin)

The general principle of the registration algorithm is an iterative search for a transformation that optimizes a similarity measure of alignment of two image volumes. We consider (a) rigid-body transformations for multi-modality intra-subject registration and (b) non-linear deformations for inter-subject spatial normalization.

Method (a) is, e.g., needed for co-registration of a PET-study and a MR-study of the same patient to fuse anatomical and functional information into one image for diagnostic purposes, see Fig. 6 for an example. Method (b) allows to transform image data of one patient into a standardized image space for statistical analysis (another Vinci plugin), e.g. comparisons with collectives of control studies.

The similarity measure of choice for co-registration of images from different modalities (method (a): MRI-CT or MR-PET) is Mutual Information (MI) which has proven to be a very reliable and precise criterion, [19]. For method (b) we use a hierarchical subdivision of the first image volume into blocks that are individually co-registered using affine transformations to the corresponding blocks of the second image volume. The resulting transformation is interpolated between those blocks.

For optimization of the similarity measure we use the downhill simplex optimization method, [20]. We have implemented this in a multi-scale approach (coarse-to-fine optimization) which reduces the computational demand significantly without loss of accuracy.

To further improve robustness and speed of the registration process, we evaluate different techniques for automatic masking of non-brain voxels (intensity thresholding using a image histogram, morphological operations, [21]).

The *Multi Modality Matching Tool* (MMM) is used routinely at the institute for a number of co-registration tasks. As mentioned in section 3.2, the registration engine is also available in a non-interactive version for Solaris and Linux systems using the same XML-based format to describe registration jobs. *BeeQ*, the queue manager we developed for the *HeinzelCluster* [8], is well suited to run these registration jobs on several servers in a user-friendly and controlled way.

5. Outlook

VINCI is the standard visualization tool for the HRRT brain scanner, [1]. It is already being used routinely in multi-modality environments and can be easily adapted to new challenges due to its modular architecture.

With newly introduced indexed volumes in VINCI, we provide accurate quantitation for current and future molecular imaging techniques making use of regions and volumes of interest. Interest in further commercial distribution of our software for scientific and clinical applications has been expressed by several parties.

References

[1] Garching Innovation, S. Vollmar, “Software for Medical Tomography”, MaxPlanck-Research, p. 94, 2/2003.

[2] U. Pietrzyk, K. Herholz, A. Schuster, H.M. von Stockhausen, H. Lucht, W.D. Heiss, “Clinical applications of registration and fusion of multimodality brain images from PET, SPECT, CT and MRI”, European Journal of Radiology 21, pp. 174-182, 1996.

[3] U. Pietrzyk, “Registration of MRI and PET images for clinical applications”, Medical Image Registration, CRC Press, 1999-216, 2001.

[4] Online Resources MPITool, http://www.atv-gmbh.de/mpitoolh/

[5] K. Herholz, H.-H. Reulen, H.-M. von Stockhausen, A. Thiel, J. Ilmberger, J. Kessler, W. Eisner, T.Yousry, W.-D. Heiss, “Preoperative activation and intraoperative stimulation of language-related areas in glioma patients”. Neurosurgery 41 (6), pp. 1253-1262, 1997.

[6] S. Vollmar et al., Proceedings of the Sixth International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine, Pacific Grove, CA, pp. 185-188, 2001.

[7] S. Vollmar, K. Wienhard, “BeeHive: Cluster Computing im NT-Netzwerk”, Neue Technologien zur Nutzung von Netzdiensten, GWDG-Bericht Nr. 55, pp. 2651-2658, 2000.

[8] S. Vollmar, C. Michel, J.T. Treffert, D.F. Newport, M. Casey, C. Knöss, K. Wienhard, X. Liu, M. Defrise, W-D. Heiss, “HeinzelCluster: accelerated reconstruction for FORE and OSEM3D”, Physics in Medicine and Biology, Vol 47, No. 15, pp. Aug 2002.

[9] S. Wolfram, The Mathematica Book 4, Cambridge University Press.

[10] J.A. Parker, R.V. Kenyon, D.E. Troxel, “Comparison of Interpolating Methods for Image Resampling”, IEEE Transactions on Medical Imaging, MI2 (1): 31-39, 1983.

[11] A. Hunt, D. Thomas, The Pragmatic Programmer, http://www.pragmaticprogrammer.com/ppbook/index.shtml.

[12] Doxygen: a documentation system for C++ and other languages. http://www.doxygen.org

[13] NSIS: a toolkit to create installers (MS Windows), http://www.nullsoft.com/free/nsis/

[14] J. Lakos, Large-Scale C++ Software Design, Addison Wesley Longman, 1996.

[15] S. Meyers, Effective C++: 50 Specific Ways to Improve Your Programs and Designs, Addison Wesley Longman, Reading, MA, 1997.

