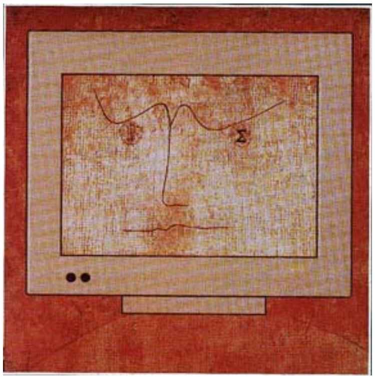
GWDG-Bericht Nr. 58

Theo Plesser, Volker Macho (Hrsg.)

Forschung und wissenschaftliches Rechnen

Beiträge zum Heinz-Billing-Preis 2001



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. 58

Volker Macho, Theo Plesser (Hrsg.)

Forschung und wissenschaftliches Rechnen

Beiträge zum Heinz-Billing-Preis 2001

Gesellschaft für wissenschaftliche Datenverarbeitung
Göttingen

2003

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: Weender Druckerei, D-37077 Göttingen

ISSN: 0176-2516

Inhalt

Vorwort

Der vorliegende neunte Band der Reihe „Forschung und wissenschaftliches Rechnen“ enthält sieben der acht für den Heinz-Billing-Preis im Jahre 2001 eingereichten Beiträge. Ein Beitrag liegt leider nur als Abstract vor. Den letztmalig in DM ausgeschriebenen Preis erhielt Herr Dr. Jörg Haber vom Max-Planck-Institut für Informatik in Saarbrücken für seine Arbeit „MEDUSA – A Facial Modeling and Animation System“.

Mit diesem Band findet eine erste Phase des „Heinz-Billing-Preis“ ihren Abschluss. Diese Zäsur wird markiert durch eine Änderung in der Ausschreibung des „Heinz-Billing-Preises“. Bis zum Preis des Jahres 2001 hieß es in der Ausschreibung einschränkend „Die Beiträge .... müssen thematisch mit einem Institut der Max-Planck-Gesellschaft verbunden sein.“ Diese Einschränkung ist mit Zustimmung von Herrn Professor Billing in der Ausschreibung für den Heinz-Billing-Preis 2002 nicht mehr enthalten.

In den Jahren 1994 bis 2001 wurden in der Reihe „Forschung und wissenschaftliches Rechnen“ 57 Arbeiten zum Heinz-Billing-Preis publiziert. Hinzu kamen zwei Beiträge zur Situation der DV in der Max-Planck-Gesellschaft und ein Beitrag zur Geschichte des „Beratenden Ausschuss für EDV–Anlagen in der Max-Planck-Gesellschaft“ anlässlich der zweihundertsten BAR Sitzung im November 2001. Die Abkürzung BAR weist auf die frühere Bezeichnung des Ausschusses „Beratender Ausschuss für Rechenanlagen in der Max-Planck-Gesellschaft“ hin.

Da der Einsatz der EDV für die Forschung sehr eng mit der technischen Entwicklung von Hard- und Software verbunden ist, war es von Anfang an ein Ziel dieser Buchreihe, Beiträge aus den Forschungslabors der einschlägigen Industrie zu integrieren. Dies ist mit insgesamt 6 Beiträgen bis 1996 einigermaßen gelungen. In den folgenden Jahren haben ihre Firmen immer wieder Bereitschaft zu einem Beitrag signalisiert, jedoch wurde dies leider aus unterschiedlichsten Gründen nicht umgesetzt.

Der erste Band der Reihe „Forschung und wissenschaftliches Rechnen" wurde mit Beiträgen zum 10. EDV-Benutzertreffen als Heft 1/94 der Berichte und Mitteilungen der Max-Planck-Gesellschaft herausgegeben. Die weiteren Bände erschienen in der Reihe der Berichte der Gesellschaft für wissenschaftliche Datenverarbeitung, Göttingen (GWDG). An dieser Stelle möchten die bisherigen Herausgeber dieser Reihe, Helmut Hayd, Stefan Heinzel, Volker Macho, Theo Plesser und Peter Wittenburg den Geschäftsführern der GWDG ganz herzlich für die Unterstützung dieses Projektes danken. Besonderen Dank gilt Herrn Günter Koch aus der Arbeitsgruppe Informationsmanagement, der in all den Jahren die eingereichten Manuskripte in die „Druckform" gebracht hat. Herr Koch hat mit seinen Anregungen und Hinweisen an die Herausgeber sehr wesentlich zum Gelingen der Buchreihe beigetragen, und wir hoffen auf weitere gute Zusammenarbeit mit Herrn Koch und der GWDG.

Der Heinz-Billing-Preis wird getragen von der Heinz-Billing-Vereinigung zur Förderung des wissenschaftlichen Rechnens e.V. Die finanzielle Basis zur Bereitstellung des Preisgeldes und zur Deckung der Publikationskosten ist durch Spenden aus der Industrie gegeben. Folgende Firmen haben die Vereinigung bisher unterstützt:

Digital Equipment GmbH, IBM Deutschland GmbH, Janz Computer AG, NEC Europa Supercomputer Systems, SiliconGraphics GmbH und SunMicrosystems GmbH.

Zur Stärkung der finanziellen Basis wurde von der Heinz-Billing-Vereinigung das Konzept des Hauptsponsors entwickelt. Nachdem 2000 und 2001 die Firma NEC Europa Supercomputer Systems Hauptsponsor war, hat 2002 die Firma IBM Deutschland GmbH diese Aufgabe übernommen.

Informationen zum Heinz-Billing-Preis und zu den Arbeiten der letzten Jahre sind unter

www.mpg.de/billing/billing.html

im Internet zu finden.

Volker Macho, Theo Plesser

Der Heinz-Billing-Preis 2001

Ausschreibung des Heinz-Billing-Preises 2001 zur Förderung des wissenschaftlichen Rechnens

Im Jahre 1993 wurde zum erstenmal der Heinz-Billing-Preis zur Förderung des wissenschaftlichen Rechnens vergeben. Mit dem Preis sollen die Leistungen derjenigen anerkannt werden, die die Hard- und Software entwickeln, die für neue Vorstöße in der Wissenschaft unverzichtbar sind und den Einsatz der EDV in den Instituten der Max-Planck-Gesellschaft fördern.

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 steht unter dem Leitmotiv

„EDV als Werkzeug der Wissenschaft“.

Für den Heinz-Billing-Preis können Arbeiten eingereicht werden, die beispielhaft dafür sind, wie die EDV als methodisches Werkzeug Forschungsgebiete unterstützt oder einen neuen Forschungsansatz ermöglicht hat. Dabei kann es sich um eine spezielle Hardware, einen eleganten Algorithmus oder um die intelligente Nutzung von Standardsoftware handeln.

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 DM 5000,- dotierten Heinz-Billing-Preis aus. Die Beiträge zum Heinz-Billing-Preis können in deutscher oder englischer Sprache abgefaßt sein und müssen über ein Max-Planck-Institut eingereicht werden. Die Beiträge, es müssen keine Originalarbeiten sein, sollen möglichst nicht mehr als fünfzehn Seiten umfassen.

Da zur Bewertung eines Beitrages 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 zur Förderung des wissenschaftlichen Rechnens wird jährlich vergeben. Die Preisverleihung findet anläßlich des 18. EDV Benutzertreffens der Max-Planck-Institute am 22. November 2001 in Göttingen statt.

Beiträge für den Heinz-Billing-Preis 2001 sind bis zum 17. September 2001 einzureichen:

Dr. Theo Plesser
Max-Planck-Institut für molekulare Physiologie
Otto-Hahn-Str. 11, D-44227 Dortmund

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

Das Kuratorium des Heinz-Billing-Preises

Prof. Dr. Heinz Billing,
Max-Planck-Institut für Astrophysik, Garching

Prof. Dr. Klaus Hahlbrock,
Max-Planck-Institut für Züchtungsforschung, Köln

Prof. Dr. Friedel Hossfeld,
Forschungszentrum Jülich GmbH, Jülich

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

Mitglieder der Jury 2001:

Prof. Dr. Wolfgang Baumeister
Max-Planck-Institut für Biochemie, Martinsried

Prof. Dr. Peter Hagoort
F.C. Donders Centre for Cognitive Neuroimaging, University of Nijmegen

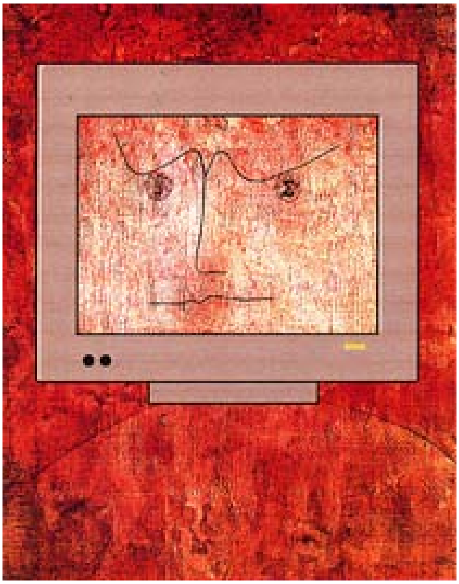
Prof. Dr. Friedrich Hertweck
Emeritiertes Wissenschaftliches Mitglied
Max-Planck-Institut für Plasmaphysik, Garching

Prof. Dr. Kurt Kremer
Max-Planck-Institut für Polymerforschung, Mainz

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

Prof. Dr. Walter Thiel
Max-Planck-Institut für Kohlenforschung, Mülheim an der Ruhr

Prof. Dr. Simon D.M. White
Max-Planck-Institut für Astrophysik, Garching



Jörg Haber
Max-Planck-Institut für Informatik, Saarbrücken

erhält den

Heinz-Billing-Preis 2001
zur Förderung
des wissenschaftlichen Rechnens

als Anerkennung seiner Arbeit

MEDUSA, ein Software-System zur Modellierung und Animation von Gesichtern

Laudatio

Der Heinz-Billing-Preis 2001 wird verliehen für das Design und die Implementierung von MEDUSA – A Facial Modeling and Animation System. MEDUSA stellt eine Sammlung von Werkzeugen bereit, um mit geringem Aufwand realistische Gesichtsmodelle lebender Personen halbautomatisch zu erzeugen und zu animieren. Die Anatomie der Schädelform, die mechanischen Eigenschaften der Haut und die Wirkungsweise der Muskulatur werden in hoher Qualität nachgebildet. Durch Unterlegung eines physikalischen Modells der Muskeln ist es möglich, eine mit der Sprache synchronisierte Animation des Mundes und der gesamten Gesichtspartie zu generieren. Die Animation erfordert einen besonders effizienten Algorithmus zur Wiedergabe der Deformation von Hautgewebe. Dieser Algorithmus ist so effizient implementiert, dass die realistische Sprachanimation in Echtzeit auf einem leistungsstarken PC mit bis zu 50 Bildern pro Sekunde erzeugt werden kann.

Herr Prof. Kurt Krämer überreicht Herrn Dr. Jörg Haber (links) die Urkunde zum Heinz-Billing-Preis 2001

Die Autoren von MEDUSA haben gezeigt, dass ein komplexer alltäglicher Vorgang unter Einbeziehung aller notwendigen Komponenten fast lebensecht in der virtuellen Welt eines Standardrechners abgebildet werden kann. Neben der Nutzung von MEDUSA als Forschungsinstrument liegen weitere Anwendungen im Bereich der Kommunikation und der kosmetischen Chirurgie sowie in der Film- und Unterhaltungsindustrie.

MEDUSA — A Facial Modeling and Animation System

Jörg Haber
Max-Planck-Institut für Informatik, Saarbrücken

Abstract

We present a system for photo-realistic facial modeling and animation, which includes several tools that facilitate necessary tasks such as mesh processing, texture registration, and assembling of facial components. The resulting head model reflects the anatomical structure of the human head including skull, skin, and muscles. Semi-automatic generation of high-quality models from scan data for physics-based animation becomes possible with little effort.

A state-of-the-art speech synchronization technique is integrated into our system, resulting in realistic speech animations that can be rendered at real-time frame rates on current PC hardware.

Keywords: *facial animation, physics-based simulation, speech synchronization*

1 Introduction

Modeling and animation of human faces is one of the most difficult tasks in computer graphics today, even more so when life is to be breathed into digitized versions of real, well-known individuals. With the advent of virtual social spaces, where people communicate face-to-face, the demand for believable virtual characters increases. Accurate reconstruction of individual faces also has major applications in medicine, e.g. cosmetic surgery, and entertainment, e.g. virtual actors.

Why is this goal so elusive? Part of the answer lies in the sensitivity of the human visual system to the nuances of facial expression that it can perceive. Another part is the sheer size of the task of reproducing the complexity of the face in the computer. An individual's face needs to be accurately captured in geometry and texture. To animate the face on a low level, appropriate animation parameters have to be chosen and the resulting face deformations must be computed. On a higher level, these animation parameters are used to produce expressions and speech. Finally, the animated face needs to be rendered. Apart from the conceptual complexity of each of these tasks, further constraints arise when real-time animation is required, as it is the case for dynamic virtual environments.

In this paper, we present an overview of our facial animation system MEDUSA, discussing each step from data acquisition to real-time rendering. We aim at the construction of realistic animatable head models from real human individuals. Geometry and images are acquired in high resolution and converted to textured polygonal geometry, simplified according to the requirements of the targeted hardware. On the lowest level, we use a physics-based approach with muscle contraction values as animation parameters. The construction of the virtual head model is based on the human anatomy, and we have developed tools to automate the task of adapting and linking this model to the actual face geometry. Multi-threaded execution of simulation and rendering results in real-time frame rates on current PC hardware. A schematic overview of the animation components of our system is depicted in Figure 1 on page 15. As an application demonstrating higher level animation, we drive the system using a state-of-the-art speech synchronization method that takes into account the complex influence of coarticulation.

2 Previous and Related Work

Animating models of the human face has remained an attractive and challenging area of research over the last 25 years [26, 30, 27]. A multitude of techniques and approaches are documented in the literature, which can be broadly classified into the following categories: parametric models, physics-based models, image-based methods, and performance-based animation.

Parametric models have been invented very early, to avoid the complexity of specifying complete key frames for each facial posture. Deformation of the skin is achieved by direct manipulation of the geometry [26, 27]. The parameterizations are often inspired by anatomical knowledge: vectors and radial functions have been used by WATERS [35] to approximate deformations caused by the facial musculature. Free-form deformations have been employed by CHADWICK *et al.* [5] to shape the skin in a multi-layer model,

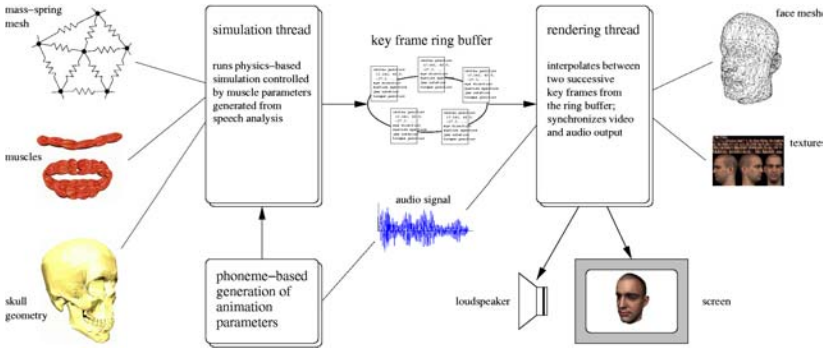

mass–spring mesh
muscles
skull geometry
simulation thread
runs physics–based simulation controlled by muscle parameters generated from speech analysis
key frame ring buffer
rendering thread
interpolates between two successive key frames from the ring buffer; synchronizes video and audio output
face mesh
texture
audio signal
phoneme–based generation of animation parameters
loudspeaker
screen

Fig. 1: Overview of the simulation and rendering components of our system. Inter-thread communication between the simulation and the rendering thread is established via the key frame ring buffer, cf. Section 5.3.

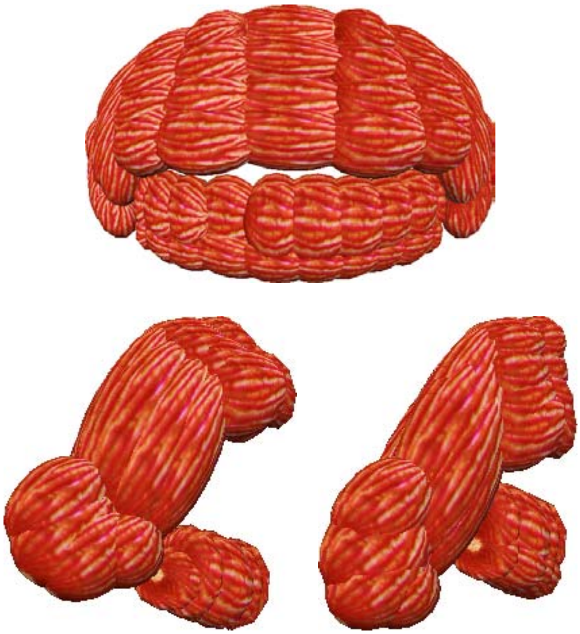


Fig. 2: Two-part orbicularis oris model. Top: relaxed state with closed mouth, front view. Bottom left: protruded lips, slightly opened mouth, side view. Bottom right: upper lip retracted, lower lip moved upward and inward.

which contains bones, muscles, fat tissue, and skin. NAHAS *et al.* [25] use a B-spline surface model to generate synthetic visual speech. A standardized set of facial animation parameters has recently been established in the form of the MPEG-4 standard [15], which has been used by GOTO *et al.* [11] to control their facial animation system.

Building on the idea of virtual muscles, physics-based approaches attempt to simulate the influence of muscle contraction onto the skin surface by approximating the biomechanical properties of skin. Typically, mass-spring or finite element networks are used [30, 21, 20]. In the context of speech animation, WATERS and FRISBIE [36] proposed a two-dimensional mass-spring model of the mouth with the muscles represented as bands. TERZOPOULOS and WATERS [33] automatically construct a three-dimensional model of the human face from an initial triangle mesh. Their model consists of three layers representing the muscle layer, dermis and epidermis. The elastic properties of skin are simulated using a mass-spring system. Employing additional volume preservation and skull penetration constraints, this approach produces realistic effects including wrinkling at interactive frame rates. This model was later simplified by LEE *et al.* [22] to two layers (dermal-fatty and muscles), which connect to the bone structure underneath. Construction of the head model from acquired surface data is largely automated. WU *et al.* focus on generation of expressive wrinkles and skin aging effects. Their face model incorporates viscoelastic properties of skin, and muscles are represented by surfaces of revolution [40] or B-spline patches [39]. Accurate simulation of skeletal muscles (not regarding the overlying skin tissue) has been developed by CHEN and ZELTZER [6] based on a finite element model. Recently, SCHEEPERS *et al.* [32] and WILHELMS and VAN GELDER [37] introduced anatomy-based muscle models for animating humans and animals, also focusing on the skeletal musculature. Skin tissue is represented only by an implicit surface with zero thickness [37].

The computational complexity and the necessary anatomical knowledge for accurate physics-based simulation of facial expressions have recently led to the rise of image-based techniques [29, 4]. These approaches result in photo-realistic images by matching a three-dimensional model to one or more photographs of an individual. Animation is still limited, though, since each expression has to be captured in advance and blending expressions is achieved by linear blends, neglecting the dynamics of facial gestures. Capturing these dynamics is the essence of performance-based methods [38, 12], where real faces are tracked to reproduce the motion on the virtual model.

Several facial animation systems that include speech synchronization have been proposed. LEWIS and PARKE presented automatic speech synchronization for recorded speech [23]. They use linear prediction to analyze the speech signal and generate a phoneme sequence from a set of twelve reference phonemes. Each phoneme is associated with parameters for lip shape and jaw rotation, which are used to render key frames of a parametric face model. An automatic approach to synthesize speech by rules and drive a parametric face model has been proposed by PEARCE, HILL *et al.* [28, 13]. Parameter values for the animation were taken from measuring and comparing front view photographs. This approach has been adopted by COHEN and MASSARO [7]. They enhanced the model by modeling coarticulation and using a synthetic tongue. KALRA *et al.* describe a multi-layered approach to specify facial animation [18]. In their model, words are mapped to phonemes using an interactively built dictionary. Phonemes and expressions are in turn translated into abstract muscle action procedures that drive facial animation. A video showing a real person talking was used by WATERS to generate control parameters for his two-dimensional mouth model [36]. He uses an optimization algorithm to determine muscle parameters that minimize the total distance between seven reference points on the digitized frames of the video and the corresponding points on the model. IP and CHAN use an English-text-to-phoneme parser to generate a sequence of phonemes [14]. Using an interactively built database, each phoneme is translated into a set of control point displacements, which are applied to a face model represented by a NURBS surface.

3 Data Acquisition

In our approach, facial data is captured from real humans. Head geometry and texture information are acquired in separate procedures. Geometry acquisition results in a high resolution triangle mesh, against which textures are registered. For real-time applications, the geometry is scaled down using mesh simplification to any desired level, and the final textured model is obtained by automatically re-registering the textures to the geometry. In this way, we can quickly create a number of models with different resolutions in terms of the polygon count.

3.1 Geometry

Data is acquired using a range scanning system based on the triangulation principle. The scanner delivers uncalibrated range images. The data is processed over a couple of steps to obtain the final head geometry:

1. Range images are registered, resulting in a point cloud.
2. An initial triangle mesh is generated by surface extraction.
3. General post-processing is usually necessary to fix holes due to missing data and to remove noise.

4. The geometry needs to be prepared for facial animation: the range scans are taken from a model with a neutral expression and a closed mouth. Thus we have to cut the mesh open between the lips. In addition, the part of the mesh representing the eyeballs is flattened to allow for proper insertion of our synthetic eye models, cf. Section 4.1.
5. Mesh decimation is applied to the once-prepared high resolution mesh as needed, to produce approximations that are both suitable for simulation and real-time rendering purposes. We have obtained good results with about 3000–4000 triangles for a complete head model.

Methods for processing polygonal geometry in this manner are well described in the literature [19]. We give some details here about the mesh decimation algorithm, which has to preserve a number of properties on the mesh, in particular:

Approximation error: the resulting face mesh has to approximate the original geometry closely, not only to serve resemblance to the original, but also to be able to re-apply the registration parameters for the textures. We found a tolerance of 1–2 mm distance error (from the point of the original mesh to the surface of the new mesh) to be practical. This is actually within the range of error present in the data obtained from the range scanner.

Triangle quality: the numerical stability of the animation system is improved by generating triangles with similar edge length. Also, the triangles should not be too long, regardless of approximation quality, so that local vertex movements during animation do not influence far away regions of the mesh.

Feature preservation: critical regions in the human face are the eyes and the mouth. Here, a higher resolution of the mesh is necessary to preserve good quality under deformation. We have found that limiting changes in mesh curvature (based on measuring face normals) are appropriate to keep these parts of the face sufficiently detailed.

Figure 7 a) shows two snapshots of our mesh decimation tool with the initial geometry and a low-resolution approximation of the head model.

3.2 Textures

To generate a *texture*[1] for a head model, we take several photographs of the person's head using different, uncalibrated camera positions. The number of photographs we use varies from four to eleven, depending on the desired quality of the combined texture and the time available for texture registration. In practice, we found four to six photographs to be sufficient. All photographs are taken with a high resolution digital camera under diffuse illumination.

[1]an electronic image that contains color values for the surface to which it is mapped

During the photo session, the facial expression of the person should resemble the neutral expression during the scanning process.

The photographs are registered and combined into a single texture suitable for rendering on graphics hardware (OpenGL) similarly to the approach in [31], which is based on the camera calibration technique developed by TSAI [34]. In contrast to the method proposed in [31], we do not rely on the property that every vertex of a face mesh is bound to an input texture. Especially in the regions inside and behind the ears, some vertices typically remain unbound. For each unbound vertex v in the mesh, we perform a region growing over the topological neighborhood of v and interpolate the texture coordinates of those vertices within that region R_v, which are bound to the same and most frequently used texture. If the texture binding within R_v is evenly distributed among several textures, we examine remaining *valid* texture bindings of the vertices in R_v (cf. [31, Sect. 3] for a definition of *valid*) and interpolate the texture coordinates within the prevailing texture. If there is still no preferred texture among all texture bindings, we randomly choose one of the most frequently used textures. The interpolated texture coordinates of vertex v need to be verified to lie within $[0,1]^2$. If this is not the case, another frequently used texture is chosen for interpolation. Figure 7 b) shows a snapshot of the GUI of our texture registration tool, where corresponding points have been interactively selected on both the 3D geometry and the 2D input photograph.

4 Construction of the Head Model

4.1 Eyes, Teeth, and Tongue

What do the human eyes, teeth, and tongue have in common? They are both important for realistic facial animation, and it is difficult to acquire data from a human being to precisely model these facial components. Thus we use generic models of these components, see Figure 8. The design of our generic models has been chosen such that they look convincingly realistic when inserted into a face mesh while still being rendered efficiently using OpenGL hardware. We have acquired a set of textures to choose from, e.g. blue, green, and brown eyes. Currently, we are working on the development of a texture synthesis algorithm, which generates a full eye texture from the (small) part of the eye that is visible in a standard portrait photograph.

Our generic facial components are animatable in a number of ways. For the eyes, the viewing direction can be specified as well as the size of the pupil and the aperture of the eyelids. The latter is also used to control the brightness of the eyes. While the position of the upper teeth remains fixed with respect to the skull position, the lower teeth can rotate along with the mandible about the axis that connects the left and right temporomandibular joint. An additional horizontal translation can be specified for the mandible and the lower teeth. Our tongue model allows rigid transformations, which are composed with the transformation of the mandible. We are currently investigating the applicability of more powerful shape-deforming transformations for the tongue. The brightness of both the teeth and the tongue is scaled by the mouth opening value.

4.2 Muscles

In our physics-based simulation, deformation of the skin mesh is caused by simulated contraction of facial muscles. For speech animation, we have built a set consisting of twelve of the most important muscles in the lower face. These muscles have been modeled from interactively specified coarse outlines on one of our head models using our muscle modeling approach as described in [17], see also Figure 7 c). Details on the behavior of the muscles during animation are given in Section 5.1.

4.3 Fitting of Components

The need to match all the parts described above to a specific head model results in a sizable amount of work. To ease the construction of the head model, we express all components in the coordinate frame of a generic reference skull (see Figure 9).

Upon creation of a new virtual head model, the reference skull is matched interactively to the skin mesh using affine transformations. By applying the skull transformation to the parts[2], we get an initial layout for all components of the head model. Adaptation of the muscle set to the details of the face geometry and linking the mesh to the muscles proceeds fully automatically. Usually only small corrections are necessary to accommodate for individual features of a face, which we support in our interactive editing tool depicted in Figure 7 c).

5 Animation and Rendering

5.1 Muscle Animation

Each muscle is built from individual fibers that are in turn composed of piecewise linear segments, given as a control polygon. An ellipsoid is aligned to

[2]only teeth and tongue are deformed, eyes are only uniformly scaled

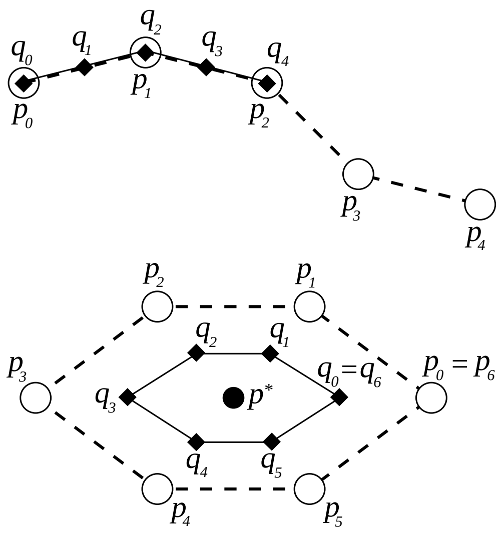


Fig. 3: Contraction ($c = \frac{1}{2}$) of a linear (top) and a sphincter (bottom) muscle fiber. The control points $\{p_i\}$ and $\{q_i\}$ represent the relaxed and contracted muscle, respectively.

each of these segments to provide shape and volume. Given a contraction parameter $c \in [0, 1]$, a new control polygon is computed from the initial one. Sphincter muscles are supported by contraction towards a center point. Through recomputation of the assigned quadrics, where we also incorporate bulging effects, the muscle is re-shaped to reflect its contracted state [17]. Figure 3 exemplifies contraction of a linear and a sphincter muscle fiber.

In speech animation, the most important muscle is the sphincter enclosing the lips, the *orbicularis oris*. This muscle is usually approximated as a closed ring, though from an anatomical point of view it consists of at least four interconnected parts. Many of the parallel muscles from the lower face merge into the orbicularis oris, e.g. the lip raisers (the *zygomaticus* and *levator labii superioris* muscles). In articulated speech, the shape of the mouth is mostly determined by the shape of the surrounding muscles. A number of observations can be made about their deformation:

Elastic behavior: real muscles straighten under tension, i.e. a parallel muscle that has a curved shape in relaxed state resembles a rather straight line under contraction (resulting from the forces applied to both ends of the muscle). Also, interconnected muscles apply forces to each other, which leads to additional deformations: for instance, when the lip raisers contract to form a smile, the upper and lower lip muscle is stretched as well, and the lower lip follows the upward movement.

Bulging: contracted muscles become thicker; this is very visible for instance in the orbicularis oris.

Non-homogeneous contraction: muscles are composed of many fibers, which usually contract by an equal or similar amount. But especially during speech, the orbicularis oris is capable of much more articulated motion resulting from different contraction of the fibers in the various segments. Lips can be protruded or retracted, and upper and lower lip are able to move independently. For instance, during the articulation of an /f/, the upper lip is protruded much more than the lower lip.

These observations suggest a much more complex muscle model based on physical properties such as elasticity. However, in the context of real-time animation, the computational overhead is prohibitive. To produce realistic speech animation, we have improved our earlier muscle model [17] in a number of ways to increase the range of expression especially around the mouth. Few additional computations are necessary, so the model is still very efficient. These modifications are:

Straightening: the control polygon of a linear muscle fiber converges to a straight line under tension. This is achieved by simple linear interpolation based on the original and current distances of the muscle end points: when the Euclidean distance between the endpoints exceeds the length of the control polygon segments in relaxed state, the muscle is completely stretched out along the line connecting the endpoints.

Central axis for sphincters: for reproducing protrusion and retraction of the orbicularis oris, a static central point of contraction is not sufficient. Thus, we specify a central *axis* instead, and muscle deformation is now controlled by two parameters: *contraction* specifies the shrinking orthogonally to the axis, and *protrusion* specifies movement along the axis in either direction, so the lips can even be “tucked in”.

Protrusion gradient: to accommodate for the non-homogeneous contraction of the orbicularis oris, we allow the fibers of a sphincter to protrude by different amounts. The outer part of the sphincter protrudes very little since around the mouth this part is constrained by other muscles and attachments to skin tissue, while the inner part takes on the full protrusion value specified, reflecting the freedom of movement at the lips. Between the extremes we linearly interpolate the protrusion value.

Constraint resolution: our original muscle model already handles merging of muscles in a simple manner: the position of the control points in the merged area of two muscles is a weighted average of the individually computed positions of contracted control points. This local mechanism of making muscles “stick together” has been extended by linking the connection points among each other by simplified springs with a rest length and a stiffness. After resolving constraints locally, the changes to the node positions induce changes in the lengths of these springs. We now directly compute new point positions from the distortion of the springs and their stiffness values over a few iterations, thus distributing the original displacements among the connected nodes. The number of iterations depends on the maximum distance of nodes in the connection graph. Typically, three or four iterations are sufficient.

In addition to these general changes to our muscle simulation model, we have split the orbicularis oris into two parts for the upper and lower lip. This allows for independent specification of contraction and protrusion values. Figure 2 shows some postures of the orbicularis oris that can be achieved with our improved model.

5.2 Speech Synchronization

A crucial point in speech synchronization is the provision for *coarticulation*. This term refers to the influence of the surrounding segments of a phoneme onto the vocal tract shape. For instance, the /k/ in ‘coin’ and the /k/ in ‘cow’ are quite distinct. In ‘coin’, the lip rounding for the /ɔi/ does not begin with the ‘o’ but already with the ‘c’, whereas the /*aʊ*/ in ‘cow’ does not affect the shape of the lips for the ‘c’. The influencing phonemes may be as many as seven segments apart [2] and may even be separated by syllable or word boundaries [3]. This interlocking of phonemes blurs the acoustic segment boundaries to such an extent that it is impossible to tell exactly where one segment begins and the previous one ends.

Our speech synchronization [1] is based on the approach of COHEN and MASSARO [7], which has also been used recently for the visual speech component of the talking face in the CSLU toolkit [8]. This toolkit was designed to assist teachers of profoundly deaf children with their daily lessons. For example by watching the artificial head pronounce a sentence, the children can improve their own pronunciation.

The lip synch method in [7] requires as input the transcription of an audio file, where the transcription contains both the phonemes and the corresponding timing information. In our system, we obtain the segmentation of the utterances from the ESPS signal processing package [9] and encode the labeling using the TIMIT database notation [10].

Coarticulation is modeled using dominance functions. They describe the influence of a speech segment on the vocal tract shape over time. In the original linguistic theory [24], each segment has a different dominance over every articulator. The approach in [7] uses facial parameters such as, for instance, lip protrusion to model the articulators. In our physics-based muscle model, we use muscle contraction and protrusion parameters for five decisive muscles around the mouth instead of a direct parameterization. In addition, we set

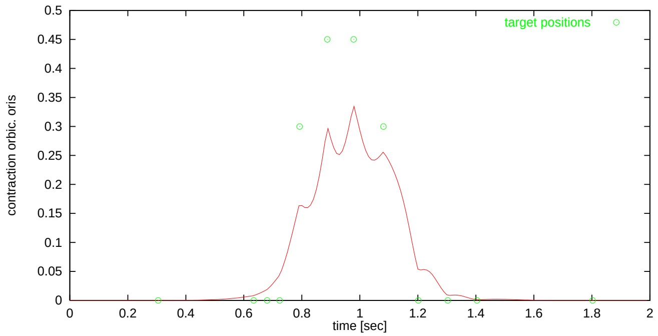


Fig. 4: Resulting function from computing the weighted average of the dominance functions multiplied with the target positions of the contraction of the orbicularis oris for each phoneme of the the utterance "hello world".

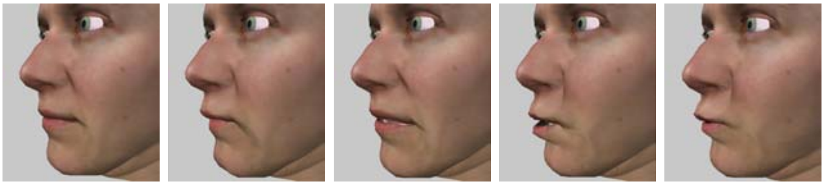


Fig. 5: Snapshots showing mouth articulation. Left to right: neutral pose, /f/, /i/, /b/, /u/. Note how upper and lower lip move independently and are able to not only contract, but protrude or retract.

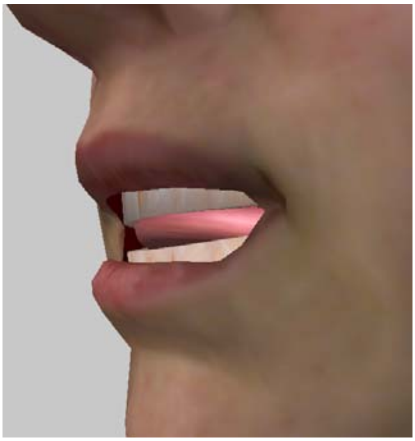


Fig. 6: Snapshot of the mouth forming a /ð/ as it is pronounced in the English word 'the'. The tip of the tongue touches the bottom of the upper teeth.

the jaw rotation angle and the transformation parameters of the tongue. Each of these parameters is controlled by an individual function that is computed as a weighted average of the dominance functions over all segments multiplied with the targets of the according muscle (or jaw rotation) for each phoneme. Figure 4 shows such a function for the contraction value of the orbicularis oris for the utterance "hello world". The target of a muscle for a certain phoneme is given by the contraction and/or protrusion the muscle should take on if the phoneme was pronounced isolatedly. As in natural speech, the targets are hardly ever fully reached. Dominance functions of nearby segments may overlap, leading to an overlapping of the corresponding speech gestures, which in turn leads to coarticulation.

The time available for tightening and relaxation of muscle contraction also determines the amount of the contraction: in fast speech there is more coarticulation than in slow speech. This is reflected by a larger overlap of the dominance functions in fast speech and hence of the speech gestures themselves. By varying the dominance functions, different ways of coarticulation can be simulated.

Figure 5 shows the articulation of some phonemes as produced by our muscle model and associated contraction values. In Figure 6, the position of the tongue during the pronunciation of a /ð/ is depicted.

5.3 Asynchronous Simulation and Rendering

Our facial animation system efficiently exploits dual processor systems by using individual *threads*[3] for the physics-based simulation and the rendering of an animation, see also Figure 1. In our implementation we use the `pthreads` library as an interface to the POSIX thread model.

The *simulation thread* is controlled by muscle parameters from an animation script or from user interaction. An animation script can be automatically generated in a speech analysis preprocessing step (cf. Section 5.2). During the simulation, the Lagrangian equations of motion for a mass-spring system are numerically integrated through time using an explicit forward integration scheme. The structure of the mass-spring system is described in detail in [17]. After a simulation step has been computed, the resulting displacements of the face mesh vertices are stored in a *key frame* entry in a ring buffer. In addition, the current animation parameters of our facial components (see Section 4.1) are stored in the same key frame. The time steps of the simulation need not be constant, but can be adapted to the needs of speech synchronization. To ensure a temporally undistorted rendering of irregularly distributed key frames,

[3]subroutines running asynchronously on different CPU's of a multi-processor machine

the time stamp t_i of each simulation step (measured in wall-clock time) is also stored within the associated key frame f_i.

The *rendering thread* reads and interpolates key frames from the ring buffer and renders the face model according to the interpolated animation and displacement parameters. Depending on the “temporal distance” $t_{i+1} - t_i$ between two successive key frames f_i and f_{i+1}, we typically generate three to ten interpolated frames from the two key frames. The blending factor α_R, which is used to generate the interpolated frame $f_R = \alpha_R f_i + (1 - \alpha_R) f_{i+1}$, is computed from the time stamps of the key frames involved and the rendering time t_R ($t_i \leq t_R \leq t_{i+1}$): $\alpha_R = (t_{i+1} - t_R)/(t_{i+1} - t_i)$. If t_R becomes larger than t_{i+1}, the key frame f_i is discarded and the new key frame f_{i+2} is read from the ring buffer. Now the interpolation takes place between f_{i+1} and f_{i+2}, accordingly. The rendering time t_R is also measured in wall-clock time, but shifted by a certain delay to the simulation time. The amount of the delay depends on the time the simulation thread needs to compute the first two key frames of the animation.

To enhance performance, we do not use any locking mechanism for the ring buffer access. However, we must make sure that neither the rendering thread $\mathcal{R}$ tries to read a key frame that has not yet been written by the simulation thread $\mathcal{S}$, nor that $\mathcal{S}$ overwrites a key frame that has not yet been read by $\mathcal{R}$. This is accomplished by using a first-in-first-out (FIFO) buffer b with mutual access and one additional variable v that is locked by a mutex. Whenever $\mathcal{S}$ has finished writing a new key frame, the index of that key frame (modulo the size of the ring buffer) is stored in the FIFO buffer b. If $\mathcal{R}$ needs to read a new key frame from the ring buffer, the respective index is read from b. This read call blocks if b is empty. On the other hand, $\mathcal{R}$ stores the index of the older key frame that is currently used in the variable v. Before writing to the ring buffer, $\mathcal{S}$ checks the content of v. If the index j of the new key frame is equal to v, writing is delayed until $v > j$.

Our ring buffer access mechanism performs very well on dual processor operating platforms for a ring buffer size of about 10–20 key frames. Neither the simulation thread nor the rendering thread had to wait for ring buffer access. Giving $\mathcal{S}$ a competitive edge of about half the size of the ring buffer minimizes the probability of blocking. In addition, our dynamic rendering approach presented in [16] can be employed within the rendering thread to improve the visual appearance of coarse base meshes especially in the region around the mouth.

5.4 Randomized Animation Parameters

To achieve a more lifelike appearance of the virtual head, we included eye blinking and small random head movements into the animation. To this effect,

activity	T_I	T_A
acquire range scans	15–20	—
acquire photographs	5–10	—
process range scans	250–350	45
prepare mesh for animation	30	1
register textures	30–60	< 5
fit skull & fine-tune muscles	20–45	< 1

Tab. 1: Time spent for diverse preprocessing activities. $\boldsymbol{T_I}$ *denotes the amount of time that was spent interactively, while* $\boldsymbol{T_A}$ *indicates the additional amount of time that went into fully automatic computations. All timings are given in minutes.*

the animation thread generates different scalar-valued noise functions that can be applied to animation parameters. We have used a function producing peaks in a randomized time interval which is applied to eyelid closure. Sine wave functions of different period and with randomized amplitude are applied to head rotation around the x-, y-, and z-axis. These effects add greatly to the realism of the animation by making the head appear much less static.

6 Results and Conclusion

We have presented a system for facial modeling and animation, which aims at the generation of high-quality models and animation with as little effort as possible. Our system includes several tools that facilitate mesh processing, texture registration, and assembling of skin, skull, muscles, and facial components (see Figure 7). Some movies demonstrating the capabilities of our system can be downloaded from the URL `http://www.mpi-sb.mpg.de/resources/FAM/`.

Table 1 shows how much time we typically spent on diverse preprocessing activities on the way from data acquisition to the final animatable head model. The acquisition and processing of range data takes considerable time and effort, mostly due to missing data that can't be obtained by the scanner. In particular, the ears are a problem as well as the inner part of the lips. Although mesh decimation results in a usable approximation of the input data, there is not enough control over the topology of the resulting mesh. This can lead to artifacts in the animation due to asymmetries between the left and right side of the face mesh and misalignment of the triangle edges. We are thus investigating fitting a previously modeled, generic head directly to the range scan data. Subdivision surfaces can be used to achieve adaptive resolution with well-known mesh topology.

Another step in the construction of the head model that introduces some manual work is registration of textures, which we would like to be fully automated. Also, modeling and rendering of hair needs to be included. Though typically no data for the hair-covered parts of the head can be obtained from range scanning devices, we are confident that geometry and texture of the haircut can be extracted from the acquired photographs.

In the context of speech synchronization, we found that the approach of COHEN and MASSARO [7] was straightforward to be adapted to our muscle-based model. Once the parameters of the dominance function and the target contraction parameters of each muscle have been interactively assigned to every phoneme, key frames for an animation synchronized to a labeled speech signal can be generated automatically. No video taping of the speakers and no neural network training is required.

The rendering performance of our system achieves real-time frame rates of about 100 fps (frames per second) on a dual processor Pentium 4 Linux-PC (2x 1.7 GHz) with a GeForce3 graphics board. On a 1 GHz single processor PC, we obtain frame rates of about 30–35 fps.

Acknowledgements

The author would like to thank all of the people involved in the development of the MEDUSA facial modeling and animation system: Kolja Kähler, Hitoshi Yamauchi, Irene Albrecht, Won-Ki Jeong, Jacques Koreman, and Hans-Peter Seidel. Without their effort, a complex system like MEDUSA could neither be developed nor maintained within a reasonable amount of time.

References

[1] Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. Speech Synchronization for Physics-based Facial Animation. In *Proc. WSCG 2002*, pages 9–16, 2002.

[2] A.-P. Benguerel and H. A. Cowan. Coarticulation of Upper Lip Protrusion in French. *Phonetica*, 30:41–55, 1974.

[3] R. A. W. Bladon and A. Al-Bamerni. Coarticulation Resistance in English /l/. *Journal of Phonetics*, 4:137–150, 1976.

[4] Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Computer Graphics (SIGGRAPH '99 Conf. Proc.)*, pages 187–194. ACM SIGGRAPH, August 1999.

[5] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered Construction for Deformable Animated Characters. In *Computer Graphics (SIGGRAPH '89 Conf. Proc.)*, volume 23, pages 243–252. ACM SIGGRAPH, July 1989.

[6] David T. Chen and David Zeltzer. Pump it up: Computer Animation of a Biomechanically Based Model of Muscle using the Finite Element Method. In *Computer Graphics (SIGGRAPH '92 Conf. Proc.)*, volume 26, pages 89–98. ACM SIGGRAPH, July 1992.

[7] Michael M. Cohen and Dominic W. Massaro. Modeling Coarticulation in Synthetic Visual Speech. In *Models and Techniques in Computer Animation*, pages 139–156. Springer–Verlag, 1993.

[8] Ron Cole, Dominic W. Massaro, Jacques de Villiers, Brian Rundle, Khaldoun Shobaki, John Wouters, Michal Cohen, Jonas Beskow, Patrick Stone, Pamela Connors, Alice Tarachow, and Daniel Solcher. New Tools for Interactive Speech and Language Training:

Using Animated Conversational Agents in the Classrooms of Profoundly Deaf Children. In *Proc. ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education*, London, UK, April 1999.

[9] Entropic Research Laboratory, Inc., Sheraton House, Castle Park, CBX 0AX Cambridge, UK. *ESPS Manual*, 1993.

[10] J. S. Garofolo. *Getting Started with the DARPA TIMIT CD-ROM: An Acoustic Phonetic Continuous Speech Database*. National Institute of Standards and Technology (NIST), Gaitherburgh, MD, 1988.

[11] Taro Goto, Marc Escher, Christian Zanardi, and Nadia Magnenat-Thalmann. MPEG-4 based Animation with Face Feature Tracking. In *Proc. Eurographics Workshop on Computer Animation and Simulation '99*, pages 89–98, 1999.

[12] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Frédéric Pighin. Making Faces. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 55–66. ACM SIGGRAPH, July 1998.

[13] David R. Hill, Andrew Pearce, and Brian Wyvill. Animating Speech: An Automated Approach using Speech Synthesised by Rules. *The Visual Computer*, 3(5):277–289, March 1988.

[14] Horace H. S. Ip and C. S. Chan. Script-Based Facial Gesture and Speech Animation Using a NURBS Based Face Model. *Computers & Graphics*, 20(6):881–891, November 1996.

[15] ISO/IEC. Overview of the MPEG-4 Standard. `http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm`, July 2000.

[16] Kolja Kähler, Jörg Haber, and Hans-Peter Seidel. Dynamic Refinement of Deformable Triangle Meshes for Rendering. In *Proc. Computer Graphics International 2001 (CGI 2001)*, pages 285–290, July 2001.

[17] Kolja Kähler, Jörg Haber, and Hans-Peter Seidel. Geometry-based Muscle Modeling for Facial Animation. In *Proc. Graphics Interface 2001*, pages 37–46, June 2001.

[18] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann. SMILE: A Multilayered Facial Animation System. In *Proc. IFIP WG 5.10, Tokyo, Japan*, pages 189–198, 1991.

[19] Leif Kobbelt, Mario Botsch, Kolja Kähler, Christian Rössl, Robert Schneider, and Jens Vorsatz. Geometric Modeling Based on Polygonal Meshes. In *Eurographics 2000 Tutorial Notes*. Blackwell Publishers, 2000.

[20] Rolf M. Koch, Markus H. Groß, and Albert A. Bosshard. Emotion Editing using Finite Elements. In *Computer Graphics Forum (Proc. Eurographics '98)*, volume 17, pages C295–C302, September 1998.

[21] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Constructing Physics-based Facial Models of Individuals. In *Proc. Graphics Interface '93*, pages 1–8, May 1993.

[22] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic Modeling for Facial Animations. In *Computer Graphics (SIGGRAPH '95 Conf. Proc.)*, pages 55–62. ACM SIGGRAPH, August 1995.

[23] John P. Lewis and Frederic I. Parke. Automated Lip-Synch and Speech Synthesis for Character Animation. In John M. Carroll and Peter P. Tanner, editors, *Proceedings of Human Factors in Computing Systems and Graphics Interface '87*, pages 143–147, April 1987.

[24] Anders Löfqvist. Speech as Audible Gestures. In W. J. Hardcastle and A. Marchal, editors, *Speech Production and Speech Modelling*, pages 289–322. Kluwer Academic Publishers, 1990.

[25] Monique Nahas, Herve Huitric, and Michel Saintourens. Animation of a B-Spline Figure. *The Visual Computer*, 3(5):272–276, March 1988.

[26] Frederic I. Parke. *A Parametric Model for Human Faces*. PhD thesis, University of Utah, Salt Lake City, UT, December 1974.

[27] Frederic I. Parke. Parameterized Models for Facial Animation. *IEEE Computer Graphics and Applications*, 2(9):61–68, November 1982.

[28] Andrew Pearce, Brian Wyvill, Geoff Wyvill, and David Hill. Speech and Expression: A Computer Solution to Face Animation. In *Proceedings of the Graphics Interface '86*, pages 136–140, May 1986.

[29] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing Realistic Facial Expressions from Photographs. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 75–84. ACM SIGGRAPH, July 1998.

[30] Stephen M. Platt and Norman I. Badler. Animating Facial Expressions. In *Computer Graphics (SIGGRAPH '81 Conf. Proc.)*, volume 15, pages 245–252. ACM SIGGRAPH, August 1981.

[31] Claudio Rocchini, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Multiple Textures Stitching and Blending on 3D Objects. In *Rendering Techniques '99 (Proc. 10th Eurographics Workshop on Rendering)*, pages 119–130, 1999.

[32] Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-Based Modeling of the Human Musculature. In *Computer Graphics (SIGGRAPH '97 Conf. Proc.)*, pages 163–172. ACM SIGGRAPH, August 1997.

[33] Demetri Terzopoulos and Keith Waters. Physically-based Facial Modelling, Analysis, and Animation. *Journal of Visualization and Computer Animation*, 1(2):73–80, December 1990.

[34] Roger Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, June 1986.

[35] Keith Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. In *Computer Graphics (SIGGRAPH '87 Conf. Proc.)*, volume 21, pages 17–24. ACM SIGGRAPH, July 1987.

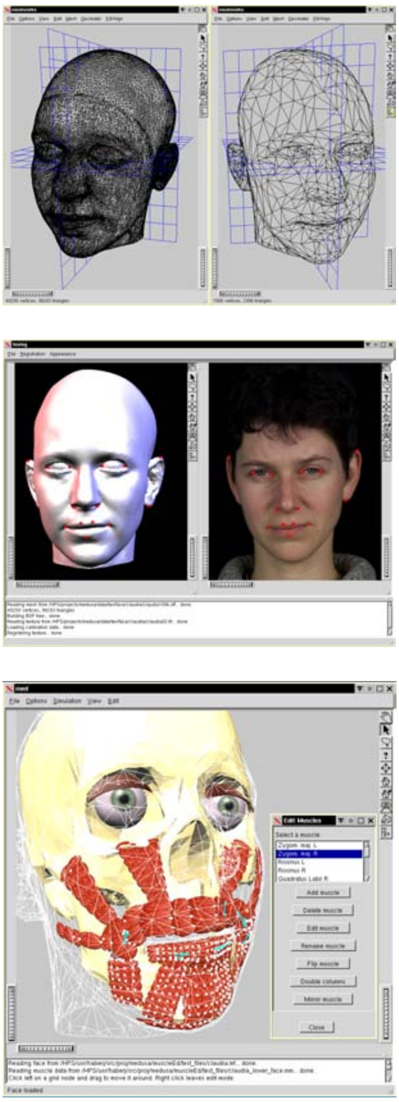
[36] Keith Waters and Joe Frisbie. A Coordinated Muscle Model for Speech Animation. In *Proc. Graphics Interface '95*, pages 163–170, May 1995.

[37] Jane Wilhelms and Allen Van Gelder. Anatomically Based Modeling. In *Computer Graphics (SIGGRAPH '97 Conf. Proc.)*, pages 173–180. ACM SIGGRAPH, August 1997.

[38] Lance Williams. Performance-Driven Facial Animation. In *Computer Graphics (SIGGRAPH '90 Conf. Proc.)*, volume 24, pages 235–242. ACM SIGGRAPH, August 1990.

[39] Yin Wu, Prem Kalra, Laurent Moccozet, and Nadia Magnenat-Thalmann. Simulating Wrinkles and Skin Aging. *The Visual Computer*, 15(4):183–198, 1999.

[40] Yin Wu, Nadia Magnenat-Thalmann, and Daniel Thalmann. A Plastic-Visco-Elastic Model for Wrinkles in Facial Animation and Skin Aging. In *Proc. Pacific Graphics '94*, pages 201–214, August 1994.



```
Reading mesh from /HPS/projects/medusa/data/[illegible] ... done
43250 vertices, 86163 triangles
Building BSP tree... done
Reading texture from /HPS/projects/medusa/data/[illegible] ... done
Loading calibration data... done
Registering texture... done
```

```
Reading face from /HPS/usr/haberj/src/proj/medusa/muscleEd/test_files/claudia.tef... done.
Reading muscle data from /HPS/usr/haberj/src/proj/medusa/muscleEd/test_files/claudia_lower_face.mm... done.
Click left on a grid node and drag to move it around. Right click leaves edit mode.
Face loaded
```

Fig. 7: Tools of our facial modeling and animation system. Top to bottom: a) Two views of our mesh decimation and alignment tool. b) Graphical user interface (GUI) of our texture registration tool. c) GUI of our muscle editor.

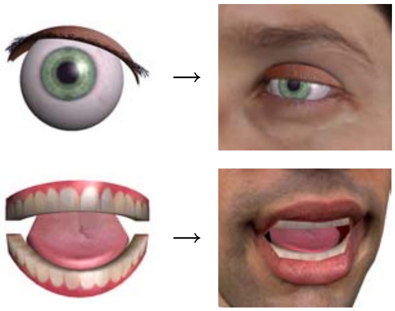


Fig. 8: Generic models of eyes, teeth, and tongue (left) are fitted into individual face meshes (right).

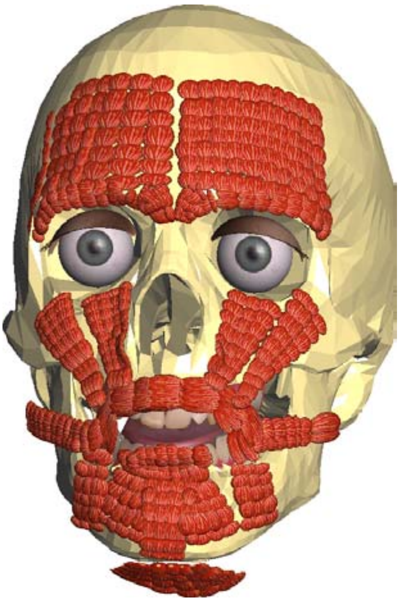


Fig. 9: Reference skull with parts: eyes, teeth, tongue, muscles.

Nominiert für den Heinz-Billing-Preis 2001

Dynamik turbulenter Gaswolken und die Entstehung von Sternen

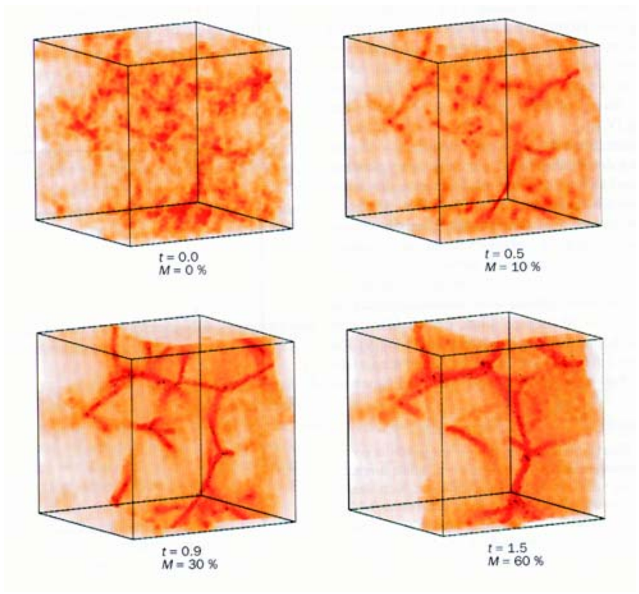
Andreas Burkert
Max-Planck-Institut für Astronomie Heidelberg

1 Einführung

Die Entstehung von Sternen gehört zu den fundamentalen und bisher ungelösten Problemen der modernen Astrophysik. Sie spielt eine bedeutende Rolle in vielen Bereichen der Astronomie, etwa bei der Entstehung von Galaxien, der galaktischen Nukleosynthese oder der Entstehung von Planetensystemen.

Trotz ihrer Bedeutung ist die Sternentstehung bisher kaum verstanden. Dies liegt an der hohen Komplexität des Problems. Junge Sternhaufen bilden sich ausschließlich in dichten kalten Gasgebieten aus molekularem Wasserstoff mit Radien von einigen hundert Lichtjahren und Massen von einer Million Sonnenmassen. Neuere Beobachtungen von verbreiterten Spektrallinien zeigen, dass die Gasbewegung in diesen *Molekülwolken* turbulent ist, mit Geschwindigkeiten die wesentlich größer sind als die Schallgeschwindigkeit. Filamentartige und klumpige Unterstrukturen werden auf allen auflösbaren Skalen beobachtet, wie zum Beispiel im 30 Doradus Nebel (Abb. 1).

Die turbulente, ungerichtete Bewegung des teilweise magnetisierten Gases verhindert den großskaligen gravitativen Kollaps der Gaswolken. Überschreitet jedoch in einem ruhigeren Gebiet die Masse einen kritischen Wert, so überwiegt die Gravitation den turbulenten Druck und der Teilbereich kolla-



biert. Bereits vorhandene Dichtefluktuationen verstärken sich, ziehen weitere Materie aus der Umgebung an und wachsen um mehr als 15 Größenordnungen, bis sich nach wenigen Millionen Jahren ein Sternhaufen gebildet hat.

Die neu enstandenen Sterne heizen und ionisieren durch ihre Strahlung das umgebende Wolkengas. Dieser Rückkopplungsprozess zerstört schließlich die Molekülwolke. Beobachtungen (siehe Abb. 2) zeigen jedoch auch, dass die Kompression des Gases in der Ionisationsfront zu weiteren Verdichtungen und damit auch zu weiterer Sternentstehung führen könnte.

2 Sternentstehung, eine Herausforderung für die numerische Astrophysik

Die Entstehung und Entwicklung turbulenter Molekülwolken, ihre Kondensation in Sterne und die anschließende Zerstörung des Sternentstehungsgebietes ist ein hoch komplexes, nichtlineares, 3-dimensionales magneto-hydrodynamisches Problem, das man im Detail nur mit Hilfe von numerischen Simulationen verstehen kann. Noch vor wenigen Jahren reichte die Rechenleistung selbst der schnellsten Rechenanlagen nicht aus, um die Dynamik des Wolkengases in 3 Dimensionen mit ausreichender numerischer Auflösung studieren zu können. Die rasante Entwicklung im Rechnerbereich und die Entwicklung neuer numerischer Methoden hat diese Situation in den letzten Jahren grund-

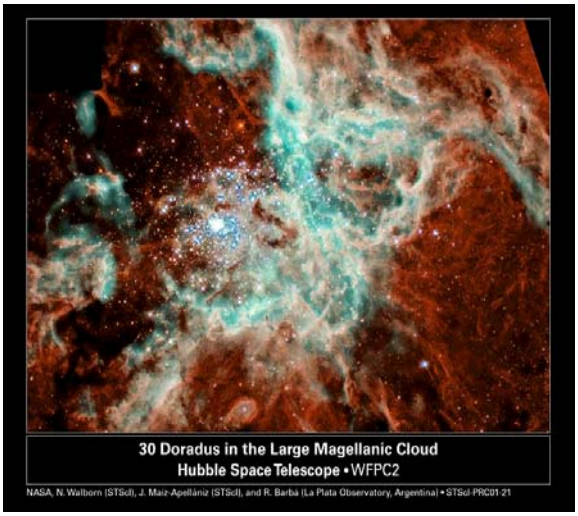

30 Doradus in the Large Magellanic Cloud
Hubble Space Telescope • WFPC2
NASA, N. Walborn (STScI), J. Maíz-Apellániz (STScI), and R. Barbá (La Plata Observatory, Argentina) • STScI-PRC01-21

Abb. 1: HST-Aufnahme eines jungen Sternhaufens im 30 Doradus Nebel

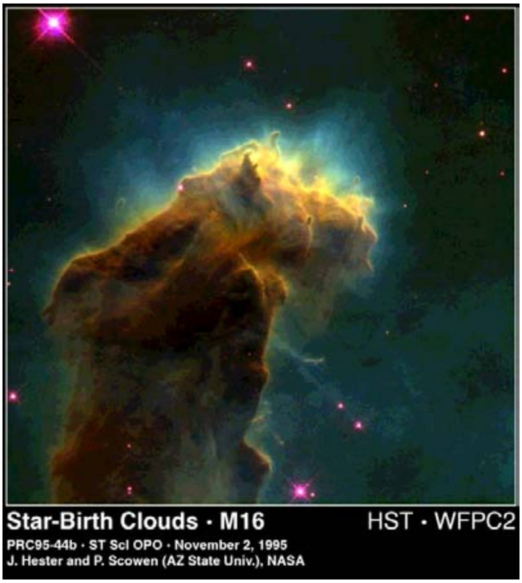

Star-Birth Clouds · M16
HST · WFPC2
PRC95-44b · ST Scl OPO · November 2, 1995
J. Hester and P. Scowen (AZ State Univ.), NASA

Abb. 2: HST-Aufnahme eines durch einen nahen Sternhaufen teilweise ionisierten, klumpigen Wolkengebietes. In den dichten, fingerartigen Knoten nahe der hellen Ionisationsfront könnten sich weitere Sterne bilden.

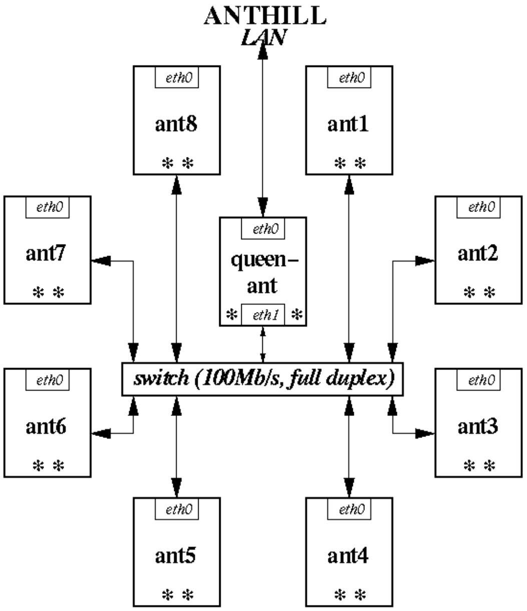


Abb. 3: Aufbau unseres PC-Clusters

legend verbessert. Ich hatte mir daher mit meiner 1995 am MPIA gegründeten Theoriegruppe das Ziel gesetzt, die wesentlichen Aspekte des Sternentstehungsprozesses detailliert numerisch zu simulieren und zu verstehen. Der gesamte Vorgang der Sternentstehung läßt sich bis heute nicht in einem einzigen Rechengang verfolgen. Wir haben daher die Entwicklung in verschiedene Phasen unterteilt, die getrennt und mit verschiedenen numerischen Methoden untersucht werden. Hierzu mußten zuerst schnelle Programme zur Integration der magneto-hydrodynamischen (MHD) Gleichungen entwickelt werden.

2.1 Der parallele MHD-Code ZEUS-MP

In den letzten 10 Jahren wurde am Laboratory for Computational Astrophysics (LCA) in Urbana-Champaign, Illinois ein frei verfügbarer, vektorisierter MHD-Gittercode entwickelt namens *Zeus-3D* (Stone & Norman 1992). Wir haben dieses Programm übernommen und mit Hilfe des LCA für die Nutzung auf unserem PC-Cluster (*Anthill*) parallelisiert.

Abbildung 3 zeigt schematisch den Aufbau von *Anthill*. 8 PCs (ants) mit jeweils 2 Prozessoren (Sterne) sind über eine 100 Mbit/s Ethernetverbindung verbunden. Ein zentraler Server (queenant) wird zur Kompilierung der Codes,

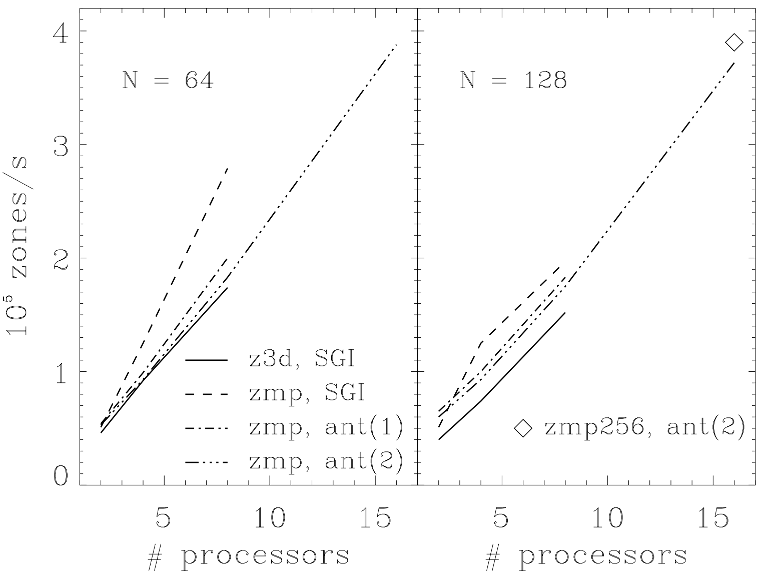


Abb. 4: Rechengeschwindigkeit als Funktion der Zahl der Prozessoren für ein magneto-hydrodynamisches Testproblem mit N^3 Gitterzellen. ant(1) steht für den PC-Cluster mit der Nutzung von einem Prozessor pro Knoten. Bei ant(2) werden beide Prozessoren jedes Knotens genutzt, was jedoch keine nennenswerte Erhöhung der Rechengeschwindigkeit liefert.

für den Filetransfer und für die Datenverwaltung genutzt. Der für den parallelen Betrieb auf dem PC-Cluster umgeschriebene *Zeus-MP*-Code skaliert linear mit der Zahl der Prozessoren (siehe Abb. 4). Die durchgezogene Line in Abb. 4 zeigt die Rechengeschwindigkeit von *Zeus-3D* auf einer SGI Origin 2000. Diese Version parallelisiert gut, allerdings nur bis zu 8 Prozessoren (loop-parallelization). Die gestrichelten und strichpunktierten Linien in Abb. 4 zeigen die Rechengeschwindigkeit von *Zeus-MP* auf der Origin und auf Anthill. Diese Version erreicht auf dem PC-Cluster nur 60% der Geschwindigkeit der Origin 2000, allerdings für ein zehntel des Preises.

2.2 Der GRAPE-SPH-TREE-Code WINE

Sobald ein Gebiet innerhalb einer Molekülwolke aufgrund der Gravitationskraft kollabiert nimmt die Gasdichte um viele Größenordnungen zu. Dieses Stadium kann mit dem gitterbasierten Verfahren von *Zeus-MP* nicht aufgelöst werden. Wir mußten daher ein weiteres Hydrodynamikprogramm entwickeln und haben uns für ”smoothed particle hydrodynamics” (SPH) entschieden. SPH (Lucy 1977; Gingold & Monaghan 1977) ist ein auf Teilchen basierendes Verfahren zur Lösung der hydrodynamischen Gleichungen. Im Gegensatz zu Gitterverfahren mit einer begrenzten räumlichen Auflösung stellen die SPH-Teilchen ein frei bewegliches Gitter dar, mit dem die Gasdynamik selbst in gravitativen Gebieten mit hohen Dichtekontrasten problemlos beschrieben werden kann, solange ein von uns entwickeltes Auflösungskriterium (Bate & Burkert 1997) erfüllt ist. Nach diesem Kriterium benötigt man typischerweise eine Auflösung von 3×10^5 bis 5×10^5 SPH-Teilchen, um den gravitativen Kollaps von Gaswolken untersuchen zu können. Diese Bedingung führte mit den verfügbaren SPH-Programmen zu extrem langen Rechenzeiten. Wir haben daher das TREE-SPH-GRAPE-Programm namens *WINE* entwickelt.

Die bei weitem zeitaufwendigste Berechnung im SPH-Algorithmus ist die Bestimmung der Gravitationskräfte und der nächsten Nachbarn. In Japan werden seit etwa 10 Jahren spezielle Hardware-Boards, sogenannte *GRAPEs* für GRAvity PipelinE (Sugimoto et al. 1990) entwickelt. Das *GRAPE*-Board berechnet nächste Nachbarn und Kräfte durch eine speziell dafür entwickelte Pipeline mit einer Geschwindigkeit pro Board von 4.8 Gflops (GRAPE-3AF), 38.5 Gflops (GRAPE-5), bzw 1 Tflop für die neue GRAPE-6 Generation. Die Entwickler haben dafür 1999 den Gordon Bell Preis gewonnen.

Wir haben einen Cluster von GRAPE-3 und GRAPE-5 Boards aufgebaut, der mit dem Hauptrechner, einem 500 MHz 21264 Alpha Prozessor verbunden ist. Da die Nachbarn und die Gravitationskräfte auf den GRAPE-Boards durch direkte Summation berechnet werden skaliert die Rechenzeit proportional zum Quadrat der Teilchenzahl N. Trotz ihrer Schnelligkeit sind GRAPE-Boards daher für Rechnungen mit großen Teilchenzahlen von $N =$

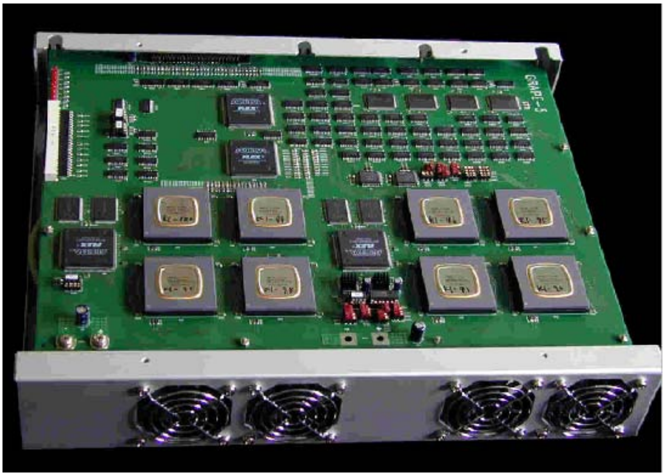


Abb. 5: Ein GRAPE-5 Board

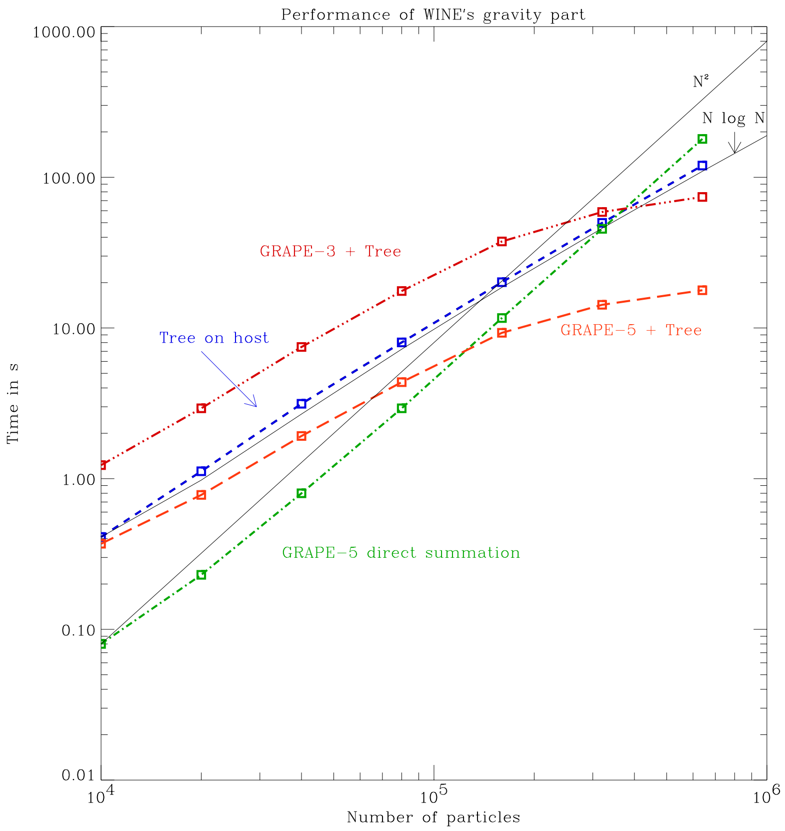


Abb. 6: Rechengeschwindigkeit für die Kraftberechnung als Funktion der Teilchenzahl.

$5 \times 10^5 - 10^6$ langsam. Wie die grüne Kurve in Abbildung 6 zeigt liegt die Rechenzeit für die Kraftberechnung bei diesen Teilchenzahlen mit einem GRAPE-5 im Bereich von einigen hundert Sekunden pro Iteration.

Wir mußten daher ein weiteres effizientes Verfahren einbauen, den sog. TREE-Algorithmus (Barnes & Hut, 1986). Hier werden Teilchen zu Gruppen zusammengefaßt und die gravitative Wechselwirkung der einzelnen Gruppen berechnet. Die blaue Kurve in Abbildung 6 zeigt die Rechenzeit des TREE-Codes, die proportional zu N log(N) skaliert. Bei großen Teilchenzahlen ist der TREE-Algorithmus demnach effizienter als GRAPE. Der von uns entwickelte *WINE* Code nutzt nun beide Verfahren effizient aus. Mit Hilfe des

TREE-Algorithmus werden zunächst Teilchengruppen gebildet, deren gravitative Wechselwirkung dann mit GRAPE berechnet wird. In ähnlicher Weise kann man die nächsten Nachbarn bestimmen. Wie die rote Kurve in Abbildung 6 zeigt lässt sich der Rechenaufwand mit diesem Trick bei $N > 10^5$ für GRAPE5+TREE um mehr als einen Faktor 10 reduzieren und die Rechenzeit nimmt mit N wesentlich langsamer zu als mit TREE oder GRAPE.

2.3 Sterne und ionisierender Strahlung

Der gravitative Kollaps der Gaswolke führt schließlich zur Bildung eines Sternhaufens, der die Umgebung durch ionisierende Strahlung aufheizt und damit die weitere Sternentstehung entscheidend beeinflußt. Um diese Phase im Sternentstehungsprozess zu beschreiben ersetzt unser SPH-Code kondensierte Gasgebiete durch akkretierende, massereiche Sternteilchen (Bate & Burkert 1997; Klessen, Burkert & Bate 1998). Mit einem neuen, von uns entwickelten Verfahren, dem sog. *SPHI* (Kessel & Burkert 2000, 2001) wird nun die Heizung der umgebenden Gasteilchen durch die Sterne berechnet. Hierzu wird die Strahlungstransportgleichung mit Hilfe des SPH-Schemas numerisch integriert und die Ionisations- und Rekombinationsrate des Gases an jedem Ort bestimmt. Das *SPHI*-verfahren skaliert proportional zu N log(N).

Mit *Zeus-MP* und *WINE/SPHI* können wir nun erstmals die Entwicklung magnetisierter turbulenter Gaswolken, ihren gravitativen Kollaps, die Entstehung eines Sternhaufens, sowie die anschließende Heizung des Wolkengases detailliert untersuchen. In den folgenden Abschnitten möchte ich einige erste interessante und teilweise unerwartete Ergebnisse zusammenfassen, die wichtige neue Erkenntnisse über den Sternentstehungsprozess liefern.

3 Dynamik turbulenter Gaswolken

Die Abbildung 7 zeigt die 3-dimensionale Struktur einer turbulenten, magnetisierten Gaswolke, die mit *ZEUS-MP* auf einem Gitter mit 256^3 Gitterzellen simuliert wurde (Mac Low et al. 1998, Burkert & Bodenheimer 2000, Burkert 2001). In diesem Modell wurde die Turbulenz auf großen Skalen getrieben um eine konstante Machzahl M der Gasgeschwindigkeit von M=4 zu erreichen. Obwohl kinetische Energie nur auf großen Skalen hinzugefügt wird überträgt sie sich durch Alfvénwellen auf kleinere Skalen und es entwickelt sich ein komplexes Dichte-und Geschwindigkeitsfeld mit zahlreichen klumpigen Verdichtungen auf allen Skalen.

Die Abbildung 8 vergleicht die Oberflächendichteverteilung der simulierten Wolken mit den Beobachtungen (Burkert & Mac Low 2001). Für Machzahlen von 4 bis 10 finden wir eine gute Übereinstimmung. Unsere Rechnungen scheinen die Dynamik magnetisierter, turbulenter Gaswolken gut beschreiben zu können.

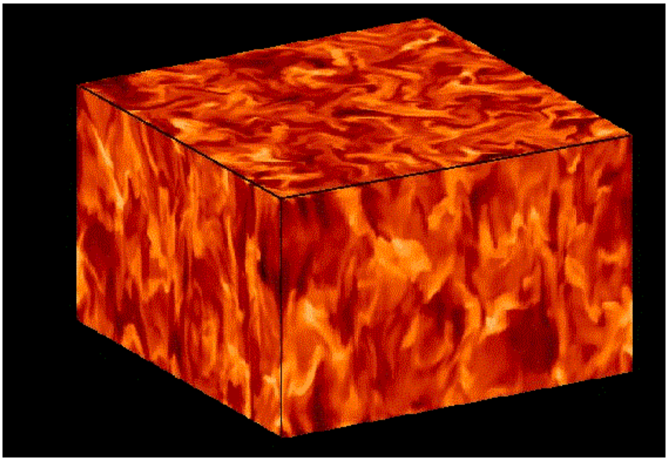


Abb. 7: Dreidimensionale Computersimulation einer turbulenten Molekülwolke. Gebiete hoher Gasdichte sind hell dargestellt. Bei dieser Rechnung wurde ein ursprünglich vertikal angeordnetes, starkes Magnetfeld angenommen.

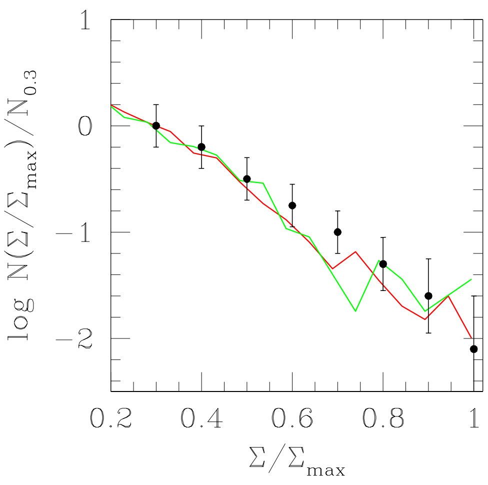


Abb. 8: Die beobachtete Oberflächendichteverteilung (Punkte mit Fehlerbalken) stimmt gut mit numerischen Modellen von auf großen Skalen getriebener Überschallturbulenz überein (rote und grüne Linien).

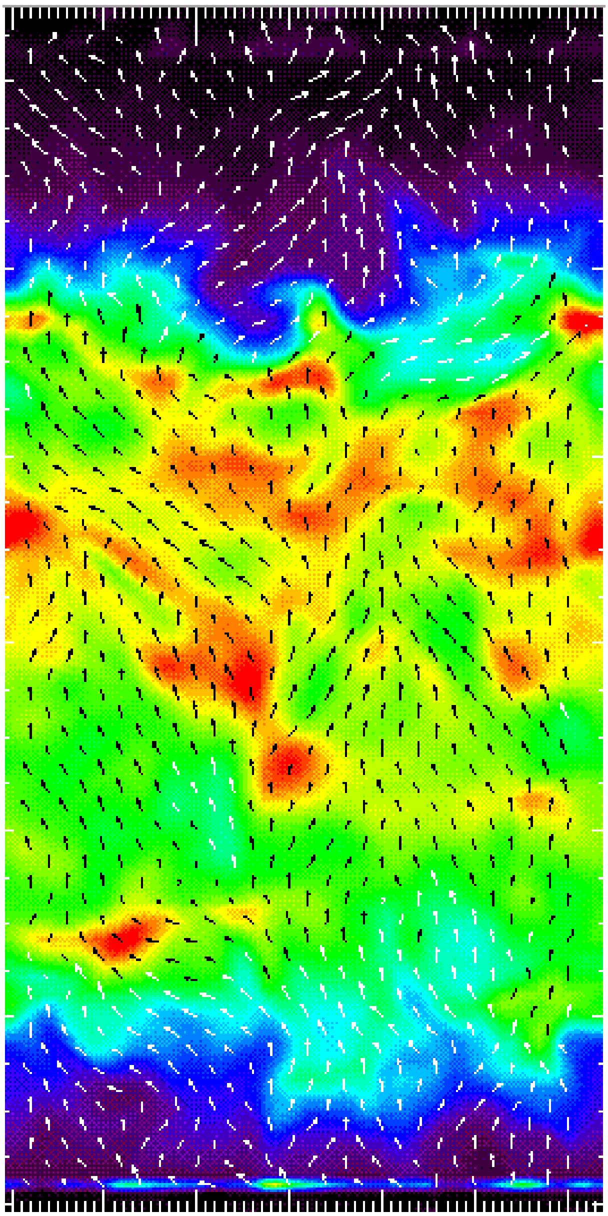


Abb. 9: Dichtestruktur einer turbulenten, von außen, d.h. am oberen und unteren Rand durch Alfvénwellen getriebenen Gaswolke. Die Pfeile zeigen die Richtung des Magnetfeldes. Gebiete des umgebenden interstellaren Mediums mit geringer Dichte sind dunkel. Im dichten inneren Bereich führen die am Rand induzierten Alfvénwellen zu irregulären Dichte- und Geschwindigkeitsfluktuationen.

Ein interessante Frage ist die Quelle der Turbulenz in Molekülwolken. Unsere Rechnungen ohne Treiber zeigen, dass im Gegensatz zu der bisherigen Vorstellung Magnetfelder nicht in der Lage sind, die Dissipation der turbulenten kinetische Energie zu verhindern. In allen Fällen, auch in dem mit starkem Magnetfeld, verebbten die turbulenten Bewegungen lange vor Ablauf der abgeschätzten Lebensdauer der Molekülwolken von einigen zehn Millionen Jahren. Um die Turbulenz aufrechtzuerhalten und dem Kollaps der Wolke effektiv entgegenzuwirken, muß dem Gas demnach mehr oder weniger kontinuierlich kinetische Energie zugeführt werden. Bisher ist der Treibmechanismus nicht bekannt. Die Sternentstehung selbst kommt nicht in Frage, wie unsere Rechnungen zeigen. Zur Zeit untersuchen wir die Möglichkeit, dass Molekülwolken von außen durch Ankopplung an das umgebende, turbulente interstellare Medium getrieben werden. Abbildung 9 zeigt die Struktur einer solchen von außen durch Alfvénwellen getriebenen Wolke (Heitsch & Burkert 2001). Weitere Untersuchungen sind notwendig um dieses interessante Problem besser zu verstehen.

4 Die Entstehung eines Sternhaufens

Sinkt die turbulente Energie des Wolkengases in einem Gebiet unter einen kritischen Wert, so überwiegt die Gravitationskraft. Das Gebiet kollabiert und kondensiert in Sterne. Wir untersuchen diese Phase mit dem *WINE*-Code. Die typische Entwicklung einer gravitativ instabilen, turbulenten Wolke wird in Abbildung 10 gezeigt (Klessen & Burkert 1998, 2000, 2001). Das Verhalten des Gases erweist sich als äußerst kompliziert. Es wird durch das Wechselspiel der dissipierenden, turbulenten Restenergie, dem Gasdruck und der Gravitation bestimmt. Kleine Dichtefluktuationen werden zunächst durch den Gasdruck zerstört, während sich gleichzeitig großräumige Fluktuationen zu länglichen Filamenten und Knoten zusammenziehen. Die ersten Sterne entstehen in diesen Filamenten. Sie wandern später in die Knoten, wo sich kleinere Sternhaufen bilden, die schließlich zu einem großen Sternhaufen verschmelzen. Unsere Simulationen zeigen, daß die Sternentstehung im einzelnen unvorhersehbar ist und sich am ehesten mit Methoden der deterministischen Chaostheorie beschreiben läßt. Die sich letztendlich einstellende Massenverteilung der Sterne wird demnach durch die anfangs vorgegebene Dichte-und Geschwindigkeitsverteilung nicht eindeutig festgelegt. Möglicherweise läßt sie sich aber mit einer statistischen Methode ermitteln.

Das gemeinsame Ergebnis mehrerer Simulationsläufe ist ein breites Massenspektrum, dessen Maximum bei etwa 0.2 Sonnenmassen liegt (Abbildung 11) und das erstaunlich gut mit den Beobachtungen der mittleren Massenverteilung von Sternen in der Milchstraße übereinstimmt.

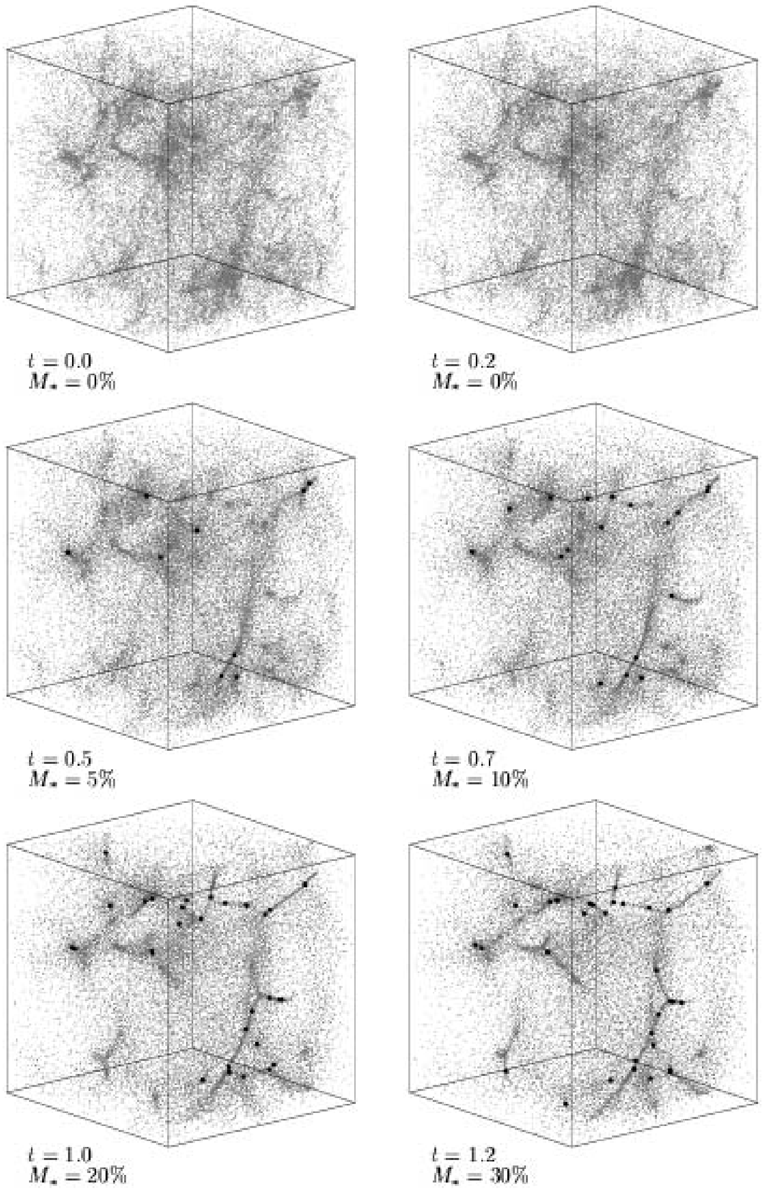


Abb. 10: Zeitliche Entwicklung und Fragmentation einer turbulenten Gaswolke und Bildung eines Sternhaufens. Die Zeit t ist in Einheiten der Freifallzeit angegeben, die Prozentzahl bezeichnet den Massenanteil der Sterne.

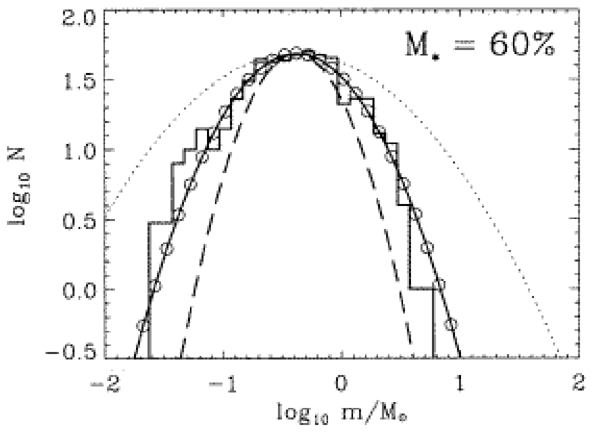


Abb. 11: Die Massenverteilung der Sterne in der Simulation (Histogramm), nachdem 60% des Gases in Sternen kondensiert ist. Die Beobachtungen liefern eine Verteilung, die durch die durchgezogene Linie und die offenen Kreise gekennzeichnet ist.

5 Heizungsprozesse und die Zerstörung von Molekülwolken

Die im letzten Abschnitt vorgestellten Rechnungen geben einen detaillierten Einblick in die Komplexität der Sternentstehung. Die jungen Sterne werden das umgebende Wolkengas durch ihre ionisierende Strahlung aufheizen (siehe Abb. 2). Mit **SPHI** sind wir erstmals in der Lage diese Spätphase der Sternentstehung genauer zu untersuchen (Geyer & Burkert 2001). Abbildung 12 zeigt die Zerstörung einer unserer klumpigen Gaswolke in der sich ein zentraler Sternhaufen gebildet hat. Man erkennt wie die nach außen laufende, heiße Gasblase eine dichte Schale von molekularem Wasserstoff aufsammelt. In dieser Rechnung wurde die Wolke in einer Million Jahren zerstört und damit die Sternentstehungphase beendet.

Es stellt sich nun die Frage, ob durch die Verdichtung des Gases in der Schale weitere Sternentstehung induziert werden kann. Dazu haben wir einen kleinen Bereich der Front mit hoher Auflösung untersucht (Kessel & Burkert 2001). Die Abbildung 13 zeigt die Entwicklung einer anfangs stabilen, sphärischen Dichtestörung, eingebettet in zunächst ungestörtes, kaltes Wolkengas. Sobald die Umgebung durch die Sternstrahlung aufgeheizt wird sinkt die Umgebungsdichte da das Gas aufgrund des hohen Druckes aus dem Gebiet getrieben wird. Der erhöhte Umgebungsdruck komprimiert nun die Dichtestörung. Hydrodynamische Instabilitäten führen an der Oberfläche zu fingerartigen Filamenten und Knoten, wie sie auch beobachtet werden (siehe Abb. 2). Diese Objekte sind jedoch nicht gravitativ instabil sondern verdampfen nach kurzer

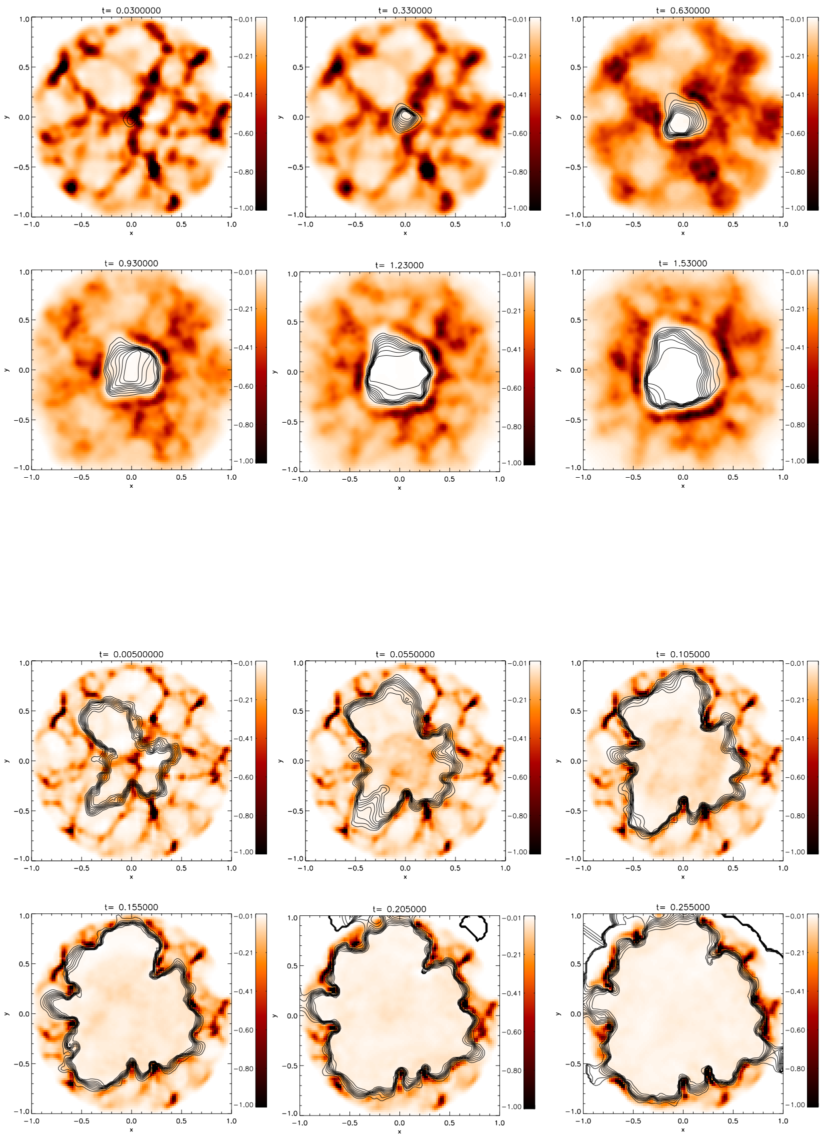


Abb. 12: Zerstörung einer turbulenten Gaswolke durch einen zentralen Sternhaufen, der die Umgebung aufheizt.

Zeit. In dieser Rechnung kollabiert der innere Teil der Fluktuation schließlich aufgrund ihrer Eigengravitation und bildet zwei Sterne, die als helle, gelbe Punkte am Ende der Simulation in Abbildung 14 zu sehen sind. Sternentstehung kann demnach tatsächlich während der Wolkenzerstörung induziert werden. Weitere Rechnungen werden zeigen, welche Masseverteilung diese Sternkomponente hat und wie häufig dieser Prozess stattfindet.

6 Zusammenfassung

Unsere Rechnungen haben gezeigt, dass wir heute in der Lage sind die Dynamik turbulenter Gaswolken, ihre Kondensation in Sterne, sowie ihre anschließende Zerstörung detailliert zu simulieren. Die Dichtestruktur der Modellwolken stimmt gut mit den Beobachtungen überein und die Massenverteilung der simulierten jungen Sternhaufen kann die beobachtete Massenverteilung der Sterne in der Milchstraße erklären. Viele Fragen sind jedoch noch offen, wie zum Beispiel die Natur des Treibers der Turbulenz in Molekülwolken, die Entstehung kalter Molekülwolken durch die Kühlung und Verdichtung des heißen interstellaren Mediums (Burkert & Lin 2000) oder die Entstehung von Doppelsternsystemen (Burkert & Bodenheimer 1993, 1996). Die Aussichten sind gut, dass diese Probleme in den kommenden Jahren mit Hilfe von numerische Rechnungen gelöst werden und wir erstmals ein ein konsistentes Modell der Sternentstehung entwickeln können.

Literatur

[1] Barnes, J.E. & Hut, P. (1986): Nature **324**, 446
[2] Bate, M.R. & Burkert, A. (1997): MNRAS **288**, 1060
[3] Burkert, A., Bodenheimer, P. (1993): MNRAS **264**, 798
[4] Burkert, A., Bodenheimer, P. (1996): MNRAS **280**, 1190
[5] Burkert, A., Lin, D. (2000): ApJ **537**, 270
[6] Burkert, A., Bodenheimer, P. (2000): ApJ **543**, 822
[7] Burkert, A. (2001): In *Star Formation and the Origin of Field Populations*, ed. E.Grebel & W. Brandner (ASP Conf. Ser: San Francisco), im Druck
[8] Burkert, A. & Mac Low, M.M. (2001): ApJ, eingereicht
[9] Geyer, M. & Burkert, A. (2001): MNRAS **323**, 988
[10] Gingold, R.A. & Monaghan, J.J. (1982): Journal of Comp. Phys. **46**, 429
[11] Heitsch, F. & Burkert, A. (2001): ApJ, in Vorbereitung
[12] Kessel, O. & Burkert, A. (2000): MNRAS **315**, 713
[13] Kessel, O. & Burkert, A. (2001): MNRAS, in Vorbereitung
[14] Klessen, R.S., Burkert, A. & Bate, M.R. (1998): ApJ **501**, L205
[15] Klessen, R.S., Burkert, A.(2000): ApJ **542**, 95
[16] Klessen, R.S., Burkert, A.(2001): ApJ **549**, 386
[17] Lucy, L.B. (1977): AJ **82**, 1013
[18] Mac Low, M.M., Klessen, R., Burkert, A. & Smith, M.D. (1998): Phys. Rev. Letters **80**, 2754
[19] Stone, J.M. & Norman, M.L. (1992): ApJS **80**, 791
[20] Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T. & Umemura, M. (1990): Nature **345**, 33

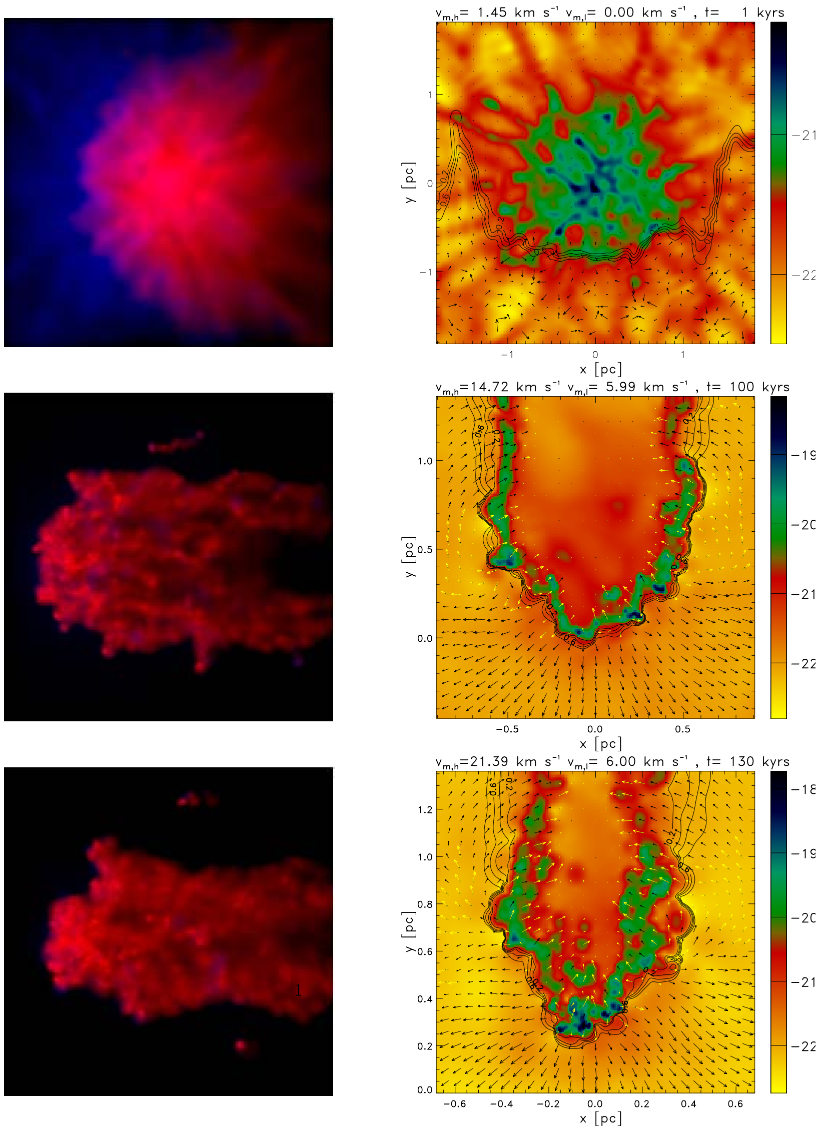


Abb. 13: Frühe Phase der Kompression einer Dichtestörung durch ein externes, ionisierendes Strahlungsfeld. Links: Dreidimensionale Projektion der Dichteverteilung. Die Strahlung fällt von links ein. Ionisiertes Gas ist blau, neutrales Gas ist rot dargestellt. Rechts: Schnitt durch das Modell. Die Dichteverteilung in [g/cm^3] ist farbig kodiert. Konturen zeigen den Ionisationsgrad, Pfeile das Geschwindigkeitsfeld.