[16] Online Resource, ActivePERL, a PERL distribution: http://www.activestate.com/

[17] Online Resource, Intel Compiler, http://www.intel.com/software/products/compilers/

[18] A.H. Jacobs, H. Li, A. Winkeler, R. Hilker, C. Knoess, A. Ruger, N. Galldiks, B. Schaller, J. Sobesky, L. Kracht, P. Monfared, M. Klein, S. Vollmar, B. Bauer, R. Wagner, R. Graf, K. Wienhard, K. Herholz, W-D. Heiss, “PET-based molecular imaging in neuroscience”, Eur J Nucl Med Mol Imaging, 2003.

[19] F. Maes, D. Vandermeulen, P. Suetens, “Comparative evaluation of multiresolution optimization strategies for multimodality image registration by maximization of mutual information”, Med Imag Anal 3 no. 4, 373–386, 1999.

[20] J.A. Nelder, R. Mead, “A simplex method for function minimization”, Computer Journal 7, pp. 308-313, 1965.

[21] J. Čížek, K. Herholz, S. Vollmar, R. Schrader, J. Klein, W.-D. Heiss, “Fast and robust registration of PET and MR images”, NeuroImage, in print

Anschriften der Autoren

Axel Arnold
Max-Planck-Institut für Polymerforschung
Ackermannweg 10, 55128 Mainz
E-Mail: axel.arnold@mpip-mainz.mpg.de

Roland Chrobok
Universität Duisburg-Essen, Institut für Physik
Lotharstr. 1, 47048 Duisburg
E-Mail: chrobok@traf1.uni-duisburg.de

Jiří Čížek
Max-Planck-Institut für neurologische Forschung
Gleuelerstr. 50, 50931 Köln
E-Mail:jiri.cizek@pet.mpin-koeln.mpg.de

Dr. Ralf Deiterding
California Institute of Technology
1200 E California Blvd., MC 158-79, Pasadena CA 91125
E-Mail: ralf@cacr.caltech.edu

Ben Greiner
Universität Köln, Staatswissenschaftliches Seminar
Albertus-Magnus-Platz, 50923 Köln
E-Mail: bgreiner@uni-koeln.de

Dr. Sigurður F. Hafstein
Universität Duisburg-Essen, Institut für Physik
Lotharstr. 1, 47048 Duisburg
E-Mail: hafstein@traf1.uni-duisburg.de

Thomas Hahn
Max-Planck-Institut für Physik
Föhringer Ring 6, 80805 München
E-Mail: hahn@mppmu.mpg.de

Prof. Dr. Karl Herholz
Max-Planck-Institut für neurologische Forschung
Gleuelerstr. 50, 50931 Köln
E-Mail:karl.herholz@pet.mpin-koeln.mpg.de

Dr. Christian Holm
Max-Planck-Institut für Polymerforschung
Ackermannweg 10, 55128 Mainz
E-Mail: christian.holm@mpip-mainz.mpg.de

Dr. Andreas Jacobs
Max-Planck-Institut für neurologische Forschung
Gleuelerstr. 50, 50931 Köln
E-Mail:andreas.jacobs@pet.mpin-koeln.mpg.de

Thies H. Jochimsen
Max-Planck-Institut für Kognitions- und Neurowissenschaften
Stephanstr. 1a, 04103 Leipzig
E-Mail: jochimse@cbs.mpg.de

Johannes Klein
Max-Planck-Institut für neurologische Forschung
Gleuelerstr. 50, 50931 Köln
E-Mail: johannes.klein@pet.mpin-koeln.mpg.de

Hanjo Limbach
Max-Planck-Institut für Polymerforschung
Ackermannweg 10, 55128 Mainz
E-Mail: hanjo.limbach@mpip-mainz.mpg.de

Bernward Mann
Max-Planck-Institut für Polymerforschung
Ackermannweg 10, 55128 Mainz
E-Mail: bernward.mann@mpip-mainz.mpg.de

Michael v. Mengershausen
Max-Planck-Institut für Kognitions- und Neurowissenschaften
Stephanstr. 1a, 04103 Leipzig
E-Mail: mengers@cbs.mpg.de

Andreas Pottmeier
Universität Duisburg-Essen, Institut für Physik
Lotharstr. 1, 47048 Duisburg
E-Mail: pottmeier@traffic.uni-duisburg.de

Michael Sue
Max-Planck-Institut für neurologische Forschung
Gleuelerstr. 50, 50931 Köln
E-Mail: michael.sue@pet.mpin-koeln.mpg.de

Stefan Vollmar
Max-Planck-Institut für neurologische Forschung
Gleuelerstr. 50, 50931 Köln
E-Mail: stefan.vollmar@ pet.mpin-koeln.mpg.de