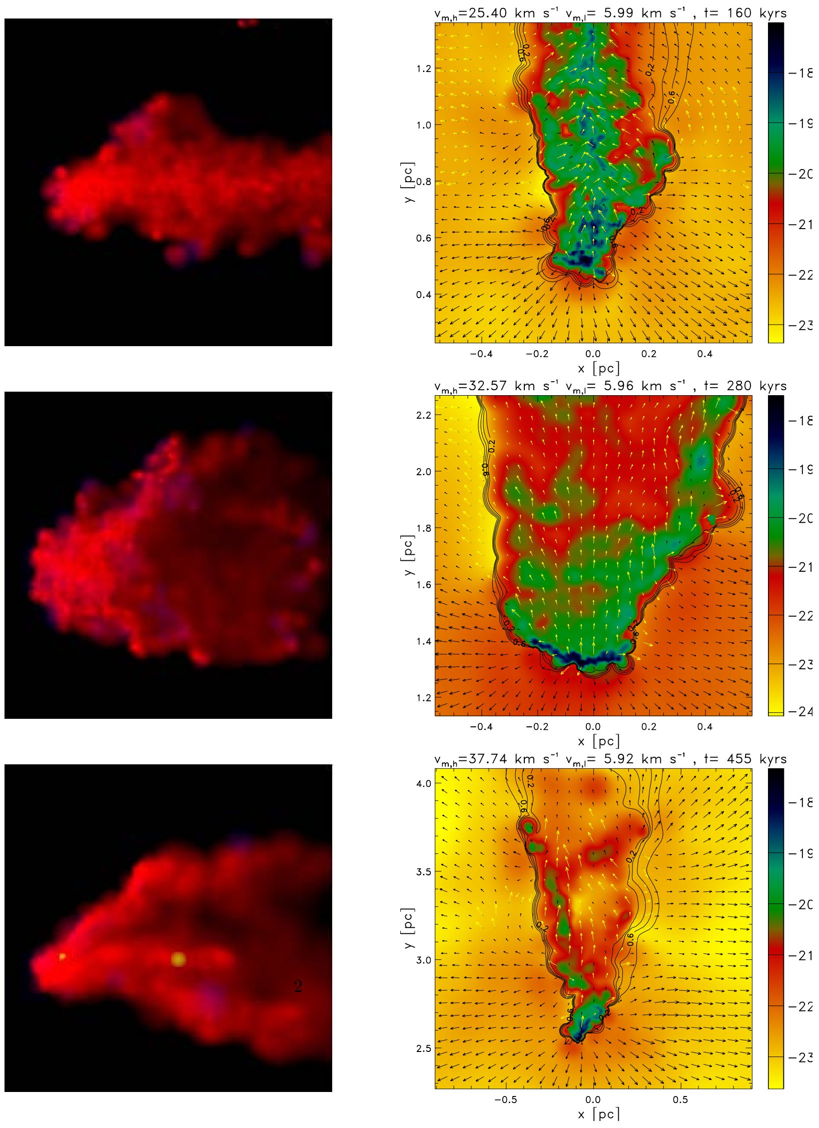


Abb. 14: Späte Phase der Kompression einer Dichtestörung durch ein ionisierendes Strahlungsfeld. Die zwei jungen Sterne sind als gelbe Punkte dargestellt.

NEST: An Environment for Neural Systems Simulations

Markus Diesmann
Dept. of Nonlinear Dynamics,
Max-Planck Inst. für Strömungsforschung, Göttingen

Marc-Oliver Gewaltig
Future Technology Research,
Honda R&D Europe (Deutschland) GmbH, Offenbach

Abstract

NEST is a framework for simulating large, structured neuronal systems. It is designed to investigate the functional behavior of neuronal systems in the context of their anatomical, morphological, and electrophysiological properties. NEST aims at large networks, while maintaining an appropriate degree of biological detail. This is achieved by combining a broad range of abstraction levels in a single network simulation. Great biological detail is then maintained only at the points of interest, while the rest of the system can be modeled by more abstract components. Here, we describe the conception of NEST and illustrate its key features. We demonstrate that software design and organizational aspects were of equal importance for the success of the project.

1 Introduction

Neuroscience is one of the fields in life science which have recently received increased attention as can be seen from the fact that the 1990's were proclaimed the *Decade of the Brain* by the US Government.

Because of the immense difficulties involved in observing neuronal activity in an intact brain, experimental data exists mostly for single or small numbers of neurons, or for very large populations ($\gg 10^6$ neurons). As a consequence, scientists increasingly employ theoretical and computational methods in order to assess the dynamical properties of individual and populations of neurons and to come up with testable predictions.

Computational neuroscience is, thus, a fast growing research field, dedicated to the investigation of the nervous system with the help of computer simulations. It is important to note, however, that the type of neuronal network model discussed here differs considerably from so-called artificial neural networks (ANN), which have become an important branch of engineering and computer science.

In Computational Neuroscience, simulations are used to investigate models of the nervous system at functional or process levels. Consequently, a lot of effort has been put into developing appropriate simulation tools and techniques, and a plethora of simulation software, specialized for the single neuron or small sized networks is available (e.g. [8, 19]).

Recently, however, there has been growing interest in large scale simulations, involving some 10^4 neurons while maintaining an acceptable degree of biological detail. Thus, there is need for simulation software, possibly parallel, which supports such simulations in a flexible way.

Here, we describe the Neural Simulation Technology (NEST) initiative, a collaborative effort to develop an open simulation framework for biologically realistic neuronal networks. The system is distinguished by at least three features. First, it aims at large structured networks of heterogeneous, biologically realistic elements at different description levels. Second, NEST employs an iterative incremental development strategy, maintaining a running system at any time (see e.g. [9, 7]). Third, the software is developed as a collaborative effort of several research groups. Since NEST is a research tool, it has to continuously adapt to the ever changing demands of the researchers who are using it. Thus, the design and development process of the software is aware of the environment in which the system has to be implemented. The mechanisms supporting the collaboration are an integral part of NEST.

The system has been continuously developed and successfully applied in research over the last few years [11].

2 Basic concepts

In this section, we introduce the conceptual framework on which NEST is based.

A computer simulation is the attempt to investigate the properties of a (complex) physical system by evaluating the behavior of a simplified model of the system on a computer. We will call such models *computer models*. In contrast to the analytical or numerical examination of mathematical models, computer models often contain algorithmic components. These make a mathematical treatment in a closed form at least difficult, if not infeasible.

A NEST simulation tries to maintain a close correspondence to an electrophysiological experiment. There are two conceptual domains which have to be mapped to the software level:

1. the description of the neuronal system, that is the network components and the network,
2. the description of the experimental setup and protocol.

2.1 Neuronal systems

It is a well established idea that information processing in the central nervous system relies on the electric discharges (*action potentials* or *spikes*) of a certain class of cells, the *neurons*, which interact at specialized points of contacts, the *synapses*. A generated spike arrives at one or more target neurons after a delay of a few milliseconds and causes a small change in the neurons' membrane potentials (so-called *post-synaptic potential*).

It is often assumed that if the superposition of sufficiently many post-synaptic potentials at a target neuron reaches a threshold value, the cell will itself generate a spike, however, details vary considerably between different neuron models. More models agree on the concept that the time-course of an action potential can be neglected, and thus the interaction between neurons can be described by the exchange of *point events*.

The standard approach to neural modeling is to consider neurons as the basic network components and describe the network in terms of the neurons, their positions and their projections. This approach can be called bottom up, since the network is built from its basic constituents upwards.

However, the brain is a heterogeneously structured system, composed of numerous areas and nuclei which are distinguished by their anatomical, physiological, and functional properties. Moreover, each of these parts has a specific architecture which is to a large extent genetically determined. The structure of the brain, therefore, conveys information beyond the mere connectivity (i.e. synaptic connections) of its constituents.

Thus, NEST adopts a top-down approach. Networks are regarded as hierarchical structures which may be represented by trees. A network is described in terms of abstract components which may either be atomic or compound.

Atomic network elements may range from sub-neuronal compartments to large populations of neurons, and may, therefore, differ considerably in complexity and degree of abstraction.

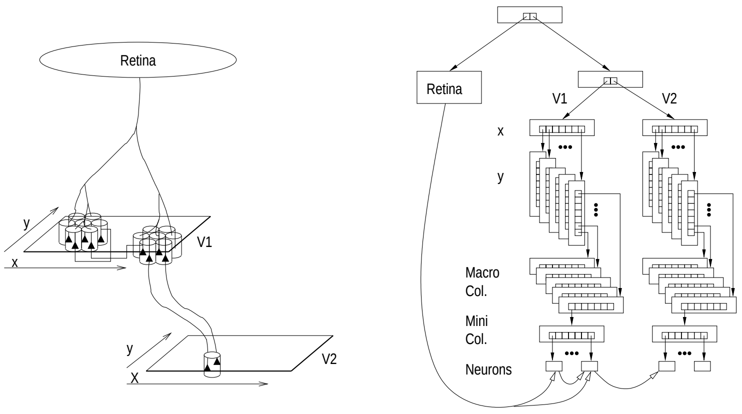


Fig. 1: Example of a structured network model (left) and how it is represented (right).

Compound network elements are defined in terms of other elements and their mutual connections. They may be nested and may, thus, be used to define concepts like areas, macro-columns, and mini-columns. Structural model concepts can therefore be mapped to the simulated network.

Fig. 1 (left) shows an example of a hierarchically structured model (see e.g. [22]). The model network consists of several sub-structures, described at different levels of abstraction: a *retina* and two model brain areas, *V1* and *V2*. The retina sends input to a first visual area V1 which consists of topographically organized macro-columns. These, in turn, consist of orientation columns. Finally, orientation columns consist of model neurons. The other area, V2, has a similar structure.

The right panel shows how the model may be represented in the framework presented here. The tree reflects the *semantic structure* of the model. The *synaptic connectivity* of the network is not shown. Atomic network elements are represented by separate objects as *leaves* of the network tree. Examples are the individual neurons of V1 and V2, or the model retina which represents a large population of neurons in terms of a single atomic element. The model areas V1 and V2 and the various columns are examples for compound elements.

2.2 Virtual experiments

In an electrophysiological experiment, a number of different devices are used to stimulate and observe the neuronal system. In our approach, the measurement and stimulation process is an explicit part of the model. Accordingly,

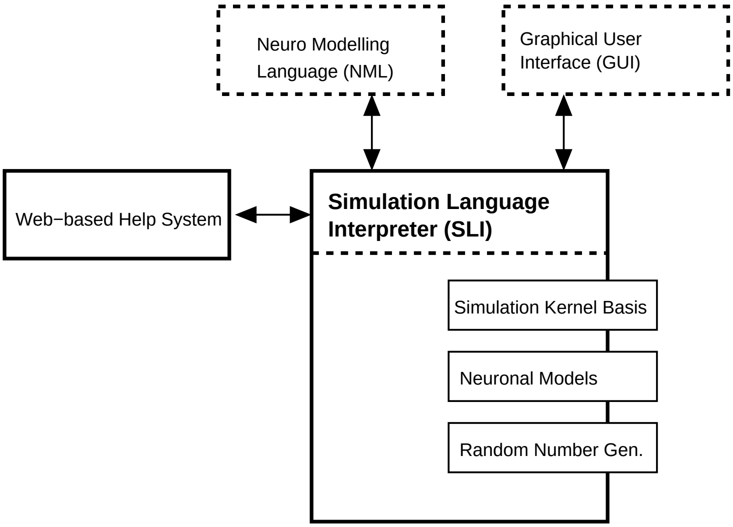

Neuro Modelling Language (NML)
Graphical User Interface (GUI)
Web−based Help System
Simulation Language Interpreter (SLI)
Simulation Kernel Basis
Neuronal Models
Random Number Gen.

Fig. 2: Structure of the simulation system. Its main parts are a simulation kernel, simulation language interpreter (SLI) and some auxiliary modules. The simulation language interpreter integrates all parts and acts as interface to the user. Additional modules extend functionality.

devices are explicitly mapped to the simulation in order to arrive at *virtual experiments*.

3 Structure of the software

At the top level, NEST consists of a number of independent modules (Fig. 2), with each module contributing some piece of functionality. NEST is written in an object-oriented style (C++). An abstract base class defines a core interface which has to be implemented by each module.

The various modules are combined by a module loader, which is part of a simulation language interpreter (SLI). The interpreter is not only the primary interface to the user, but also provides a set of basic data structures which are used to exchange data between modules.
The simulation language interpreter is a simple stack machine, optimized for the fast processing of high-level commands and objects. It is discussed in detail in section 6.

The second important component of NEST is the simulation kernel. It implements all functionalities to define and simulate large structured networks of biologically realistic neural networks. It provides

1. base classes for neuronal models,
2. derived implementations of important models,
3. a network driver class to manage the temporal update of all simulation elements,
4. a SLI module to provide high-level access to the kernel, and
5. a SLI module to provide a high-level interface to the available models.

Items 1 through 3 are combined in a single C++ library which may also be used without the simulation language interpreter. The simulation kernel is discussed in detail in section 5.

In addition to the simulation kernel, there are a number of other modules which provide important functionality. Some modules solve well-defined mathematical problems like differential equations, some perform high-level operations on arrays and lists or provide random number generators. Other modules implement interfaces with the host operating system or on-line help facilities.

The modular architecture of NEST is an essential ingredient to the aforementioned incremental/iterative development model.

4 Structure of the project

Apart from the technical consideration of the last section, the structure of the project is strongly shaped by the context in which the software needs to be developed and maintained. Important constraints come from the human resources available to the project, the expected life time of individual components, conditions for software development in a scientific environment, and funding.

4.1 Constraints

Software which is used as a research tool usually never reaches a steady state. Scientific progress constantly opens up new and unexpected questions which may require improvement and probably partial redesign of software tools. Sometimes a new generation of hardware enables the investigation of a new class of problems (see section 11). In any case, the research tool should be flexible enough to be easily adapted to the new demands of the researcher.

The long time scale of software projects like NEST usually conflicts with two other time scales. First, researchers stay in a laboratory for only 2-5 years. This encourages researchers to write software specifically for their immediate purpose at the expense of reusability. Thus, this software and know-how is lost to the group when the researcher leaves. Second, the time scale on which a particular hardware platform becomes obsolete has dropped to a few years.

The same is true for parts of the software environment, like graphical user interfaces or even operating systems.

4.2 Consequences

These considerations shaped the structure of our simulation tool from the very beginning and resulted in a number of design decisions:

Platform independence The software should not rely on specific hardware or a specific software manufacturer. We achieve this by using open standards where possible, never relying on proprietary software components. In addition the GNU developer tools [32] enable us to design the compile/build process in a platform independent way.

Implementation language An object oriented programming language is used to allow a clear separation of concepts. It helps to hide implementation details, to enhance reusability of code, and helps new users to understand existing code.

Although at the time the project was started (1994) there was no finished standard available, we chose C++ as implementation language. The combination of object oriented programming and static type checking turned out to be an ideal choice for simulation software where the time needed by a simulation run severely limits the feasible research topics.

Cascaded interfaces Changes and extensions of the software are routinely required. To minimize the effort needed to arrive at the desired functionality, NEST provides a cascade of interfaces. The idea is that simple improvements should be simple to implement with little internal knowledge. However, substantial improvements are also possible. Both are supported by programming interfaces at the level of the simulation kernel and the interpreter.

Collaborative effort The large investment required to develop the infrastructure of NEST led us to the conclusion that simulation software should be developed in collaboration with several researchers with the additional benefit that the different interests of researchers automatically tests the generality of new concepts. Moreover, it reduces the risk that the software becomes unusable when one researcher leaves the team. Fig. 3 indicates the research groups which currently participate in the development of NEST.

Open Source All software tools on which NEST depends are freely available under the GNU general public license (GPL). An important insight from

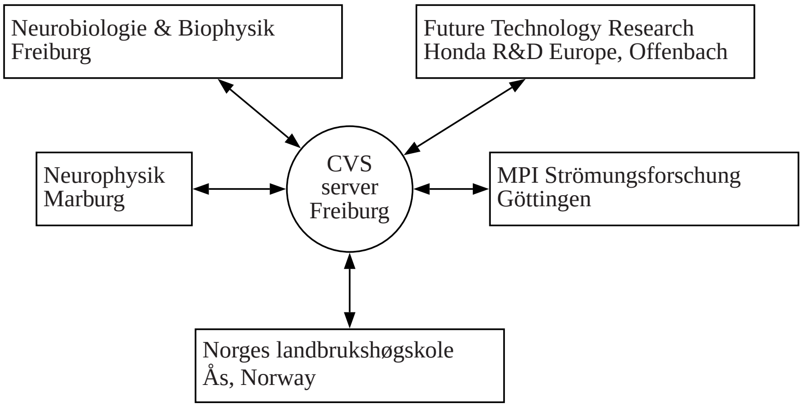


Fig. 3: Distributed development using a common source base. Boxes indicate the locations at which the software is currently developed and used. The circle indicates a server from which the members of the project can update their sources at any time and submit their changes and additions. The system performs automatic version control and detects conflicts between changes made by different users (CVS, [6]).

Open Source projects is that their characteristic rapid speed of progress and the high quality of the software are achieved by developing the software in a distributed team of developers. Thus, distributions of NEST will also be released under the GPL.

Iterative/Incremental development This development model is supported by two principles. First, each researcher implements those parts which he needs for his research. Second, every change and every contribution is immediately submitted to all others. Because each change made by one developer becomes incorporated in the simulation software used by others, errors and unwanted side effects are quickly detected and resolved. Moreover recompilation in different software environments and with different compilers rapidly uncovers non-standard or insecure code.

If one change breaks the simulation project of others and it is not easily possible to resolve the problem, it is always possible to revert to an older version. Thus, the rapid spread of new code between different sites does not affect usability of the software in (scientific) production.

We use the concurrent version system CVS to coordinate this process (see Fig. 3). Using CVS effectively initiates a continuous *review process* which improves the quality of the code and reduces the time taken for errors to be detected and removed. The motivating effect of this policy cannot be overestimated. Each developer is motivated to produce high quality code because he is aware that his contribution is immediately reviewed by others. In our experience most problems are solved within 1 to 2 days by the exchange of a e-mails and cooperative work on the sources.

Documentation Because of the distributed development and the growing number of developers, an efficient mechanism is required to write and distribute documentation. The measures we have taken are described in section 7.

5 The simulation kernel

5.1 Overview

NEST uses an object-oriented approach. The system to be simulated is conceptually broken down into *components* which are then represented by *classes* at the C++ level. As outlined in section 2, neuronal systems are composed of objects like neurons, synapses, columnar circuits, or whole cortical areas. In a NEST simulation, all these components are called *elements*. Elements share a common interface, but may differ considerably in their purpose, functionality, or their level of abstraction. Different element types may co-exist in the same network. This design also supports the implementation of element types at different levels of abstraction, ranging from sub-neuronal components (e.g. compartments) to supra-neuronal elements (e.g. populations of neurons).

A simulation is then built from the set of pre-defined components which are dynamically (during runtime) combined and configured to represent the neuronal model network. While the components are hard coded in C++, their configuration is usually not. Configuration generally takes place at the level of the simulation language SLI.

The simulation kernel has two main parts. The first part is a set of abstract base classes which provide the starting points for neuronal elements at different abstraction levels. The second part is the simulation driver, the administrative center of the simulation kernel.

5.2 Simulation driver

The simulation driver has four main tasks:

1. administration of the network structure,
2. administration of network elements,
3. organizing the temporal update of each element, and
4. orchestrating the communication between elements.

Networks and elements As mentioned above, network elements fall into two categories: atomic elements and compound elements. The most important compound element is the sub-network. It is used to group arbitrary elements. Since sub-networks may be nested, a network can be thought of as a tree structure with one root sub-network at its base.
The concept of nest-able sub-networks allows the modeler to group functionally related neurons into circuits which may themselves be grouped into higher order circuits. This conceptual grouping introduces a hierarchical structure to the network.

The simulation driver controls the root of this network tree. To the user the network tree is comparable to a directory tree in a file system: sub-networks correspond to *directories* and atomic elements to *files*. The driver provides functions to operate on the network tree similar to the UNIX commands `cd`, `ls`, and `mkdir`, with the difference that network elements are numbered, rather than named. If the user enters a sub-network, the driver will perform any subsequent operation local to this sub-network. If the user creates an element, it will be placed in the current sub-network. Note that GENESIS [8] uses this approach to navigate through the compartments of a model neuron.

Element update After the neuronal model system is configured, the driver is responsible for running the simulation for a specified amount of simulation time. It also controls the global clock of the simulation.

Simulation time proceeds on an evenly spaced grid $t_0, t_1, \ldots, t_n$ with

$$t_i := t_0 + i \cdot h$$

where h is the *simulation step size*.

During each time slice, the state of the network is updated by updating each network element. The actual computation performed during update is encapsulated in the respective element class. Different elements may possess very different internal dynamics and may still live in one network. All the driver has to know is that all elements have a common interface which is used during update.

Thus, during each time-slice, the driver iterates over all network elements and calls a specific member function which updates the element's state. After all elements have been updated, the central clock is advanced. This is repeated until the desired simulation time has elapsed.

5.3 Network elements

Computational neuroscience is faced with the problem that there is no standard model neuron; nor is there agreement at which scale neuronal systems

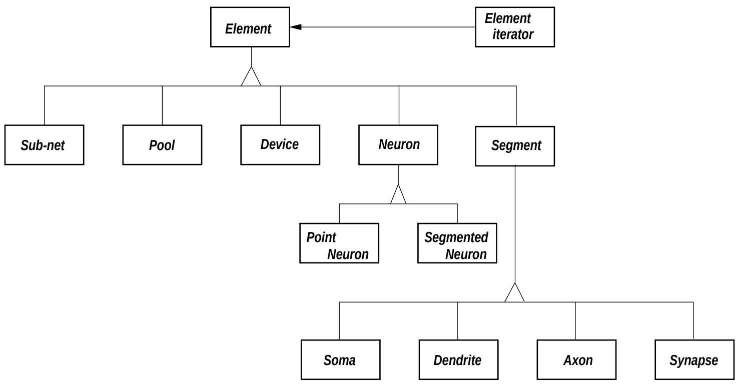


Fig. 4: Base class hierarchy for network elements. All elements are derived from a common base class, the abstract network element. Derived elements add new properties, and inherit the properties of their ancestors. There are elements at the sub- and supra-neuronal level. The distinction between point neurons and segmented neurons maintains the possibility of efficiently simulating "simple" neuron models. Measurement and manipulation devices inherit from the common base class `Device`*. See [16] for notation.*

should be investigated. This is reflected in the class hierarchy of network elements which are currently used in NEST.

Element hierarchy The range of possible abstractions is covered by deriving several abstract classes from the element base class, each class representing a different range of possible abstractions. Figure 4 shows a simplified excerpt of the element hierarchy.

Configuration interface The element base class defines the minimal interface that each element has to implement. However, neuron models have different parameters and it is not trivial to define a generic interface which is applicable to all implemented neurons and which can be used by the interpreter to access an element's internal parameters.

One option is that for each element there exists a separate function, taking exactly those parameters required by the element. This leaves the task of remembering the number of parameters and what is worse the correct ordering of the parameters to the user. Moreover, specific functions have to be added to the interpreter whenever a new model class is introduced. Another option is to provide access to all thinkable parameters by separate member functions of base class for network element introducing all the problems and ugliness of a "fat interface" (see discussion in [30]).

The solution to the problem is to use named or keyword parameters [14].

This also opens the possibility to use meaningful names like “Resistance” or “Capacity” for the parameters. This technique is used by a number of interpreted languages where functions can have a large number of arguments, many of them typically having default values. In order to implement a named-parameter interface, the simulation kernel uses the dictionaries (section 6.1) of the SLI interpreter.

Now an element only has to provide two interface functions: one function which takes a dictionary as argument and changes the element’s parameters accordingly; and one function which returns a dictionary with the element’s current parameter settings.

From the interpreter, a single command can extract a dictionary from an element. The user can now operate on this dictionary. A second SLI command transmits the dictionary back to the network object.

5.4 Communication

Elements in a network are able to exchange information. While simple neurons may just exchange point events (spikes), more powerful element types also need to exchange more complex information. NEST provides a generic event mechanism which allows elements to exchange arbitrarily complex objects on demand. In order to ensure causality, all interactions must have a minimal delay of one simulation step.

A connection between two neurons usually involves a synapse. A synapse determines the biophysical properties of a connection. For example, it determines how strongly an incoming spike affects the target neuron.

Spike events As described above, a large class of neuron models communicate by exchanging spikes. During update, a neuron model combines all incoming spikes and determines whether it should itself generate a spike. When a spike is emitted, it is simultaneously sent to all target neurons. The emission of spikes by a single neuron is sparse (in each time slice the probability of spike generation is $\ll 1$). Therefore it is efficient to place the dynamics of synapses on the post-synaptic side of the connection and to let the pre-synaptic neuron notify the post-synaptic site whenever a pre-synaptic spike occurs.

Time driven vs. event driven simulation The fact that in a NEST simulation, neurons communicate by discrete events appears to be ideally suited to a paradigm which is called *event driven simulation* [15]. In such a paradigm, all events carry a stamp with the time when the event should arrive at the target. A scheduler collects all events and queues them until they have to be delivered. Consequently, it is possible to drive the simulation time according to the time-stamps of the delivered messages. If all messages with stamps $t \leq T$ have been handled, the simulation time may be advanced to time T.

In contexts where the evaluation of each element is expensive compared to the costs of communication, e.g. if each element sends its messages to only a few targets, event driven strategies offer great potential for optimizations.

However, neuronal networks usually consist of a very large number of relatively simple elements with high connectivity (about 10^4 targets per neuron), where each spike event is sent to all targets. In this case an event driven strategy leads to an immense overhead, because a single spike results in some 10,000 event objects which have to be allocated, queued, sorted, delivered and deallocated. Thus, a large number of undelivered events accumulate in the queue.

By contrast, a time driven strategy is useful if the communication is expensive compared to the update of the element. In this case, it is possible to deliver events as they are sent and queue the information where there is enough information to perform optimizations: at the target. Moreover, as the event information is still available at the sender, it is also possible to send an event reference rather than objects. Thus, unnecessary replication of events is not needed.

6 The simulation language interpreter

The simulation language has five basic tasks:

1. to describe the structure of the neural system and the setup of the required manipulation and probing devices,
2. to describe the protocol of the virtual experiment,
3. to provide support for pre- and post-processing of data,
4. to provide a safe user interface with the simulation kernel for interactive sessions, and
5. to provide the interface between kernel and external programs.

This section is concerned with the general properties of the interpreter. An example for a complete simulation script can be found in section 8. We decided to implement a stack machine [5] which interprets a language very similar to PostScript [4]. Such a language is easy to implement and allows us to manipulate high-level objects. At the same time the language is fast and simple enough to serve as the target language for machine generated code. Thus, the simulation language interpreter (SLI) can be used as a virtual machine (section 3). The interpreter required to execute PostScript and how it operates is described in the excellent “Red Book” [4]. Closely following the literature for the implementation of the interpreter has minimized our development time and reduced the risk of of design errors. Several extensions with respect to the PostScript machine were needed to meet our requirements. The most relevant extensions are

– the sizes of arrays, procedures, and strings are dynamic. At any time appending, inserting, or deleting elements can alter the size of a container object. Preallocating memory to increase the performance, e.g. if the size of an array is successively increased by appending elements, is possible.
– optional type checking and operator overloading.

Whereas the basic commands for manipulation of the stacks are the same in SLI and in PostScript, obviously a different set of commands is required for neural simulations than for the description of graphics. The commands introduced to control the simulation kernel are mainly constructors for different network elements, commands for navigating in the network and commands for inspecting and configuring network elements.

Although PostScript provides some commands to manipulate arrays, a much richer set of commands was desired for setting up neural simulations and data pre- and post-processing . The high-level mathematics programming language Mathematica [34] has a consistent set of commands for data manipulation. The style of Mathematica operations fits well to the capabilities of our stack machine. The combination of heterogeneous arrays as universal containers and pure (or anonymous) functions [14] is present in Mathematica as well as in SLI and promotes the definition of powerful operators and a functional programming style. We therefore designed the commands for data manipulation according to the Mathematica model. Doing so, we could again reduce the development time and the risk of design flaws. Many of the current users have several years of experience with Mathematica and can therefore start to work with SLI without having to learn new commands for the same thing. It is one of the principles of NEST that, if later generalization is not severely impaired, basically only those features should be implemented which are immediately used. In particular, all mathematical operations are carried out numerically, although the stack machine itself fully supports the manipulation of symbols.

6.1 Language properties

In the scope of the present paper it is neither necessary nor possible to discuss the capabilities of SLI in detail. However, to illustrate the resulting language we briefly summarize a few useful properties and commands

Heterogeneous arrays In particular an array element can itself be an array of arbitrary length. Thus multidimensional data structures of arbitrary shape are supported (with lazy evaluation, copy-on-write).

Dictionaries A data type called dictionary represents a special form of an associative array [14, 20]; array elements are accessed by symbols. Dictionaries have multiple uses in SLI. In fact, similar to PostScript, in SLI all variables and named functions are managed by dictionaries (a named function is nothing else but a pure function assigned to a symbol). SLI dictionaries do not take strings, but symbols as keys for dictionary lookup. When a SLI expression is parsed, the terminal (character) representation of a symbol is transformed into an efficient internal representation which allows fast dictionary lookup. Dictionaries are also used to communicate with the simulation kernel.

Functions as first class values As in some other interpreted languages, it is possible to assemble an expression from a string at runtime and execute it by invoking the full scanner, parser, stack machine sequence. However, in SLI a much more efficient alternative exists. Similar to some other languages (e.g. LISP, see [3] and references therein) SLI procedures are first class values [14]. Therefore it is possible to assemble a procedure with the same efficiency and with the same operators as one assembles an array.

Error handling SLI supports exception handling [14] which allows the user to protect a certain piece of code by a construct similar to the try-catch block of C++. If an exception occurs, the execution stack (call stack) is appropriately unwinded and the innermost exception handler is called. This allows the developer of a simulation to handle errors at locations in the program where sufficient information is still available, and cleanly separate error handling code from the code describing the simulation. The SLI interpreter itself uses the exception mechanism such that application code can be well integrated with existing code implemented in SLI or C++. In normal operation, the interpreter executes code protected by a default exception handler that prints some diagnostics for otherwise uncaught exceptions.

6.2 Levels of extendability

The incremental development strategy (section 4) puts strong demands on the extendability of the interpreter. There are three different levels upon which functionality can be added to the interpreter. At each level the developer is presented with an interface which hides as much of the interpreter machinery as possible and requires only localized changes. The top level interface is the SLI language itself. New functions can be defined in individual or collective text files and loaded or discarded as needed. In fact, a large part of the interpreter's functionality is implemented in SLI itself and loaded at start up. Only a small set of primitive operations is required to drive the stack machine. On the second level, the developer can define new “micro programs” for the stack machine. Operators influencing program control like `loop`, or the operator actually executing a procedure, are typically not implemented by a single C++ function. These micro programs have full access to the execution stack. This is not allowed for SLI programs in order to enforce separation of levels of abstraction. Providing support for writing micro programs allows the developer to implement new control structures efficiently. The third level is the straight implementation of an operator on the C++ level. Important examples are the operators interfacing with the simulation kernel. Arguments need to be taken from the operand stack and appropriately passed to a member function of a kernel object. The interpreter provides an interface that gives the developer access to the stacks and the framework for handling error conditions. The ordering in the three levels does not necessarily reflect the difficulty of implementation (from low to high), but rather the amount of available SLI functionality that can be used (from high to low). The level of micro programs may be considered to be the most difficult. It is a common strategy to implement a new function in SLI (level 1) first. Only when serious performance problems are detected is the function re-implemented in C++ (level 3).

6.3 Implementation issues

We have shown in section 5 that choosing C++ as the implementation language for the simulation kernel proved to be very useful for a number of reasons. Although most of the properties of C++ are also advantageous for the implementation of the interpreter, one aspect constituted a serious challenge. The strength of SLI’s machinery is the heterogeneous nature of its stacks and arrays. This is in stark contrast to the philosophy of static typing in C++ which is so useful in generating high performance code. Considerable design efforts were required to arrive at a solution which is at the same time computationally efficient and comfortable for the developer. In this area of code virtual member functions and templates are extensively used.

6.4 The Language is the protocol

We would like to end the section on the simulation language interpreter with a remark found in [18]. Considering the communication between a process performing some computation and a process providing the GUI, one is tempted to think that some format needs to be defined for the data and requests to be exchanged. Even worse, in this case a considerable amount of code on both sides needs to be devoted to the interpretation of the format. However, there is a much more elegant solution. If the programs on both ends of the communication channels are executed by interpreters, the sender can simply send its data in the language of the receiver. We have successfully tested this approach using Tcl/Tk [25, 33] as a graphics engine for SLI. The GUI sends requests as SLI statements and the simulation engine responds with Tcl/Tk statements.

7 Documentation

The first documentation appeared as a technical report of the Weizmann Institute of Science in 1995 [11] consisting of some 60 pages. While the basic principles of the simulator remained practically unchanged in the following years, the number of individual commands and their scope has grown swiftly.

Different types of documentation are required for the user and the developer. Both types consist of introductory parts describing concepts of usage and design respectively. In addition, both need a reference part describing individual functionality in the first case, and its implementation in the second. Whereas the introductory parts only change slowly over the life time of the project, the reference parts are subject to much more rapid extension and change. In the NEST project, all documentation is integrated in a web-based (hyper-linked) help-desk. In the present section we focus on the mechanism we developed to generate and maintain the user-level reference documentation. We return to the documentation of source code in section 11.

Encouraging experience was gathered with the style of documentation in Matlab [24], a tool for numerical data analysis and visualization. The startup message of SLI informs the user of the availability of a command, which contacts the web-browser and displays the introductory help-desk page of the documentation. The user has access to an index of SLI functions organized by name, and an index organized by subject. Hyper-links lead to the documentation of individual functions. The documentation page of an individual function, in turn, may provide links to the documentation of related functions. Alternatively, the documentation for function `f` can be viewed on the console by issuing `/f help`. On an X-terminal, `help` can be configured to present the documentation in a separate window using an application like `xless`.

We observed that functions were routinely commented. Here, the comments in the code primarily served as specification and reminder for the author. In contrast, developing a separate user documentation turned out to be a slow process. The technique used by Matlab overcomes this problem by generating documentation from the source code. If the user issues the command to ask for documentation of a specific function `f`, the file `f.m` is searched for the first comment, and this comment is returned as documentation of `f`.

Exploring this technique, we found that the combination of two aspects is essential in motivating developers to write and maintain documentation. First, documentation is written directly in the source code. Second, after typing a single command at the interpreter prompt the new text is directly visible, integrated in the online help.

In our project, functions can be defined in C++ or in SLI. In contrast to Matlab, several functions can be defined in the same file. A mechanism for automated documentation should work homogeneously for both languages. Because of the rapid development cycle, the documentation also contains information about the location of the source file. In addition to the PostScript (and Matlab) comment sign `%` which protects a single line of comment, SLI also uses the C/C++ style comment block `/* ... */` (however, see [29]). This allows us to specify a homogeneous documentation block in both languages which should precede any definition of a SLI operator, implemented in C++ or SLI:

```
/*BeginDocumentation
Name: Flatten - flattens a nested list
Synopsis:
         array         Flatten -> array
         array integer Flatten -> array
Description:
Flatten called with one argument flattens out all
levels of the argument. Flatten called with two
arguments flattens out the first n levels.
Examples:
   [3 [4 [5 [6]]] 7]   Flatten -->  [3 4 5 6 7]
   [3 [4 [5 [6]]] 7] 1 Flatten -->  [3 4 [5 [6]] 7]
   [3 [4 [5 [6]]] 7] 2 Flatten -->  [3 4 5 [6] 7]
Author: Gewaltig, Diesmann
...
SeeAlso: Partition
*/
```

The documentation block starts with `BeginDocumentation` to distinguish it from other comments in the file. A set of keywords (e.g. `Name` and `SeeAlso`) structures the documentation.

Our documentation generator works as follows. The command

```
(filename) makehelp
```

extracts the documentation comments of all functions defined in the file and writes them as separate files into a special directory. Information about the name and the location of the source file is appended to the documentation file. In the next step the file is converted to an HTML file. The function names following keyword `SeeAlso` are used to create hyper-links. Command `makeallhelp` searches the source tree of NEST and calls `makehelp` for all appropriate source files. Subsequently, `makeallhelp` generates an index of all the functions. Similar to the "H1" line of Matlab, the text in the line where keyword `Name` appears following the minus sign is used as a short description of the function in the index.

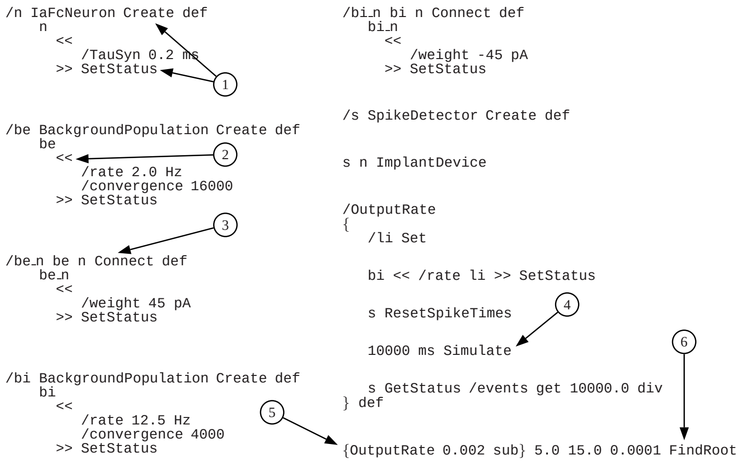
The index file is then converted to an HTML file, with function names representing hyper-links to the documentation files. HTML files are designed to integrate with the remainder of the HTML documentation of the simulator (appropriate links). When the simulator is built from the sources, in the last step of the make process, the interpreter is started to execute `makeallhelp`. Thus, consistency of the documentation with the sources is automatically enforced. A list of SLI functions ordered by subject is currently maintained manually. It should be noted that the documentation engine is fully implemented in the SLI language.

Since the mechanism described in this section has been in operation the documentation has grown steadily and new functions have practically always been commented. The clearness of the HTML documentation improves the detectability of errors. The availability of the location of the sources increases the probability that errors are removed, and documentation is improved by persons different from the initial developer. Thus, the process of writing user level documentation is integrated into our iterative/incremental strategy of software development.

8 A complete simulation session

In this section we give a complete example of a simulation script illustrating key features of both kernel (section 5) and interpreter (section 6). We simulate a neuron receiving spike input from large pools of excitatory and inhibitory neurons. In each pool the neurons fire randomly at a constant homogeneous spike rate. As a result of this random input, the neuron under study also generates spikes at random times. We are looking for a self-consistent solution, where the firing rate of our neuron equals the firing rate of the neurons in the excitatory pool, using the firing rate of the inhibitory pool as a parameter.

Fig. 5 shows the corresponding simulation script. First an object representing a neuron (described by a particular model `IaFcNeuron`) is created. A handle for this object is stored in variable `n`. This allows us to configure a parameter of this object (the rise-time of synaptic currents `TauSyn` is set to $0.2\,\mathrm{ms}$). Using the same syntax, an object, `be`, is constructed, representing the pool of excitatory neurons. This pool consists of 16000 neurons, all of them firing at $2.0\,\mathrm{Hz}$. We are not interested in the detailed dynamics of these



```
/n IaFcNeuron Create def
    n
      <<
         /TauSyn 0.2 ms
      >> SetStatus


/be BackgroundPopulation Create def
    be
      <<
         /rate 2.0 Hz
         /convergence 16000
      >> SetStatus


/be_n be n Connect def
    be_n
      <<
         /weight 45 pA
      >> SetStatus


/bi BackgroundPopulation Create def
    bi
      <<
         /rate 12.5 Hz
         /convergence 4000
      >> SetStatus
```

```
/bi_n bi n Connect def
   bi_n
     <<
        /weight -45 pA
     >> SetStatus


/s SpikeDetector Create def


s n ImplantDevice


/OutputRate
{
   /li Set

   bi << /rate li >> SetStatus

   s ResetSpikeTimes

   10000 ms Simulate

   s GetStatus /events get 10000.0 div
} def


{OutputRate 0.002 sub} 5.0 15.0 0.0001 FindRoot
```

Fig. 5: Example of a complete simulation script (SLI code presented in two columns). A neuron receives random spike input from an excitatory and an inhibitory population of neurons. The simulation script searches for a fixed point where the output spike rate of the modeled neuron equals the spike rate of the neurons in the excitatory population. Arrows indicate where key features of the interpreter and the kernel-interpreter interaction are used: (1) uniform methods for object creation and configuration, (2) communication with kernel by associative arrays, (3) type safe connector, (4) control passed to kernel, (5) pure function object, (6) functional operator searches for fixed point.

neurons. Therefore, the neurons of the pool are represented by a single object `BackgroundPopulation` describing the collective properties of the pool. Here, we exploit the fact that objects describing components of a neural system at different levels of abstraction can be used simultaneously in a single simulation. Using the two handles `be` and `n`, a connection is established by command `Connect`. Subsequently, the amplitude of the post-synaptic current elicited in `n` by a spike emitted from `be` is specified (`/weight 45 pA`). The same sequence is repeated to create and connect the inhibitory pool, differing from the excitatory pool only in terms of parameter values and the sign of the interaction (`/weight -45 pA`). In the last step of the specification a measurement device (spike detector), `s`, is created, a handle assigned to variable `s`, and the device is implanted into neuron `n`.

The next step is to specify the simulation procedure or protocol. To this end a new SLI function `OutputRate` is defined which takes the firing rate of the inhibitory pool as an argument, and returns the firing rate of neuron `n`. Without discussing the details, we observe that this is achieved as follows. First, the parameter `rate` of `bi` is changed. Then the neuronal dynamics is simulated for 10000 ms. Finally, the number of spike events recorded by detector `s` is read out and converted to a rate.

SLI operator `FindRoot` numerically searches for the root of a function on a specified interval (here $[5.0, 15.0]$) with a predefined precision (here 0.0001). The function supplied to `FindRoot` is an unnamed (pure) function having a root at the desired output rate $0.002\,\mathrm{ms}^{-1}$.

Thus, we have described an example simulation with a non-trivial interaction between simulation kernel and interpreter. In the search for a self-consistent solution the interpreter repeatedly passes control to the kernel to simulate the neuronal system with altered parameter settings. The efficiency of the kernel in simulating the dynamics, and the flexibility of the interpreter in expressing algorithms are exploited in the same task.

Let the simulation script be stored in a file `searchfxpt.sli`. The fact that the simulation language is interpreted expresses itself in the following interactive session:

```
SLI ] (searchfxpt.sli) run
SLI [1] /x Set
SLI ] 400 ms Simulate
```

The first line carries out the computation we have described above. `SLI ]` indicates the prompt of the interpreter in an interactive session. When `Find-Root` terminates it leaves the requested root on the stack. The prompt now reads `SLI [1]`, visualizing that one object is available on the stack. The sequence `/x Set` stores the object in variable `x` and the prompt returns to its initial appearance. The neuronal system created during the execution of `searchfxpt.sli` still exists and in the exact state where the last call of `Simulate` left it. Therefore, we can continue to manipulate and investigate the neural system in interactive fashion (e.g. advance system time by another $400\,\mathrm{ms}$).

9 Applications

The simulation software has already supported a number of publications. Let us discuss the problem of spike synchronization in feed-forward subnetworks to illustrate the different types of simulations that have been performed while investigating such systems. Fig. 6A is a sketch of a feed-forward subnetwork known as a *synfire chain* [2]. Groups of neurons are connected into a chain-like structure. Each neuron receives excitatory input from all neurons in the preceding group. The structure is a subnetwork in the sense that it is embedded in a large random network of excitatory and inhibitory neurons. Thus, in addition to the input from the preceding group, each neuron receives on the order of 10^4 inputs from the remainder of the network. In this setting, the membrane potential of an individual neuron exhibits large random fluctuations (Fig. 6B). Synfire chains were introduced to explain the occurrence

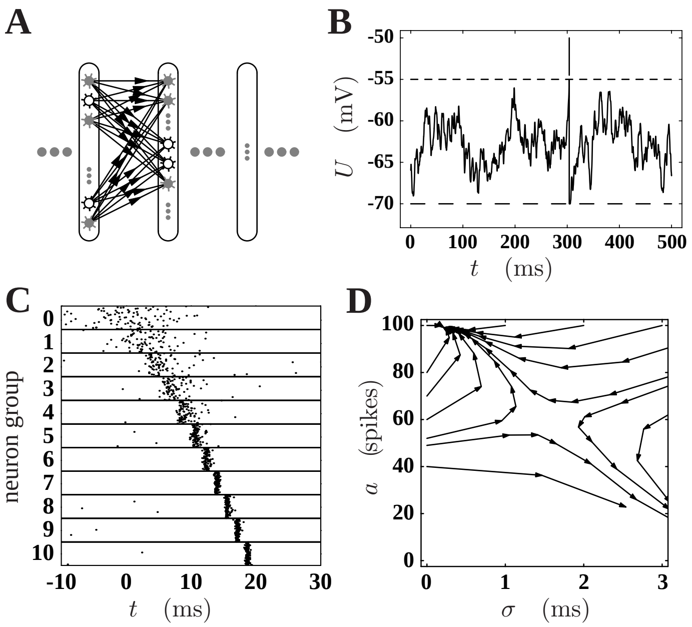


Fig. 6: Examples for the different types of simulations required in investigating a neural system. (A) Feed-forward network of groups of neurons [2]. (B) Simulation of membrane potential fluctuations in a single neuron. Upper dashed line indicates the threshold at which a spike is emitted (a spike occurs at $t \approx 300\,\mathrm{ms}$). In the absence of any input the membrane potential would reside at resting level (lower dashed line). (C) Spike synchronization in a feed-forward network of thousands of neurons. Panels labeled 1 to 10 contain the spike times (dots) of the neurons in the consecutive neuron groups of A. Panel 0 contains the spikes (centered at $t = 0\,\mathrm{ms}$) constituting a stimulus applied to group 1. (D) State space constructed for the dynamics in C (assuming 100 neurons per group). Number of spikes a in a packet vertical and temporal spread of spikes σ horizontal. Trajectories are obtained from a deterministic iterative mapping computed from statistics obtained in single neuron simulations. The activity in C corresponds to a trajectory reaching the attractor from the regime of large σ.

of spatio-temporal spike patterns with millisecond precision in the cortex [1]. The underlying question is how precise spike timing can occur, given the large membrane potential fluctuations and the weakness of individual synaptic connections. Investigating the relationship between parameters determining the sub-threshold dynamics of the membrane potential and spike timing connects single neuron properties with network effects. Measures obtained on the two levels need to be simultaneously consistent with experimental results. Consequently, a moderately realistic neuron model needs to be employed [13].

Fig. 6C shows the result of a network simulation where the first group of a chain is stimulated with a broad packet of input spikes. Following the stimulation a packet of spikes travels along the chain. The spike packet synchronizes until some residual spread is reached and is then stably propagated. Such simulations require a large number of neurons but only a simple protocol.

Describing a spike packet by two variables, number of spikes a and temporal spread σ, we were able to construct a two-dimensional transmission function characterizing the single neuron response to spike packet input. Surprisingly, this allowed us to construct a two-dimensional iterative mapping predicting the synchronization dynamics in a synfire chain (Fig. 6D).

We found that there is indeed an attractor for synchronous spiking activity at millisecond precision [12]. Thus, synfire activity seems to be a possible mode of activity in the cortex. To allow numerical state space analysis, the transmission function needs to be available at high precision. Here, only a single neuron needs to be simulated. However, the simulation protocol is complex and requires a large number of iterations at each parameter setting. In a later publication [17] we demonstrated that activity does indeed develop as predicted by our theory, which reduces the thousands of variables of the network to just two variables . In addition, we characterized the variability observed in individual trials not captured by our deterministic iterative mapping.

10 Teaching computational neuroscience

Although the software is primarily used in research, it is also suitable for teaching Computational Neuroscience. Since 1997 we have used our simulation tool in different courses for students of physics and biology. Starting from the material presented in the course, students can continue to explore neuronal dynamics and build more complex systems on their own initiative. Knowledge can be directly applied and deepened in later research. It is unlikely that software specifically developed for teaching is updated as thoroughly and regularly as software used in research. For the teacher, the advantage of using the same software for research and teaching is that with minor pedagogically motivated improvements, the setup for the research can be used in the classroom. In our experience, education in theoretical neuroscience requires teaching at three different levels.

The first level is concerned with the topic of the research: the structure and dynamics of neural systems. At this level the simulator should support investigation of the dynamical properties of the model. The performance of "virtual experiments" should be comfortable (see section 8) and not be obfuscated by technicalities of the simulation. Depending on the course, implementations of the models used and simulation scripts may even be supplied by the teacher.

The second level is concerned with computational physics topics. Here we deal with the question of how a model can be represented in a computer and how the system can be solved accurately and efficiently. This involves standard numerical techniques (e.g. [27]) and techniques specific to neuronal modeling (e.g. [21, 28]).

The third level is concerned with computer science topics. Access to the sources of the simulation software, including the implementation of the infrastructure (e.g. the simulation language interpreter), enables a discussion of standard computer science techniques in a concrete and applied context. Reviewing many lines of code from neuroscientists, we have learned that neuroscientists usually still do not have access to the appropriate computer science literature and are therefore not aware of standard solutions to the problem at hand. This often results in unnecessarily long development times and suboptimal code.

A particular course can focus on any of the three levels or combine aspects of several levels.

11 Discussion

In the present paper we have described a software environment for the simulation of neural systems which has evolved and been heavily used over the course of several years [11] and led to new insights into the dynamics of neural networks (see section 9).

After 25 years of intensive neuroscience research with computers [23], no clear picture has emerged of how simulations of neural systems should be expressed in software and be made communicable and reproducible. Reasons for this situation are manifold: the inhomogeneity of models describing neural systems at different levels of abstraction, the rapid progress in experimental research, the rapid changes in computer technology and the short term funding of research have all contributed. We have argued that due to the design decisions we made (section 3), we arrived at an efficient simulation system that is able to cope with the demands of the coming years. The four most important aspects are:

– the use of object oriented language and technology,
– the possibility of having network elements at different levels of abstraction and different interactions,
– the view that devices like a spike detector and the protocol are an integral part of a simulation, and
– the strict separation into software layers (e.g. kernel, interpreter, graphics).

However, equally important to the success of our project was gaining an understanding about the conditions under which the software is developed, the resources available, and the risks of a long term project (section 4). The four most important consequences are:

– the software is developed in a collaboration of research groups,
– an iterative/incremental strategy is used avoiding long periods where the software cannot be used,
– free software and open standards are used, and
– mechanisms are devised that support rapid availability of documentation.

The next scientific target is to investigate networks of on the order of 10^5 neurons (see below). Being able to simulate networks of this size enables us for the first time to simulate the volume of cortex from which multiple single unit recordings are usually experimentally obtained. This allows us to make better predictions for spike synchronization and to compare sub- and supra-threshold dynamics in a realistic network environment. Artificial down-scaling of the number of inputs per neuron is no longer necessary.

At the functional level some of our collaborators are interested in modeling large parts of the visual system. This will lead to an increase of network elements on level of abstraction above the single neuron level. At the order of 10^5 neurons, parallel or distributed methods are required to solve network dynamics in appropriate time, and to have sufficient memory available. With a test implementation of a distributed version of the simulation kernel we have already performed preliminary scientific studies. This parallel kernel uses the MPI library for distributed computing [26]. The next task in the development of our software is to re-integrate these parallel methods with the simulation language interpreter, in order to be able to use parallel methods routinely, and perform complex experimental setups and protocols. However, with SMP computers having large homogeneously addressable memory becoming available at reasonable costs an implementation using POSIX threads [10] also becomes an interesting option.

The code base is constantly growing, as is the number of users. Therefore, some of our management procedures will have to be adapted. Until now, no formal bug-tracking mechanism has been introduced.

To get an overview of the code base it is instructive to look at the distribution of code lines and files over main components of the project Tab. 1. In terms of the amount of C++ code, simulation kernel and SLI interpreter seem to be of equal complexity. The SLI support library are the parts of the simulation software implemented in SLI itself. Note again, that the documentation is extracted from the respective SLI or C++ source code (section 7). The figures for simulation kernel, SLI interpreter, and SLI startup library, thus, include both code and documentation.

Especially in the C++ components, large parts of the code are not described by user level documentation, because the basic machinery is only relevant for the developer. In the future it will become increasingly important to introduce

Part	Lines	Files
Simulation kernel	20 000	137
SLI interpreter	26 000	127
SLI support library	10 000	18
Documentation from SLI code	3 700	177
Documentation from C++ code	4 000	249

Tab. 1: The code base of NEST.

a process to generate developer level documentation similar to the technique we currently use to generate user level documentation. Doxygen [31] is a candidate tool already used in parts of the project.

Instead of using an adapted general purpose language like SLI, a dedicated *Neuro Modeling Language* could solve a number of problems related to the description of neuronal network simulations. Moreover, such a language could lift the degree of collaboration to a new level, because many teams would use the same language to define their models. Up to the present day no Neuro Modeling Language exists. However, recently considerable research in this direction has started in which we are actively participating (`www.neuroml.org`). These efforts are promising since they have strong support in the research community.

In the last two years we have managed to install a formal collaboration, the Neural Simulation Technology Initiative (NEST Initiative), between several labs (Fig. 3) which regulates the exchange of concepts and software. In this collaboration the simulation software serves as the reference implementation for new models and methods. Having these resources available has considerably accelarated the development process. The software that is made publicly available (GPL) continues to be distributed under the name SYNOD, the name of the initial version. Some of our results may also be of interest for the development of software tools in other areas of research.

Acknowledgements

This paper documents the results of a collaborative work in which our names are representative for all members of the NEST collaboration (alphabetical by institution): Agricultural University of Ås, Norway: Hans Ekkehard Plesser; University of Freiburg, Germany: Prof. Ad Aertsen, Ulrich Hehl, Carsten Mehring, and Stefan Rotter; Max-Plack Institut für Strömungsforschung, Göttingen: Prof. Theo Geisel; Honda R&D Europe (Deutschland) GmbH, in Offenbach/Main: Prof. Edgar Körner and Andreas Richter; University of Marburg: Rüdiger Kupper.

We thank Prof. Ad Aertsen whose ideas on a *zoom-in simulator* strongly influenced the concepts presented in this paper. He supports and shapes this project since its very beginning.

We gratefully acknowledge the help of Prof. Moshe Abeles, who kindly provided us with the source code and personal notes of his synfire chain simulator. His help has been invaluable in the design of the first version of the simulation kernel.

Partial funding for this project was obtained from the Deutsche Forschungsgemeinschaft (DFG), the German-Israeli Foundation for Scientific Research and Development (GIF), and the Human Frontier Science Program (HFSP).

References

[1] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia. Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of Neurophysiology*, 70(4):1629–1638, 1993.

[2] Moshe Abeles. *Corticonics: Neural Circuits of the Cerebral Cortex*. Cambridge University Press, Cambridge, 1st edition, 1991.

[3] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*. professional computing. Addison-Wesley, 1998.

[4] Adobe Systems Inc. *The PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, 1991.

[5] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers, principles, techniques, and tools*. Addison-Wesley, Reading, Massachusetts, 1988.

[6] Moshe Bar and Karl Fogel. *Open Source Development with CVS*. Coriolis Group Books, 2nd edition, 2001.

[7] Grady Booch. *Managing the Object-Oriented Project*. Addison-Wesley, Reading, Massachusetts, 1996.

[8] James M. Bower and David Beeman. *The Book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System*. TELOS, Springer-Verlag, New York, 2nd edition, 1997.

[9] Frederick P. Brooks. *The mythical man-month: essays on software engineering*. Addison-Wesley Longman, anniversary edition, 1995.

[10] David R. Butenhof. *Programming with POSIX Threads*. Addison-Wesley, Boston, 1997.

[11] Markus Diesmann, Marc-Oliver Gewaltig, and Ad Aertsen. SYNOD: An environment for neural systems simulations - language interface and tutorial. Technical report, The Weizmann Institute of Science, 76100 Rehovot, Israel, 1995. Technical Report GC-AA/95-3.

[12] Markus Diesmann, Marc-Oliver Gewaltig, and Ad Aertsen. Stable propagation of synchronous spiking in cortical neural networks. *Nature*, 402:529–533, 1999.

[13] Markus Diesmann, Marc-Oliver Gewaltig, Stefan Rotter, and Ad Aertsen. State space analysis of synchronous spiking in cortical neural networks. *Neurocomputing*, 38–40:565–571, 2001.

[14] Raphael A. Finkel. *Advanced programming languages*. Addison-Wesley, Menlo Park, California, 1996.

[15] Richard M. Fujimoto. *Parallel and distributed simulation systems*. Wiley, New York, 2000.

[16] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Professional Computing Series. Addison-Wesely, 1994.

[17] Marc-Oliver Gewaltig, Markus Diesmann, and Ad Aertsen. Propagation of cortical synfire activity: survival probability in single trials and stability in the mean. *Neural Networks*, 14:657–673, 2001.

[18] Mark Harrison and Michael McLennan. *Effective Tcl/Tk programming: writing better programs with Tcl and Tk*. Addison-Wesley, Reading, Massachusetts, 1998.

[19] Michael Hines and N. T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[20] Nicolai Josuttis. *The C++ Standard Library: A Tutorial and Reference*. Addison-Wesley, Reading, Massachusetts, 1999.

[21] Christof Koch and Idan Segev, editors. *Methods in neuronal modeling: from ions to networks*. A Bradford Book, MIT Press, Cambridge, 2nd edition, 1998.

[22] Edgar Körner, Marc-Oliver Gewaltig, Ursula Körner, Andreas Richter, and Tobias Rodemann. A model of computation in neocortical architecture. *Neural Networks*, 12(7–8):989–1005, 1999.

[23] Ronald J. MacGregor and Edwin R. Lewis. *Neural Modeling, Electrical Signal Processing in the Nervous System*. Plenum Press, New York, 1977.

[24] MathWorks. *MATLAB The Language of Technical Computing: Using MATLAB*. Natick, MA, 1998. 24 Prime park Way, Natick, Mass. 01760-1500.

[25] John K. Ousterhout. *Tcl and the Tk Toolkit*. Professional Computing. Addison-Wesley, 1994.

[26] Peter S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, California, 1997.

[27] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[28] Sefan Rotter and Markus Diesmann. Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. *Biological Cybernetics*, 81:381–402, 1999.

[29] Bjarne Stroustrup. *The Design and Evolution of C++*. Addison-Wesely, New York, 1994.

[30] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesely, New York, 3rd edition, 1997.

[31] Dimitri van Heesch. Doxygen, 1997. http://www.stack.nl/ dimitri/doxygen.

[32] Gary V. Vaughan, Ben Elliston, Tom Tromey, and Ian Lance Taylor. *GNU Autoconf, Automake, and Libtool*. New Riders, 2000.

[33] Brent B. Welch. *Practical Programming in Tcl and Tk*. Prentice Hall, 3rd edition, 2000.

[34] Stephen Wolfram. *The Mathematica Book*. Wolfram Media/Cambridge University Press, Cambridge, UK, 3rd edition, 1996.

Weitere Beiträge für den Heinz-Billing-Preis 2001

Drift Tube Based Pseudorapidity Assignment of the Level-1 Muon Trigger for the CMS Experiment at CERN

Markus Brugger, Massimiliano Fierro, Claudia-Elisabeth Wulz
Institute for High Energy Physics, Austrian Academy of Sciences, Vienna

Abstract

CMS, the Compact Muon Solenoid experiment under construction at the CERN Large Hadron Collider, will explore new physics at high energies. Proton-proton collisions at a centre of mass energy of 14 TeV and heavy-ion collisions will be studied. Muons with large transverse momenta are expected to be among the decay products of many new particles. Their identification and selection is the task of the trigger system. Specifically, the Level-1 Muon Trigger has to search for muon candidates and to determine their parameters at a rate of 40 MHz, corresponding to a beam crossing interval of 25 ns. The precise knowledge of the spatial parameters of a track allows to fully make use of the possibility to select topological trigger conditions already at Level-1 in CMS.

Track segments measured in orthogonal layers of drift tube chambers are combined to form a muon candidate. Its transverse momentum is calculated from the track curvature in the (r/φ)-projection caused by a magnetic field along the beam direction z. The azimuthal angle φ in the plane transverse to the beam is also determined. The pseudorapidity η is a function of the track angle θ •relative to the beam axis. Using the information from the bending plane projection only allows a coarse assignment of η• in the central region of CMS by determining which chambers were crossed by the track. A method to assign η-values of much greater precision is presented. It relies on track finding performed in the non-bending plane (r/z) and on matching the found tracks with those of the (r/ φ)-projection. The requirements, the chosen algorithm, its simulated performance and the feasibility for a hardware implementation are described.

1. Introduction

Compact Muon Solenoid (CMS) [1] is a general-purpose experiment designed to study proton-proton and heavy-ion collisions at the Large Hadron Collider (LHC) of CERN. Its main feature is a strong solenoidal magnetic field ensuring high momentum resolution for charged particles. Apart from the superconducting coil [2] the apparatus consists of a silicon inner tracker with an embedded pixel detector [3] [4] , a lead tungstate crystal electromagnetic calorimeter [5] , a copper-scintillator sandwich hadron calorimeter [6] and a sophisticated four station muon system [7] featuring dedicated trigger chambers, the Resistive Plate Chambers (RPC), and tracking chambers, which are also used in the trigger. The latter are made up of Drift Tubes with bunch crossing identification capability (DT) in the central region and Cathode Strip Chambers (CSC) in the forward parts of the experiment. The RPC trigger chambers are mounted on the tracking chambers. To improve detector coverage a forward calorimeter consisting of a copper matrix with embedded quartz fibres is added in the forward regions [6] . A perspective view of the CMS experiment is shown in Figure 1.

The purpose of the trigger system of the experiment is the selection of all interesting events in the presence of an overwhelming background. Potentially interesting events at LHC are those including a Standard Model Higgs particle, supersymmetric or any other previously unknown particles as well as top events, WW, ZZ, Wγ and many others. Background events originate from collisions that carry no new information for the study of matter and its origin. The trigger also has to select events used for calibration, synchronization or testing purposes.

If the trigger discards a bunch crossing, the data is lost forever. For lepton colliders, such as the e^+e^- collider LEP at CERN, where the probability of collisions in a bunch crossing is far below 1 (about 3 events per second) dead times of the trigger system usually do not cause severe inefficiencies. This is because the probability of a second interaction within the trigger processing time is low. On the contrary, since the LHC beam crossing rate will be 40 MHz for protons, with about 18 superposed events per beam crossing, the trigger system is of crucial importance to the physics to be discovered.

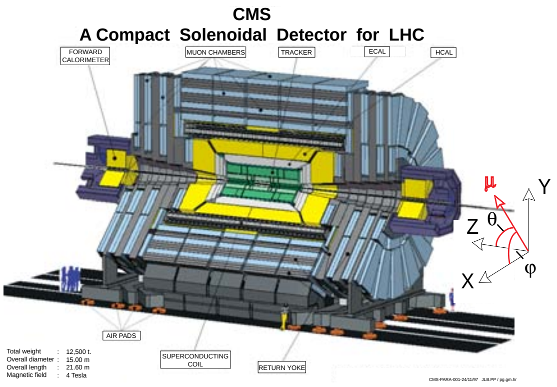


Figure 1: Perspective view of the CMS detector

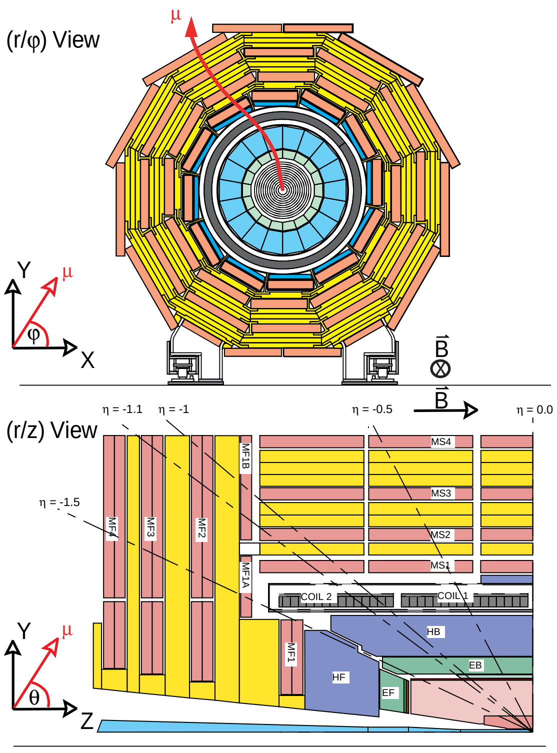


Figure 2: The two different projections of the CMS detector

Muons (μ) are important signatures for new particle production due to their power to penetrate large amounts of material. They usually have a clean signal in the muon detectors and can be identified even in the presence of hadronic jets.

The CMS muon system triggers on and reconstructs muon candidates. The drift tube chambers in the central region (barrel) measure the tracks independently in two projections. In this region the magnetic field lines are parallel to the beam axis, which defines the z coordinate. In the bending plane (r/φ) tracks form helices, and in the plane along the beam (r/z) they are almost straight lines originating at the interaction point (Figure 2). As a consequence, two different systems are used to detect muon tracks in each projection. In the first projection the transverse momentum (p_T) and the azimuthal angle (φ) transverse to the beam axis are measured. In the second projection the so-called pseudorapidity (η) is determined, which is a function of the angle (θ) between the track and the beam axis.

Since the LHC particle accelerator delivers proton-proton collisions at a rate of 40 million events per second, the trigger electronics needs to provide a decision to retain or to reject an event for each bunch crossing interval (bx) of 25 ns. The CMS trigger has two distinct stages, the Level-1 Trigger and the High Level Triggers. The latency of the CMS Level-1 Trigger is restricted to 128 bx corresponding to 3.2 μs. Because of radiation levels, access and maintenance, most of the trigger electronics is not located on the detector itself but after a 90 m cable run in the counting room. The consequence of this arrangement is that a large part of the trigger latency is actually due to propagation delays in the optical links between the detector and the counting room, and only a fraction of the total latency time can be used for the processing of the actual trigger algorithms.

Since the computing time available is too short to use ready-made programmable devices such as microprocessors or digital signal processors, the Level-1 Trigger has to use custom built processors. If hints of unforeseen new physics appear it must be flexible enough to adapt the trigger algorithms as needed. It must also be able to accommodate higher backgrounds. It must cope with noisy and dead trigger cells by providing the possibility to selectively mask off individual channels.

A block diagram of the CMS Level-1 Trigger is shown in Fig. 3. Details about the Muon Trigger are given in section 2.4. The Global Muon Trigger combines the two independent components of the Muon Trigger, the DT/CSC based Track Finder and the RPC based Pattern Comparator Trigger (PACT) as described in [12] and section 2. The Level-1 Trigger decision is performed at the level of the Global Trigger. The earlier stages only provide information for this decision. The Global Trigger (refs. [8] and [9]) uses only the information of the Global Calorimeter Trigger and the Global Muon Trigger. Particle rates in the inner tracker are very high,

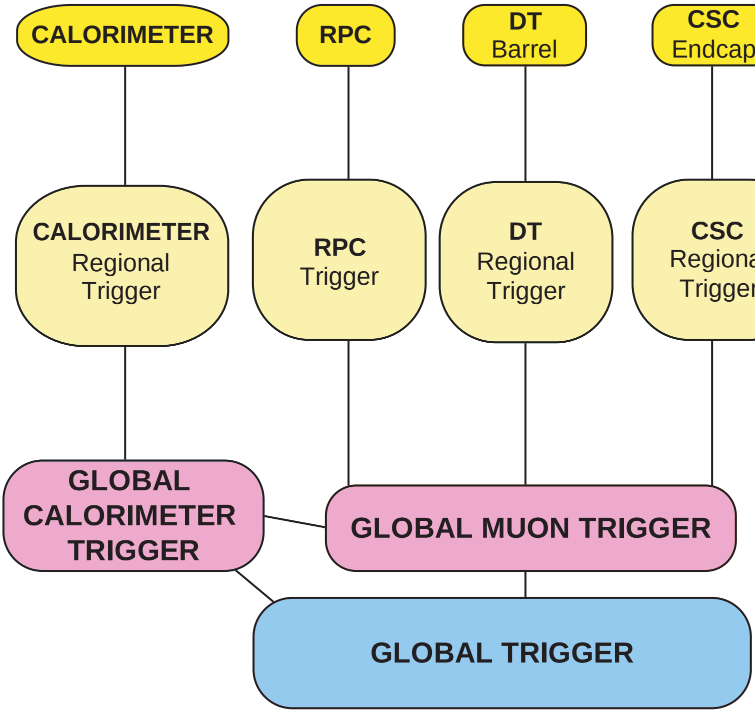


Figure 3: Logical structure of the Level-1 Trigger components

therefore track finding in the inner tracker is too slow to be used in the Level-1 Trigger.

CMS requires all components of the Level-1 Trigger to be dead-time free. Therefore, a trigger decision based on trigger objects delivered by the detector subsystems is provided for each beam crossing, at a rate of 40 MHz. The Global Trigger comprises a novel concept where event selection is not only based on objects exceeding energy or momentum thresholds but also on complex event topology which is made possible by the availability of space, charge and quality information in the algorithm calculations. The trigger logic is largely programmable in order to satisfy evolving physics requirements.

In the following an algorithm for calculating precise η values for all muon tracks found in the central part of the CMS detector is presented. It will be implemented in the DT based Track Finder electronics.

2. Barrel Muon Trigger

The CMS muon detectors consist of two different components: the barrel chambers in the center and the endcap chambers in the two forward regions. In order to detect a passing muon different types of detectors form a redundant system consisting of Drift Tubes (DT) located outside the magnet coil in the barrel region and Cathode Strips Chambers (CSC) located on the sides. The CMS muon system is also equipped with Resistive Plate Chambers (RPC) dedicated to triggering. The positions of the three detector types are schematically shown in Figure 4.

The barrel as well as the endcap systems consist of four muon stations interleaved with the iron of the magnetic flux return yoke. The iron decouples adjacent stations in case of muon-induced electromagnetic background: A muon can emit a bremsstrahlung photon which gives rise to an electromagnetic cascade. The iron between adjacent stations has to be at least 30 cm thick to absorb all these cascade particles and to prevent them from penetrating into the next station.

Since the central magnetic field has no bending power in the (r/z) plane, the muon tracks form straight lines in this projection. They originate at the vertex. The detector measures the pseudorapidity η using the angle θ

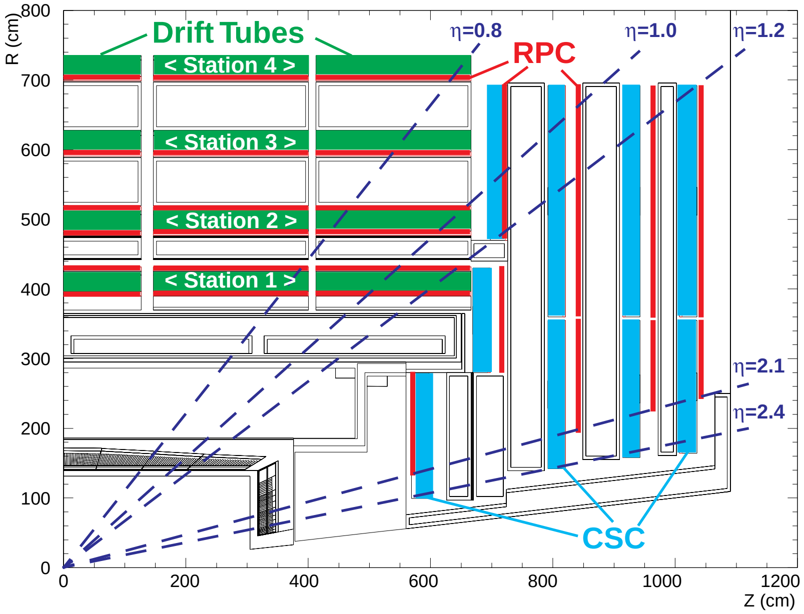


Figure 4: The CMS muon detectors in longitudinal view

between the track projection in the longitudinal plane and the beam direction. η is defined as:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right)$$

Directions parallel to the beam axis, i.e. $\theta = 0°$ or $\theta = 180°$, have pseudorapidities η of $+\infty$ or $-\infty$ respectively.

2.1. Drift Tube Chambers

The barrel muon system [7] is composed of four muon stations. It is divided into five wheels along the z-axis, each approximately 2.5 m long. Each wheel consists of twelve azimuthal sectors, therefore one sector covers approximately 30°. All sectors within the same φ range across all five detector wheels are grouped together and called a “wedge” (Figure 5).

Due to the strong magnetic field outside the coil - the magnetic flux density in the return yoke corresponds to the saturation value of 1.8 T - a standalone muon momentum measurement up to high energies is possible. Calculating the curvature of the muon tracks allows the assignment of p_T values. This is essential for the trigger and useful for the off-line matching of a reconstructed track to its image inside the inner tracker.

Every muon station contains modules of drift tubes measuring the track coordinates in the (r/φ) plane. Only the inner three stations are equipped to measure in addition the track position in (r/z).

As shown in Figure 7 a single chamber in stations one, two or three consists of twelve layers (L) of drift tubes arranged in three superlayers (SL) of four layers each. The tubes of one superlayer are arranged in four layers with an offset of half a tube width each. The outer two (r/φ) superlayers measure the φ coordinate in the bending plane, whereas the central (r/z) superlayer measures the z coordinate along the beam axis. The chambers are 2.56 m long in z, and the width in (r/φ) increases from 2 m in the innermost station to 4 m in the outermost station. The position of the chambers can be seen in Figure 4 .

In the endcap region the muon detector also consists of four muon stations but mounted vertically and using a different method for particle detection. The working principle of the CSC detector at CMS is described in [7] .

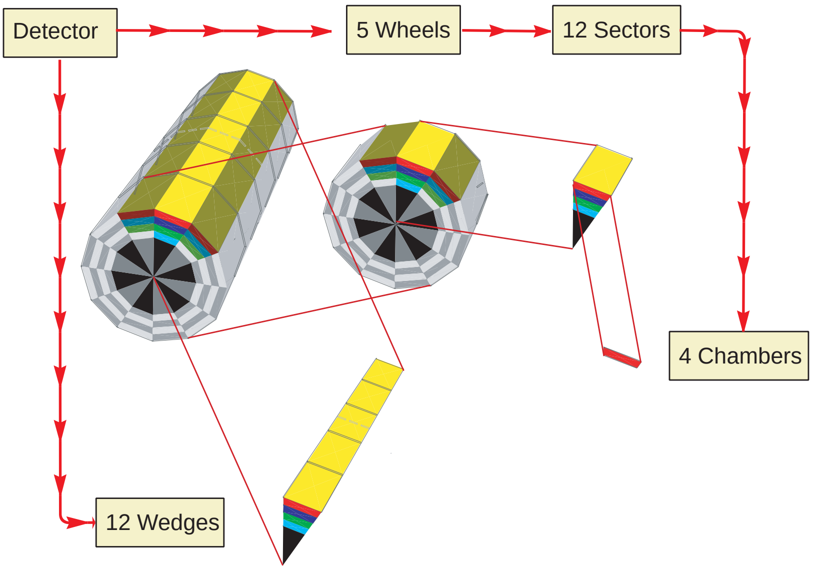


Figure 5: Classification of the different detector parts

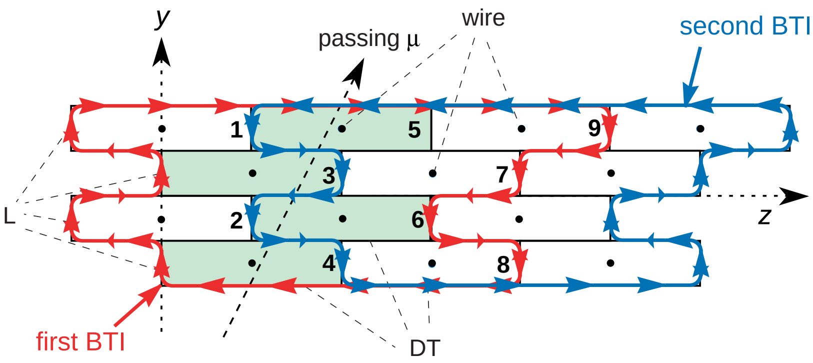


Figure 6: BTI layout consisting of nine DT cells. A passing muon hits different wires which are then recognized by the BTI.

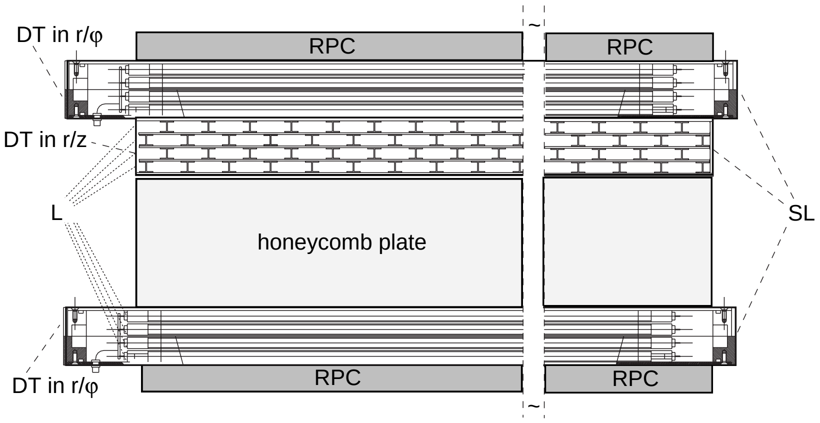


Figure 7: Cross section of a muon chamber in the (r/z) view (station 1 or 2)

DT	...	*Drift Tube*
L	...	*Layer*
SL	...	*Super Layer*
RPC	...	*Resistive Plate Chamber*

To calculate the η value in the barrel data coming from the central superlayer of the DTs are used. This superlayer consists of drift cells with wires oriented along the beam direction and is read out by a special logic, housed in the DT Trigger Electronics.

2.2. Drift Tube Local Trigger Electronics

The maximum drift time of the particles in the drift tubes of 400 ns is much longer than the delay between two consecutive bunch crossings (25 ns). This means that the muon causing the electron avalanche in the tube will be detected by the wire only after several bunch crossings have occurred. Therefore it is necessary to identify the bunch crossing to which the detected muon belongs. The method is based on a generalized meantimer technique described in [10] .

The so-called ‘Bunch and Track Identifier (BTI)’ was explicitly developed to apply the technique working on groups of four layers of staggered drift tubes, aiming to identify tracks giving signals in at least three of the four layers. This method is based on the fact that - considering the small width (6 cm) of one superlayer - the particle’s path is a straight line. The distances of the wires along the path, which give rise to the measured “points” are equal.

Each BTI is connected to nine wires grouped as shown in Figure 6. To enable an effective measurement the BTI's overlap by five wires, i.e. the next BTI will start from the cell labelled 5 in Figure 6. This overlap assures the redundancy needed to limit the inefficiency in case of a BTI failure.

The BTI's deliver the input for both the (r/φ) Track Finder and the η Track Finder. In the former case, the outputs of the BTI's from two superlayers are correlated by a device called Track Correlator to so-called "track segments", thus reducing the noise and improving the angular resolution. Track segments contain not only the hit location, but in addition angular information of the particle's path. In the latter case there is only one superlayer per chamber, therefore only the hit coordinates are retrieved, which are then sent to the θ Trigger Server.

2.3. *Trigger Server*

The Trigger Server (TS) is a device that forwards the track segments to the Regional Muon Trigger, which has to construct complete tracks with the local trigger information and to determine their parameters.

2.3.1. *Transverse View (r/φ)*

The Trigger Server of the φ-layer selects the best two track segments per chamber out of all segments delivered by the Track Correlator and delivers their location and direction angles to the Regional Trigger.

2.3.2. *Longitudinal View (r/z)*

The DT θ Trigger Server in the longitudinal view (TSL) has to group the information coming from approximately 64 BTI's of each θ-superlayer. It sends a corresponding bit pattern to the η Assignment Unit according to the output of the BTI's of one chamber. Studies on the Regional Muon Trigger [11] have shown that the minimum pattern resolution, without significant loss of track finding efficiency, corresponds to the space covered by eight BTI's. As a consequence only 8 different values are calculated per chamber. The TSL receives two bits from each BTI, one for the trigger strobe pulse and one for the corresponding quality. For both of them the TSL performs a logical OR in groups of eight BTI's. The output is formed by two bits for each group: in total there are eight bits for the trigger pattern plus eight bits for the corresponding quality information. The "quality" bit gives information whether three or four out of four wires registered a passing muon. In the former case a low trigger signal (LTRG) is issued, whereas a high trigger signal (HTRG) is assigned in the case of the latter. If fewer than three wires detected a hit, no signal is produced.

2.4. Regional Muon Trigger (Track Finder)

The Regional Muon Trigger in (r/φ)-projection is subdivided in units covering 30°-sectors in φ, the so-called "Sector Processors". Their task is to find muon tracks originating at the interaction point and to measure their transverse momenta p_T and their location in φ. If a track passes a chamber, the chamber trigger logic forwards a track segment to the Track Finder. The track finding is done by extrapolating from one track segment to the next. For p_T measurement this is sufficient, because in the barrel region the magnetic field is uniform, thus the p_T value can be calculated from the bending angle of the track. Chambers have inefficiencies - in only ~85% of the cases a crossing muon gives rise to at least three track segments belonging to one track. However, only two compatible track segments are already sufficient to form a track candidate, leading to a track acceptance of about 96% [12] .

Since tracks may cross detector wheel and sector boundaries, adjacent Sector Processors have to perform intercommunication. Due to the design of the trigger system every Sector Processor determines the best two muon tracks. All processors of one wedge forward their selected tracks to the Wedge Sorter (WS), which selects up to two muon tracks per wedge and forwards them to the DT Sorter. The DT Sorter in turn selects the best four muon candidates of all wedges and sends them to the Global Muon Trigger (GMT) [12] .

Although the Regional Trigger in (r/φ)-projection, which does not use the θ superlayer, is by definition unable to measure the z coordinate along the beam direction, it can nevertheless assign a coarse η value due to the information, on which chambers were crossed by a track. With this method a coarse, location dependent, resolution in η can be achieved. Moreover, the resolution depends on the number of track segments a track consists of. If a change of wheel has not taken place, the η value will be particularly coarse, especially in the central wheel. The final hardware implementation of the DT Regional Trigger therefore foresees a separate track finding processor in the (r/z)-plane for each wedge. The tracks thus found are then matched with those seen in the bending plane projection, resulting in a more accurate determination of η than the one by the Track Finder in (r/φ) alone. The η assignment scheme is also more flexible, as will be explained later.

3. η Assignment Unit

3.1. *Design Principle*

There are two possibilities to assign an η value to a track: (a) using the coarse information of the (r/φ) Track Finder, taking advantage of the known position where a track crosses detector wheel boundaries; (b) using the θ superlayer data from the first three muon stations to calculate a precise η value. Both of them are foreseen in the design of the device that calculates η. Originally the coarse η assignment was planned to be implemented on the (r/φ) Sector Processor boards, and the precise η assignment on a separate board. However, it turned out to be more practical to calculate both the coarse and the precise η values on a single electronic board, the so-called η Assignment Unit (EAU), which performs two steps. The first one is the stand-alone track finding in the (r/z) plane, and the second one is the matching with the (r/φ) track information. Since the track finding is done independently in the two projections of the detector, the EAU tries to match the η track to the muon tracks found in (r/φ). If the match is successful, the improved η value is definitely assigned to the track, otherwise the coarse assignment is used. The sub-unit of the EAU that finds the tracks in (r/z) is called η Track Finder (ETF), and the sub-unit that tries to combine the muon track information of both projections, namely (r/φ) and (r/z), is called the η Matching Unit (EMU).

Although the η Assignment Unit could theoretically find additional tracks not seen by the (r/φ) Track Finder in the barrel, this possibility has not been implemented due to background considerations. Therefore only the tracks selected by the (r/φ) Sector Processors have an η value assigned.

Table 1: Summary of the input data for the η Assignment Unit from each muon chamber

Content	# of Bits
Position	**8**
Quality	**8**

3.2. Input and Output

Two different inputs are relevant for the η assignment. One comes from the θ superlayers for the η track finding and the other comes from the (r/φ) Track Finder for the matching. A special Sector Collector sends the θ data to the Regional Muon Trigger using an optical link with the data format described in Tab. 1.

In principle, each chamber can resolve around 64 different positions in the z direction. However, as already mentioned before, a study [11] has demonstrated that there is little impact on the system resolution using even fewer positions. Therefore, only 8 instead of 64 different θ positions per chamber are forwarded to the η Assignment Unit (EAU). In order to obtain these 8 values per chamber eight adjacent BTI's are grouped together in a logical OR. Since the θ data contains only the grouped information coming from the BTI's in one superlayer, no track correlation like in the (r/φ) plane is possible, or necessary. Therefore the data arrives 5 bx earlier at the input of the EAU than the corresponding track segments arrive at the (r/φ) Sector Processors.

For the matching and also for the calculation of the coarse η values there has to be an intercommunication between the Sector Processors and the EAU. In each wedge there is a single EAU, which receives track segments coming from the first three muon stations in all five wheels belonging to one wedge. The chamber hits are sent in the form of two eight-bit patterns, one for the quality and one for the position (cf. Tab. 1). In addition, signals for calibration and control purposes are transmitted to the Sector Collector.

In total, this results in 5 x 3 x 16 = 240 bits for each bx and wedge, forwarded from the Sector Collector using eight optical links per wedge. This results in 96 optical links for the whole detector. In addition, for each found track the (r/φ) Sector Processor sends a five-bit information of chamber addresses to the EAU. Since the (r/φ) Track Finder consists of two modules in the central wheel, in total six Sector Processors are sending the track address data of up to two muons each, resulting in 12 x 3 bits per wedge.

Since each (r/φ) Sector Processor selects the best two muon candidates (if present), this results in at most 12 muon tracks, for which the η Assignment Unit has to calculate an η value. Each track is assigned a six bit η value defining an η range from -1.3 to 1.3. An additional bit is foreseen to indicate whether the matching was successful or not, resulting in (6 x 2 x 6)+(6 x 2) = 84 bits. Table 2 shows a summary of all input and output signals for one EAU.

Table 2: Input and output of the η Assignment Unit

	Input from		
	Information	***Format***	***# Bits***
DT-TSL	θ superlayer hit data	8 bit Position 8 bit Quality	240
(r/φ) Track Finder	Found track address	5 bit address	60
	Output to		
Wedge Sorter	Assigned η values	12 x 6 bit η Track Data 12 x 1 bit Assignment Identifier	84

The η Track Finder optical link receivers will be mounted on daughter boards (mezzanine cards) that will be plugged on to the Sector Processor board.

4. Algorithm

4.1. Track Finding

In [11] it is shown that for the track finding in the (r/z) plane a pattern matching method is the most suitable one. When a track passes different chambers detecting hits the information is sent through corresponding electronic systems (refs. [12] Chap. 9.1) to the η Assignment Unit. Each combination of hits is interpreted as a pattern pointing back to the vertex (cf. Figure 8), which is compared to a precalculated list.

If a matching pattern is found a corresponding η value is assigned to the track in the (r/z) plane. The calculation of this pattern list, based on simulation data as long as no real data are available, is the task of the simulation software described in [13] . Each η track has an assigned quality, depending on the number and the quality of the hits used to form the track (cf. Tab. 3).

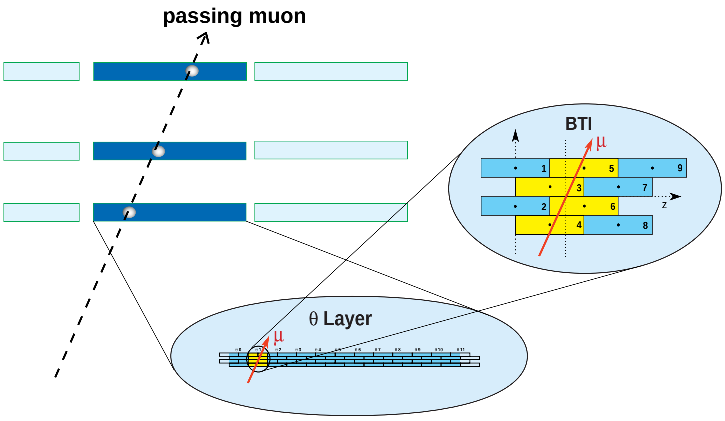


Figure 8: For the track finding in the (r/z) plane a pattern matching method is used

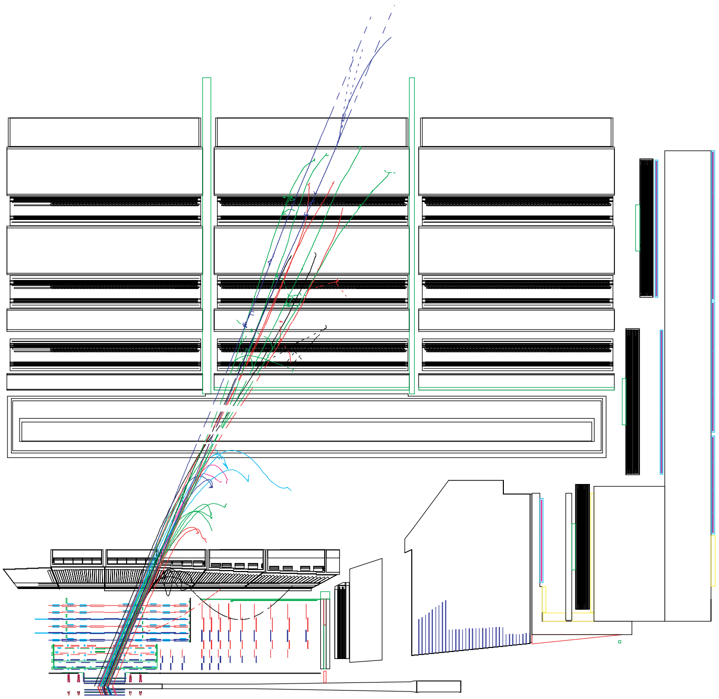


Figure 9: Muon tracks with different values for p_T

4.1.1. Requirements for a valid η Track

The tracks viewed in the z projection are approximately straight lines (cf. Figure 9). Therefore, knowing the position of the vertex, a single hit indicating the z coordinate where a particle passed the chamber is sufficient to assign an η value to this track. The question, how many hits should be required to form a valid track, arises.

Since the design of the muon detector foresees only the inner three muon stations to be equipped with a θ superlayer and considering the efficiency of the chamber trigger logic shown in [11] , it is clear that requiring only two hits results in a very low track acceptance. In other words, too few η tracks would be found and as a result too few tracks seen in the r/φ projection would receive an improved η value. In ref. [11] it was shown that a single low quality track segment has to be accepted as a valid track in order to achieve an acceptable matching rate. This can also be derived from the η Track Finder efficiency shown later in Figure 16.

This is not as bad as it appears at first sight. What could happen is that a wrong η value is assigned to a (r/φ) track which still must be in an acceptable range. This range is defined in such a way that the unphysical track matches the track found in (r/φ). In this way no additional tracks are forwarded to the Global Muon Trigger by the η Assignment Unit. If an η track cannot be matched the track is simply discarded. From this point of view even the η tracks with the lowest quality can be used for the matching. The η Assignment Unit merely sorts the track information the chamber trigger logic delivers rather than filtering out ghosts by throwing away data. It is in the Matching Unit where the unmatched η tracks will be filtered out.

Another question is whether a single hit can belong to more than one track. In contrast to the input in (r/φ), the granularity of the trigger primitives is rather coarse. Therefore, the probability for two muons to overlap is higher than in the bending plane. As a consequence, two tracks must be allowed to share the same hit. The last requirement is the vertex constraint: every found muon track must point back to the interaction region. In the case of the η Track Finder only patterns that belong to tracks coming from the interaction point are stored, thus the vertex constraint is embedded in the pattern list by construction.

4.1.2. The η Track Finder (ETF)

The reconstruction of tracks using (r/z) plane hits, which lie on a straight line pointing back to the interaction vertex, is the task of the η Track Finder. A pattern matching method has been chosen. A list of possible patterns has been generated using simulated data samples, each pattern belonging to a certain η value. Therefore, if a set of hits matches a particular pattern, a track is found and the pattern η value is assigned to the track.

Since a track can consist of different subpatterns, it may occur that using only part of the hits multiple tracks will be found for a single real track. Therefore, a cancellation has to be implemented ensuring that the track with the highest quality is taken. Thus, each track belonging to a pattern also has a quality information attached.

The following scheme is used to define a unique quality for each η hit combination. The different combinations of the η hit quality (LTRG or HTRG) in the three different stations result in 26 quality numbers shown in Tab. 3.

This list of η track qualities, sorted according to all combinations of hit qualities, is used in the calculation of the pattern lists. The matching logic uses the same scheme and provides a simple correlation between the track information coming from the (r/φ) Sector Processors and the found η tracks.

Table 3: Definition of all possible η track quality numbers. They are grouped together according to their quality significance (0...no TS, 1...LTRG, 2...HTRG)

Ver	Slice			Ver	Slice		
	1	2	3		1	2	3
1	0	0	1	**14**	1	2	0
2	0	1	0	**15**	2	0	1
3	1	0	0	**16**	2	1	0
4	0	1	1	**17**	1	1	2
5	1	0	1	**18**	1	2	1
6	1	1	0	**19**	2	1	1
7	0	0	2	**20**	0	2	2
8	0	2	0	**21**	2	0	2
9	2	0	0	**22**	2	2	0
10	1	1	1	**23**	1	2	2
11	0	1	2	**24**	2	1	2
12	0	2	1	**25**	2	2	1
13	1	0	2	**26**	2	2	2

Table 4: Definition of all possible η track classes according to their number of hits and hit quality used to form the η track.

Track Class #	η Track containing at least
1	... one LTRG
2	... one HTRG
3	... two LTRG
4	... one LTRG and one HTRG
5	... two HTRG
6	... two LTRG and one HTRG
7	... one LTRG and two HTRG
8	... three HTRG

For the analysis of the EAU performance a simpler scheme is more useful, grouping the tracks according to their number of hits and hit qualities, while discarding the station information. This scheme is shown in Tab. 4 and used in Figure 16 and Figure 20.

4.2. Matching

Each muon track found in the bending plane needs to have an η value assigned. The muon tracks in the (r/z) plane are found independently, thus the computed η values need to be matched to the (r/φ) tracks. To perform the matching with the tracks found in the bending plane, each Sector Processor of a wedge sends track address data for all found tracks to the η Assignment Unit board. Using these address data a coarse η value is calculated in parallel and later on used if the match was not successful. In other words, if no track of the (r/z) plane fits the track found in (r/φ), the coarse η value will be used.

The calculation of the coarse η value is also based on a pattern matching method, in analogy to the η Track Finder. The pattern of the chambers used to reconstruct the track in the (r/φ) plane is searched in a precalculated list containing the corresponding η values. These η values are determined with the simulation software package.

In order to perform the matching, the η Assignment Unit receives coded track address data from the (r/φ) Track Finder. Either the muon chambers used to reconstruct each muon track, or the corresponding η value can be used in the comparison. Thus, in the first case a track can be matched if the used chambers can be combined, whereas in the second case only the possible η ranges are checked. The (r/φ) address data contain the necessary wheel, station and sector information of all (r/φ) track segments (TS) used to reconstruct each track found in the (r/φ) plane as shown in Tab. 5. For-

Table 5: Track address data used for matching. Since each track found in (r/φ) belongs to one Sector Processor, the sector and wheel numbers of the starting track segment are known. Only the remaining information has to be defined, e.g. if the track stays in the same wheel ("S") or changes to the next one ("N"). The combinations that must not be used because of bending back ("b"), not found by the Sector Processor ("n"), being equal ("e") or single tracks ("s") are shaded.

Id #	St. 1	St. 2	St. 3	St. 4	Reason	Id #	St. 1	St. 2	St. 3	St. 4	Reason
0	0	0	0	0			S	0	0	0	s
	0	0	0	S	S	**9**	S	0	0	S	
	0	0	0	N	S	**10**	S	0	0	N	
	0	0	S	0	S	**11**	S	0	S	0	
1	0	0	S	S		**12**	S	0	S	S	
2	0	0	S	N		**13**	S	0	S	N	
	0	0	N	0	S	**14**	S	0	N	0	
	0	0	N	S	B		S	0	N	S	b
	0	0	N	N	N		S	0	N	N	e
	0	S	0	0	S	**15**	S	S	0	0	
3	0	S	0	S		**16**	S	S	0	S	
4	0	S	0	N		**17**	S	S	0	N	
5	0	S	S	0		**18**	S	S	S	0	
6	0	S	S	S		**19**	S	S	S	S	
7	0	S	S	N		**20**	S	S	S	N	
8	0	S	N	0		**21**	S	S	N	0	
	0	S	N	S	B		S	S	N	S	b
	0	S	N	N	E		S	S	N	N	e
	0	N	0	0	S	**22**	S	N	0	0	
	0	N	0	S	B		S	N	0	S	b
	0	N	0	N	N		S	N	0	N	e
	0	N	S	0	B		S	N	S	0	b
	0	N	S	S	B		S	N	S	S	b
	0	N	S	N	B		S	N	S	N	b
	0	N	N	0	N		S	N	N	0	e
	0	N	N	S	B		S	N	N	S	b
	0	N	N	N	N		S	N	N	N	e

tunately a lot of combinations turn out to be not possible because of one of the following reasons:

- the track would have to bend back in η
- the track consists only of a single track segment
- the track would not be found by the Sector Processor
- the track lies inside the same η boundaries.

In the next step this track information is compared to the reconstructed η tracks. If a suitable combination is found, an η value can be selected.

In the second case only the corresponding η values are used. Hence, what is compared is the η value belonging to the improved pattern and the possible η range corresponding to the raw pattern. This kind of matching is called “virtual matching” in the following.

Generally, if more than one η track fits the (r/φ) track, the η track with the higher quality is used. If no match is possible, the coarsely calculated η value - retrieved from the position of the used muon chambers in (r/φ) - is taken.

The next question is the impact of the minimal quality of the reconstructed η track and the minimal number of muon chambers requested to be equal. In the simulation different matching schemes have been studied in order to verify the impact of the quality of the η track and the minimal matching criteria between the tracks found by the (r/φ) and η Track Finders. Setting a minimal threshold results in up to 26 x 4 different matching schemes that can be used. What is called matching scheme is therefore a combination of setting different minimal thresholds for track qualities and matching combinations as shown in Figure 10.

4.3. η Assignment

If the two tracks can be matched, the η value calculated by the η Track Finder is assigned to the (r/φ) track candidate. Otherwise, if a matching is not possible, the coarse η value is assigned to the (r/φ) track candidate. Therefore it is possible to assign an η value to each track found in the bending plane. It is important that an η value is assigned and forwarded to the Wedge Sorter simultaneously with the p_T and φ values, whenever a track is found by the DT Track Finder. An additional bit is forwarded to the Global Muon Trigger with the information whether the assigned η value corresponds to the coarse or to the improved assignment method.

In Figure 19 the matching efficiencies for the various schemes are shown and compared with different resolutions. Choosing one of the different possible schemes defines the matching probability and the relative η resolution.

The settings for the matching can be adjusted in the simulation software which then generates different lists of patterns including the information on which (r/φ) track can be matched with which η track. This option allows to change the matching and track identification scheme very quickly. Together with the modularity of the simulation software and the algorithm itself it offers a flexible η assignment method.

After the matching the last step is the assignment of the calculated η value.

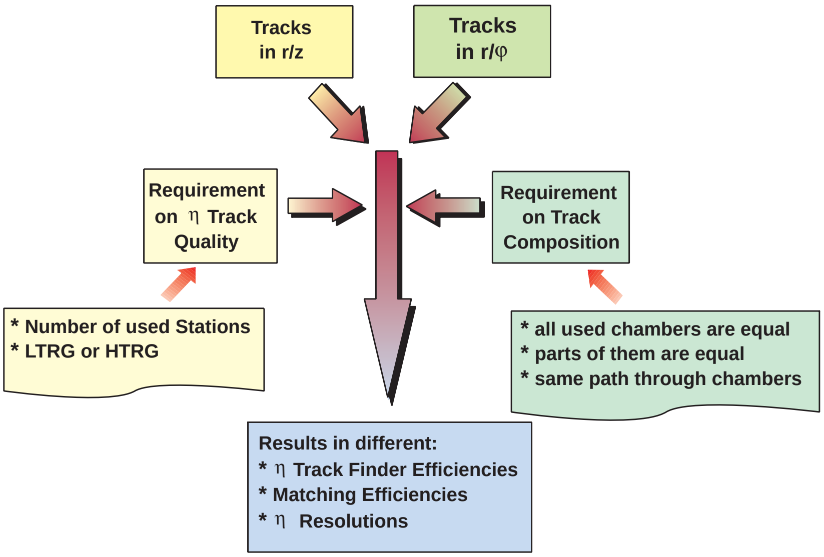

Tracks in r/z
Tracks in r/φ
Requirement on η Track Quality
Requirement on Track Composition
* Number of used Stations
* LTRG or HTRG
* all used chambers are equal
* parts of them are equal
* same path through chambers
Results in different:
* η Track Finder Efficiencies
* Matching Efficiencies
* η Resolutions

Figure 10: Different possible combinations of η track quality and the matching of tracks found in (r/φ) and (r/z)

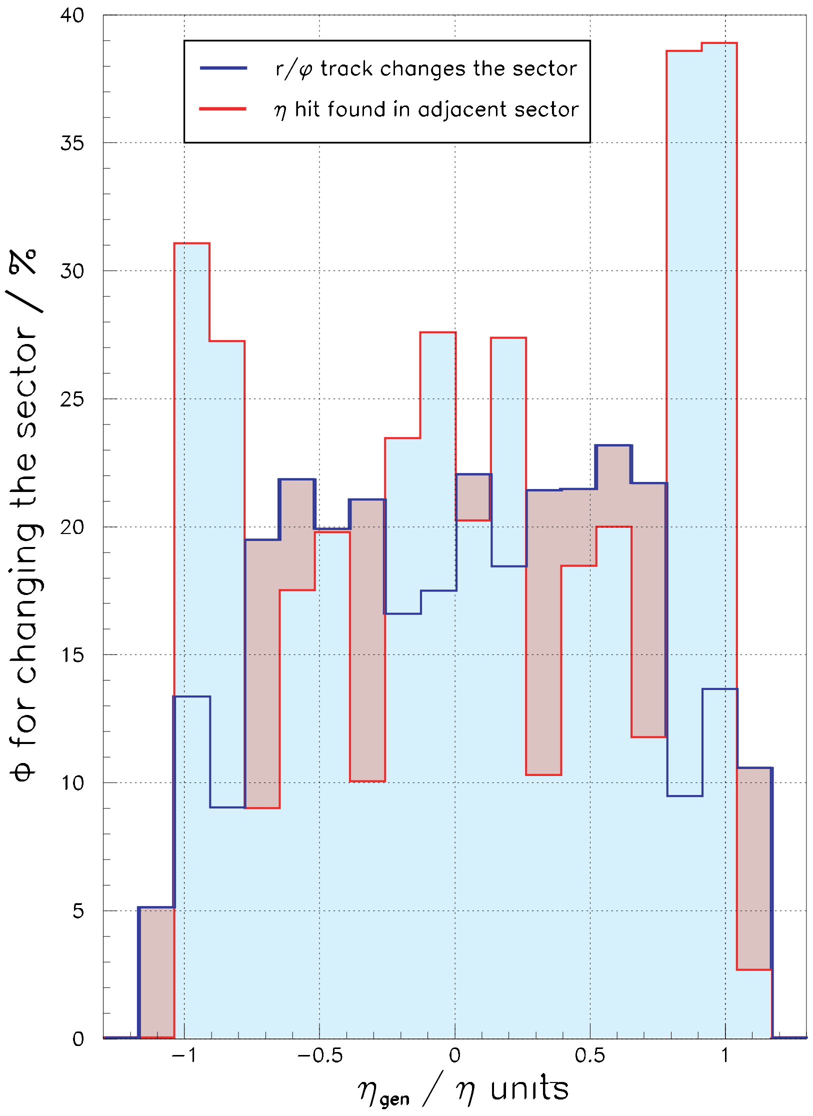


Figure 11: The probability (F) for a track to change the sector is shown for both projections. The probability for tracks found in (r/φ) and (r/z) is compared and plotted versus the generated η value

5. Simulation Results

In order to calibrate and test the performance of the η Assignment Unit a sample of one million positively and negatively charged single muon events was generated. The muon sample is flatly distributed in the following parameters:

- p_T: (3.5 ... 100.0) GeV
- φ: (0.0 ... 2π) rad
- θ: (0.0 ... π) rad

Since the input to the η Track Finder is restricted to a single wedge, some tracks can neither be found nor matched. The probability (Φ) for changing the sector in both views is shown and compared in Figure 11.

Most of these tracks have a track segment or a hit in an inner station. They are not lost for matching because they can be reconstructed using the track data of the remaining chambers. However, depending on the minimal η track quality to be required and the matching scheme applied, they can be rejected. In the actual scheme, using only one LTRG for a valid η track, the impact of lost tracks is very low as is shown in Tab. 6.

Table 6: Probabilities for a track changing the sector between different stations in the case of the (r/φ) Track Finder (a). If a track changes the sector in (r/φ) it is checked whether the track in (r/z) changes the sector as well (b) and the loss for the matching is calculated (c).

Between stations	(a) Probability for a (r/φ) track / %	(b) Probability for a (r/ z) track / %	(c) matching loss / %
1 – 2	5.86	19.35	2.17
2 - 3	1.45	0.11	0.04
3 - 4	6.55	0.00	0.04

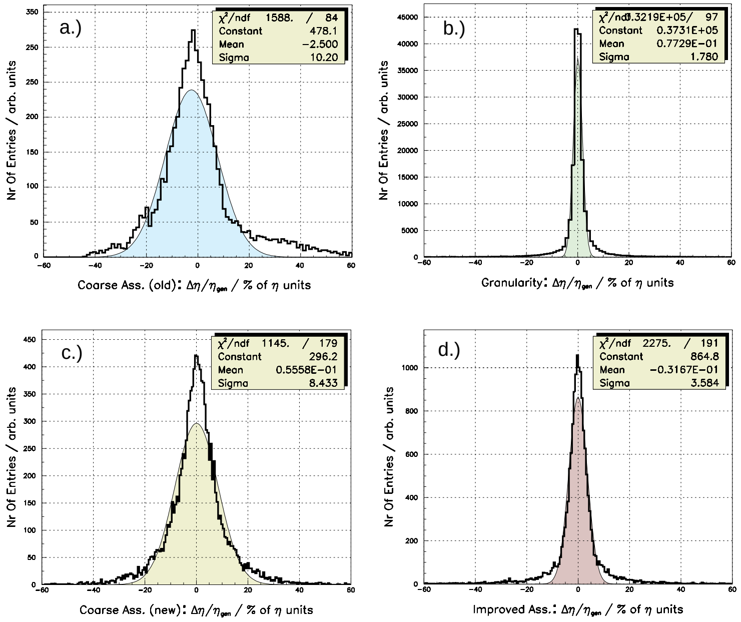


Figure 12: Resolution plots for four different cases:
a.) The resolution of the η assignment as it is currently implemented in the detector simulation.
b.) The resolution of the θ superlayer using the maximal granularity of 64 different η values per chamber.
c.) The new coarse η assignment using all combinations of the chambers used from a track found in (r/φ).
d.) The improved η assignment using 8 different η values coming from each chamber and calculating 64 different η values.:

It has to be kept in mind that the percentage of lost tracks increases if the EAU is forced into a higher quality scheme. This can have a considerable impact if the EAU would be used for other purposes than the η assignment, e.g. the suppression of non-prompt muons. Adding the input from adjacent wedges - at least from station two - would then certainly be an improvement.

5.1. Resolution Studies

The maximal resolution that the EAU could theoretically achieve is the intrinsic resolution of the muon chamber, corresponding to approximately 64 different values as shown in Figure 12.b. Since these 64 values are OR-connected to only eight different values per chamber, this basic resolution decreases and sets already a limit for the EAU.

The η assignment, which was using only the information of the station where the track changed the wheel, is improved by a new coarse scheme using all combinations of chambers to assign an η value to the track. This scheme, which uses partially overlapping η ranges, offers a better resolution and is important for the matching. To summarize, Figure 12 shows a comparison between the resolutions for the system granularity, the old coarse assignment and both the new coarse and improved η assignments.

Not only the number of hits delivered by each chamber influences the resolution of the η assignment, but also the number of different η values which are finally assigned. This occurs because the simulation can produce more patterns than fit into the maximal number of different η values. Since the algorithm is implemented in hardware, only numbers mapping in a bitwise scale are of interest. At the moment the number of output bits is set to 6, resulting in a maximal number of 64 different η values for the whole η region. Figure 13 shows the influence of the different granularities of the assigned η scale on the η resolution.

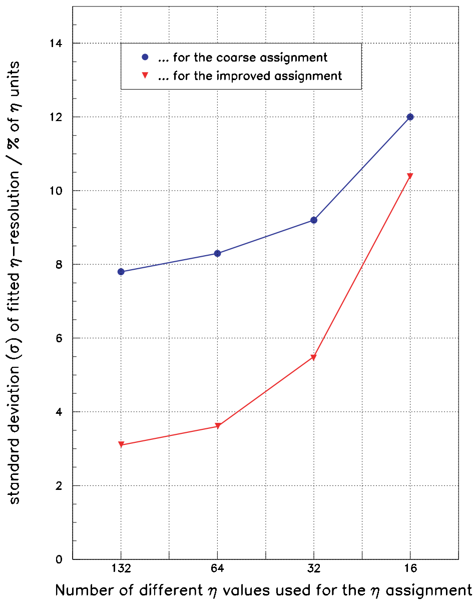


Figure 13: Depending on the number of assigned η values the resolution of the η assignment changes. The chosen 6 bit η range - corresponding to 64 different η values - results in an acceptable η resolution

Thus, the choice of a 6 bit η range corresponds to an insignificant loss in resolution, while keeping the hardware implementations reasonable.

Figure 14 shows the η resolution achieved by the η Track Finder as a function of the generated η values. The improved resolution - using the middle superlayer of the muon stations (circles) - is compared to the raw resolution achieved with the coarse η assignment (triangles).

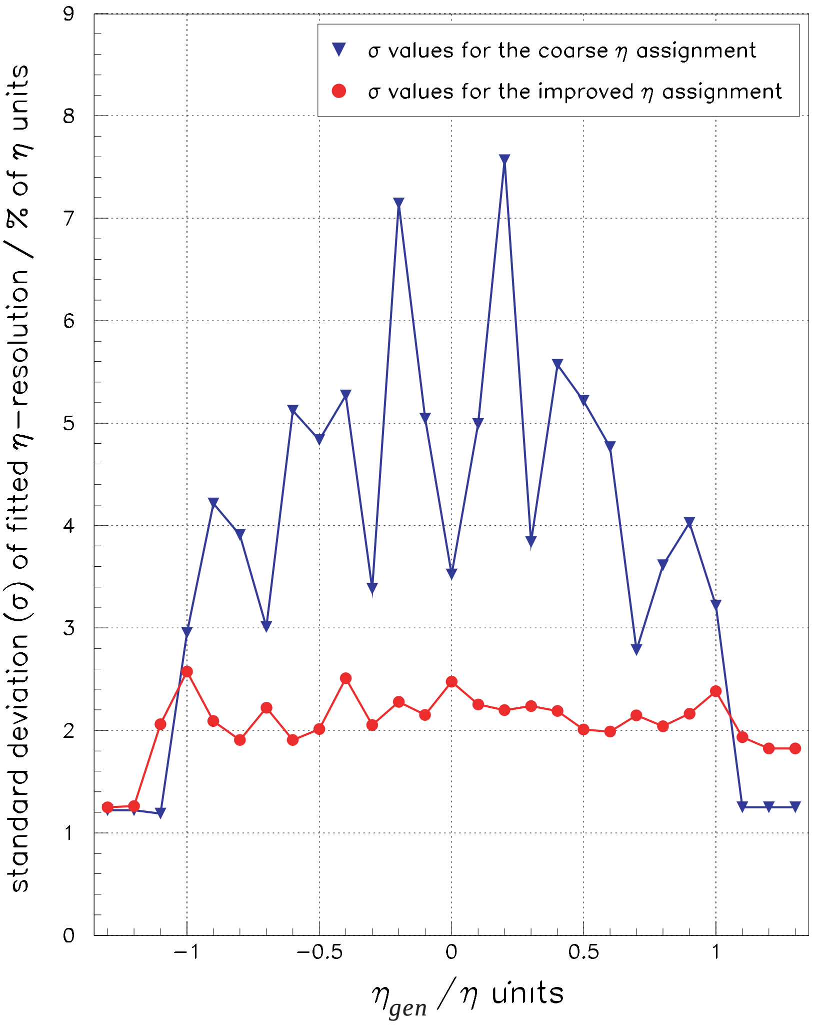


Figure 14: Pseudorapidity resolution versus the η coordinate for both the coarse (blue triangles) and the improved (red circles) η assignment

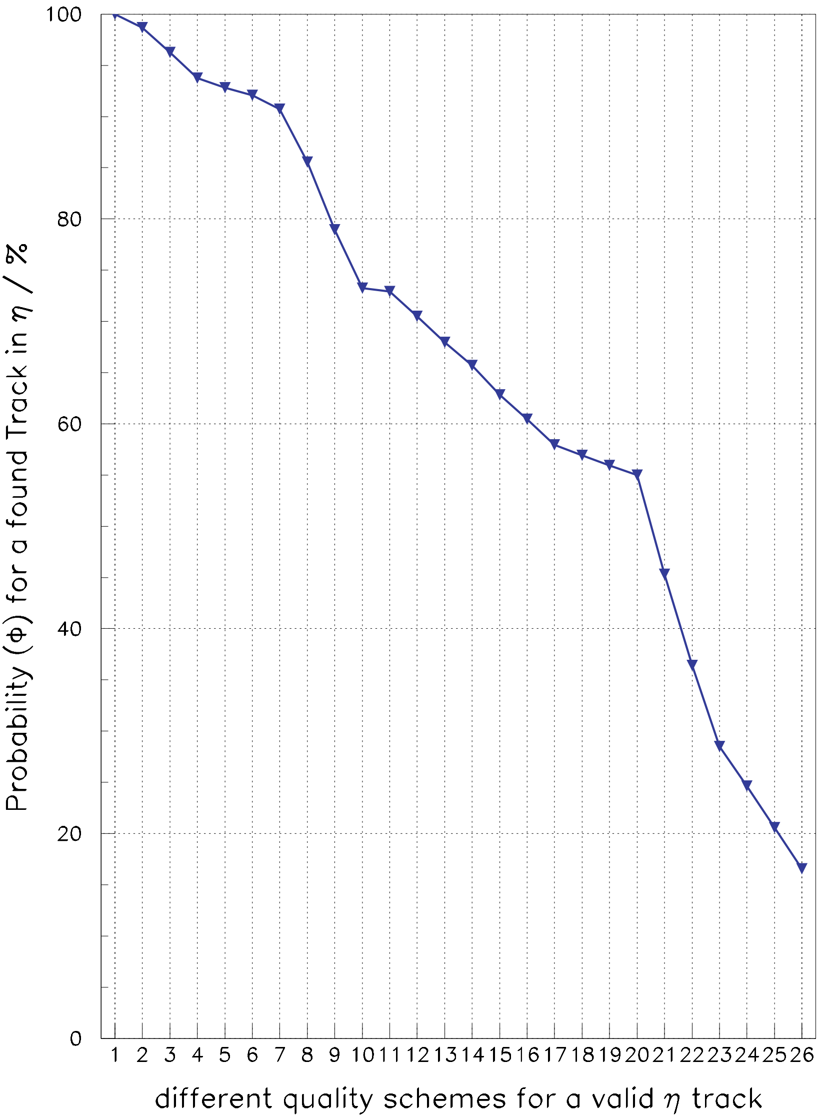


Figure 15: η Track Finder efficiency versus different track qualities

5.2. *Track Finding Efficiency*

According to the definition of the different η track qualities Figure 15 shows the probability to find an η track with respect to the minimal η track quality used in the EAU simulation. The efficiency drastically decreases when higher quality tracks are required. By comparing efficiency with different track classes it can be shown that requiring more than a LTRG for a valid track introduces inefficiencies of the order of 10% and higher to the EAU (c.f. Figure 16).

Therefore, it is strongly recommended to use an η Track Finder method in which a LTRG is allowed to be a valid track, as long there is no improvement in the efficiencies of the chambers or any other change in the muon detector system performance.

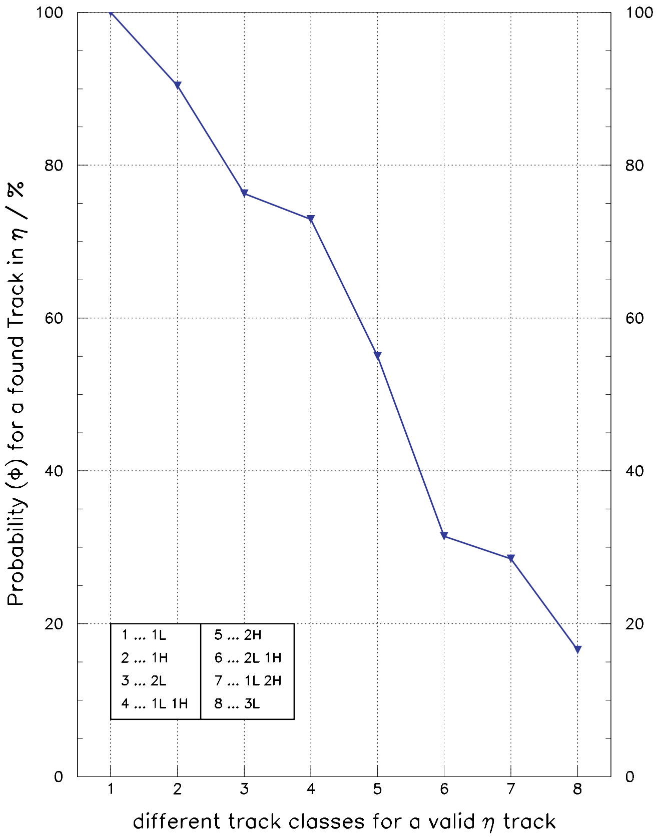


Figure 16: η Track Finder efficiency versus different track classes

5.3. Matching Efficiency

Figure 19 presents the calculated standard deviation (σ) of the η resolution for each η assignment scheme. The resolution improves with a higher matching probability because of the large fraction of improved η values assigned. Studies on the mis-matching probability - i.e. a wrong η track is matched to the (r/φ) track - due to multiple scattering or delta rays have been performed, but they did not show a significant impact. Anyhow, the assignment scheme can easily be tuned in case of a much higher expected background rate which influences the η resolution.

To illustrate the same dependence in a more compact way, Figure 20 shows the matching efficiencies as a function of the different track classes. In this plot the resolution values almost remain constant, because the requirement for a certain track class contains different track qualities.

In order to reach a highly efficient η assignment, the best choice is the so called “virtual matching” scheme, by comparing only the η value of the improved pattern to the corresponding η range of the coarse pattern. In case of the minimal quality scheme for the η track finding, a single LTRG is enough for efficient track reconstruction. Therefore the calculated matching efficiency reaches 100 percent if a match is possible in principle. Another option for the matching is to use at least one chamber to reconstruct the track in both projections. This causes inefficiencies, however, as can be seen in Figure 19 to Figure 18.

Figure 17 shows the dependence of the matching efficiency versus the transverse momentum p_T. Figure 18 compares the matching efficiency to the generated pseudorapidity η.

To summarize, an optimal algorithm implementation can be set using a single LTRG as minimal η track quality and using the “virtual matching” for the comparison between the two Track Finders. In this case all possible tracks will be matched to the (r/φ) track candidate and the best possible resolution will be obtained.

6. Conclusion

An algorithm to assign pseudorapidity values to muon tracks found in the barrel by the Level-1 Trigger of the CMS experiment has been developed. Detailed simulation studies have been performed in order to determine the choice of its parameters. The feasibility of the hardware implementation has also been validated.

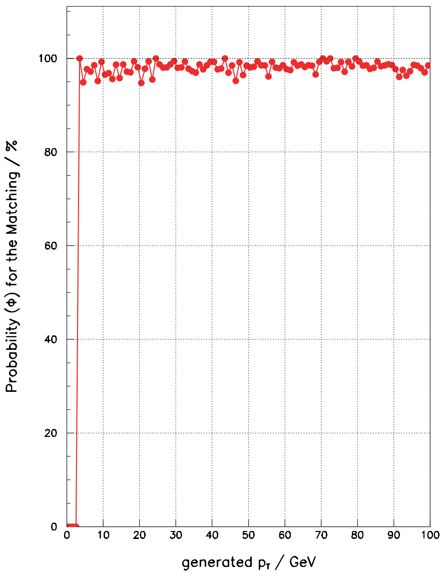


Figure 17: p_T dependence on the matching efficiency, requiring at least one LTRG for an η track and at least one chamber to have one TS in (r/φ) and one in (r/z) for a successful match

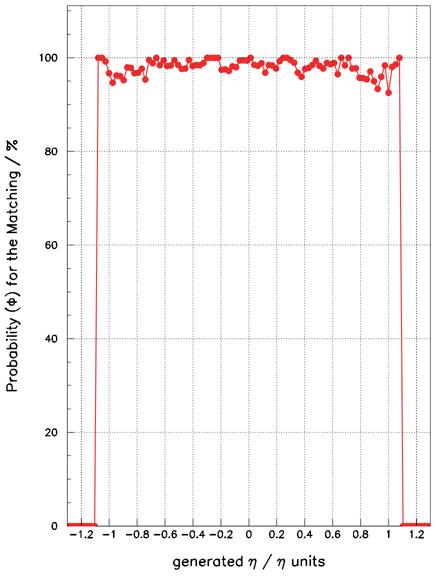


Figure 18: η dependence on the matching efficiency, choosing one LTRG to be required for an η track and at least one chamber to have one TS in (r/φ) and one in (r/z) for a successful match

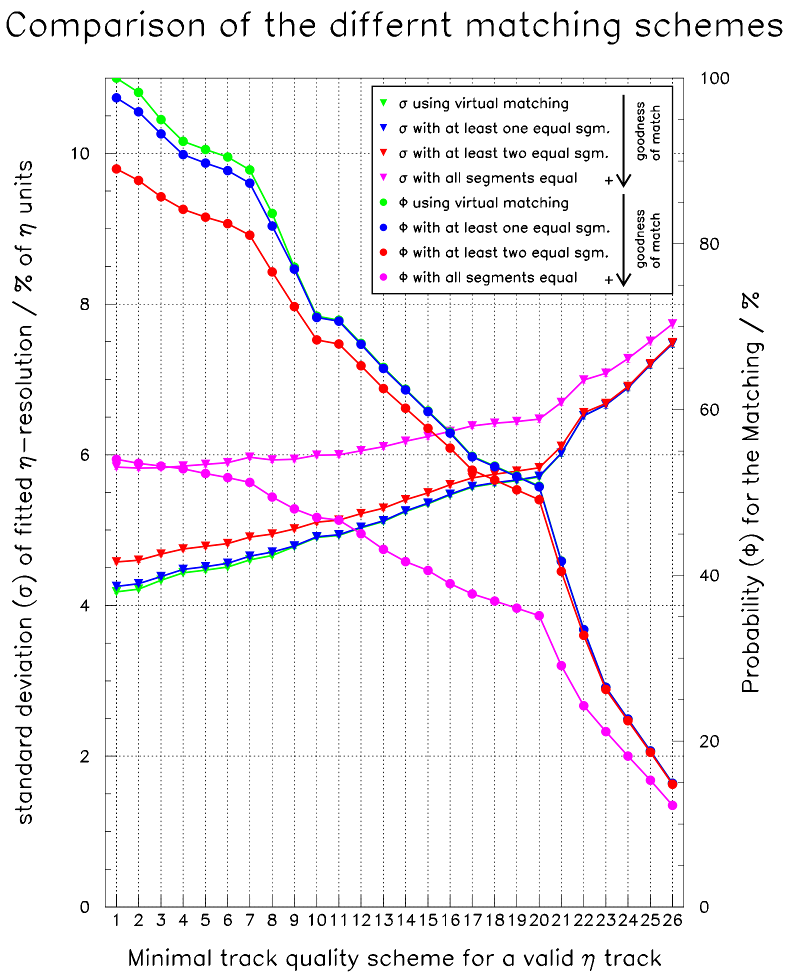


Figure 19: Using the calculated standard deviation the resolution for each assignment scheme and its impact on efficiency and overall resolution can be compared (for the definition of the matching schemes see Tab.4)

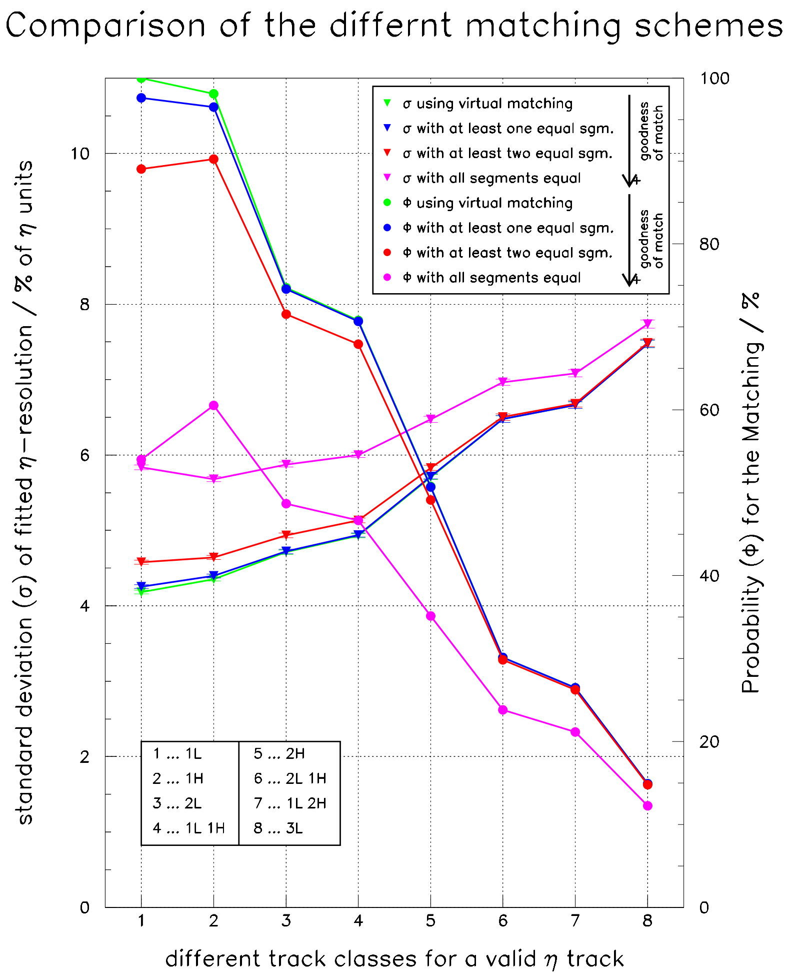


Figure 20:Using the calculated standard deviation, the resolution for each track class and its impact on efficiency and overall resolution can be compared (for the definition of the matching classes see Tab. 3)

7. Acknowledgements

The authors gratefully acknowledge support by their colleagues, especially J. Erö, who is in charge of building the η Assignment Unit electronics and whose technical advice was essential in order to succeed in the development.

8. References

[1] CMS Technical Proposal, CERN/LHCC 94-38 (1994).

[2] The CMS Magnet Project - Technical Design Report, CERN/LHCC 97-10 (1997).

[3] The CMS Tracker Project - Technical Design Report, CERN/LHCC 98-6 (1998).

[4] Addendum to the CMS Tracker Technical Design Report, CERN/LHCC 2000-016 (2000).

[5] The CMS Electromagnetic Calorimeter Project - Technical Design Report, CERN/LHCC 97-33 (1997).

[6] The CMS Hadron Calorimeter Project - Technical Design Report, CERN/LHCC 97-31 (1997).

[7] The CMS Muon Project - Technical Design Report, CERN/LHCC 97-32 (1997).

[8] Claudia-E. Wulz, Concept of the First Level Global Trigger for the CMS Experiment at LHC, CERN CMS Note 2000/052 (2000), in print at Nucl. Instr. Meth. A.

[9] A. Taurok, H. Bergauer, M. Padrta, Implementation and Synchronisation of the First Level Global Trigger for the CMS Experiment at LHC, CERN CMS Note 2000/ 057 (2000), in print at Nucl. Instr. Meth. A.

[10] M. Andlinger et al., Bunch Crossing Identification at LHC using a Mean Timer Technique, Nucl. Instr. Meth. A336 (1993) 91-97.

[11] M.Kloimwieder, Improving the • Assignment of the DTBX Based First Level Regional Muon Trigger, CMS Note 1999/054 (1999).

[12] CMS Collaboration, C M S The Trigger and Data Acquisition project, Volume I. The Level-1 Trigger, CERN LHCC 2000-038 (2000).

[13] Markus Brugger, Software Description of the Standalone η Track Finder Simulation Package, CERN, (2000) http://wwwhephy.oeaw.ac.at/p3w/cms/trigger/muonTrigger/EAU/Simulation/-SimuDocu/start.html.

Chaotic ionization of non-hydrogenic alkali Rydberg states

Andreas Krug[1,2]
Andreas Buchleitner[1]
[1] Max-Planck-Institut für Physik komplexer Systeme, Dresden,
[2] Max-Planck-Institut für Quantenoptik, Garching

Abstract

We report the first ab initio quantum treatment of microwave driven alkali Rydberg states. We find that nonhydrogenic atomic initial states exhibit fingerprints of classically chaotic excitation, and identify the cause of their experimentally observed enhanced ionization, as compared to Rydberg states of atomic hydrogen.

Rydberg states of atomic hydrogen are the perfect quantum analog of classical planetary motion. They incarnate the correspondence principle via the equivalence of the classical principal action and the principal quantum number n_0, as well as of the classical Kepler frequency and the local energy spacing in the Rydberg progression. Also under strong periodic driving do they exhibit the essential features of the underlying classical dynamics [1], which go chaotic as the driving field amplitude F is increased. However, classically chaotic motion is tantamount to the destruction of symmetries and, hence, of good quantum numbers in the quantum system, what prevents us from identifying a single quantum state with a single chaotic classical trajectory. Therefore, a large number of quantum states, i.e. a large spectral density is needed as a prerequisite for quantum dynamics to resolve the intricate phase space structures of classically chaotic dynamics [1]. Furthermore, the periodic driving induces multiphoton transitions of variable order between atomic bound and continuum states, which finally gives rise to the ionization of the atom [2]. Indeed, it is the resulting ionization yield which is typically used as the experimental probe of the chaotic bound state dynamics [3].

In laboratory experiments, it is state of the art to produce atomic Rydberg states with principal quantum numbers n_0 between 20 and 120 [3, 4, 5], and, hence, large spectral densities (scaling as n_0^5). However, for driving frequencies in the microwave regime which are near resonant with the transition $n_0 \simeq 57 \to n_0 + 1$ [6], chaotic ionization is mediated by multiphoton transitions of the order $k = 1/2n_0^2\omega \geq 29$ (simply given by the ratio of the initial state ionization potential to the photon energy). An exact theoretical/numerical treatment of the corresponding quantum mechanical eigenvalue problem defined by the T-periodic Hamiltonian

$$\begin{aligned} H(t) &= \frac{\vec{p}^2}{2} + V_{\text{atom}}(r) + \vec{r}\cdot\vec{F}(t),\ r > 0, \qquad (1) \\ & \vec{F}(t+T) = \vec{F}(t),\ T = \frac{2\pi}{\omega}, \end{aligned}$$

in dipole approximation and using atomic units, with ω and $\vec{F}$ the driving field frequency and amplitude, remained untractable so far, simply due to its dimension [2]. Furthermore, for reasons of experimental convenience, a considerable part of experiments was performed using Rydberg states of alkali [4, 5, 7] rather than of hydrogen atoms [3, 6], starting out from the hypothesis that the driven dynamics of the highly excited Rydberg electron should not undergo any relevant changes due to the presence of a multielectron core which modifies the potential $V_{\text{atom}}(r)$ only in the immediate vicinity of the nucleus. All available data prove, nontheless, the opposite: nonhydrogenic alkali Rydberg states appear to ionize at considerably lower driving field amplitudes than Rydberg states of atomic hydrogen [3, 5, 7]. This assertion, however, is hitherto based on a somewhat questionable comparison of the available hydrogen and alkali data, since relevant experimental parameters such as the atom-field interaction time or the driving field frequency are typically different for different experiments. As the experimentally observed ionization yield was shown to be explicitely time dependent [5], and since the scale invariance of the classical dynamics of the periodically driven two-body Coulomb problem prevails – *a priori* – only in the driven hydrogen atom [3, 9] (where it allows to realize the same classical ionization scenario for different driving field frequencies ω), the experimentally observed enhanced ionization of nonhydrogenic states of alkali atoms remains a puzzle for more than one decade [3, 5]. Only a (laboratory or numerical) experiment which allows for the exclusive change of the atomic species (from hydrogen to alkali), fixing *all* other experimental parameters, can settle this longstanding problem. Yet, whilst laboratory experiments on alkali atoms are easier performed than on atomic hydrogen, the converse is true for a theoretical/numerical treatment. The alkali ionization problem is complicated by the fact that it additionally involves quantum mechanical scattering of the Rydberg electron off the multi-electron core [8, 9], on top of the mere size of the eigenvalue problem defined by (1).

Only with the advent of the most powerfull supercomputers, which provide us with the means to treat the described atomic ionization process in its full complexity, without any essential approximations nor adjustable parameters, are we now able to tackle this challenging problem. It is the purpose of this letter to present results of the first numerical experiment on the microwave ionization of non-hydrogenic states of lithium. These are compared to laboratory results on atomic hydrogen, obtained under identical experimental conditions.

Combining group theoretical methods for the description of the atomic degrees of freedom (including the continuum), R-matrix theory to include the non-vanishing quantum defects induced by the multielectron core [9], the Floquet theorem (a generalized version of the Bloch theorem familiar from solid state physics) to account for the periodic driving, complex dilation to extract the energies and ionization rates of the atomic eigenstates in the field [2], and an intelligent, parallel implementation of the Lanczos algorithm with excellent scalability, we have been able to perform the numerical experiment on nonhydrogenic $\ell_0 = 0$ angular momentum Rydberg states of lithium, for a linearly polarized microwave field, i.e., $\vec{r} \cdot \vec{F}(t) = Fz\cos(\omega t)$ in (1). We precisely chose the experimental parameters employed in laboratory experiments on atomic hydrogen [6], with the *only exception* of a finite quantum defect of the $\ell = 0, 1, 2, 3$ angular momentum states, and a well-defined atomic initial state $\mid n_0,\ \ell_0 = m_0 = 0\rangle$, with m_0 the angular momentum projection along the fixed field polarization axis. Furthermore, we assumed a constant microwave amplitude F experienced by the atoms, thereby neglecting pulse-induced switching effects. These, however, are of minor importance in this kind of experiments [2, 3]. To be specific, the parameters were: the driving field frequency $\omega/2\pi = 36$ GHz, the atom-field interaction time $t = 327 \times 2\pi/\omega$, principal quantum numbers in the range $n_0 = 28, \ldots, 80$, and lithium quantum defects $\delta_{\ell=0} = 0.39947$, $\delta_{\ell=1} = 0.04726$, $\delta_{\ell=2} = 0.00213$, $\delta_{\ell=3} = -8 \cdot 10^{-5}$ [10]. Correspondingly, the order of the multiphoton process to ionize the atomic initial state extends from $k \geq 15$ (for $n_0 = 80$) to $k \geq 120$ (for $n_0 = 28$), where we account for the shift of the effective ionization threshold to a finite value of the principal quantum number, $n_c \simeq 105$, due to the finite size of the basis used for our numerical computations [2]. Note that such a shift also occurs in the laboratory experiment, due to unavoidable stray fields [3].

All calculations were performed on the CRAY T3E Garching (for $n_0 \geq 40$), and on the HITACHI SR8000-F1 Munich (for $n_0 < 40,\ k > 50$) – the latter one occupies position 1 in the international ranking [11] of top-level parallel computing facilities in the academic realm. To obtain converged low-n_0-results, the availability of the HITACHI SR8000-F1 was absolutely crucial. As an example, the eigenvalue problem which describes the ionization of $\mid n_0 = 28,\ \ell_0 = m_0 = 0\rangle$, for a given F, is defined by a banded, complex symmetric matrix of dimension 1000000 and width 6000. Approx. 4000 eigenvalues and eigenvectors have to be generated to gain all relevant spectral information, for the final calculation of the ionization yield.

Fig. 1 shows a comparison of our numerical experiment to laboratory data obtained in the Stony Brook group [3, 6]. The plot uses "scaled variables" [1] F_0 and ω_0, i.e. the driving amplitude F and the driving frequency ω are measured in units of the Coulomb force ($\sim n_0^{-4}$) and of the Kepler frequency ($\sim n_0^{-3}$) along the unperturbed classical two-body Coulomb trajectory. Plotted is the ionization threshold field $F_0(10\%)$, i.e. the field amplitude needed to induce an ionization probability of 10% at given interaction time t and frequency ω, for different principal quantum numbers n_0 [1, 2, 3, 4, 5]. In a classical description of the driven two-body Coulomb problem, F_0 and ω_0 completely determine the classical phase space structure, what allows for an immediate interpretation of the experimental hydrogen data in Fig. 1 in terms of the underlying classical dynamics [3]. Most prominently, the hydrogen ionization threshold decreases almost monotonously in the frequency windows (II) and (III), and exhibits local maxima around $\omega_0 \simeq 1.0 \ldots 1.3$ and $\omega_0 \simeq 2.0 \ldots 2.5$, on top of a global increase in the frequency window (I). The decrease in (III) and (II) essentially follows the classical ionization threshold which can be extracted, e.g., from classical 3D Monte Carlo simulations of the hydrogen experiments [3], whereas the global increase in (I) is a signature of dynamical localization [5, 6], the analog of Anderson localization in the quantum transport of driven, chaotic, low-dimensional Hamiltonian systems [12]. The local maxima around $\omega_0 \simeq 1.0$ and $\omega_0 \simeq 2.0$ are a quantum signature of locally integrable [3] or "sticky" [13] classical dynamics, within or in the vicinity of nonlinear resonance islands in classical phase space.

Comparison of the laboratory to the numerical experiment immediately leads to the following observations:

1. In the scaled frequency window (I), the ionization thresholds of nonhydrogenic Rydberg states of lithium follow the global trend of the hydrogen data, in particular with the *same* absolute values of the driving field amplitude. Furthermore, the lithium data exhibit local maxima at $\omega_0 \simeq 1.1$ and $\omega_0 \simeq 2.2$, very much as the hydrogen data.
2. There is a dramatic difference between hydrogen and lithium thresholds in the frequency window (II), where the lithium thresholds continue to

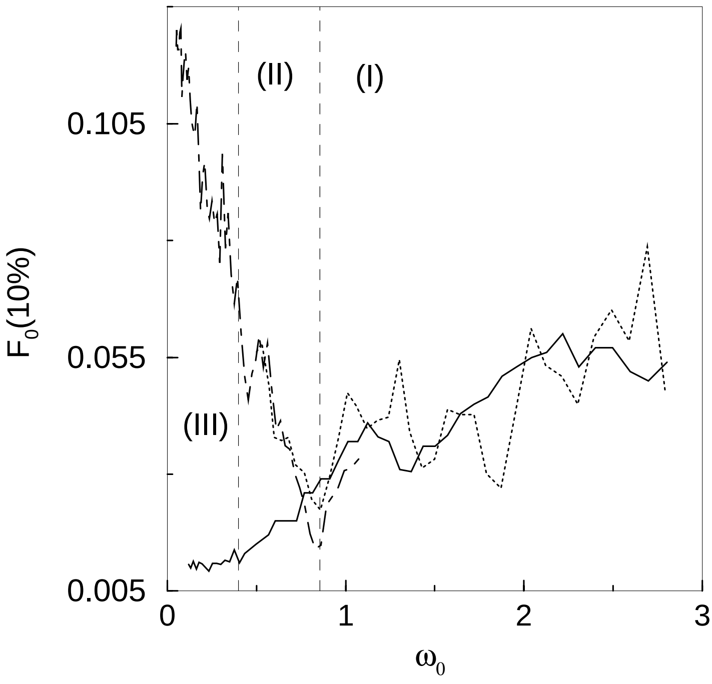


Fig. 1: Experimentally measured, scaled ionization threshold fields $F_0(10\%) = F(10\%)n_0^4$ of atomic hydrogen (dotted [6] and dashed-dotted [3] curves), compared to our numerical experiment on the $| n_0, \ell_0 = m_0 = 0\rangle$ state of lithium (full curve), as a function of the scaled frequency $\omega_0 = \omega n_0^3$. The dotted curve was obtained under precisely *the same conditions as the lithium data, i.e. at frequency $\omega/2\pi = 36$ GHz, interaction time $t = 327 \times 2\pi/\omega$, and principal quantum numbers within the range $n_0 = 28 \dots 80$. Only ℓ_0 and m_0 were not well-defined in the laboratory experiment, but microcanonically distributed over the energy shell [3, 6]. Furthermore, the atoms experienced a finite rise and fall time of the microwave pulse in the laboratory (approx. 80 field cycles each [6]), whereas the numerical experiment assumes a constant value of F. These finite switching times of the microwave pulse are responsible for the more pronounced structures of the laboratory data as compared to the lithium results, but do not affect the globally very good agreement in the ω_0-interval (I). We also reproduce the experimental results obtained with $\omega/2\pi = 9.923$ GHz [3], to cover the low-ω_0 regime for hydrogen. (I), (II), and (III) distinguish scaled frequency ranges where $\omega > \Delta_{\rm hyd}$, $\Delta_{\rm alk} < \omega < \Delta_{\rm hyd}$, and $\omega < \Delta_{\rm alk}$, respectively, compare Fig. 2.*

increase with ω_0, whereas the hydrogen thresholds decrease from rather high values, in accordance with classical results [1, 3].

3. The scaled lithium thresholds are essentially n_0-independent at very low values in the interval (III), whereas hydrogen exhibits (classical) thresholds approx. 10 to 20 times as large as for lithium, in this low-frequency regime.

The clue for understanding these features lies in the level structure of the unperturbed alkali spectrum which is illustrated in Fig. 2: For driving frequencies ω larger than or comparable to the energy difference $\Delta_{\rm hyd}(n_0) \sim n_0^{-3}$ between neighbouring hydrogen manifolds, and this is precisely in the regime (I), the driving field can efficiently mix hydrogenic and nonhydrogenic levels of the lithium atom, even for initial atomic states with nonvanishing quantum defect. Since in this frequency domain the essential symmetry properties of the driven two-body Coulomb problem below the ionization threshold prevail in the alkalis [9], also the transition to chaotic transport in the classical dynamics (i.e., the destruction of these symmetries) dominates the ω_0- or n_0-dependence of the ionization threshold of nonhydrogenic alkali Rydberg states, *despite* their manifestly non-classical character due to the scattering of the Rydberg electron off the multielectron core [8].

For driving frequencies ω larger than the closest, energy gaining dipole transition frequency $\Delta_{\rm alk}(n_0)$ starting out from the nonhydrogenic initial state, but smaller than $\Delta_{\rm hyd}(n_0)$ (regime (II)), the driving field is still capable to efficiently couple different hydrogenic and nonhydrogenic states, whereas no comparable excitation mechanism is available in the hydrogen atom. Consequently, the general dependence of $F_0(10\%)$ on ω_0 is unaltered for lithium in this regime, whereas the hydrogen threshold is now determined by the classical ionization process in the absence of near-resonant quantum mechanical transition amplitudes [1]. Accordingly, the alkali data display the general trend of increasing threshold with ω_0, i.e. the signature of dynamical localization [6], over both frequency intervals (I) and (II), where only the former has a classical counterpart, not the latter. As a matter of fact, all currently available experimental data which exhibit dynamical localization in the ionization of nonhydrogenic alkali Rydberg [5, 7] states have been obtained in regime (II) and (III), what explains the apparent discrepancy (reaching a factor 10) between hydrogen and alkali thresholds to the largest extent (A second, though secondary, reason are the systematically longer atom-field interaction times in the alkali as compared to the hydrogen experiments, which account for a correction of the observed threshold value which depends algebraically on t [5]. We can easily change the interaction time in our numerical experiment and verified that an increase of t by a factor 10 does not affect the qualitative difference observed in regime (II).).

Finally, at driving frequencies $\omega < \Delta_{\rm alk}(n_0)$, even the alkali spectrum doesn't offer any atomic transition which allows for an efficient one-photon-coupling of the initial atomic state to any other Rydberg level. This is clearly reflected by the transition from decreasing to essentially constant scaled threshold fields, which defines the border line between (II) and the low-frequency range (III) in Fig. 1. Only higher-order processes can induce efficient depletion of the initial state population, with subsequent ionization. The depletion rate of the nonhydrogenic initial state via a m-photon upward transition to the next Rydberg level can be estimated perturbatively, leading to a scaling of the threshold field as $F(10\%) \sim n_0^{-\alpha}$, where α grows from 4 to 4.6 for m increasing from 2 to 3. Such an estimation is consistent with our lithium data in Fig. 1, which cover the range from $n_0 = 39$ $(m = 2)$ to $n_0 = 28$ $(m = 3)$, in regime (III). However, it does not yet provide a very satisfactory picture of the ionization process in the low-frequency-domain, since m is rapidly changing with n_0, and no reliable estimation of the absolute value of $F(10\%)$ (depending on the coherent interplay of multiphoton transition amplitudes of different order connecting $| n_0 \, \ell_0 \, m_0\rangle$ to the continuum [14]) is included.

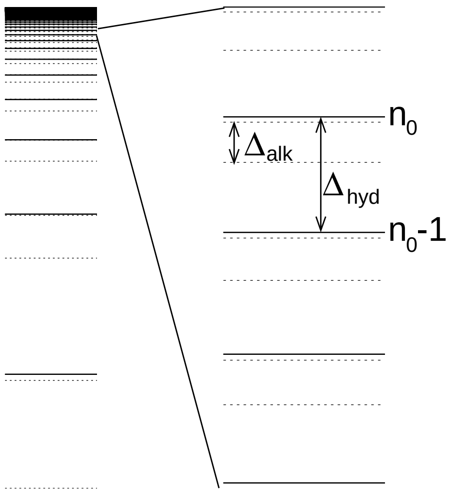


Fig. 2: Hydrogenic (full lines) and nonhydrogenic (dotted lines) energy levels of the unperturbed lithium atom, with a detail of the Rydberg progression on the right. In alkali atoms, the angular momentum degeneracy of the hydrogen spectrum is lifted by the multielectron core. The latter induces nonvanishing quantum defects δ_ℓ [10] of the low angular momentum states, which lead to the apparent energy shift of the nonhydrogenic with respect to the hydrogenic eigenstates. The relevant frequency scales for the ionization process, which define the intervals (I), (II), and (III) in Fig. 1 are $\Delta_{\rm hyd}(n_0)$, the spacing of adjacent hydrogenic manifolds, and $\Delta_{\rm alk}(n_0)$, the energy splitting corresponding to the first dipole-allowed upward transition leaving the nonhydrogenic initial state of the atom.

In summary, we performed an exact numerical experiment which allows – for the first time, in the experimentalist's as well as in the numerical laboratory – for the direct comparison of complex energy transport in Rydberg states of atomic hydrogen and of nonhydrogenic alkali atoms. Whereas laboratory experiments on atomic hydrogen and alkalis always differ by more than the finite quantum defects alone (interaction time, driving field frequency), the numerical experiment now allows for the appreciation of quantum scattering of the Rydberg electron off the multielectron core, at otherwise *precisely identical* experimental conditions. Furthermore, our prediction of a marked transition from similar to distinct ionization dynamics of nonhydrogenic lithium Rydberg states as compared to atomic hydrogen can immediately be verified in the laboratory [4].

Acknowledgements

CPU time was provided by the Rechenzentrum Garching der Max-Planck-Gesellschaft, on a CRAY T3E, and by the Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, München, on a Hitachi SR8000-F1. This letter was published in Physical Review Letters under Phys. Rev. Lett. **86**, 3538 (2001), copyright APS.

References

[1] G. Casati, B. V. Chirikov, D. L. Shepelyansky, and I. Guarneri, Phys. Rep. **154**, 77 (1987).
[2] A. Buchleitner, D. Delande, and J.-C. Gay, J. Opt. Am. Soc. **B12,** 505 (1995).
[3] P. M. Koch, and K. A. H. Leeuwen, Phys. Rep. **255**, 289 (1995).
[4] M. W. Noel, W. M. Griffith, and T. F. Gallagher, Phys. Rev. A **62**, 063401 (2000).
[5] M. Arndt, A. Buchleitner, R. N. Mantegna, and H. Walther, Phys. Rev. Lett. **67**, 2435 (1991).
[6] E. J. Galvez, B. E. Sauer, L. Moorman, P. M. Koch, and D. Richards, Phys. Rev. Lett. **61**, 2011 (1988).
[7] Panming Fu, T. J. Scholz, J. M. Hettema, and T. F. Gallagher, Phys. Rev. Lett. **64**, 511 (1990).
[8] T. Jonckheere, B. Grémaud, and D. Delande, Phys. Rev. Lett. **81,** 2442 (1998).
[9] A. Krug, and A. Buchleitner, Europhys. Lett. **49**, 176 (2000).
[10] C. J. Lorenzen, and K. Niemax, Phys. Scr. **27,** 300 (1983).
[11] http://www.top500.org
[12] S. Fishman, D. R. Grempel, and R. Prange, Phys. Rev. Lett. **49**, 509 (1982).
[13] J. G. Leopold, and D. Richards, J. Phys. B **27**, 2169 (1994).
[14] A. Buchleitner, I. Guarneri, and J. Zakrzewski, Europhys. Lett. **44,** 162 (1998).

SENECA: A Platform-Independent, Distributed and Parallel System for Computer-Assisted Structure Elucidation in Organic Chemistry Titel

Christoph Steinbeck

Max-Planck-Institut für Chemische Ökologie, Jena

ABSTRACT

The program package SENECA for Computer-Assisted Structure Elucidation (CASE) of organic molecules is described. SENECA is written completely in the programming language Java and divided into a server, a client and a gatekeeper part. While the client serves allow for input of spectroscopic information, the server part performs the actual structure elucidation by stochastically walking through constitution space while optimizing the molecule towards agreement with given spectral properties. The convergence is guided by Simulated Annealing. The gatekeeper administers a list of server processes, which can be retrieved by the client. The package is completely platform-independent and its server part can be distributed over the Internet or an intranet using a heterogeneous network of almost any number and type of computers, thus allowing for parallel CASE computations on ordinary networks, present in almost any institution.

Reference:

Steinbeck C: SENECA: A platform-independent, distributed, and parallel system for computer-assisted structure elucidation in organic chemistry.
Journal of Chemical Information & Computer Sciences 2001, 41:1500-1507.

Influence of Head Tissue Conductivity Anisotropy on Human EEG and MEG using Fast High Resolution Finite Element Modeling, based on a Parallel Algebraic Multigrid Solver

Carsten H. Wolters[1,2]
Alfred Anwander[2]
Martin A. Koch[2,3]
Stefan Reitzinger[4]
Michael Kuhn[4,5]
Markus Svensén[2]
[1] Max Planck Institute for Mathematics in the Sciences, Leipzig
[2] Max Planck Institute of Cognitive Neuroscience, Leipzig
[3] Universitätsklinikum Hamburg-Eppendorf, Neurologische Klinik, Hamburg
[4] Institute for Comp. Math., J. Kepler Univ., Linz, Austria
[5] Bruker Saxonia Analytik GmbH, Leipzig

Abstract

Accuracy and time play an important role in medical and neuropsychological diagnosis and research. The inverse problem in the field of Electro- and MagnetoEncephaloGraphy requires the repeated simulation of the field distribution for a given dipolar source in the human brain using a volume-conduction model of the head. High resolution finite element head modeling allows the inclusion of tissue conductivity inhomogeneities and anisotropies. We will present new approaches for individually determining the direction-dependent conductivities of skull and brain white matter, based on non-invasive multimodal magnetic resonance imaging data, and for generating a high resolution realistic anisotropic finite element model of the human head. Error estimations will indicate the necessity of the chosen complex forward model. The finite element approach within the inverse problem leads to a sparse, large scale, linear equation system with many different right hand sides to be solved. The presented solution process is based on a parallel algebraic multigrid method. It is shown that very short computation times can be achieved through the combination of the multigrid technique and the parallelization on distributed memory computers. The iterative solver approach is shown to be stable towards modeling of tissue anisotropy. A solver time comparison to a classical parallel Jacobi preconditioned conjugate gradient method is given.

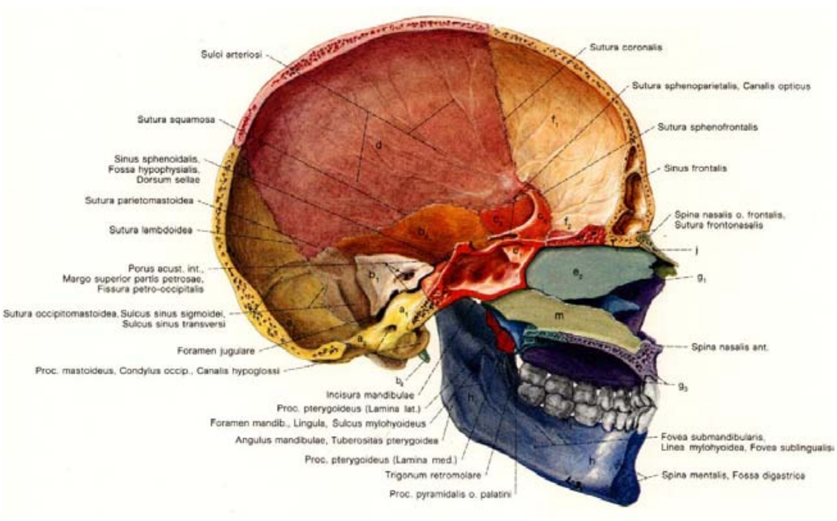
Acknowledgements

This work has been supported by Prof. E. Zeidler, director of the MPI for Mathematics in the Sciences Leipzig, by the Leibniz-Prize of the German Research Foundation awarded to Prof. A. D. Friederici, director of the MPI of Cognitive Neuroscience Leipzig, by the IST-program of the European Commission, project No.10378, SimBio (http://www.simbio.de), and by the Austrian Science Fund – 'Fonds zur Förderung der wissenschaftlichen Forschung' – within the SFB F013 "Numerical and Symbolic Scientific Computing". We are particularly thankful to the following colleagues: S.Burkhardt (Munich University of Technology), F. Kruggel, G. Lohmann and B.Maess (MPI of Cognitive Neuroscience Leipzig).

1 Introduction

Nowadays devices and tools are available for analyzing and monitoring the human brain at a high level of detail. These details are necessary, e.g., for successful surgery or, more generally, for basic brain research. Often computational methods are used in the diagnostic and pre-surgical phase. Such non-invasive tools are of course preferable to invasive methods, e.g., surgery, with high risks for patients. In fundamental brain research, most often there is no other choice besides computational methods. However, the acceptance of tools depends very much on their reliability and robustness and on their speed. In this article it will be shown how non-invasive imaging methods deliver necessary data for such a tool and how advanced numerical methods enhance its accuracy and speed. The article brings together clinical diagnosis, pre-surgical planning, clinical and cognitive research and numerical mathematics, and describes the needed imaging data and the requirements of necessary algorithms and software.

It is common practice in cognitive research and in clinical routine and research to localize current sources in the human brain by means of the induced electric potentials, measured with electrodes which are fixed on the scalp (ElectroEncephaloGraphy, EEG) and/or the induced magnetic flux density, measured in a distance of a few centimeters from the head surface (Magne-



Sulci arteriosi
Sutura squamosa
Sinus sphenoidalis, Fossa hypophysialis, Dorsum sellae
Sutura parietomastoidea
Sutura lambdoidea
Porus acust. int., Margo superior partis petrosae, Fissura petro-occipitalis
Sutura occipitomastoidea, Sulcus sinus sigmoidei, Sulcus sinus transversi
Foramen jugulare
Proc. mastoideus, Condylus occip., Canalis hypoglossi
Incisura mandibulae
Proc. pterygoideus (Lamina lat.)
Foramen mandib., Lingula, Sulcus mylohyoideus
Angulus mandibulae, Tuberositas pterygoidea
Proc. pterygoideus (Lamina med.)
Trigonum retromolare
Proc. pyramidalis o. palatini
Sutura coronalis
Sutura sphenoparietalis, Canalis opticus
Sutura sphenofrontalis
Sinus frontalis
Spina nasalis o. frontalis, Sutura frontonasalis
i
g1
Spina nasalis ant.
g3
Fovea submandibularis, Linea mylohyoidea, Fovea sublingualis
Spina mentalis, Fossa digastrica
d
f1
f2
b1
b2
c
e2
m
a
h

Fig. 1: The human skull: Suture lines and the tri-layeredness [Platzer,1994].

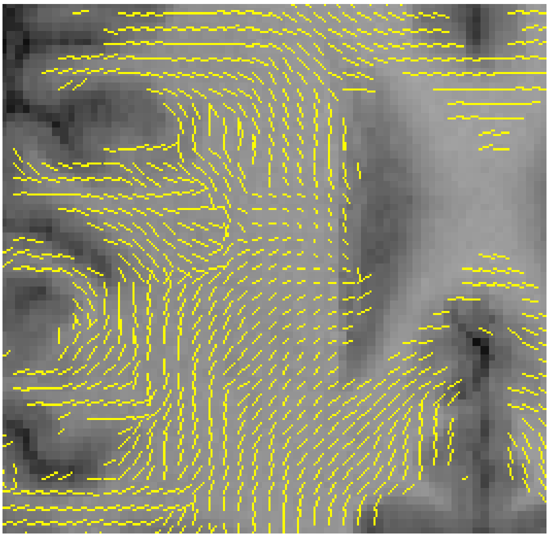


Fig. 2: Fibre orientation map from a DT-MRI experiment [Wolters et al.,1999b]. Eigenvector orientations corresponding to the largest eigenvalue are projected onto the imaging plane, and overlaid on a T1 weighted MRI. Eigenvector directions were suppressed in voxels with $FA < 0.2$ (see Definition for FA in Equation (11))

toEncephaloGraphy, MEG). The localization of the underlying source distribution is an inverse problem whose solution requires the repeated simulation of the electric/magnetic fields in the head for a varying source in the brain (forward problem). For the forward problem, the volume-conductor head has to be modeled. An overview of the head tissues with different conductivities can be found in [Haueisen, 1996]. The human skull consists of a soft bone layer (spongiosa) enclosed by two hard bone layers (compacta). Since the spongiosa have a much higher conductivity than the compacta [Akhtari *et al.*, 2000], the skull shows a direction-dependent (anisotropic) conductivity with an anisotropy ratio of up to 1:10 (radially:tangentially to the skull surface) [Akhtari *et al.*, 2000]. Skull anisotropy was shown to have an impact on the inverse problem in EEG [Marin, 1997; Marin *et al.*, 1998]. Figure 1, taken from [Platzer, 1994], illustrates geometrical features of the human skull.

It is known that brain white matter has an anisotropic conductivity with a ratio of about 1:9 (normal:parallel to fibers) [Nicholson, 1965], but still, no technique exists for a robust and non-invasive direct measurement of conductivity anisotropy in the whole brain. Recently, formalisms have been described for relating the effective electrical conductivity tensor of white matter tissue to the effective water diffusion tensor as measured by Diffusion Tensor Magnetic Resonance Imaging (DT-MRI). Fig. 2 shows a white matter fibre orientation map from a DT-MRI experiment [Wolters *et al.*, 1999b]. [Basser *et al.*, 1994b] introduced the assumption that the effective electrical conductivity tensor shares the eigenvectors with the effective diffusion tensor of water, which can be measured for white matter tissue by DT-MRI. [Tuch *et al.*, 1998; Tuch *et al.*, 1999] proposed a linear relationship between the eigenvalues of both tensors for small intracellular diffusion and high resistivity of the cell membrane. Their proposition is based on a self-consistent differential Effective Medium Approach (EMA) for the generalized dielectric constant (for low frequencies the conductivity) of porous media, derived from a multiple scattering formula from solid state physics developed by [Sen *et al.*, 1981]. [Latour *et al.*, 1994] derived a similar EMA for the effective water diffusion in biological cells. The coupling of both EMA-formulae through the unknown porosity variable led to the linear dependence of the eigenvalues described in [Tuch *et al.*, 1998]. In a first study, white matter conductivity anisotropy was shown to have an influence on EEG and MEG [Haueisen *et al.*, 2002].

In this article, we will present measurement techniques and methods for obtaining a realistically shaped high resolution volume conductor model of the human head in a non-invasive way with anisotropically conducting compartments skull and white matter. Our goal is the study of the influence of tissue anisotropy on EEG and MEG.

A bottleneck for such sensitivity studies towards the different inverse source reconstruction techniques and especially for broad application of high resolution volume conductor modeling to inverse reconstructions in the application fields is the time for calculating the 3D potential distributions during the various forward problems that have to be solved. [Waberski *et al.*, 1998], e.g., conclude that for the achievement of the final goal in epilepsy source localization, i.e., the general clinical use, realistically shaped high resolution head models are necessary and parallel computing has to speed up the computation. Finite Element (FE) models for the electromagnetic field simulation in the head have been developed by various research groups (see e.g. [Bertrand *et al.*, 1991; Haueisen, 1996; van den Broek *et al.*, 1997; Buchner *et al.*, 1997; Awada *et al.*, 1997; Marin *et al.*, 1998]). The FE method is able to treat geometries of arbitrary shape and inhomogeneous and anisotropic material parameters. Generally iterative solvers like the preconditioned Conjugate Gradient (CG) method with conventional preconditioners on single processor machines have been used for the large linear equation system arising from this approach. The hundred or even thousand times repeated solution of such a system with a constant stiffness matrix and varying right hand sides is the major time consuming part within the inverse localization process. These calculation times limited the resolution of the models or, even worse, the broader application of anisotropic FE based head modeling to practical source localization problems got stuck.

Geometric MultiGrid (GMG) methods have proven to be of optimal order with respect to arithmetic costs and memory requirement, see e.g. [Hackbusch, 1985]. In [Jung and Langer, 1991], it was shown that multigrid methods are efficient preconditioners for the conjugate gradient method. For a parallel implementation see for instance [Bastian *et al.*, 1997]. GMG methods suffer from the requirement of a grid hierarchy, which is not available in our case. By contrast, Algebraic MultiGrid (AMG) methods use only single grid information (see e.g. [Ruge and Stüben, 1986; Braess, 1995; Kickinger, 1998; Haase *et al.*, 2000; Reitzinger, 2001] and for parallel versions [Falgout *et al.*, 1999; Krechel and Stüben, 2001; Haase *et al.*, 2002; Wagner, 2000]) while mostly preserving the properties of the geometric version. Many numerical studies have shown a good performance of AMG preconditioners. Furthermore, AMG preconditioners were successfully applied to source localization recently ([Wolters *et al.*, 2000; Johnson *et al.*, 2000]). Even if AMG preconditioned CG (AMG-CG) was shown to be very fast in comparison to standard methods, additional speedup is required. In [Wolters *et al.*, 2002], we described for realistically shaped isotropic high resolution tetrahedra and cube head models how the latter can be achieved by using a parallel computer with a moderate number of processors. Within this article, we will show that our approach is stable towards realistic head tissue anisotropy.

Subsection 1.1 of this article will give an overview of different application fields of source localization and will present an exemplary reconstruction result for Somatosensory Evoked Potentials (SEP). It aims at giving further motivation for readers from outside the bioelectromagnetism area and it can be skipped otherwise. In Section 2, the modeling aspects for the forward problem will be described. Subsection 2.1 gives an overview of a physical model for the source and for the field distribution in the head volume conductor. In 2.2, we will describe the generation process of a realistically shaped anisotropic 5-tissue head model. We focus on the modeling aspect for obtaining an anisotropically conducting skull in 2.2.1 and an anisotropically conducting white matter compartment in 2.2.2, using multimodal Magnetic Resonance Imaging (MRI) data. The section terminates with FE meshing and discretization aspects for EEG and MEG in 2.3. In Section 3, the AMG-CG solver is described as a fast solver for the large linear equation system arising from the FE approach. The partitioning of the meshes and a parallelization strategy for distributed memory computers will then be presented. Section 4 describes the new software developments. In the first part of Section 5, we will present results concerning the influence of tissue conductivity anisotropy on EEG and MEG for various simulated sources. Performance results of the parallel AMG solver and its sensitivity to tissue anisotropy will be discussed in the second part. The parallelized multigrid method will be compared with a parallel Jacobi-preconditioned CG method (Jacobi-CG), which is a well-known solver method in FE source localization. It will be shown that high speedups can be achieved which open the possibility for a broader application of high resolution realistically shaped anisotropic FE based source localization in the human brain. The article ends with the conclusions in Section 6.

1.1 Overview of applications of source localization

This subsection is only meant to be a further motivation and to list some references for readers who would like to know more about the inverse problem and some well-established application fields of EEG/MEG-source localization.

An overview of the different application fields of source localization can be found in [Andrä and Nowak, 1998]. Various inverse reconstruction techniques for continuous and discrete source parameter spaces are described, e.g., in [Scherg and von Cramon, 1985; Buchner *et al.*, 1997; Knösche, 1997; Wagner, 1998; Wolters *et al.*, 1999a; Schmitt and Louis, 2002; Schmitt *et al.*, 2002].

A first example is the study of functional cortical organization by means of evoked fields of the somatosensory system. The different evoked signal components of interest in such studies appear during the first 100 ms post-

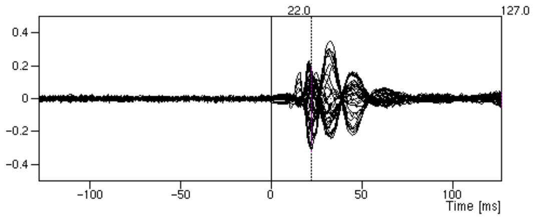


Fig. 3: SEP example dataset, taken from CURRY: Butterfly plot of averaged EEG data from -0.4 *to* $0.4\ \mu V$*. The P22 signal component is marked.*

stimulus. Since the components are well time-locked and not dependent on the attention of the subjects, the signal can be averaged over a large number of trials so that the signal components of interest are obtained with a relatively good signal-to-noise ratio. Figure 3 shows the averaged EEG measurements for SEP in 31 channel butterfly plot from [Fuchs *et al.*, 1998], included as an example dataset in the software package [CURRY, 2000]. As an example for a medically interesting source localization result, the continuous dipole fit method, introduced by [Scherg and von Cramon, 1985], with two dipoles at the peak of the SEP-P22 signal component is shown in Figure 4 (see [Fuchs *et al.*, 1998]). The result has been calculated using the example dataset and methods within [CURRY, 2000]. Source localization methods have also been

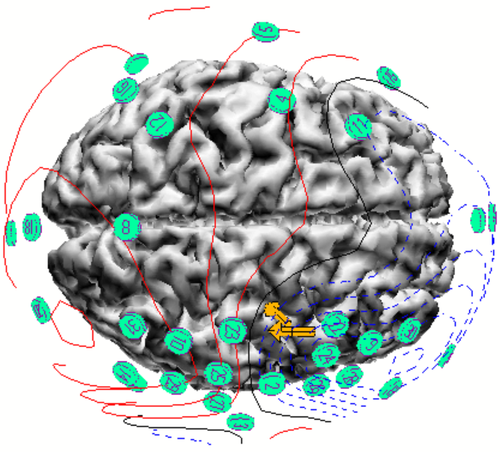


Fig. 4: SEP source localization example, computed with CURRY: Results of the continuous dipole fit method with two dipoles at the peak of the P22 signal component.

introduced to characterize the generators of signals related to higher cognitive function. An example is a recent study showing equivalences between speech and music processing in the brain [Maess *et al.*, 2001].

The non-invasive EEG/MEG-source localization diagnosis method is successfully used in clinical research and application. For instance tumors may distort brain anatomy so that the presurgical localization of sensory or motor areas on the basis of anatomical landmarks is impossible. In [Sutherling *et al.*, 1988], the agreement between invasive and non-invasive methods was evaluated and an "excellent precision of the source localization results" was found. About 0.25 % of the world population suffers from drug-resistant epilepsy and about 10 to 15 % would profit from a surgical removement of the epileptogenic tissue [Andrä and Nowak, 1998]. As opposed to alternative invasive diagnostic procedures, i.e., opening the skull and implanting electrodes near the assumed focus (ECoG surface electrodes or depth electrodes) which put the patient under a considerable risk and is cost intensive, source localization procedures are non-invasive and can give a more "global" overview since the sensors can be placed around the whole head. [Waberski *et al.*, 1998], e.g., found a high congruence of source reconstruction and invasive determination of the focus of epileptiform activity using realistically shaped head models.

2 The forward problem

2.1 Physical modeling

The sources to be localized during the inverse problem and to be modeled in the forward problem are electrolytic currents within the dendrites of the large pyramidal cells of activated regions in the cortex sheet of the human brain. Stimulus-induced activation of a large number of excitatory synapses of a whole pattern of neurons leads to a negative monopole under the brain surface, whereas the cells in rest form a positive monopole quite closely underneath. The stimulus can have various forms, e.g., any visual or auditory stimulus in neuropsychological experiments or an epilepsy- or tumor-induced stimulus as clinical examples. The resulting primary current is generally formulated as a mathematical dipole

$$\vec{j}^p\,(\vec{x}) = \vec{M}\delta(\vec{x} - \vec{x}_0) \tag{1}$$

at the position $\vec{x}_0$ with the moment $\vec{M}$ (see e.g.[de Munck *et al.*, 1988]). The dipole source establishes an electric field $\vec{E}$ and a return current $\sigma\vec{E}$ in the whole head with σ denoting the 3×3 conductivity tensor. The total current distribution $\vec{j}$ in the head is then modeled as

$$\vec{j} = \vec{j}^p + \sigma\vec{E}.$$

Since in the considered low frequency band (frequencies below 1000 Hz) the capacitive component of tissue impedance and the electromagnetic propagation effect can be neglected [Plonsey and Heppner, 1967], the fields are quasistatic and $\vec{E}$ can be expressed as the negative gradient of a scalar potential Φ, so that

$$\vec{j} = \vec{j}^p - \sigma\nabla\Phi.$$

Because the divergence of $\vec{j}$ must be zero, we arrive at the quasistatic approach of Maxwell's equations of electrodynamics

$$\nabla \cdot (\sigma\nabla\Phi) = J^p = \nabla \cdot \vec{j}^p \tag{2}$$

in Ω with appropriate boundary conditions

$$\sigma \frac{\partial\Phi}{\partial\vec{n}}\bigg|_{\Gamma} = 0 \tag{3}$$

with Ω denoting the head, Γ the head surface and $\vec{n}$ the surface normal. Additionally, a reference electrode with given potential is assumed, i.e.,

$$\Phi_{ref} = 0\,. \tag{4}$$

For the magnetic problem, linear material equations are used and it is assumed, that the magnetic permeability, μ, is constant over the whole volume and equal to the permeability of vacuum, so that the following quasistatic Maxwell equation for the flux density $\vec{B}$ is valid [Sarvas, 1987]:

$$\nabla \times \vec{B} = \mu\vec{j}. \tag{5}$$

Since the divergence of $\vec{B}$ is zero, a magnetic potential $\vec{A}$ with $\vec{B} = \nabla \times \vec{A}$ can be introduced and, using Coulomb's gauge $\nabla \cdot \vec{A} = 0$, Equation (5) transforms to

$$\mu\left(\vec{j}^p - \sigma\nabla\Phi\right) = \nabla \times \left(\nabla \times \vec{A}\right) = \nabla\left(\nabla \cdot \vec{A}\right) - \Delta\vec{A} = -\Delta\vec{A}.$$

The source term is vanishing outside the volume conductor, so that the solution of Poisson's equation is [Nolting, 1992]

$$\vec{A}(\vec{x}) = \frac{\mu}{4\pi}\int_{\Omega} \frac{\vec{j}^p(\vec{y}) - \sigma(\vec{y})\nabla\Phi(\vec{y})}{|\vec{x} - \vec{y}|} d\vec{y}. \tag{6}$$

Finally, the magnetic flux Ψ through an MEG magnetometer flux transformer Υ (see, e.g., Figure 13, left) is determined as an integral over the coil area F enclosed by Υ, or, using Stokes theorem, as

$$\Psi = \int_F \vec{B}(\vec{x})d\vec{x} = \oint_{\Upsilon} \vec{A}(\vec{x})d\vec{x}.$$

With

$$\vec{C}(\vec{y}) = \oint\limits_{\Upsilon} \frac{1}{|\vec{x}-\vec{y}|} d\vec{x}, \tag{7}$$

the final equations for primary magnetic flux, Ψ_p, and secondary magnetic flux, Ψ_s, emerging from primary and secondary (return) currents, resp., are given by

$$\Psi_p \overset{(1)}{=} \frac{\mu}{4\pi} < \vec{M}, \vec{C}(\vec{x}_0) > \tag{8}$$

$$\Psi_s = \frac{-\mu}{4\pi} \int\limits_{\Omega} < \sigma(\vec{y})\nabla\Phi(\vec{y}), \vec{C}(\vec{y}) > d\vec{y} \tag{9}$$

$$\Psi = \Psi_p + \Psi_s.$$

[de Munck and Peters, 1993] derived series expansion formulas for problem (2) with boundary conditions (3) and reference potential (4) in order to calculate the potential distribution for a dipolar source in a multi-layer spherical shell model with constant isotropic or anisotropic conductivity values/tensors within each layer. It is now widely known that realistically shaped models of the human head are necessary to keep the localization error at an acceptable level (see e.g. [Waberski *et al.*, 1998]).

2.2 Generation of a realistic 5 tissue anisotropic head model

A prerequisite for a realistic modeling of the volume conductor is the segmentation of head tissues with different conductivity properties. MRI is known as a safe and non-invasive method for imaging the human head and does not expose subjects to radiation load like in Computed Tomography. The identification of the CerebroSpinal Fluid(CSF)-skull boundary based on T1-MRI (T1-weighted MRI) is problematic, and PD-MRI (Proton-Density-weighted MRI) is most appropriate for this task. Figure 5 shows an axial slice of the segmented 5-tissue head model and the corresponding slices of T1- and PD-MRI. The model will be used throughout this article. For the segmentation of skin, white and gray matter surfaces, we only refer to [Wolters, 2002], whereas we will now focus on the description of the modeling for the two anisotropic compartments, skull and white matter.

2.2.1 Generation of an anisotropic skull layer

The exact modeling of the low-conducting anisotropic human skull is of particular importance for EEG source localization [Burkhardt *et al.*, 2002; Huiskamp *et al.*, 1999]. The skull can be seen as an isolating layer which

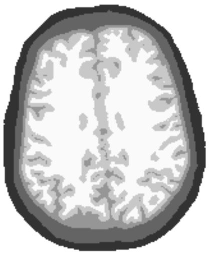
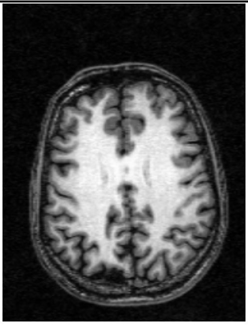
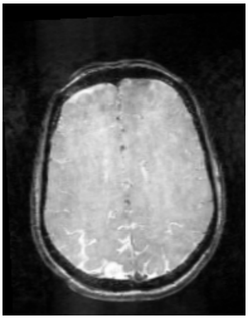
5-tissue model	T1-MRI	registered PD-MRI
		

Fig. 5: Axial slice of the 5-tissue segmentation result and the corresponding slice of the T1-MRI and of the registered PD-MRI.

leads to a strong decrease and a blurring of the potential distribution towards the measurement electrodes.

A first step in the modeling process is the segmentation of inner and outer skull surfaces. We will now shortly summarize our results, see [Burkhardt *et al.*, 2002] for a deeper study. The registration of a bimodal data set is a fundamental step in order to exploit the information in both images in the segmentation process. Inspired by [Maes *et al.*, 1997], we used a voxel-similarity based registration method, yielding high accuracy in matching both modali-

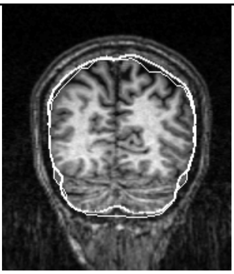
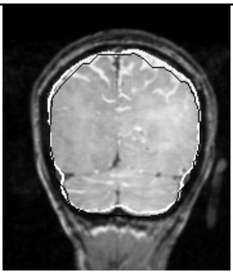
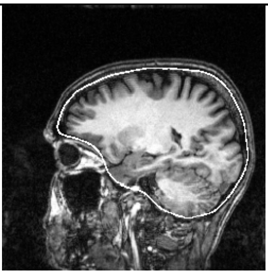
ISS/EISS on T1-MRI	ISS/EISS on PD-MRI	SSSM on T1
		

Fig. 6: Left and middle: Comparison of the segmented Inner Skull Surface (ISS) from bimodal MR images (bold white) and the Estimated Inner Skull Surface (EISS) by means of a T1-MRI based closing and inflation procedure (white on underlying T1 MRI and black on underlying registered PD MRI). Right: Smooth Surface Spongiosa Model (SSSM) on underlying T1-MRI for modeling the eigenvectors of the skull conductivity tensors.

ties [Burkhardt *et al.*, 2002]. Figures 5 and 6 show the PD-MRI, which was registered onto the T1 image. Our segmentation approach for inner and outer skull surfaces uses a combination of basic 3D image operations, a fuzzy segmentation method which compensates for image intensity inhomogeneities, an extended region growing concept and a deformable model, exploiting the registered bimodal data set [Burkhardt *et al.*, 2002]. The bimodal MRI approach was shown to substantially improve the Inner Skull Surface (ISS) segmentation when compared to a T1-MRI based method. The latter estimates the inner skull from a segmentation of the cortical surface. The cortex surface is closed and inflated by a fixed distance via mathematical morphology to provide the Estimated Inner Skull Surface (EISS). The segmentation improvement was found to be large in particular in regions of the skull base, but also in those neurocranial roof areas with larger deviation between chosen global inflation parameter and realistic local thickness of the cerebrospinal fluid compartment [Burkhardt *et al.*, 2002]. The comparison of ISS and EISS segmentation results is shown in Figure 6 (left and middle). Errors in EEG source localization of up to 1cm in mesial-temporal and basal-frontal regions, resulting from inaccurate skull segmentation, were found in [Huiskamp *et al.*, 1999]. Since these regions are of particular importance in epilepsy surgery, it was concluded that this imprecision may be detrimental in clinical applications.

As mentioned in the introduction, the human skull is an anisotropically conducting layer, if it is regarded as one unit. [Marin, 1997; Marin *et al.*, 1998] pointed out the importance of well-defined skull conductivity tensor eigenvectors and reported larger errors for the EEG potential in the case of erroneous tensor directions. We therefore based our determination of tensor eigenvectors on the resulting mesh of a discrete deformable surface model, whose pseudo-code is shown in Algorithm 1. This model was also used for a segmentation improvement of inner and outer skull surface [Burkhardt *et al.*, 2002]. The deformable model was applied here in order to generate a Smooth Surface Spongiosa Model (SSSM), i.e., a strongly smoothed triangular mesh, which was shrunken from the outer skull mask onto the outer spongiosa surface. Therefore, the binary mask of the outer skull surface was shrunken by 2 under the assumption of a 2 mm thick outer compacta layer [Akhtari *et al.*, 2000], resulting in the initial binary $mask$. In a first step, the operation $EXTRACT$ extracts a triangle mesh from $mask$, using the marching tetrahedra method [Payne and Toga, 1990]. Then, the wrapper algorithm [Gueziec and Hummel, 1994] is applied to achieve a reduction in the vertex count to N_v vertices, while retaining the shape of the surface, denoted by the operation $SIMPLIFY$. $N_v = 30,000$ was found to be a good choice, since $mask$ is only moderately curved. The operation outputs a mesh with mesh vertices $\vec{v}^0$, normal vectors $\vec{n}^0$ and triangle elements Δ, where the normal vector for

Algorithm 1 DEFORM : $(mask, N_v, T1, I_{lim}, \omega_{in}, \omega_{ex}, it) \rightarrow SSSM$

```
$(\vec{v}^0, \vec{n}^0, \Delta) = SIMPLIFY(EXTRACT(mask), N_v)$, $i = 0$, $\kappa = 1/3$
**for** $i = 1$ up to $i = iters$ **do**
  **for** each of the $N_v$ vertices $\vec{v}^i$ **do**
    $\vec{F}_{in}(\vec{v}^i) = 0$                    /* calculate smoothing force */
    **for** EACH OF THE $N$ EDGE-CONNECTED NEIGHBORS $\vec{v}^i_j$ OF $\vec{v}^i$ **do**
      $\vec{F}_{in}(\vec{v}^i) = \vec{F}_{in}(\vec{v}^i) + \frac{1}{N}\left(\vec{v}^i_j - \vec{v}^i\right)$
    **end for**
    $F_{ex,1}(\vec{v}^i) = \langle -\nabla I^{T1}_g(\vec{v}^i), \vec{n}^i \rangle$                    /* gradient force */
    $F_{ex,2}(\vec{v}^i) = \tanh\left(\kappa(I^{T1}(\vec{v}^i) - I_{lim})\right)$            /* capturing force */
    $\vec{v}^{i+1} = \vec{v}^i + \omega_{in}\vec{F}_{in}(\vec{v}^i) + \omega_{ex}\left(F_{ex,1}(\vec{v}^i)F_{ex,2}(\vec{v}^i) + F_{ex,2}(\vec{v}^i)\right)\vec{n}^i$
  **end for**
  $i = i + 1$
**end for**
$SSSM = (\vec{v}^i, \vec{n}^i, \Delta)$
```

each vertex was calculated as the arithmetic mean of the neighboring triangle normals. The internal force $\vec{F}_{in}(\vec{v}^i)$ for a vertex in the i^{th} iteration, $\vec{v}^i$, is chosen as a force, which pulls the vertex to the centroid of its N edge-connected neighbor vertices. In order to force the SSSM to remain inside the segmented skull layer, the intensity values of all voxels inside the inner skull surface and outside the outer skull surface were set to the highest grey value 255, resulting in the modified image $T1$. The magnitude of the external force, acting along the vertex normal $\vec{n}^{\,i}$, is divided into two parts. The first, $F_{ex,1}$, is attached to the local intensity value gradient of the Gauss smoothed modified image, ∇I^{T1}_g, while the second term, $F_{ex,2}$, captures the surface within a narrow range around an image intensity I_{lim}. We chose I_{lim} as the arithmetic mean of the intensities of skull compacta and spongiosa. The parameter $\kappa \in \mathbb{R}$ defines the capturing range, which is related to the amount of noise in the MR data. It was fixed to $\kappa = 1/3$, following [Burkhardt *et al.*, 2002]. We performed $it = 100$ iterations with a strong weighting of the smoothing force, $\omega_{in} = 0.07$ and a moderate weighting for the external force, $\omega_{in} = -0.002$. The resulting triangle mesh $SSSM$ was voxelized and shown in Figure 6 (right). For each point in the skull layer, we are now able to determine the radial skull direction by means of the normal vector $\vec{n}^{\,k}$ of the SSSM vertex $\vec{v}^k$ with minimal distance. The two tangential directions in the perpendicular plane to the radial direction can then be determined using vector product. Within the simulations in Section 5, we use conductivity tensors of the form $\sigma = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$ with $\mathbf{S}$ the orthogonal eigenvector matrix, built of the two tangential and the radial direction vectors, and simulated eigenvalues

$\mathbf{\Lambda} = \text{diag}(\lambda_{tang}, \lambda_{tang}, \lambda_{rad})$, as described in Table 1 in Subsection 5.1.1.

2.2.2 Generation of an anisotropic white matter compartment

We performed whole-head-DTI using a 4-slice displaced U-FLARE [Norris and Börnert, 1993] protocol with centric phase-encoding. Diffusion weighting was implemented as a Stejskal-Tanner type spin-echo preparation [Koch, 2000]. Although echo planar imaging (EPI) is being widely applied for DTI purposes, U-FLARE was preferred to EPI in order to avoid spatial misregistration between the DTI data and the 3D data sets due to magnetic field inhomogeneities. The effective echo time was $T_{eff} = 120$ ms, and $TR = 11$ s. The diffusion weighting gradient pulses had a duration of 22 ms, and their onset was separated by 40 ms. Four different b values evenly spaced between 50 and 800 s/mm^2 were applied through variation of the gradient strength [Koch, 2000]. The slices were axially oriented and $5mm$ thick. In-plane resolution was 2×2 mm^2. In order to increase the signal-to-noise ratio, 5 to 16 images (depending on the b value) with identical diffusion weighting were averaged. Due to the long measurement time (50 min for 4 slices) data acquisition was split into 8 sessions. Diffusion tensor calculation [Basser *et al.*, 1994a] was based on a multivariate regression algorithm in IDL (Interactive Data Language, Research Scientific, Bolder, Colorado/USA). Figure 2 shows a detail of an axial slice of the measured DTI data with 2×2 mm^2 resolution on an underlying coregistered T1-MRI. The coregistered T1 images of the same slices allowed the registration of the DTI data on the 3D T1 data set. The registered DT data were then resampled to $1 \times 1 \times 1$ mm^3. In order to handle the orientation information in the registered DT images appropriately, each diffusion tensor $\mathbf{D}'$ was rotated with the rotation matrix $\mathbf{R}$ of the respective registration process via the similarity transform $\mathbf{D} = \mathbf{R}\mathbf{D}'\mathbf{R}^T$ [Alexander *et al.*, 2001]. Figure 7 shows $Tr(\mathbf{D})$, i.e., the sum of the diagonal tensor ele-

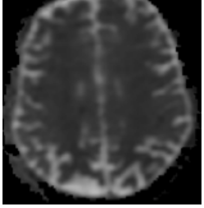
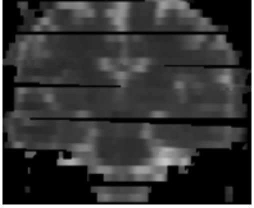
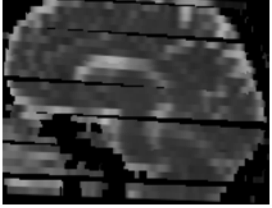
Axial	Coronal	Saggital
		

Fig. 7: $Tr(\mathbf{D})$ of the 8 registered DT-MRI sessions. Water diffusion coefficients in CSF (white) are much larger than in the brain, allowing a quality check of the registration.

ments, of the 8 registered DTI sessions. Since water diffusion coefficients in CSF are much larger than in the brain, a large contrast is achieved at the brain surface, which allows a quality check of the registration. As the figure shows, the registered DTI slices are not exactly parallel. Later in Subsection 2.3.1 for the generation of the FE volume conductor model, the gaps were filled with isotropic white matter conductivity tensors.

When extracting the anisotropic part of the diffusion tensor by means of

$$\mathbf{A} = \mathbf{D} - \frac{Tr\mathbf{D}}{3}\mathbf{I}, \tag{10}$$

we can define the "fractional anisotropy index" [Basser and Pierpaoli, 1996] as

$$FA = \sqrt{\frac{3}{2}}\frac{\sqrt{\mathbf{A}:\mathbf{A}}}{\sqrt{\mathbf{D}:\mathbf{D}}} \qquad \text{with} \quad \mathbf{B}:\mathbf{C} \equiv \sum_{i,j} B_{ij}C_{ij}. \tag{11}$$

Figure 8 shows a map of the fractional anisotropy index of the registered DT data, masked with the white matter mask of the 5-tissue segmentation result. The highest value for fractional anisotropy was found in the splenium of the corpus callosum, where $FA = 0.74$.

Within the simulations in Section 5, we will not make use of the EMA for the conductivity tensor eigenvalues as described in the introduction. We are rather interested in using conductivity tensors of the form $\sigma = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$ with $\mathbf{S}$ the orthogonal matrix of eigenvectors of the measured diffusion tensors, but with simulated eigenvalues $\mathbf{\Lambda} = \text{diag}(\lambda_{long}, \lambda_{trans}, \lambda_{trans})$ as described in Table 2 in Subsection 5.1.1. λ_{long} is the eigenvalue parallel (longitudinal) and λ_{trans} perpendicular (transverse) to the fibre directions. Figure 9 shows a detail of the projection of the conductivity tensor ellipsoids $\sigma = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$ with an eigenvalue choice of $\lambda_{long} = 0.65$ and $\lambda_{trans} = 0.065$ onto a coronal slice of the T1-MRI. The eigenvalue choice corresponds to the modeled anisotropy

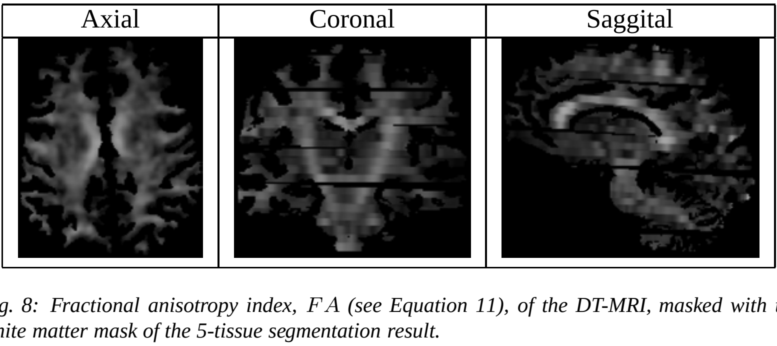


Fig. 8: Fractional anisotropy index, FA (see Equation 11), of the DT-MRI, masked with the white matter mask of the 5-tissue segmentation result.

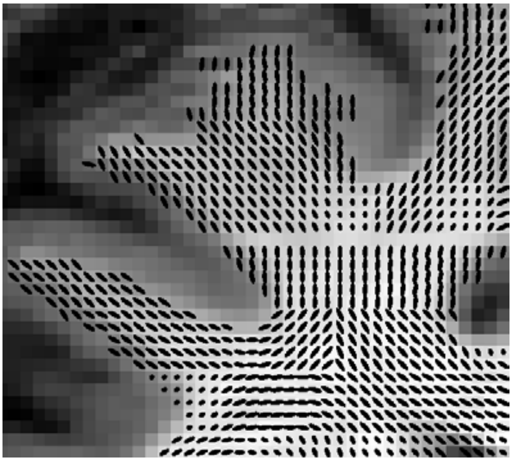


Fig. 9: Detail of the projection of the conductivity tensor ellipsoids σ onto a coronal cut of the T1-MRI through the Commissura anterior. The gap, where no measurement data was available, is not yet filled with isotropic conductivity tensors. Top right: Fibres going up to the Gyrus frontalis superior; top left: Fibres going up to the Gyrus frontalis medius; bottom right: Fibres going through the Truncus corporis callosi; bottom left: Fibres going up to the Gyrus precentralis and down to Gyrus frontalis inferior, pars opercularis. Tensor validation and visualization was carried out with the VM tool [SIMBIO, 2000].

ratio of 1:10 in Table 2 in Subsection 5.1.1. In Figure 9 the gap, where no measurement data was available, is not yet filled with isotropic conductivity tensors. The visualization of tensors in Figures 9, 10 and 11 was carried out with the software tool VM (Visualization Module), developed within the project [SimBio, 2000].

2.3 Meshing and discretization aspects

Numerical methods are needed for field simulations in realistically shaped anisotropic volume conductors. Within this article, we use the FE method.

2.3.1 FE mesh generation

An essential prerequisite for FE modeling is the generation of a mesh which represents the geometric and electric properties of the head volume conductor. Our approach uses a surface-based tetrahedral tessellation of the relevant compartments skin, skull, CSF, brain gray and white matter, and ventricular system, as described in [Wagner, 1998]. Auxiliary surfaces with a distance d_1 from the given compartment borders are generated so that a set of layered surfaces is obtained. In a next step, the vertices of the tetrahedral mesh are

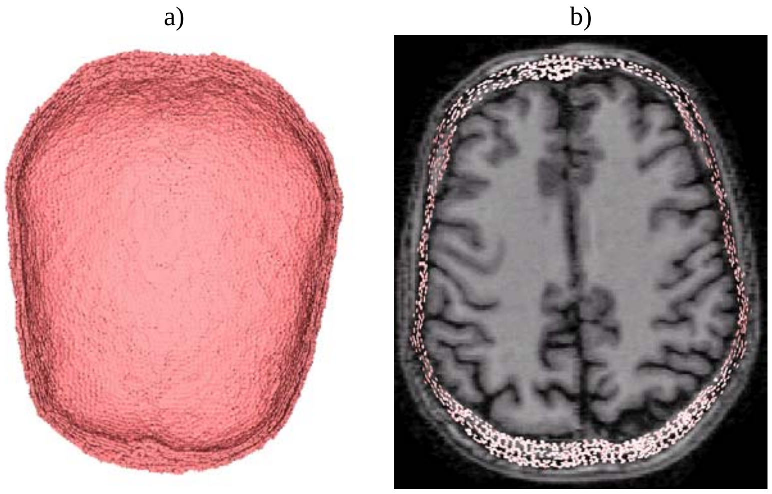


Fig. 10: Conductivity tensors in the barycenters of the skull elements for 1:5 anisotropy: a) Tensors of the skull roof. b) Tensors of an axial cut through the skull model on underlying T1-MRI. A stronger scaling of the eigenvalues of the tensors was chosen in a) compared to b). Tensor validation and visualization was carried out with the VM tool [SIMBIO, 2000].

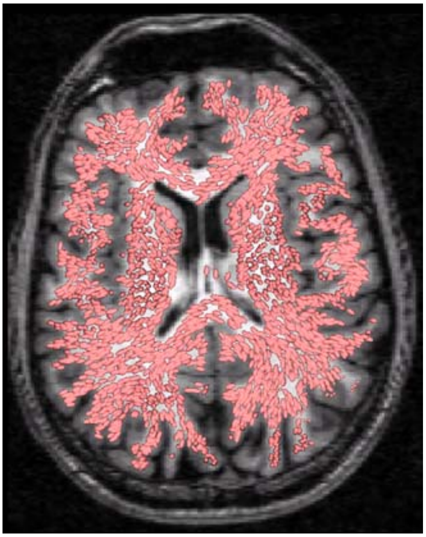


Fig. 11: Conductivity tensor ellipsoids in the barycenters of white matter elements for 1:10 anisotropy on underlying T1-MRI: The eigenvalues were chosen as $\lambda_{long} = 0.65$ *and* $\lambda_{trans} = 0.065$ *according to Table 2 in Subsection 5.1.1. Tensor validation and visualization was carried out with the VM tool [SIMBIO, 2000].*

generated by means of a thinning of the surfaces with thinning-distance d_1 for auxiliary and d_2 for compartment surfaces. $d_2 = 2mm$ enabled a very exact representation of the volume conductor. A distance of 1.3 times d_2 was chosen for d_1. This resulted in 147287 nodes. After a three-dimensional Delaunay triangulation, each of the 892115 tetrahedra was labeled according to its compartment. The tetrahedral FE mesh was generated using the software package [CURRY, 2000].

According to the procedure described in 2.2.1, an anisotropic conductivity tensor was assigned to the barycentre of each finite element in the skull. Figure 10 shows the conductivity tensors of the skull roof (left) and of an axial cut through the skull model on underlying T1-MRI (right) for 1:5 anisotropy. Tensor validation and visualization was carried out with the VM tool [SimBio, 2000]. To the barycentre of each finite element in the white matter compartment, we assign the anisotropic conductivity tensor derived from the measured diffusion tensor image with $1mm^3$ resolution, presented in Subsection 2.2.2. Figure 11 shows the conductivity tensor ellipsoids in the barycenters of the white matter finite elements for 1:10 anisotropy on the underlying T1 MRI. Again, the tensors were validated and visualized by means of the VM tool [SimBio, 2000].

2.3.2 The blurred dipole model

A direct approach for the discretization of Equation (2) is used within this article. Therefore, the blurred dipole model was introduced for FE based source localization in [Buchner *et al.*, 1997], which will be shortly summarized below. The blurred dipole is made up from monopole sources $J_k^b := J^b(\vec{x}_k)$, calculated for all neighboring FE mesh nodes $\vec{x}_k$ around the location $\vec{x}_i$ of a mathematical dipole $\vec{M}_i := \vec{M}\delta(\vec{x} - \vec{x}_i)$, so that

$$\begin{aligned} F &= \frac{1}{2}\left({}^{n_0}\vec{M}_i^r - \left(\Delta\bar{\vec{x}}_{ki}^r\right)^{n_0} J_k^b\right)\left({}^{n_0}\vec{M}_i^r - \left(\Delta\bar{\vec{x}}_{si}^r\right)^{n_0} J_s^b\right) \\ &\quad +\lambda\frac{1}{2}J_k^b g_{ks} J_s^b \overset{!}{=} min \\ g_{ks} &:= \begin{cases} \left(\Delta\bar{\vec{x}}_{ki}^r \Delta\bar{\vec{x}}_{si}^r\right)^{n_s/2} & if\ k = s \\ 0 & if\ k \neq s \end{cases} \end{aligned}$$

with $\Delta\bar{\vec{x}}_{ki}^r = \Delta\vec{x}_{ki}^r/a$ the r cartesian component of the a-weighted vector from node i to node k, $\Delta\vec{x}_{ki}$, n_0 the order of the source model, n_s the dipole smoothness and λ the dipole regularization parameter. The first part of the functional F ensures a minimal difference between the resultant moment of the blurred dipole and the one of the mathematical dipole, while the second part, a Tikhonov-Phillips regularizer, smoothes the monopole distribution and enables a unique minimum for F. The differentiation of F with respect to J^b is used to express the minimum condition which leads to a system of linear equations:

$$\begin{aligned}\left[\left(\Delta\vec{\bar{x}}^{r}_{ki}\right)^{n_0}\left(\Delta\vec{\bar{x}}^{r}_{si}\right)^{n_0}+\lambda g_{ks}\right]J^b_s &= \left(\Delta\vec{\bar{x}}^{r}_{ki}\right)^{n_0}\vec{M}^{n_0}_i \\ \sum_k J^b_k &= 0\end{aligned} \tag{12}$$

Together with $J^b_t = 0$ for all non-neighbor indices t of dipole index i, the monopole distribution of the blurred dipole model is defined. See [Buchner *et al.*, 1997] for a motivation of this source model and for accuracy tests in a sphere model, where the numerical results were compared with results of an analytical formula from [Smythe, 1989] for two closely neighbored monopoles, a source and a sink.

2.3.3 FE formulation for EEG forward computation

The direct application of variational and FE techniques to equation (2) with boundary conditions (3) together with the blurred dipole model yields a system of linear equations

$$\mathsf{K}_{h}{}_{h} = \underline{\mathsf{J}}_h \tag{13}$$

with $\mathsf{K}_h \in \mathbb{R}^{N_h \times N_h}$ denoting the stiffness matrix, $\underline{\mathsf{J}}_h \in \mathbb{R}^{N_h}$ the source load and ${}_h \in \mathbb{R}^{N_h}$ the solution vector for the total potential. The stiffness matrix is given by

$$\mathsf{K}^{[i,j]}_h = \int_\Omega \nabla\psi_j \sigma \nabla\psi_i \, d\Omega \tag{14}$$

and the right hand side entries for the direct method by

$$[\underline{\mathsf{J}}_h]^i = -\int_\Omega J^b_i \psi_i \, d\Omega \tag{15}$$

for an FE space $\mathbb{V}_h = span\{\psi_i\}_{i=1}^{N_h}$. The subscript h denotes the average meshsize and $N_h = O(h^{-3})$ is the number of unknowns. The condition number of the stiffness matrix behaves asymptotically like $O(h^{-2})$.

In contrast to the described direct discretization method in combination with the blurred dipole model, the subtraction method (see e.g. [Awada *et al.*, 1997; van den Broek *et al.*, 1997; Schimpf *et al.*, 2002]) splits the total potential into two parts, the singularity potential and the correction potential. The singularity potential is the analytically calculated solution for a mathematical current dipole (equation (1)) in an unbounded homogeneous conductor with constant isotropic conductivity. The correction potential is a solution to equation 2 in the closed sourceless domain under boundary conditions that correct

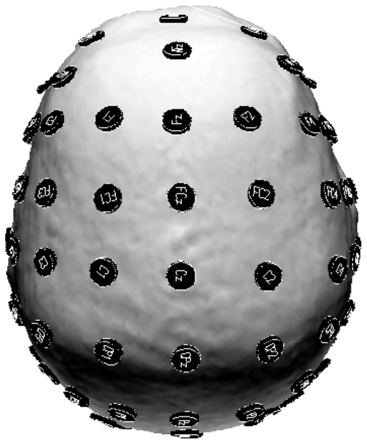


Fig. 12: EEG sensors: 71 electrodes of the chosen EEG system. Electrode size was enlarged for visualization purposes.

the movement of current across boundaries between regions of different conductivity. The correction potential is calculated by means of an FE approach, leading to a linear equation system with the same stiffness matrix (14), but with different right hand side. Since the solvers are independent of the right hand side of the equation system, the results presented in the following are also valid for the subtraction method in combination with the mathematical dipole.

For the EEG forward computation, 71 electrodes were placed interactively on the head surface according to the international 10/20 system [Pastelak-Price, 1983]. The electrode configuration is shown in Figure 12. The sensors were projected onto the FE head model, i.e., we model the electrode potential with the value of the closest neighboring FE mesh node.

2.3.4 MEG forward computation

For the magnetic forward problem, the flux transformers of the MEG device have to be modeled. The Max-Planck-Institute of Cognitive Neuroscience Leipzig is equipped with a BTI 148 channel whole-head MEG system. [Pohlmeier, 1996] modeled each coil Υ of this system (see Figure 13, left) by means of a thin, closed conductor loop with a diameter of 11.5 mm, using 8 isoparametric quadratic finite row elements. Errors slightly below the data noise between this realistic and a point-like coil representation were reported, showing the necessity of the chosen approach. With regard to these

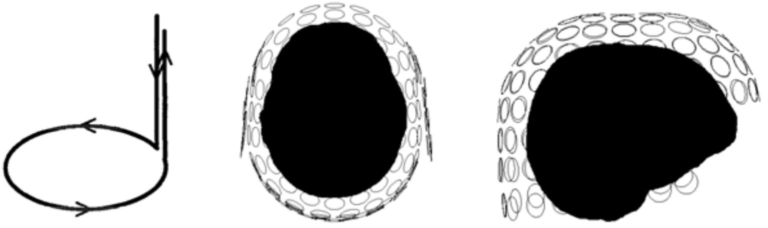


Fig. 13: MEG sensors: Magnetometer flux transformer (left) and the chosen whole head BTI-148-channel MEG system together with the FE head model, top view (middle) and side view (right).

results, equation (7) was discretized by means of

$$\vec{C}(\vec{y}) = \sum_n \int_{-1}^{1} \frac{1}{|\sum\limits_{i(n)} \chi_i(\xi)\vec{x}_i - \vec{y}|} \sum_{i(n)} \frac{\partial \chi_i(\xi)}{\partial \xi} \vec{x}_i d\xi, \tag{16}$$

where n denotes a finite row element and an isoparametric FE Ansatz with quadratic (parabolic) Ansatz functions χ_i was made for the coil position vector, $\vec{x}(\xi) = \sum_i \chi_i(\xi)\vec{x}_i$, with $\vec{x}_i$ the vertices of the row element. The determination of the primary flux Ψ_p in Equation (8) is then straight-forward. After the FE calculation of the potential distribution, the secondary flux Ψ_s in Equation (9) is computed by means of a Gauss integration, where the integrand consists of the interpolated functions in the FE space [Pohlmeier, 1996]. In Figure 13, the position of the 148 magnetometer coils, each represented by its row elements, are visualized together with the FE head model.

3 Parallel Algebraic Multigrid Solver

The inverse reconstruction process requires the solution of hundreds or even thousands of large scale systems of equations (13) with the stiffness matrix (14). In [Wolters *et al.*, 2000], condition numbers of about 10^7 have been calculated for high resolution realistically shaped head stiffness matrices, causing severe accuracy and convergence problems for classical iterative solvers. These problems were recovered by applying appropriate preconditioners for the CG method such that the condition number of the resulting preconditioned stiffness matrix was small. The AMG preconditioner was shown to be superior to incomplete Cholesky factorization with threshold. In [Johnson *et al.*, 2000], AMG-CG was found to be superior to a successive overrelaxation method.

If we are going to solve the entire localization problem with many calls of the solver, the results cannot be produced within an acceptable time. However, a parallel computer may provide sufficient capacity such that time limitation can be fulfilled. In [Haase *et al.*, 2002; Wolters *et al.*, 2002] it has been shown that AMG-CG solvers exhibit high speedups on parallel computers including PC clusters and an SGI ORIGIN 2000. The speedup was especially good for the solver-part of the algorithm. Since the setup of the preconditioner has to be carried out only once per head geometry, its calculation time and speedup can be neglected. Our following description of the parallel AMG is taken from [Wolters *et al.*, 2002].

3.1 Algebraic Multigrid Method

As in Geometric MultiGrid (GMG, see [Hackbusch, 1985] for a theoretical overview), the basic idea in AMG is to reduce high and low frequency components of the error by the efficient interplay of smoothing and coarse grid correction, respectively. In AMG, both, the matrix hierarchy and the prolongation operators are constructed just from the stiffness matrix K_h. In analogy, we will speak of "coarse grids" although these are purely virtual and do not have to be constructed explicitly as coarse FE meshes. Since the automatic generation of a grid-hierarchy for GMG and especially the proper assembling of all components would be a very difficult task with respect to conductivity inhomogeneities and anisotropies in a realistically shaped head model, the automatic algebraic construction of a virtual grid is a big advantage. A general concept of AMG methods for FE discretizations can be found in [Haase *et al.*, 2000]. Each AMG algorithm consists of the following components:

(a)Coarsening: define the splitting $\omega_h = \omega_C \cup \omega_F$ of ω_h (the index set of nodes) into sets of coarse and fine grid nodes ω_C and ω_F, respectively.
(b)Transfer operators: prolongation $\mathfrak{P}_h : V_H \mapsto V_h$ and restriction $\mathfrak{R}_h := \mathfrak{P}_h^T$.
(c)Definition of the coarse matrix by Galerkin's method, i.e.,
$\mathsf{K}_H := \mathfrak{R}_h \mathsf{K}_h \mathfrak{P}_h$.
(d)Appropriate smoother for the considered problem class.

The most important issue to be discussed is the setup phase, i.e., the construction of the matrix hierarchy and the prolongation operators. We will give the explanation for a two grid method where h is related to the fine grid and H to the coarse grid.

In our case, the stiffness matrix K_h can be associated with an FE grid, i.e., the diagonal entry of the i^{th} row of the matrix K_h is related to a grid point in ω_h and an off-diagonal entry is related to an edge in an FE grid (see

Figure 14). First we look at the coarsening process which has the task to

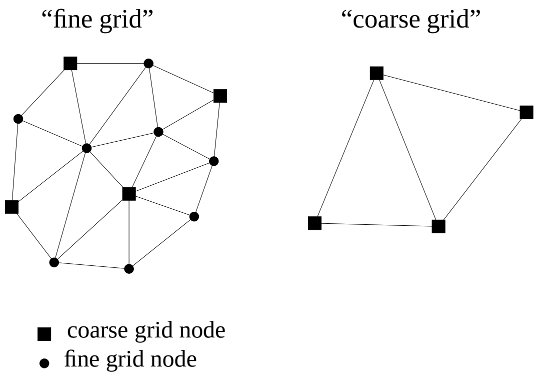


Fig. 14: Illustration of a two grid method.

reduce the nodes such that $N_H = |\omega_C| < N_h = |\omega_h|$. Here, $|\omega|$ denotes the number of elements in the set ω. Motivated from Figure 14, the grid points ω_h can be split into two disjoint subsets ω_C (coarse grid nodes) and ω_F (fine grid nodes), i.e.,

$$\omega_h = \omega_C \cup \omega_F, \quad \omega_C \cap \omega_F = \emptyset$$

such that there are (almost) no direct connections between any two coarse grid nodes and the resulting number of coarse grid nodes is as large as possible. Instead of considering all connections between nodes being of the same rank, we introduce the following sets

$$\begin{aligned} N_h^i &= \left\{ j \mid |\mathsf{K}_h^{[i,j]}| \neq 0,\, i \neq j \right\} \qquad (17) \\ S_h^i &= \left\{ j \in N_h^i \mid |\mathsf{K}_h^{[i,j]}| > coarse(i,j,\mathsf{K}_h) \right\} \\ S_h^{i,T} &= \left\{ j \in N_h^i \mid i \in S_h^j) \right\} \end{aligned}$$

where N_h^i is the index set of neighbors, S_h^i denotes the index set of nodes with a "strong connection" from node i and $S_h^{i,T}$ is related to the index set of nodes with a "strong connection" to node i (see [Ruge and Stüben, 1986]). In addition $coarse(i,j,\mathsf{K}_h)$ is an appropriate cut-off (coarsening) function, e.g.,

$$coarse(i,j,\mathsf{K}_h) := \alpha \cdot \max_j \{|\mathsf{K}_h^{[i,j]}|\}\,, \qquad (18)$$

Algorithm 2 (Parallel) $V(\nu_F, \nu_B)$-cycle $\textsc{mg}(\mathsf{K}_h, , \underline{\mathsf{J}})$

```
if CoarseGrid then
    $\Leftarrow$ DirectSolve $(\mathsf{K} \cdot \ = \underline{\mathsf{J}})$
else
    $\tilde{}\Leftarrow \nu_F$ times Smooth$(\mathsf{K}_h, , \underline{\mathsf{J}})$
    $\underline{\mathsf{d}} \leftarrow \underline{\mathsf{J}} - \mathsf{K}_h \cdot \tilde{}$
    $\underline{\mathsf{d}}^H \leftarrow \mathfrak{P}^T \cdot \underline{\mathsf{d}}$
    $\underline{\mathfrak{w}}^H \leftarrow 0$
    $\underline{\mathfrak{w}}^H \Leftarrow \textsc{mg}(\mathsf{K}_H, \underline{\mathfrak{w}}^H, \underline{\mathsf{d}}^H)$
    $\underline{\mathfrak{w}} \leftarrow \mathfrak{P} \cdot \underline{\mathfrak{w}}^H$
    $\hat{}\leftarrow\tilde{}+\underline{\mathfrak{w}}$
    $\Leftarrow \nu_B$ times Smooth$^T(\mathsf{K}_h, \hat{}, \underline{\mathsf{J}})$
end if
```

with $\alpha \in [0, 1]$. With those definitions a splitting into coarse and fine grid nodes can be done. For our computations we used a modified splitting algorithm of [Ruge and Stüben, 1986]. Next the prolongation operator has to be defined correctly. We require that the prolongation operator $\mathfrak{P}_h : V_H \mapsto V_h$ has full rank. There are a lot of possibilities to define such transfer operators with pure algebraic information. For the construction we refer to [Ruge and Stüben, 1986; Braess, 1995; Kickinger, 1998; Wagner, 2000]. A possible setting and the one which turned out to be the most efficient for the presented application is given by

$$(\mathfrak{P}_h)^{[i,j]} = \begin{cases} 1 & i = j \in \omega_C \\ 1/|S_h^{i,T} \cap \omega_C| & i \in \omega_F,\ j \in S_h^{i,T} \cap \omega_C \\ 0 & \text{else}\,. \end{cases} \tag{19}$$

The coarse grid matrix K_H is defined by the classical Galerkin method, i.e.,

$$\mathsf{K}_H = \mathfrak{P}_h^T \mathsf{K}_h \mathfrak{P}_h \in \mathbb{R}^{N_H \times N_H},$$

being again symmetric and positive definite (see e.g. [Ruge and Stüben, 1986]).

After the proper definition of the prolongation and coarse grid operators a matrix hierarchy can be setup in a recursive way. Finally, a multigrid cycle can be assembled, see Algorithm 2. Therein the variable CoarseGrid denotes the level where a direct solver is applied.

For our application we use AMG-CG, i.e., AMG is applied as a preconditioner for the CG method (see [Jung and Langer, 1991]). For the m-$V(\nu_F, \nu_B)$-cycle AMG preconditioned CG method, the operation $\underline{\mathfrak{w}} =$

Algorithm 3 (Par.) PCG algorithm PCG($\mathsf{K}_h, , \underline{\mathsf{J}}, \mathsf{C}_K$)

```
  $\underline{\mathsf{r}} \leftarrow \underline{\mathsf{J}} - \mathsf{K}_h$
  $\underline{\mathfrak{w}} \Leftarrow \mathsf{C}_K^{-1} \cdot \underline{\mathsf{r}}$
  $\underline{\mathfrak{s}} \leftarrow \underline{\mathfrak{w}}$
  $\gamma \Leftarrow \langle \underline{\mathfrak{w}}, \underline{\mathsf{r}} \rangle$
  **repeat**
    $\underline{\mathsf{v}} \leftarrow \mathsf{K}_h \cdot \underline{\mathfrak{s}}$
    $\alpha \Leftarrow \gamma / \langle \underline{\mathfrak{s}}, \underline{\mathsf{v}} \rangle$
    $\leftarrow + \alpha \underline{\mathfrak{s}}$
    $\underline{\mathsf{r}} \leftarrow \underline{\mathsf{r}} - \alpha \underline{\mathsf{v}}$
    $\underline{\mathfrak{w}} \Leftarrow \mathsf{C}_K^{-1} \cdot \underline{\mathsf{r}}$
    $\gamma \Leftarrow \langle \underline{\mathfrak{w}}, \underline{\mathsf{r}} \rangle$
    $\beta \leftarrow \gamma / \gamma_{\text{OLD}}$ , $\gamma_{\text{OLD}} \leftarrow \gamma$
    $\underline{\mathfrak{s}} \leftarrow \underline{\mathfrak{w}} + \beta \underline{\mathfrak{s}}$
  **until** TERMINATION
```

$\mathsf{C}_K^{-1} \underline{\mathsf{r}}$ is realized by m calls of MG($\mathsf{K}_h, \underline{\mathfrak{w}}, \underline{\mathsf{r}}$). For the Jacobi-preconditioner, it is $\mathsf{C}_K = \mathsf{D}$ with D the diagonal of K_h. The Preconditioned CG (PCG) method is shown in Algorithm 3.

3.2 *Data Partitioning*

The aim of parallelization is to split both data and operations to the P processors available. The consistency of the algorithms is preserved by message passing. In our case, the parallelization is based on a non-overlapping domain decomposition, i.e., we decompose $\overline{\Omega}$ into P subdomains $\overline{\Omega}_s$ such that

$$\overline{\Omega} = \bigcup_{s=1}^{P} \overline{\Omega}_s$$

with

$$\Omega_s \cap \Omega_q = \emptyset \quad \forall q \neq s, \quad s, q = 1, \ldots, P$$

holds. Each subdomain Ω_s is discretized by a mesh $\tau_{h,s}$ such that the whole triangulation

$$\tau_h = \bigcup_{s=1}^{P} \tau_{h,s}$$

of Ω forms a conforming mesh. A global FE space $\mathbb{V}_h$ is defined with respect to τ_h and the local spaces $\mathbb{V}_{h,s}$ are restrictions of $\mathbb{V}_h$ onto $\tau_{h,s}$.

The mesh partitioning of realistic FE geometries with unstructured meshes is critical for the efficiency of the parallel solver method. The distribution

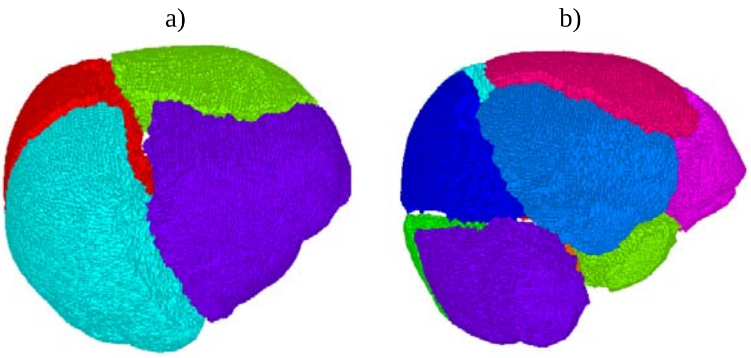


Fig. 15: Realistic tetrahedral FE head model, 892115 elements, partitioned with METIS and visualized with PMVIS: a) for 4 processors b) for 12 processors.

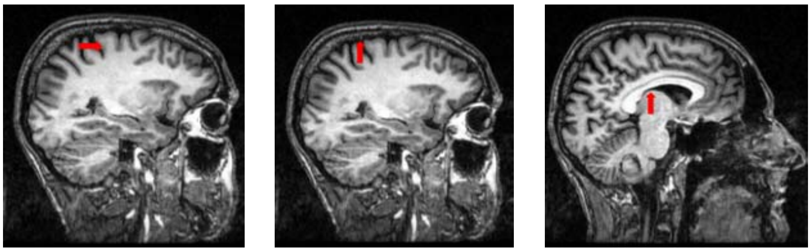


Fig. 16: Simulated sources on underlying T1-MRI: Almost tangentially oriented somatosensory source (left), somatosensory source with large radial orientation component (middle) and left thalamic source (right).

must be done so that the number of elements assigned to each processor is the same and the number of adjacent elements assigned to different processors is minimized in order to balance the computation amount among the processors and to minimize the communication between them, respectively. Therefore, graph partitioning algorithms were used which model the FE mesh by a graph (V, E) with vertices V and edges E. Since we are interested in an "element-wise-" in contrast to a "node-wise-" distribution, the dual graph of the FE mesh was partitioned. The finite elements are the vertices of the dual graph and adjacent elements are the corresponding edges. A balanced k-way partitioning was used, minimizing the number of edges which straddle partitions. No weighting of the edges, e.g. with regard to jumping conductivities between elements at tissue-boundaries, was used. The algorithm is based on a multilevel approach, first reducing the size of the dual graph by collapsing vertices and edges, then partitioning the dual graph on the lowest level and further refine during the uncoarsening steps. For the described mesh-partitioning, the software package METIS was used [Karypis and Kumar, 1998]. The results were achieved in a few seconds on a single processor SGI workstation. A first examination of the partitioning result was carried out by means of zooming, rotating, translating, scaling, and applying explosion factors. Figure 15 shows a visualization of the partitioned geometries for 12 processors (see [Öztekin *et al.*, 1998]). Later, the number of interface and inner nodes and the number of elements were controlled during the calculations. The interface nodes are those nodes which belong to at least two processors, whereas inner nodes only belong to one. In all cases, the quality of the partitioning results were very satisfactory.

3.3 Parallel AMG

The mapping of a vector $\underline{\Phi}_h \in \mathbb{R}^{N_h}$ in global numbering onto a local vector $\underline{\Phi}_s \in \mathbb{R}^{N_s}$ in subdomain $\overline{\Omega}_s$ $(s = 1, \ldots, P)$ is represented symbolically by subdomain connectivity matrices $\mathcal{A}_s : \mathbb{R}^{N_h} \mapsto \mathbb{R}^{N_s}$ with entries

$$\mathcal{A}_s^{[i,j]} := \begin{cases} 1 & \text{if} \quad j = \textsc{Loc2Glob}(i) \\ 0 & \text{else} \end{cases} \qquad \forall i \in \omega_s \, , \, \forall j \in \omega_h$$

where $\textsc{Loc2Glob}(\cdot)$ maps a local index to the global index. The transpose $\mathcal{A}_s^T$ of these binary matrices $\mathcal{A}_s$ maps a local vector back onto the global one. The index set of all those subdomains to which an unknown $\Phi_h^{[j]}$, $j \in \omega_h$ belongs, is denoted by

$$\sigma^{[j]} := \{ s \,|\, \Phi_h^{[j]} \in \overline{\Omega}_s \} \, . \tag{20}$$

We store the data related to the i^{th} node in the subdomain Ω_s if $s \in \sigma^{[i]}$. This approach results in local data denoted by index s of two types [Haase, 1999]: *accumulated data* (vector , matrix $\mathfrak{P}$) represented by

$$_s := \mathcal{A}_s \cdot \;\; , \qquad \mathfrak{P}_s := \mathcal{A}_s \cdot \mathfrak{P} \cdot \mathcal{A}_s^T \tag{21}$$

and *distributed data* (vector $\underline{\mathsf{d}}$, matrix K_h) represented by

$$\underline{\mathsf{d}} = \sum_{s=1}^{P} \mathcal{A}_s^T \cdot \underline{\mathsf{d}}_s, \qquad \mathsf{K}_h := \sum_{s=1}^{P} \mathcal{A}_s^T \cdot \mathsf{K}_s \cdot \mathcal{A}_s \,. \tag{22}$$

It turns out, that in Algorithms 2 and 3, the functionals are represented as distributed data ($\underline{\mathsf{J}}$, $\underline{\mathsf{v}}$, $\underline{\mathsf{r}}$, $\underline{\mathsf{d}}$, K_h), whereas functions are represented as accumulated data (, $\underline{\mathfrak{s}}$, $\underline{\mathfrak{w}}$, $\mathfrak{P}$). The local FE accumulation with respect to $\mathbb{V}_{h,s}$ produces automatically distributed matrices K_s. For instance, it can be shown that the multiplication of a distributed matrix K_h with the accumulated vector $\underline{\mathfrak{s}}$ in Algorithm 3 results in a distributed vector $\underline{\mathsf{v}}$:

$$\begin{aligned} \mathsf{K}_h \cdot \underline{\mathfrak{s}} = \sum_{s=1}^{P} \mathcal{A}_s^T \mathsf{K}_s \mathcal{A}_s \cdot \underline{\mathfrak{s}} &= \sum_{s=1}^{P} \mathcal{A}_s^T \left(\mathsf{K}_s \cdot \underline{\mathfrak{s}}_s \right) \\ = \sum_{s=1}^{P} \mathcal{A}_s^T \underline{\mathsf{v}}_s &= \underline{\mathsf{v}} \end{aligned}$$

The realization requires no communication at all because we only have to compute $\underline{\mathsf{v}}_s = \mathsf{K}_s \cdot \underline{\mathfrak{s}}_s$ locally.

If an accumulated matrix $\mathfrak{M}$ fulfills the condition

$$\forall i \in \omega_h \,, \forall j \in \omega_C \;:\; \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{M}^{[i,j]} = 0, \tag{23}$$

then the operations $\underline{\mathfrak{w}} = \mathfrak{M} \cdot \underline{\mathfrak{w}}^H$, $\underline{\mathsf{d}}^H = \mathfrak{M}^T \cdot \underline{\mathsf{d}}$ and $\mathsf{K}_H = \mathfrak{M}^T \mathsf{K}_h \mathfrak{M}$ can be performed locally without any communication [Haase, 1999].

In AMG the coarsening and prolongation operators are components which can be chosen. The main idea in the design of parallel AMG is to choose these components such that the resulting prolongation operators $\mathfrak{P}$ are of accumulated type satisfying the pattern condition (23). For this purpose, a local node ordering is introduced by means of a grouping and ordering of the index sets (20) according to $|\sigma^{[j]}|$. The coarsening then starts at interfaces involving more than 2 processors and continues with faces between two processors and finally the coarsening of inner nodes is realized. In addition the coarsening has to be synchronized such that the coarse grid problem is conforming across interfaces between processors. This synchronization requires next neighbor communication. Note that the partitioning of the nodes has only been performed on the finest grid. For a detailed discussion we refer to [Haase *et al.*, 2002].

Now we observe that Algorithm 2 and Algorithm 3 are also the appropriate parallel formulations, where double-line arrows "$\Leftarrow$" indicate that communication is required for the corresponding operation. In Algorithm 2, the coarse grid system is accumulated globally once in the setup phase. During the iteration only a vector has to be assembled for computing the coarse grid solution. Furthermore, the smoother requires communication and has to be adapted appropriately. We use a Gauss-Seidel smoother for the inner nodes and a Jacobi smoother for the interface nodes. The Jacobi-smoother involves a vector conversion from distributed to accumulated type, i.e., one next neighbor communication across interfaces is required per smoothing step. In this way we get a sophisticated smoother which can be found in [Haase, 1999]. In Algorithm 3, only inner products involve communication besides the preconditioning operation. Since for the inner product of different type vectors it is

$$\begin{aligned}\langle \underline{\mathfrak{w}}, \underline{\mathfrak{r}} \rangle &= \underline{\mathfrak{w}}^T \sum_{s=1}^{P} \mathcal{A}_s^T \underline{\mathfrak{r}}_s = \sum_{s=1}^{P} (\mathcal{A}_s \underline{\mathfrak{w}})^T \underline{\mathfrak{r}}_s \\ &= \sum_{s=1}^{P} \langle \underline{\mathfrak{w}}_s, \underline{\mathfrak{r}}_s \rangle,\end{aligned}$$

only one global reduce operation is needed.

4 Software developments

A new FE software package NeuroFEM was developed, based on the package CAUCHY (see [CAUCHY, 1997; Buchner *et al.*, 1997]). Since it would have been difficult to integrate the FORTRAN77-CAUCHY code using quasistatic memory management in a new C++ class structured inverse toolbox, the old software was redesigned. The inverse toolbox contains a variety of state-of-the-art current source localization methods ([SimBio, 2000], see also [Knösche, 1997; Wolters *et al.*, 1999a]). Another argument for the code development was the possibility for a proper interface to the software package PEBBLES including the parallel AMG solver ([Reitzinger, 1999; Haase *et al.*, 2002]). The solver code exploits C++ principles of overloading and inheritance.

Therefore, C++ class structured software concepts replace old CAUCHY kernel routines. The storage management within NeuroFEM is fully dynamical so that a recompilation of the software is no longer necessary when changing the problem- and thus memory- size. The new structure facilitated parallel programming on distributed memory computers using the Message-Passing Interface (MPI) standard. The integrated software allows future comparisons with Boundary Element (BE) method based forward simulations (see e.g.[Zanow and Peters, 1995; Fuchs *et al.*, 1998]) or series expansion formulas in spherical shell models [de Munck and Peters, 1993].

The coupling to the parallel solver-package is carried out through an "element by element" interface. The root-process determines the index set (20) for each node of the partitioned geometry and scatters the corresponding data together with the conductivity tensors to the processors. The arrangement of the nodes to groups according to their index-sets, the ordering of the groups and the allocation of corresponding MPI-communicator groups and the local node numbering is then a fully parallel process. Element-stiffness-matrices are computed on each processor and stored in the local stiffness matrices in FE compact row format. These matrices automatically have the distributed data format (22). The global Dirichlet-node information is scattered to all processors and implemented with a penalty approach in local numbering to those local stiffness matrices whose processor-number is part of the global Dirichlet-node index-set. The coarsening can then be carried out and the hierarchy of stiffness and prolongation matrices can be determined in the parallel setup-phase of the AMG preconditioner as described in Section 3.

5 Results and discussion

After discussion of parameter settings and definition of sensitivity error measures, we present simulation results concerning the influence of tissue conductivity anisotropy on EEG and MEG for typical current sources in the brain. Performance results of the parallel AMG solver and its sensitivity to tissue anisotropy will be discussed in the last part of the section.

5.1 Parameter settings and error measures

5.1.1 Simulated sources

Simulation studies were carried out with three blurred dipolar current sources at two different locations in the brain (see Figure 16 before p. 134). The first two sources, one of them almost tangentially oriented (in y-direction, Figure 16, left) and the other radially (in z-direction, Figure 16, middle), were chosen in the right somatosensory cortex as an example for eccentric, i.e., superficial sources. The second location was chosen in the left thalamus as an example for deeper sources, where the orientation is always almost radial (Figure 16, right). For the parameters of the blurred dipoles (see equation

(12)), we chose $n_0 = 2$ for the order of the source model, $n_s = 2$ for the dipole smoothness, $\lambda = 10^{-6}$ for the regularization parameter and $a = 20.0$ for the dipole scale, effecting a spatial concentration of monopole loads J_k in the dipole axis around the dipole node. This choice also led to best results in sphere model accuracy tests, when comparing the numerical results with an analytical formula from [Smythe, 1989] for two closely neighboring monopoles, a source and a sink, see [Buchner *et al.*, 1997].

5.1.2 Volume conductor modeling

	Volume constraint		Wang's constraint	
$\lambda_{rad} : \lambda_{tang}$	λ_{rad}	λ_{tang}	λ_{rad}	λ_{tang}
1:1 (iso)	0.0042	0.0042	0.0042	0.0042
1:2	0.0026	0.0053	0.003	0.0058
1:5	0.00143	0.0072	0.00188	0.00938
1:10	0.000905	0.00905	0.00133	0.01326
1:100	0.000195	0.0195	0.00042	0.042

Tab. 1: Skull conductivity tensor eigenvalue settings: λ_{rad} is the conductivity tensor eigenvalue for radial and λ_{tang} for tangential direction.

The conductivity σ of the chosen isotropic skin- and grey matter-elements was set to 0.33 1/Ωm. An isotropic conductivity value of 1.79 1/Ωm was assigned to elements in the CSF [Baumann *et al.*, 1997], i.e., within the compartment between brain and skull and within the ventricular system. The conductivity tensor eigenvectors of skull elements were determined by means of the procedure, described in Subsection 2.2.1. The conductivity eigenvalues for an isotropic skull layer were set to 0.0042 1/Ωm, resulting in the well-known conductivity ratio of about $1 : 80$ between skull and skin layer (see, e.g., [Haueisen, 1996]). For a chosen anisotropy ratio, $\lambda_{rad} : \lambda_{tang}$, radial ($\lambda_{rad}$) and tangential ($\lambda_{tang}$) eigenvalues were calculated, obeying two different constraints: The first constraint, namely retaining $\lambda_{rad} \cdot \lambda_{tang}$ between the isotropic and anisotropic models, was proposed in [Wang *et al.*, 2001; van den Broek, 1997]. Our second constraint tries to retain the geometric mean of the eigenvalues and thus the volume of the conductivity tensor, $4/3\pi\lambda_{rad}\lambda_{tang}^2$. Table 1 shows the 5 chosen anisotropy ratios and the calculated eigenvalues under constraint of the respective approach. We chose anisotropy ratios of $1 : 2$, $1 : 5$, $1 : 10$ and $1 : 100$, where the last should be considered to be out of the realistic range. For the white matter compartment, the eigenvectors of a measured water diffusion tensor at the barycentre of a white matter finite element were taken as its conductivity tensor eigenvectors. The conductivity eigenvalues for an isotropic white matter compartment

	Volume method		Wang's method	
$\lambda_{trans} : \lambda_{long}$	λ_{trans}	λ_{long}	λ_{trans}	λ_{long}
1:1 (iso)	0.14	0.14	0.14	0.14
1:2	0.111	0.222	0.099	0.19798
1:5	0.0818	0.41	0.0626	0.31309
1:10	0.065	0.65	0.04427	0.4427
1:100	0.03016	3.016	0.014	1.4

Tab. 2: White matter conductivity tensor eigenvalue settings: λ_{long} is the conductivity tensor eigenvalue for longitudinal and λ_{trans} for transverse fibre direction.

were set to 0.14 1/Ωm (see, e.g., [Haueisen, 1996]). The anisotropy of the white matter compartment was varied through a variation of the eigenvalues assigned to the longitudinal fibre direction (parallel to white matter fibre bundles, i.e., the eigenvector direction with the largest eigenvalue in the water diffusion tensor), λ_{long}, and transverse direction, λ_{trans}, again obeying the two constraints which were described above. Table 2 shows the chosen anisotropy ratio, $\lambda_{trans} : \lambda_{long}$, and the resulting eigenvalues. Again, we chose the anisotropy ratios $1 : 2$, $1 : 5$, $1 : 10$ and $1 : 100$, where the last should again be considered to be out of the realistic range.

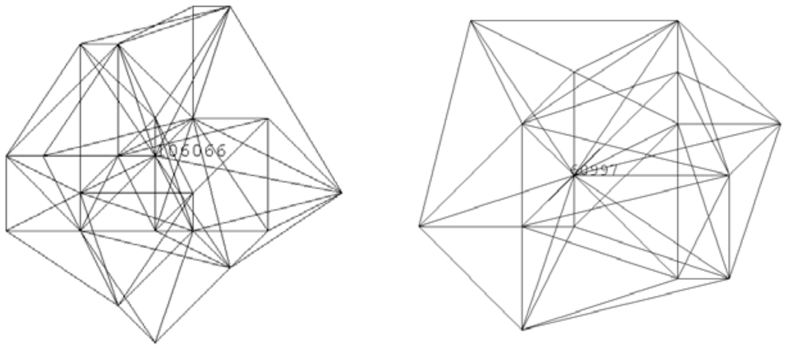


Fig. 17: Determined 41 finite elements in the neighborhood of the eccentric source at vertex 106066 (left) and the 29 neighbored finite elements of the deep source at vertex 60997 (right).

In [Haueisen *et al.*, 2000], a strong influence of local conductivity changes around the source location to EEG and MEG was reported. Therefore, in our model with 1:10 anisotropy of white matter and skull (volume constraint), we first determined 41 finite elements in the neighborhood around the eccentric source, 5 of which were isotropic CSF, 30 isotropic grey matter and 6 anisotropic white matter elements (Fig. 17, left). To the latter 6 elements, we then assigned the isotropic white matter conductivity. This model is denoted by EALC (Eccentric 1:10 Anisotropic Locally Changed). For the deep source, we determined 29 neighbored finite elements (Fig. 17, right), all of which were anisotropic white matter elements. The isotropic white matter conductivity tensor was then assigned to these elements, resulting in the DALC (Deep 1:10 Anisotropic Locally Changed) model.

5.1.3 Settings for the FE solution process

The zero starting vector $\Phi_0 = \vec{0}$ was chosen for the iterative solution process. The FE space $\mathbb{V}_h$ consisted of piecewise linear Ansatz-functions. Note that the solver-speed of the algorithms in Section 3 is only dependent on the stiffness matrix (14), so that the following results are valid for both the direct and the subtraction method (see Subsections 2.1 and 2.3.3) and all possible source configurations (i.e., independence of the right-hand side of the linear equation system). For the AMG-CG, we used the 1-$V(1,1)$-cycle AMG-preconditioner. Equation (18) was taken as the cut-off coarsening function with $\alpha = 0.01$ and the prolongation was chosen as in (19), respecting the pattern condition (23). The factorization in Algorithm 2 was carried out, if the size of the coarsest grid (COARSEGRID) in the preconditioner-setup was below 800. The coarse system is solved by means of a Cholesky-factorization. We observed only a small influence of the coarse grid size towards the solver times, when, e.g., increasing from COARSEGRID=800 to 1000.

The process of determining the index set (20) for each node and scattering the data to the processors, both of which are carried out by the root processor, and the local arrangement of nodes to groups according to their index-set and the allocation of corresponding communicator groups takes about half a minute. This duration can be neglected since these processes have to be performed only once per head model.

The solver process was stopped after the i^{th} iteration if the relative error in the controllable $\mathsf{K}_h\mathsf{C}_K^{-1}\mathsf{K}_h$-energy norm was below $\epsilon = 10^{-8}$, i.e.,

$$\frac{\langle \underline{\mathfrak{w}}^i, \underline{\mathsf{r}}^i \rangle}{\langle \underline{\mathfrak{w}}^0, \underline{\mathsf{r}}^0 \rangle} \leq \epsilon^2.$$

The simulations were run on an SGI ORIGIN 2000 with R10000, 195 MHz processors and overall 6GB of main memory. The solver speedup for 1 up to 12 processors was investigated.

5.1.4 Error criteria for sensitivity determination

We now define the two error criteria, which shall describe the influence of skull and white matter anisotropy on the field distribution. These criteria were introduced for Boundary Element calculations in [Meijs *et al.*, 1989]. The first error criterion, the Relative Difference Measure (RDM) at M measurement sensors, defined as

$$\text{RDM} = \sqrt{\sum_{n=1}^{M} \left(\frac{\Phi_n^{iso}}{\sqrt{\sum_{n=1}^{M} \left(\Phi_n^{iso}\right)^2}} - \frac{\Phi_n^{aniso}}{\sqrt{\sum_{n=1}^{M} \left(\Phi_n^{aniso}\right)^2}} \right)^2}, \qquad (24)$$

is a measure for the topography error (Optimum RDM $= 0$). The second error measure, the MAGnification factor (MAG), defined as

$$\text{MAG} = \frac{\sqrt{\sum_{n=1}^{M} \left(\Phi_n^{aniso}\right)^2}}{\sqrt{\sum_{n=1}^{M} \left(\Phi_n^{iso}\right)^2}}, \qquad (25)$$

gives an indication of errors in the magnitude.

5.2 *Influence of tissue conductivity anisotropy on EEG and MEG*

5.2.1 *Results for a tangentially oriented eccentric source*

Forward calculations for the eccentric source with a large tangential orientation component (Figure 16, left) and a strength of 10 nAm were carried out in the 5-tissue model and the sensitivity of EEG and MEG towards anisotropy of the skull layer, the white matter compartment, and of both skull and white matter was determined. Figure 18 shows the resulting RDM (left) and MAG (right) errors for an increasing anisotropy ratio, when either obeying the volume constraint or Wang's constraint.

Our EEG results concerning 1:10 anisotropic skull in combination with an isotropic white matter compartment are generally in agreement with the observations in [Marin *et al.*, 1998], where a topography error of about 10% and a magnification factor larger than 1 was reported for an eccentric tangentially oriented dipole in a realistic FE head model. However, [Marin *et al.*, 1998] modeled 1:10 skull anisotropy by fixing the radial and increasing the tangential conductivity eigenvalues by a factor of 10. If we do so, we observe a much larger topography error (RDM=20%) and a much larger magnification factor (MAG=1.72) (see also Figure 19, bottom row, right). This difference could be attributed to different simulation parameters like skull thickness and source location, and to the definition/evaluation of RDM and MAG. [Marin *et al.*, 1998] evaluated an integral over the whole head surface, whereas we considered only the potentials at the 71 EEG electrodes. We focus on the influence of anisotropy on the inverse EEG problem, where in practice, only

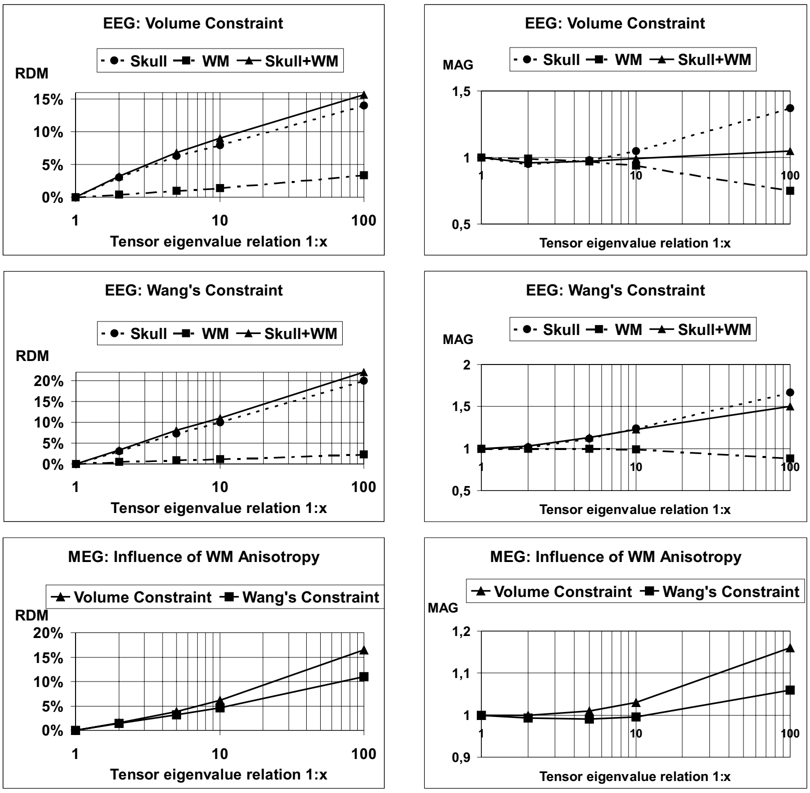


Fig. 18: Eccentric source, large tangential orientation component: EEG/MEG topography error (left) and magnitude error (right) for different anisotropy ratios: For EEG, errors due to anisotropy effects of skull, white matter (WM) and both skull and WM are presented for the tensor volume retaining (top row) and Wang's constraint [Wang et al., 2001] (middle row). For MEG, only WM anisotropy effects for both constraints are presented, since skull anisotropy was found to have no influence.

a limited number of sensors is available. The electrodes are often placed around the "center of interest" in order to sample the whole measurable dipolar pattern (see Figure 4). Figure 19 shows the isopotential distribution of the somatosensory source interpolated on the head surface of various FE forward calculations in the 5-tissue model. The isopotential distribution for the isotropic skull layer is shown in Figure 19, top row, left, and the result for 1:10 anisotropy with volume constraint in Figure 19, top row, right. The two 1:10 skull anisotropy isopotential distributions, presented in the bottom row of Figure 19, were calculated either by means of a fixation of the tangential

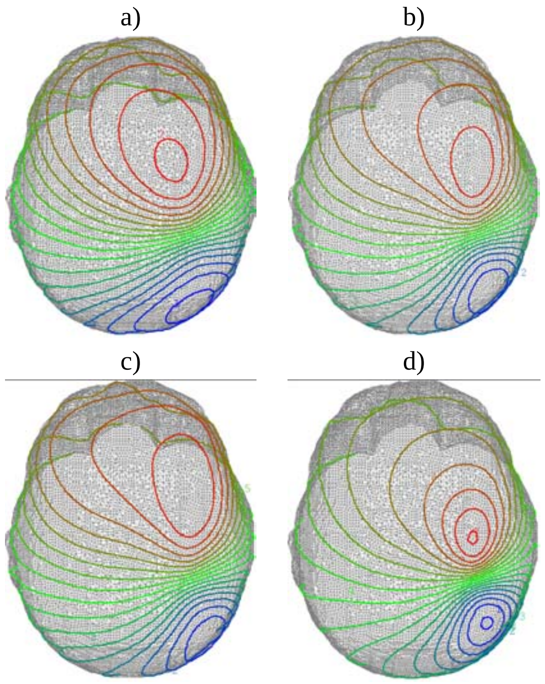


Fig. 19: Topography of isopotential distribution of the eccentric source with large tangential orientation component on upper part of the head a) isotropic skull ($\lambda_{rad} = 0.0042$, $\lambda_{tang} = 0.0042$); from -0.9 to 0.2 μV b) 1:10 anisotropic skull; volume method: $\lambda_{rad} = 0.000905$ and $\lambda_{tang} = 0.00905$); from -1.1 to 0.3 μV c) 1:10 anisotropic skull; 10 times lower radial cond.: $\lambda_{rad} = 0.00042$ and $\lambda_{tang} = 0.0042$; from -0.7 to 0.1 μV d) 1:10 anisotropic skull; 10 times higher tangential cond.: $\lambda_{rad} = 0.0042$ and $\lambda_{tang} = 0.042$; from -2.2 to 0.9 μV.

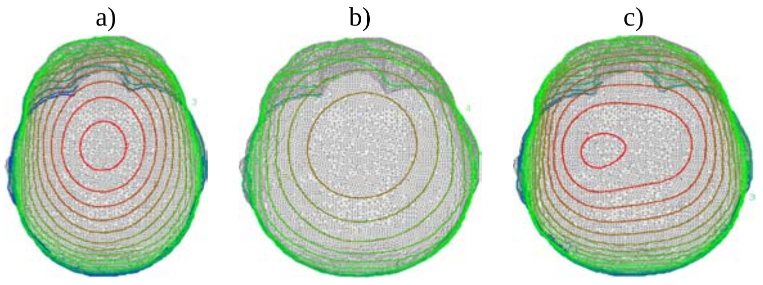


Fig. 20: Thalamic source: Isopotential distribution from -0.6 to 0.4 μV on upper part of the head a) isotropic model: $\lambda_{rad} = 0.0042$, $\lambda_{tang} = 0.0042$ and $\lambda_{trans} = 0.14$, $\lambda_{long} = 0.14$. b) 1:10 anisotropic white matter (volume constraint): $\lambda_{trans} = 0.065$ and $\lambda_{long} = 0.65$; isotropic skull. c) 1:10 anisotropic skull (volume constraint): $\lambda_{rad} = 0.000905$ and $\lambda_{tang} = 0.00905$); isotropic white matter.

conductivity eigenvalues and a reduction of the radial conductivity eigenvalue by a factor of 10 (left) or by means of a fixation of the radial conductivity eigenvalue and an increase of the tangential conductivity eigenvalues by a factor of 10 (right).

When comparing our results on skull anisotropy in realistic FE models with anisotropy examinations on multilayered sphere models [Zhou and van Oosterom, 1992; Marin *et al.*, 1998], we find the important difference that the EEG topography error between isotropic and 1:10 anisotropic skull modeling in our realistic FE head model (in agreement with the realistic FE head model in [Marin *et al.*, 1998]) is much larger than the EEG topography error between isotropic and 1:10 anisotropic spherical layer modeling, reported in [Zhou and van Oosterom, 1992; Marin *et al.*, 1998]. Additionally, a MAG close to 1 was pointed out for the spherical model [Zhou and van Oosterom, 1992], whereas for the realistic FE model, we found a MAG larger 1, like [Marin *et al.*, 1998]. Spherical symmetry effects could thus play a role, reducing topography and magnitude errors for the multilayered sphere model. However, these errors seem to be apparent in realistically shaped head models.

1:10 skull anisotropy was found to have no influence (RDM $<$ 1%, MAG $\approx$ 1) on the MEG topography and magnitude for both volume and Wang's constraint. This is in agreement with the generally accepted idea that volume currents in the skull and scalp layer give negligible contributions to the magnetic field [Hämäläinen and Sarvas, 1987] and with results in realistic FE head models [van den Broek *et al.*, 1997].

Our observations regarding the influence of white matter anisotropy on EEG/MEG for an isotropic skull layer agree well with the results in [Haueisen *et al.*, 2002]. [Haueisen *et al.*, 2002] reported small topography and magnitude errors for EEG and MEG for an eccentric source with large tangential orientation component. We observed a negligible influence for a ratio of 1:10 on the EEG topography with an RDM close to 1% and on the MEG topography with an RDM of about 5%. Our observed MAG is close to the optimum in the realistic anisotropy range for both EEG and MEG (see Figure 18).

We computed EEG and MEG in the EALC model (see definition in 5.1.2) and compared these forward solutions to the corresponding model with 1:10 anisotropy of white matter and skull (volume constraint). For both EEG and MEG, an RDM smaller than 1% and a MAG of about 1 was found.

In summary, for the EEG, 1:10 anisotropy of both skull and white matter layer, leads to a topography error of about 9% for the volume constraint and 11% for Wang's constraint. The topography error is mainly due to skull anisotropy, whereas white matter anisotropy has only a small influence on the potential distribution for the chosen eccentric source with large tangential orientation component. When choosing the volume constraint, the magnitude error MAG for 1:10 anisotropy of skull and white matter layer is kept close to the optimum of 1.0, whereas Wang's constraint leads to a MAG of 1.23. An increase of radial or tangential skull conductivity contracts (in Figure 19, compare top row, left with bottom row, right), whereas a decrease spreads out the isopotential distribution on the surface. The pattern is also distorted (in Figure 19, compare top row, left with bottom row, left), so that an approximation of skull anisotropy effects by means of an increase or a decrease of the scalar isotropic skull conductivity value in BE head models seems to be impossible. This is in particular true, because further computations with different source locations and orientations showed that contraction or spreading out depends on parameters such as location/orientation of the source and skull shape and thickness (not shown).

For the MEG, no influence of skull anisotropy was observed. The influence of white matter was small for realistic anisotropy ratios.

5.2.2 Results for a radially oriented eccentric source

Forward calculations for the eccentric source with large radial orientation component (Figure 16, middle) and a strength of 10 nAm were carried out in the 5-tissue model. Figure 21 shows the resulting RDM (left) and MAG (right) errors for EEG and MEG.

1:10 realistic white matter anisotropy for a radially oriented eccentric source has a strong influence on the topography of EEG and MEG, a result which again mainly agrees with the results in [Haueisen *et al.*, 2002]. [Haueisen *et al.*, 2002] reported a large topography error for both EEG and MEG (our MEG results have to be compared to the flux density component B_y in Table 2 in [Haueisen *et al.*, 2002]) and a moderate magnitude error for an eccentric almost radially oriented source. We observed an EEG topography error of RDM=23% for the volume and RDM=20% for Wang's constraint and an MEG topography error of about 15% for both constraints. The large MEG topography error can be explained by the fact that white matter anisotropy influences the secondary (return) currents and that the ratio of the secondary to the whole magnetic flux increases with increasing ratio of the radial dipole orientation component [Haueisen *et al.*, 1995]. Remember that for radially oriented sources in spherical head models, the primary magnetic flux (and because of spherical symmetry effects also the secondary magnetic flux) is zero outside the model. For both EEG and MEG, the MAG was again close to the optimum 1.0 for realistic white matter anisotropy ratios except for the MEG in combination with the volume constraint, where the error was close to 1.1.

We again found an RDM smaller than 1% and a MAG of about 1 between forward results in the EALC (see 5.1.2) and the corresponding locally unchanged model for both EEG and MEG.

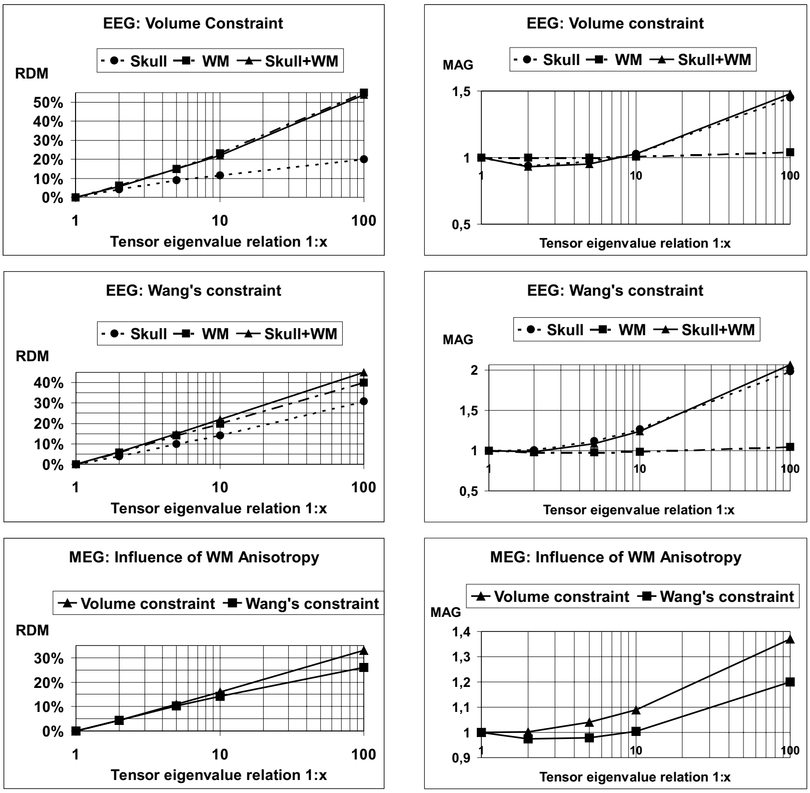


Fig. 21: Eccentric source, large radial orientation component: EEG/MEG topography error (left) and magnitude error (right) for different anisotropy ratios: For EEG, errors due to anisotropy effects of skull, white matter (WM) and both skull and WM are presented for the tensor volume retaining (top row) and Wang's constraint [Wang et al., 2001] (middle row). For MEG, only WM anisotropy effects for both constraints are presented, since skull anisotropy was found to have no influence.

Our EEG results concerning 1:10 anisotropy of the skull agree well with the observations in [Marin *et al.*, 1998]. With an RDM of 12% for the volume and 14% for Wang's constraint, the influence on the potential topography is in the range as seen for the tangential dipole. We achieved a MAG close to 1 for the volume and a MAG of 1.27 for Wang's constraint. For MEG, as for the tangential eccentric source, no influence (RDM $<$ 1%, MAG $\approx$ 1) of skull anisotropy was found (see [Hämäläinen and Sarvas, 1987], [van den Broek *et al.*, 1997]).

In summary, 1:10 anisotropy of both considered compartments leads to

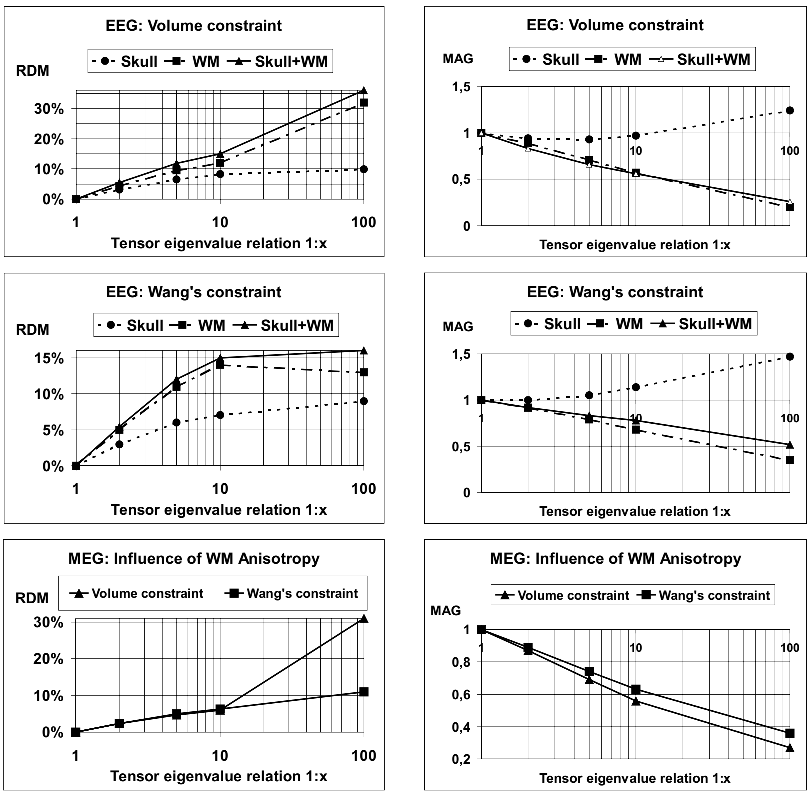


Fig. 22: Deep thalamic source: EEG/MEG topography error (left) and magnitude error (right) for different anisotropy ratios: For EEG, errors due to anisotropy effects of skull, white matter (WM) and both skull and WM are presented for the tensor volume retaining (top row) and Wang's constraint [Wang et al., 2001] (middle row). For MEG, only WM anisotropy effects for both constraints are presented, since skull anisotropy was found to have no influence.

a non-negligible topography error for both EEG and MEG. This error is mainly due to white matter anisotropy. Skull anisotropy has no influence on the MEG, but a non-negligible influence on the EEG. For EEG, the volume constraint led to a smaller MAG error than Wang's constraint, whereas we observed the inverse for the MEG. The local change from anisotropic to isotropic conductivity of white matter elements in the neighborhood of the source did not influence our error considerations.

5.2.3 Results for a deep source

In a last simulation, forward calculations in the 5-tissue-model were carried out for a deep and thus almost radial source (Figure 16, right) with a strength of 10 nAm. In the thalamus, tissue structure is almost radially oriented. Figure 22 shows the resulting RDM and MAG errors. Figure 20 (before p. 143) shows the isopotential distribution of the thalamic source for three different volume conductor models. The isopotentials for the isotropic 5-tissue model are shown on the left. In the middle, we present the result for an isotropic skull layer and a 1:10 anisotropic white matter compartment (volume constraint) and on the right the result for 1:10 anisotropic skull (volume constraint) and isotropic white matter compartment.

It can be observed from the Figs. 22 and 20, that 1:10 anisotropy of white matter in combination with an isotropic skull layer leads to a non-negligible topography error larger 10% for the EEG, whereas with 6%, the error is moderate for the MEG, but it is then strongly increasing for larger anisotropy ratios. White matter anisotropy strongly decreased the surface potential (Figure 20, middle, and MAG=0.57 in Figure 22, top row, right) and the magnetic fields (MAG$\approx$0.6 in Figure 22, bottom row). The former is related to the results of [Zhou and van Oosterom, 1992], who reported a decreased (increased) potential magnitude for increased (decreased) radial conductivity in the inner sphere of a multilayer sphere model, whereas for a change in the tangential conductivity component, no influence was found.

When comparing forward computation results between DALC (see definition in 5.1.2) and the corresponding locally unchanged model, we found an RDM of 16% and a MAG of 1.34 for the EEG and an RDM of 14% and a MAG of 1.19 for the MEG. In contrast to the eccentric source, these results show the importance of local conductivity changes, as reported by [Haueisen *et al.*, 2000].

For EEG, we observed an RDM of about 7% for 1:10 anisotropic skull in combination with an isotropic white matter compartment (Figure 22 and Figure 20, right) and an increase in the magnitude for Wang's constraint (Figure 22, middle row, right, MAG = 1.14), while the magnitude was kept close to the optimum for the volume constraint (Figure 22, top row, right, MAG = 0.97). For the MEG, we only observed a small influence (RDM $<$ 3%, MAG=0.98) for Wang's constraint (not shown).

In summary, for the deep thalamic source, a non-negligible influence of 1:10 skull and white matter anisotropy on the EEG topography was found. The EEG topography error is mainly due to white matter, but to a minor extent also to skull anisotropy. For the MEG, we assume that white matter anisotropy has a non-negligible influence on the topography, whereas skull anisotropy can be neglected. White matter anisotropy strongly reduced EEG and MEG field magnitude. Local conductivity changes were strongly influencing our error considerations.

5.3 *Performance results and sensitivity of the FE solution process towards tissue conductivity anisotropy*

In [Wolters *et al.*, 2002], nearly linear speedups of the parallel AMG method were reported for high resolution realistically shaped isotropic tetrahedra and cube head models. In comparison to a classical parallel Jacobi preconditioned conjugate gradient method, large improvements in the computation time were found for the parallel AMG. In the following, the performance of both solver methods will be examined for the high resolution realistically shaped anisotropic tetrahedra model in comparison to the corresponding isotropic one. The stability of both preconditioners with regard to realistic tissue anisotropy will be analysed.

5.3.1 *The isotropic tetrahedra model*

We first discuss the FE solution process within the isotropic tetrahedra model. We chose the deep thalamic source. The local accumulation of the geometry matrix K_s on 1 processor took 225 seconds, parallelized on 12 processors a setup time of 19 seconds was achieved.

Figure 23 shows the wall-clock time of the parallel AMG-CG solver compared to the parallel Jacobi-CG. The number of iterations for both solvers, necessary for the required accuracy, is shown over the curves. The time for the setup of the preconditioner is not included, since it has to be carried out only once per head model and is thus negligible with regard to the solution of the inverse problem. To give an impression, the setup of the AMG on 1 processor took 39 seconds and parallelized on 12 processors 7.6 seconds. The 3D potential distribution was calculated on one processor within 370 seconds with the Jacobi-CG method, whereas the parallel AMG-CG method on 10 processors needed 4.2 seconds. This is a factor of about 88 (10.2 through multigrid preconditioning and 8.6 through parallelization on 10 processors).

The speedup results from 1 to 12 processors are shown in Figure 24. The matrix generation is purely local and gives the reference curve for the quasi optimal speedup. This curve can also be seen as an indicator for the quality of the mesh-partitioning, described in Subsection 3.2. The speedups for the parallel AMG-CG solver, for one iteration of this solver and for the parallel Jacobi-CG solver are compared.

Since the coarsening process and the determination of the prolongation matrix $\mathfrak{P}$ respecting pattern condition (23) in the setup of the parallel AMG-preconditioner and the smoother-component of the solver depend on the de-

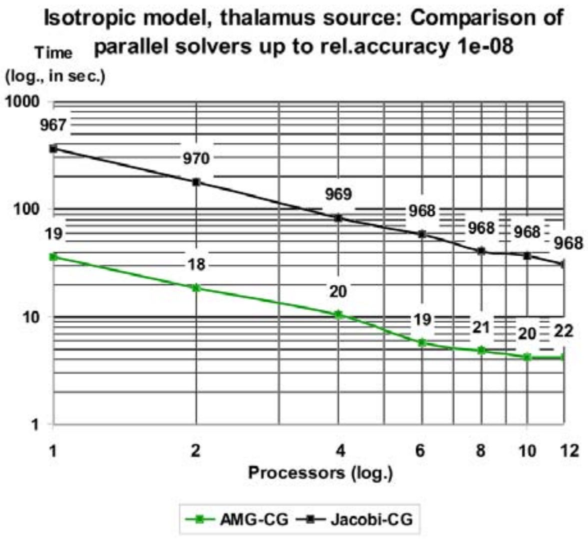


Fig. 23: SGI ORIGIN: Wall-clock time from 1 to 12 processors for the solver part of the parallel AMG-CG compared to the parallel Jacobi-CG up to an accuracy of 10^{-8} *for the realistic tetrahedra head model, 147287 nodes and a dipole source in the thalamus. The number of iterations are shown over the curves.*

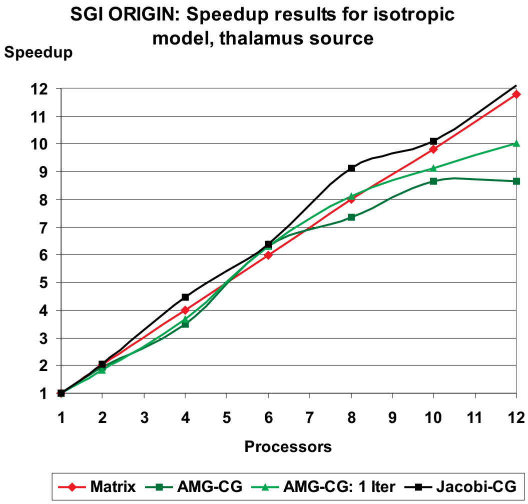


Fig. 24: SGI ORIGIN: Speedup results from 1 to 12 processors for the tetrahedra head model, 147287 nodes and a deep thalamic source.

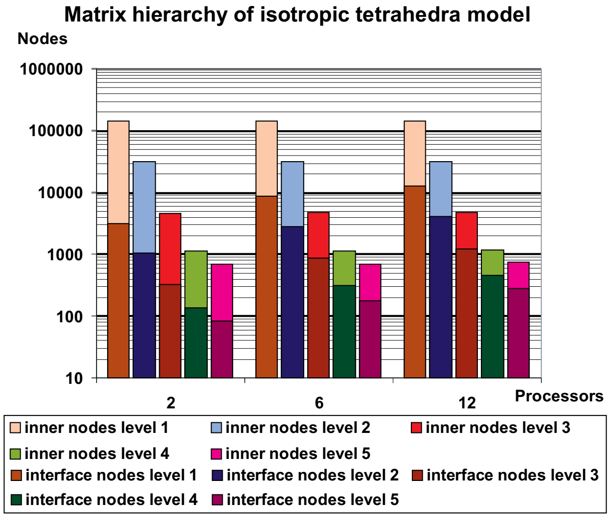


Fig. 25: Isotropic tetrahedra head model, 147287 nodes: Relation interface nodes to all nodes (interface plus inner nodes) on the four levels of the algebraic multigrid, exemplarily for the decompositions for 2, 6 and 12 processors.

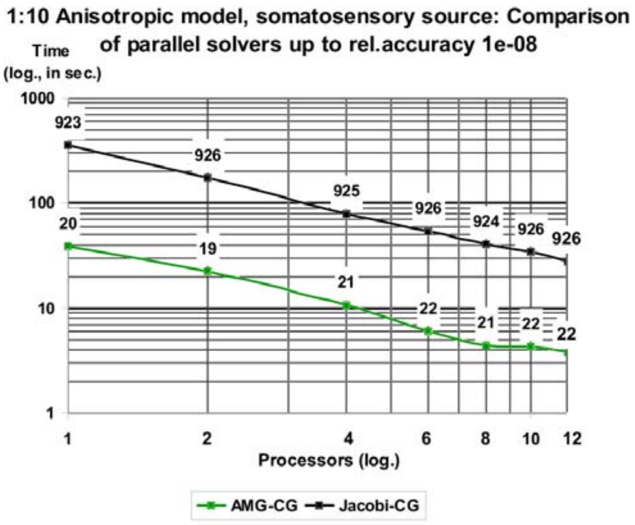


Fig. 26: SGI ORIGIN: Wall-clock time from 1 to 12 processors for the solver part of the parallel AMG-CG compared to the parallel Jacobi-CG up to an accuracy of 10^{-8} *for 1:10 anisotropic tetrahedra head model, 147287 nodes and the almost tangentially oriented eccentric source. The numbers of iterations are shown over the curves.*

composition into subdomains and a strongly increasing number of interface-nodes would spoil the preconditioning effect, it is interesting to have a look at the relation of interface nodes to all nodes (interface plus inner nodes) on the different levels of the multigrid. Figure 25 shows this relation exemplarily for 2, 6 and 12 processors. The decomposition into two domains lead to 3084 and thus 2% interface nodes on the finest level. On the fourth level (their is no more smoother component on the fifth and coarsest virtual grid), 133 out of 1117 nodes were interface nodes and thus a percentage of 12%. On 12 processors, 12462 and thus 8% were interface nodes on the finest level and on the fourth level, 461 out of 1164, i.e., 40%. In summary, it can be concluded that the results are close to what we found for the isotropic tetrahedra head model in [Wolters *et al.*, 2002].

5.3.2 1:10 anisotropic tetrahedra model

We now discuss the solution process for the potential distribution within the anisotropic tetrahedra model with 1:10 anisotropic skull and white matter layer for the tangentially oriented eccentric source. The accumulation of the local geometry matrices on 1 and parallelized on 12 processors takes the same times as it was presented for the isotropic model above.

Figure 26 shows the wall-clock time of the parallel AMG-CG solver compared to the parallel Jacobi-CG for the anisotropic model. Again, the number of iterations is shown over the curves. As for the isotropic model, the time for the setup of the preconditioner is not included. The setup of the AMG on 1 processor took 39 seconds and parallelized on 12 processors 7 seconds in the anisotropic case. The 3D potential distribution was calculated on one processor within 361 seconds with the Jacobi-CG method, whereas the parallel AMG-CG method on 12 processors needed 3.8 seconds. This is a factor of about 95 (9.3 through multigrid preconditioning and 10.2 through parallelization on 12 processors).

In [Wolters *et al.*, 2000], condition numbers of high resolution 4 layer sphere FE models were computed. No remarkable difference between isotropic models and models with 1:10 anisotropic third (“skull”) layer were reported. It was observed, that the anisotropy lead to a slight increase of Jacobi iterations, whereas the iteration count of the AMG-CG was constant. For the realistic tetrahedra head model, we now observe the inverse behavior, a slight decrease of Jacobi-CG iterations (967 for the isotropic and 923 for the anisotropic) and a slight increase of AMG-CG iterations (19 for the isotropic and 20 for the anisotropic case). If we expect that the condition numbers of isotropic and anisotropic models are again in the same range, we explain the slight changes of the iteration count to be in the range of $O(1)$. Remember also that different sources were chosen for the isotropic and the anisotropic model. The speedup results from 1 to 12 processors are shown in Figure 27. Again, the matrix generation gives the reference curve for the quasi optimal speedup.

It is also interesting to consider the number of interface nodes as a fraction of all nodes on the five levels of the multigrid. Figure 28 shows these node numbers for 2, 6 and 12 processors. The decomposition into two domains lead to 3084 and thus again 2% interface nodes on the finest level. On the 4^{th} level, 120 out of 1022 nodes were interface nodes and thus a percentage of 12%. On 12 processors, 12462 and thus 8% were interface nodes on the finest level and on the fourth level, 443 out of 1066, i.e., 42%.

6 Conclusions

Head tissue conductivity anisotropy has a non-negligible influence on the EEG/MEG field distribution and should therefore be taken into account. These findings in our article are in agreement with results of [van den Broek *et al.*, 1997; Marin *et al.*, 1998; Haueisen *et al.*, 2002]. We presented non-invasive measurement techniques and image processing methods for obtaining a high resolution anisotropic map of the two tissue compartments skull and white matter. For solving the EEG/MEG inverse problem, the field distribution has to be simulated repeatedly for given dipolar sources in the brain. The chosen modern numerical approach in combination with high performance computing on a parallel machine was shown to yield computation times, which should push high resolution realistically shaped anisotropic forward modeling within the EEG/MEG inverse problem into the application fields.

Our presented head model was constructed from an MRI segmentation of the five tissue compartments skin, skull, cerebrospinal fluid and brain grey and white matter. The modeling of skull anisotropy was based on results for an improved segmentation of inner and outer skull surfaces, using bimodal MRI data. Realistic skull conductivity tensors were obtained by means of the surface normals of a smooth surface spongiosa model, for which the segmentation results of inner and outer skull were exploited. Whole head DT-MRI measurements were used to determine the conductivity tensor eigenvectors for each point in the white matter compartment. The diffusion tensor eigenvector with largest measured eigenvalue was taken for modeling the high conductive longitudinal white matter fibre direction.

Boundary element method based volume conductor models (see, e.g., [Fuchs *et al.*, 1998; Zanow and Peters, 1995]) cannot take tissue conductivity anisotropy into account. In multilayered sphere models [de Munck and Peters, 1993; Zhou and van Oosterom, 1992], anisotropy of the inner spherical

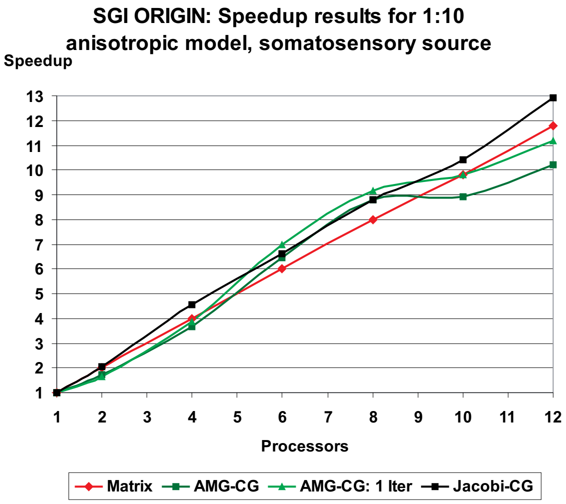


Fig. 27: SGI ORIGIN: Speedup results for 1:10 anisotropic tetrahedra head model and the almost tangentially oriented eccentric source, 147287 nodes, relative accuracy 10^{-8}.

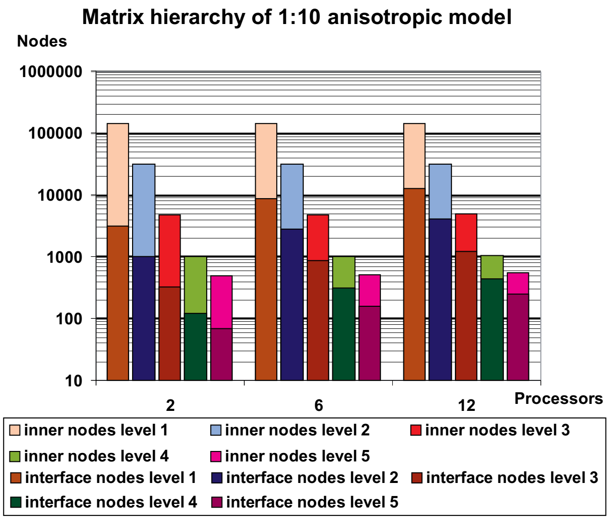


Fig. 28: 1:10 anisotropic tetrahedra head model, 147287 nodes: Relation interface nodes to all nodes (interface plus inner nodes) on the five levels of the algebraic multigrid, exemplarily for the decompositions for 2, 6 and 12 processors.

"white matter compartment" can only be represented by a single tensor, which is not realistic. We used high resolution realistic finite element head modeling, where a measured or realistically modeled tensor-valued conductivity was assigned to each finite element in the skull and white matter compartment. In order to test the influence of tissue anisotropy on EEG and MEG, different anisotropy ratios were modeled, following a constraint, proposed in [Wang *et al.*, 2001], and a tensor volume retaining constraint for achieving the desired ratio.

For the EEG, our influence results agree to a large extent with [van den Broek *et al.*, 1997; Marin *et al.*, 1998; Haueisen *et al.*, 2002]. For a tangentially oriented eccentric source, we found a topography error of about 10% between the isotropic and a 1:10 anisotropic model. The errors are mainly due to the skull, whereas we found that white matter anisotropy was negligible. The magnitude error for the volume retaining constraint was close to the optimum, whereas with increasing anisotropy ratio, it increased for Wang's constraint. An increase of radial or tangential skull conductivity contracts, whereas a decrease spreads out the isopotential distribution on the surface. The isopotential pattern is also distorted, so that an approximation of skull anisotropy effects by means of an increase or a decrease of a scalar isotropic skull conductivity value in boundary element head models seems to be impossible. Thus, dipole mislocalizations have to be expected for tangentially oriented eccentric sources, if skull layer anisotropy is not taken into account.

For a radially oriented eccentric source, we found an EEG topography error of more than 20% between the isotropic and a 1:10 anisotropic model. In this case, the topography errors are mainly due to white matter anisotropy, the error of the skull layer was only about half the one of the white matter compartment. The magnitude error for the volume retaining constraint was again close to the optimum, while, again, it was larger for Wang's constraint. A change of the local conductivities in the neighborhood of the eccentric source did not influence our error considerations. Larger dipole mislocalizations thus have to be expected for radially oriented eccentric sources if white matter or skull anisotropy is not taken into account.

For a deep source, we found an EEG topography error of about 15%, mainly due to white matter, but to a smaller extent also to skull layer anisotropy. White matter anisotropy strongly reduced the magnitude of the surface potential. A change of the local conductivities in the neighborhood of the deep source strongly influenced our error considerations, which is in agreement with results of [Haueisen *et al.*, 2000]. Larger dipole mislocalizations and errors in dipole strength estimations have to be expected for deeper sources if white matter or skull anisotropy is not taken into account..

We conclude that for robust EEG-based dipole source reconstruction in the human brain, tissue conductivity anisotropy of the skull as well as the white matter compartment has to be taken into account so that high resolution FE head modeling is needed.

We found, that skull anisotropy has no influence on the MEG, which is in agreement with the results of [van den Broek *et al.*, 1997] and with the generally accepted idea that volume currents in the skull and scalp layer give negligible contributions to the magnetic field [Hämäläinen and Sarvas, 1987]. In contrast to the skull compartment, white matter anisotropy was found to have a non-negligible influence on the MEG. In agreement with [Haueisen *et al.*, 2002], the topography error was moderate for an eccentric source with almost tangential orientation (about 5% for a ratio of 1:10), whereas it was much larger for an eccentric source with almost radial orientation (about 15%). The larger error can be explained by the fact that tissue anisotropy only influences the secondary (return) currents and that the ratio of the secondary to the whole magnetic flux increases with increasing ratio of the radial dipole orientation component [Haueisen *et al.*, 1995]. As for the EEG, a change of the local conductivities in the neighborhood of the eccentric source did not influence the error considerations. The magnitude error was again close to the optimum for both eccentric sources. For the deep source, in particular the magnitude error was large (MAG of about 0.6 for a ratio of 1:10), and from our results we also deduce that the topography error is not negligible. A change of the local conductivities in the neighborhood of the deep source strongly influenced the MEG error considerations, which is in agreement with results of [Haueisen *et al.*, 2000]. For MEG we conclude that white matter conductivity anisotropy has to be taken into account, if sources with a non-negligible radial orientation component should be reconstructed or if statements about the strength of deeper sources have to be made.

The bottleneck for a broader application of finite element method based anisotropic forward modeling is the time for solving the large linear equation system with thousands of different right hand sides arising from the FE discretization. Within this article, an efficient and memory-economical way was presented to face this problem. Very short calculation times were achieved through the combination of AMG preconditioning techniques and its parallelization on distributed memory platforms. We compared the presented AMG-CG with the Jacobi-CG, the latter being a well-known solver method in FE-based source localization. Our findings with regard to anisotropic head modeling strongly agree with the results in isotropic volume conductors [Wolters *et al.*, 2002]. If the Jacobi-CG on a single processor is taken as a reference, we achieved a speedup of 95 for a realistically shaped high resolution 1:10 anisotropic tetrahedra head model with 147287 nodes when comparing to the parallel AMG-CG on 12 processors, 9.3 through multigrid preconditioning and 10.2 through parallelization on 12 processors. The solver process was thus shown to be stable with respect to realistic tissue anisotropy.

The partitioning of the dual graph of a convex head geometry generally leads to a relatively large percentage of interface nodes. Nevertheless, for the examined moderate processor numbers between 1 and 12, the AMG-preconditioner was found to be stable, i.e., a sensible increase of the number of subdomains did not result in a deterioration of the AMG-preconditioner and thus an increasing need for iterations. We showed that the latter is true both for the isotropic and for the anisotropic model.

The presented methods and software concepts provide higher accuracy in EEG/MEG source localization by accounting for tissue conductivity anisotropy on the basis of a high resolution realistically shaped FE head model. Our methods allow the study of the influence of anisotropy on the inverse source localization results, where many forward solutions have to be computed. This will be done in future examinations. A first performance test of the NeuroFEM software on a Linux PC-cluster with 100 MBit ethernet showed very good parallelization speedups and, because of the 1 GB processors, a strongly reduced computation time compared to the presented results on the SGI ORIGIN 2000. The presented algorithms can thus be used on a simple PC-cluster. Moreover, the overall solver CPU-time for the inverse source reconstruction will be accelerated through the use of techniques for multiple right hand sides [Chan and Wan, 1997; Haase and Reitzinger, 2002]. First studies on a sequential computer have shown a good performance.

References

[Akhtari *et al.*, 2000] M. Akhtari, H.C. Bryant, A.N. Marmelak, L. Heller, J.J. Shih, M. Mandelkern, A. Matlachov, D.M. Ranken, E.D. Best, and W.W. Sutherling. Conductivities of three-layer human skull. *Brain Top.*, 13(1):29–42, 2000.

[Alexander *et al.*, 2001] D.C. Alexander, C. Pierpaoli, P.J. Basser, and J.C. Gee. Spatial transformations of diffusion tensor magnetic resonance images. *IEEE Trans. Biomed. Eng.*, 20(11):1131–1139, 2001.

[Andrä and Nowak, 1998] W. Andrä and H. Nowak. *Magnetism in medicine – a handbook*. Wiley–VCH, 1998.

[Awada *et al.*, 1997] K.A. Awada, D.R. Jackson, J.T. Williams, D.R. Wilton, S.B. Baumann, and A.C. Papanicolaou. Computational aspects of finite element modeling in EEG source localization. *IEEE Trans. Biomed. Eng.*, 44(8):736–751, 1997.

[Basser and Pierpaoli, 1996] P.J. Basser and C. Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitive-diffusion-tensor MRI. *J. Magn. Reson. B*, 111:209–219, 1996.

[Basser *et al.*, 1994a] P.J. Basser, J. Mattiello, and D. LeBihan. Estimation of the effective self-diffusion tensor from the NMR spin echo. *J. Magn. Reson. B*, 103:247–254, 1994.

[Basser *et al.*, 1994b] P.J. Basser, J. Mattiello, and D. LeBihan. MR diffusion tensor spectroscopy and imaging. *Biophysical Journal*, 66:259–267, 1994.

[Bastian *et al.*, 1997] P. Bastian, K. Birken, K. Johannsen, S. Lang, N. Neuss, H. Rentz-Reichert, and C. Wieners. UG – A flexible Software Toolbox for Solving Partial Differential Equations. *Comput. Vis. Sci.*, 1:27 – 40, 1997.

[Baumann *et al.*, 1997] S.B. Baumann, D.R. Wozny, S.K. Kelly, and F.M. Meno. The electrical conductivity of human cerebrospinal fluid at body temperature. *IEEE Trans. Biomed. Eng.*, 44(3):220–223, 1997.

[Bertrand *et al.*, 1991] O. Bertrand, M. Thévenet, and F. Perrin. 3D finite element method in brain electrical activity studies. In J. Nenonen, H.M. Rajala, and T. Katila, editors, *Biomagnetic Localization and 3D Modelling*, pages 154–171. Report of the Department of Technical Physics, Helsinki Univ. of Technology, 1991.

[Braess, 1995] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[Buchner *et al.*, 1997] H. Buchner, G. Knoll, M. Fuchs, A. Rienäcker, R. Beckmann, M. Wagner, J. Silny, and J. Pesch. Inverse localization of electric dipole current sources in finite element models of the human head. *Electroenc. Clin. Neurophysiol.*, 102:267–278, 1997.

[Burkhardt *et al.*, 2002] S. Burkhardt, C.H. Wolters, and D. Saupe. Segmentation of human skull surfaces from bimodal MRI volumes. 2002. submitted to Med.Imag.Anal.

[CAUCHY, 1997] CAUCHY. Anatomic source reconstruction of EEG/MEG-data. http://www.rwth-aachen.de/ neurologie/, 1997.

[Chan and Wan, 1997] T.F. Chan and W. L. Wan. Analysis of projection methods for solving linear systems with multiple right-hand sides. *SIAM J. Sci. Comput.*, 18(6):1698 – 1721, 1997.

[CURRY, 2000] CURRY. Current reconstruction and imaging. NeuroScan Labs, http://www.neuro.com/ neuroscan/ prod05.htm, 2000.

[de Munck and Peters, 1993] J.C. de Munck and M. Peters. A fast method to compute the potential in the multi sphere model. *IEEE Trans. Biomed. Eng.*, 40(11):1166–1174, 1993.

[de Munck *et al.*, 1988] J.C. de Munck, B.W. van Dijk, and H. Spekreijse. Mathematical dipoles are adequate to describe realistic generators of human brain activity. *IEEE Trans. Biomed. Eng.*, 35(11):960–966, 1988.

[Falgout *et al.*, 1999] R. Falgout, Van E. Henson, J. E. Jones, and U. Meier Yang. Boomer AMG: A parallel implementation of algebraic multigrid. Techn. Report UCRL-MI-133583, Lawrence Livermore National Laboratory, March 1999.

[Fuchs *et al.*, 1998] M. Fuchs, M. Wagner, H.A. Wischmann, T. Köhler, A. Theißen, R. Drenckhahn, and H. Buchner. Improving source reconstructions by combining bioelectric and biomagnetic data. *Electroenc. Clin. Neurophysiol.*, 107:93–111, 1998.

[Gueziec and Hummel, 1994] A. Gueziec and R. Hummel. The wrapper algorithm: surface extraction and simplification. In *IEEE Workshop on Biomedical Image Analysis, Seattle*, pages 204–213. IEEE Computer Press, Los Alamitos, 1994.

[Haase and Reitzinger, 2002] G. Haase and S. Reitzinger. Cache issues of algebraic multigrid methods for linear systems with multiple right-hand sides. Technical Report 02-05, J. Kepler Univ. Linz, SFB "Numerical and Symbolic Scientific Computing", 2002.

[Haase *et al.*, 2000] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl. A general approach to algebraic multigrid. Technical Report 00-33, J. Kepler Univ. Linz, SFB "Numerical and Symbolic Scientific Computing", 2000.

[Haase *et al.*, 2002] G. Haase, M. Kuhn, and S. Reitzinger. Parallel AMG on distributed memory computers. *SIAM J. Sci.Comp.*, 2002. in press.

[Haase, 1999] G. Haase. *Parallelisierung numerischer Algorithmen für partielle Differentialgleichungen*. Teubner, 1999.

[Hackbusch, 1985] W. Hackbusch. *Multigrid Methods and Application*. Springer Verlag, 1985.

[Hämäläinen and Sarvas, 1987] M.S. Hämäläinen and J. Sarvas. Feasibility of the homogeneous head model in the interpretation of neuromagnetic fields. *Phys.Med.Biol.*, 32:91–97, 1987.

[Haueisen *et al.*, 1995] J. Haueisen, C. Ramon, P. Czapski, and M. Eiselt. On the influence of volume currents and extended sources on neuromagnetic fields: A simulation study. *Annals of Biomedical Engineering*, 23:728–739, 1995.

[Haueisen *et al.*, 2000] J. Haueisen, C. Ramon, H. Brauer, and H. Nowak. The influence of local conductivity changes on MEG and EEG. *Biomedizinische Technik*, 45(7-8):211–214, 2000.

[Haueisen *et al.*, 2002] J. Haueisen, D.S. Tuch, C. Ramon, P.H. Schimpf, V.J. Wedeen, J.S. George, and J.W. Belliveau. The influence of brain tissue anisotropy on human EEG and MEG. *NeuroImage*, 15:159–166, 2002.

[Haueisen, 1996] J. Haueisen. *Methods of Numerical Field Calculation for Neuromagnetic Source Localization*. PhD thesis, Technical Univ.Ilmenau, 1996.

[Huiskamp *et al.*, 1999] G. Huiskamp, M. Vroeijenstijn, R. van Dijk, G. Wieneke, and A.C. van Huffelen. The need for correct realistic geometry in the inverse EEG problem. *IEEE Trans. Biomed. Eng.*, 46(11):1281–1287, 1999.

[Johnson *et al.*, 2000] C.R. Johnson, M. Mohr, U. Rüde, A. Samsonov, and K. Zyp. Multilevel methods for inverse bioelectric field problems. In R. Bank, T. Barth, and T. Chan, editors, *Yosemite Educational Symposium*, http://raphael.mit.edu/yosemite/, Oct.29–Nov.1, 2000.

[Jung and Langer, 1991] M. Jung and U. Langer. Applications of multilevel methods to practical problems. *Surv.Math.Ind.*, 1:217–257, 1991.

[Karypis and Kumar, 1998] G. Karypis and V. Kumar. METIS – User's Guide. University of Minnesota, http://www-users.cs.umn.edu/ ∼karypis, 1998.

[Kickinger, 1998] F. Kickinger. Algebraic multigrid for discrete elliptic second-order problems. In W. Hackbusch, editor, *Multigrid Meth. V. Proc. of the 5th Europ. Multigrid conf.*, pages 157–172. Springer Lect. Notes Comput. Sci. Eng. 3, 1998.

[Knösche, 1997] T.R. Knösche. *Solutions of the neuroelectromagnetic inverse problem*. PhD thesis, Univ. of Twente, The Netherlands, 1997.

[Koch, 2000] M. Koch. *Measurement of the Self-Diffusion Tensor of Water in the Human Brain*. PhD thesis, Univ. of Leipzig, Germany, 2000.

[Krechel and Stüben, 2001] A. Krechel and K. Stüben. Parallel algebraic multigrid based on subdomain blocking. *Parallel Comp.*, 27(8):1009 – 1031, 2001.

[Latour *et al.*, 1994] L.L. Latour, K. Svoboda, P.P. Mitra, and C.H. Sotak. Time-dependent diffusion of water in a biological model system. *Proc.Natl.Acad.Sci.USA*, 91:1229–1233, 1994.

[Maes *et al.*, 1997] F. Maes, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. *IEEE Trans. Med. Imag.*, 16(2):187–198, 1997.

[Maess *et al.*, 2001] B. Maess, S. Koelsch, T.C. Gunter, and A.D. Friederici. Musical syntax is processed in broca's area: an MEG study. *Nature Neuroscience*, 4(5):540–545, 2001.

[Marin *et al.*, 1998] G. Marin, C. Guerin, S. Baillet, L. Garnero, and Meunier G. Influence of skull anisotropy for the forward and inverse problem in EEG: simulation studies using the FEM on realistic head models. *Human Brain Mapping*, 6:250–269, 1998.

[Marin, 1997] G. Marin. *Utilisation de la méthode des éléments finis pour le calcul des champs électromagnétiques à l'aide d'un modèle réaliste de tête en MEG et EEG*. PhD thesis, Université de Paris-Sud, U.F.R. Scientifique d'Orsay, France, 1997.

[Meijs *et al.*, 1989] J.W.H Meijs, O.W. Weier, M.J. Peters, and A. van Oosterom. On the numerical accuracy of the boundary element method. *IEEE Trans. Biomed. Eng.*, 36:1038–1049, 1989.

[Nicholson, 1965] P.W. Nicholson. Specific impedance of cerebral white matter. *Exp.Neurol.*, 13:386–401, 1965.

[Nolting, 1992] W. Nolting. *Grundkurs: Theoretische Physik, Elektrodynamik*. Zimmermann-Neufang, Ulmen, 1992.

[Norris and Börnert, 1993] D.G. Norris and P. Börnert. Coherence and interference in ultra-fast rare experiments. *J. Magn. Reson. A*, 105:123–127, 1993.

[Öztekin *et al.*, 1998] B.U. Öztekin, G. Karypis, and V. Kumar. PMVIS – User's Guide. University of Minnesota, http://www-users.cs.umn.edu/ ∼oztekin/ pmvis/, 1998.

[Pastelak-Price, 1983] C. Pastelak-Price. The international 10-20-system of electrode placement: Its rationale and a practical guide to measuring procedures and electrode placement. *EEG-Labor*, 5:49–72, 1983.

[Payne and Toga, 1990] B.A. Payne and A.W. Toga. Surface mapping brain function on 3D models. *IEEE Computer Graphics & Applications*, 10(5):33–41, 1990.

[Platzer, 1994] W. Platzer, editor. *Pernkopf Anatomie*. Urban & Schwarzenberg, München, 1994.

[Plonsey and Heppner, 1967] R. Plonsey and D. Heppner. Considerations on quasi-stationarity in electro-physiological systems. *Bull.math.Biophys.*, 29:657–664, 1967.

[Pohlmeier, 1996] R. Pohlmeier. Lokalisation elektrischer Gehirnaktivität durch inverse Analyse des Magnetoenzephalogramms (MEG) mit Finite-Elemente-Modellen des Kopfes. Diplomarbeit in Elektrotechnik, RWTH Aachen, 1996.

[Reitzinger, 1999] S. Reitzinger. PEBBLES – User's Guide. SFB F013 "Numerical and Symbolic Scientific Computing", http://www.sfb013.uni-linz.ac.at., 1999.

[Reitzinger, 2001] S. Reitzinger. *Algebraic Multigrid Methods for Large Scale Finite Element Equations*. Number 36 in Schriften der Johannes-Kepler-Universität Linz, Reihe C - Technik und Naturwissenschaften. Universitätsverlag Rudolf Trauner, 2001.

[Ruge and Stüben, 1986] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. McCormick, editor, *Multigrid Methods*, volume 5 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, 1986.

[Sarvas, 1987] J. Sarvas. Basic mathematical and electromagnetic concepts of the biomagnetic inverse problem. *Phys.Med.Biol.*, 32(1):11–22, 1987.

[Scherg and von Cramon, 1985] M. Scherg and D. von Cramon. Two bilateral sources of the late AEP as identified by a spatio-temporal dipole model. *Electroenc. Clin. Neurophysiol.*, 62:32–44, 1985.

[Schimpf *et al.*, 2002] P.H. Schimpf, C.R. Ramon, and J. Haueisen. Dipole models for the EEG and MEG. *IEEE Trans. Biomed. Eng.*, 49(5):409–418, 2002.

[Schmitt and Louis, 2002] U. Schmitt and A.K. Louis. Efficient algorithms for the regularization of dynamic inverse problems-part I: Theory. *Inverse Problems*, 2002. in press.

[Schmitt *et al.*, 2002] U. Schmitt, A.K. Louis, C. Wolters, and M. Vauhkonen. Efficient algorithms for the regularization of dynamic inverse problems-part II: Applications. *Inverse Problems*, 2002. in press.

[Sen *et al.*, 1981] P.N. Sen, C. Scala, and M.H. Cohen. A self-similar model for sedimentary rocks with application to the dielectric constant of fused glass beads. *Geophysics*, 46(5):781–795, 1981.

[SimBio, 2000] SimBio. A generic environment for bio-numerical simulation. IST-program of the European Commission, Project No.10378, http://www.simbio.de, 2000.

[Smythe, 1989] W.R. Smythe. *Static and dynamic electricity*. Hemisphere Publishing, New York, 1989.

[Sutherling *et al.*, 1988] W.W. Sutherling, P.H. Crandall, T.M. Darcey, D.P. Becker, M.F. Levesque, and D.S. Barth. The magnetic and electrical fields agree with intracranial localizations of somatosensory cortex. *Neurology*, 38:1705–1714, 1988.

[Tuch *et al.*, 1998] D.S. Tuch, V.J. Wedeen, A.M. Dale, and J.W. Belliveau. Electrical conductivity tensor map of the human brain using NMR diffusion imaging: An effective medium approach. *ISMRM, 6th Scientific Meeting, Sydney*, 1998.

[Tuch *et al.*, 1999] D.S. Tuch, V.J. Wedeen, A.M. Dale, J.S. George, and J.W. Belliveau. Conductivity mapping of biological tissue using diffusion MRI. *Ann. NYAS*, 888:314–316, 1999.

[van den Broek *et al.*, 1997] S.P. van den Broek, M. Donderwinkel, and M.J. Peters. The influence of inhomogenities in a realistically shaped volume conductor on EEG and MEG. In H. Witte, U. Zwiener, B. Schack, and A. Doering, editors, *Quantitative and Topological EEG and MEG Analysis*, pages 456–458. Druckhaus Mayer Verlag GmbH Jena · Erlangen, 1997.

[van den Broek, 1997] S.P. van den Broek. *Volume Conduction Effects in EEG and MEG*. PhD thesis, Proefschrift Universiteit Twente Enschede, The Netherlands, 1997.

[Waberski *et al.*, 1998] T.D. Waberski, H. Buchner, K. Lehnertz, A. Hufnagel, M. Fuchs, R. Beckmann, and A. Rienäcker. The properties of source localization of epileptiform activity using advanced headmodelling and source reconstruction. *Brain Top.*, 10(4):283–290, 1998.

[Wagner, 1998] M. Wagner. *Rekonstruktion neuronaler Ströme aus bioelektrischen und biomagnetischen Messungen auf der aus MR-Bildern segmentierten Hirnrinde*. PhD thesis, Shaker-Verlag Aachen, 1998.

[Wagner, 2000] C. Wagner. On the algebraic construction of multilevel transfer operators. *Computing*, 65:73–95, 2000.

[Wang *et al.*, 2001] Y. Wang, D.R. Haynor, and Y. Kim. An investigation of the importance of myocardial anisotropy in finite-element modeling of the heart: Methodology and application to the estimation of defibrillation efficacy. *IEEE Trans. Biomed. Eng.*, 48(12), 2001.

[Wolters *et al.*, 1999a] C.H. Wolters, R.F. Beckmann, A. Rienäcker, and H. Buchner. Comparing regularized and non-regularized nonlinear dipole fit methods: A study in a simulated sulcus structure. *Brain Top.*, 12(1):3–18, 1999.

[Wolters *et al.*, 1999b] C.H. Wolters, U. Hartmann, M. Koch, F. Kruggel, S. Burkhardt, A. Basermann, D.S. Tuch, and J. Haueisen. New methods for improved and accelerated FE volume conductor modeling in EEG/MEG-source reconstruction. In J. Middleton, M. Jones, N. Shrive, and G. Pande, editors, *4th Symp. on Computer Methods in Biomech. and Biomed.Eng.*, pages 489–494, Lisboa, Okt.31–Nov.3 1999. Gordon & Breach.

[Wolters *et al.*, 2000] C. Wolters, S. Reitzinger, A. Basermann, S. Burkhardt, U. Hartmann, F. Kruggel, and A. Anwander. Improved tissue modeling and fast solver methods for high resolution FE-modeling in EEG/MEG-source localization. In J. Nenonen, R.J. Ilmoniemi, and T. Katila, editors, *Proc. of the 12th Int.Conf.of Biomagnetism*, pages 655–658, http://biomag2000.hut.fi/papers_all.html, Aug.13–17, 2000.

[Wolters *et al.*, 2002] C.H. Wolters, M. Kuhn, A. Anwander, and S. Reitzinger. A parallel algebraic multigrid solver for finite element method based source localization in the human brain. *Comp.Vis.Sci.*, 2002. in press.

[Wolters, 2002] C. Wolters. *Influence of tissue conductivity inhomogeneity and anisotropy to EEG/MEG based source localization in the human brain*. PhD thesis, University of Leipzig, 2002. in preparation.

[Zanow and Peters, 1995] F. Zanow and M.J. Peters. Individually shaped volume conductor models of the head in EEG source localisation. *Med. & Biol. Eng. & Comput.*, 7:151–161, 1995.

[Zhou and van Oosterom, 1992] H. Zhou and A. van Oosterom. Computation of the potential distribution in a four-layer anisotropic concentric spherical volume conductor. *IEEE Trans. Biomed. Eng.*, 39(2):154–158, 1992.

Anschriften der Autoren

Dr. Alfred Anwander
Max-Planck-Institut für neuropsychologische Forschung
MEG Gruppe
Muldentalweg 9, 04828 Bennewitz
E-Mail: anwander@cns.mpg.de

Markus Brugger
Institut für Hochenergiephysik
Nikolsdorfer Gasse, 1050 Wien, Österreich
E-Mail: Markus.Brugger@cern.ch

Dr. Andreas Buchleitner
Max-Planck-Institut fuer Physik komplexer Systeme
Nöthnitzer Str. 38, 01187 Dresden
E-Mail: abu@mpipks-dresden.mpg.de

Dr. Andreas Burkert
Max-Planck-Institut für Astronomie
Königstuhl 17, 69117 Heidelberg
E-Mail: burkert@mpia-hd.mpg.de

Markus Diesmann
Max-Planck-Institut für Stroemungsforschung
Bunsenstrasse 10, 37073 Göttingen
E-Mail: diesmann@chaos.gwdg.de

Massimiliano Fierro
Institut für Hochenergiephysik
Nikolsdorfer Gasse, 1050 Wien, Österreich
E-Mail: Massimiliano.Fierro@cern.ch

Marc-Oliver Gewaltig
Future Technology Research, Honda R&D Europe GmbH
Carl-Legien Str. 30, 63073 Offenbach/Main
E-Mail: Marc-Oliver.Gewaltig@hre-ftr.f.rd.honda.co.jo

Dr. Jörg Haber
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken
E-Mail: haberj@mpi-sb.mpg.de

Martin Koch
Max-Planck-Institut für neuropsychologische Forschung
Stephanstraße 1b, 04103 Leipzig
E-Mail: kochm@cns.mpg.de

Dr. Andreas Krug.
Max-Planck-Institut für Physik komplexer Systeme
Noethnitzer Str. 38, 01187 Dresden
E-Mail:apk@mpipks-dresden.mpg.de

Dr. Michael Kuhn
Bruker Saxonia Analytik GmbH
Permoserstrasse 15, 04318 Leipzig
E-Mail: mkn@bsax.de

Dr. Stefan Reitzinger
Institut für Numerische Mathematik
Johannes Kepler Universität Linz
Altenbergerstr. 69, 4040 Linz, Österreich
E-mail: reitz@numa.uni-linz.ac.at

Dr. Christoph Steinbeck
Max-Planck-Institut für chemische Ökologie
Carl-Zeiss-Promendae 10, 07745 Jena
E-Mail: steinbeck@ice.mpg.de

Markus Svensén
Max-Planck-Institut für neuropsychologische Forschung
Stephanstraße 1b, 04103 Leipzig

Carsten H. Wolters
Max-Planck-Institut für Mathematik in den Naturwissenschaften
Inselstr.22-26, 04103 Leipzig
E-Mail: wolters@cns.mpg.de

Dr. Claudia-Elsabeth Wulz
Institut für Hochenergiephysik
Nikolsdorfer Gasse, 1050 Wien, Österreich
E-Mail: Claudia.Wulz@cern.ch