

Modeling the Toning Process in Electrophotographic Copiers/Printers *

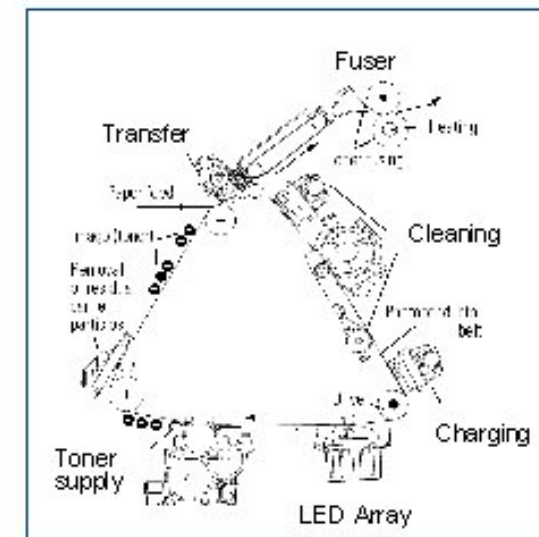
Ulrich Mutze, Bad Ditzenbach, www.ulrichmutze.de

- The toning process
- Modeling the dynamics of irregularly shaped particles
- Some aspects of the toning model
- Results

* Talk given at the Institute for Computer Applications 1 (ICA1) of the University of Stuttgart October 15th 2004. Extended version with later additions.

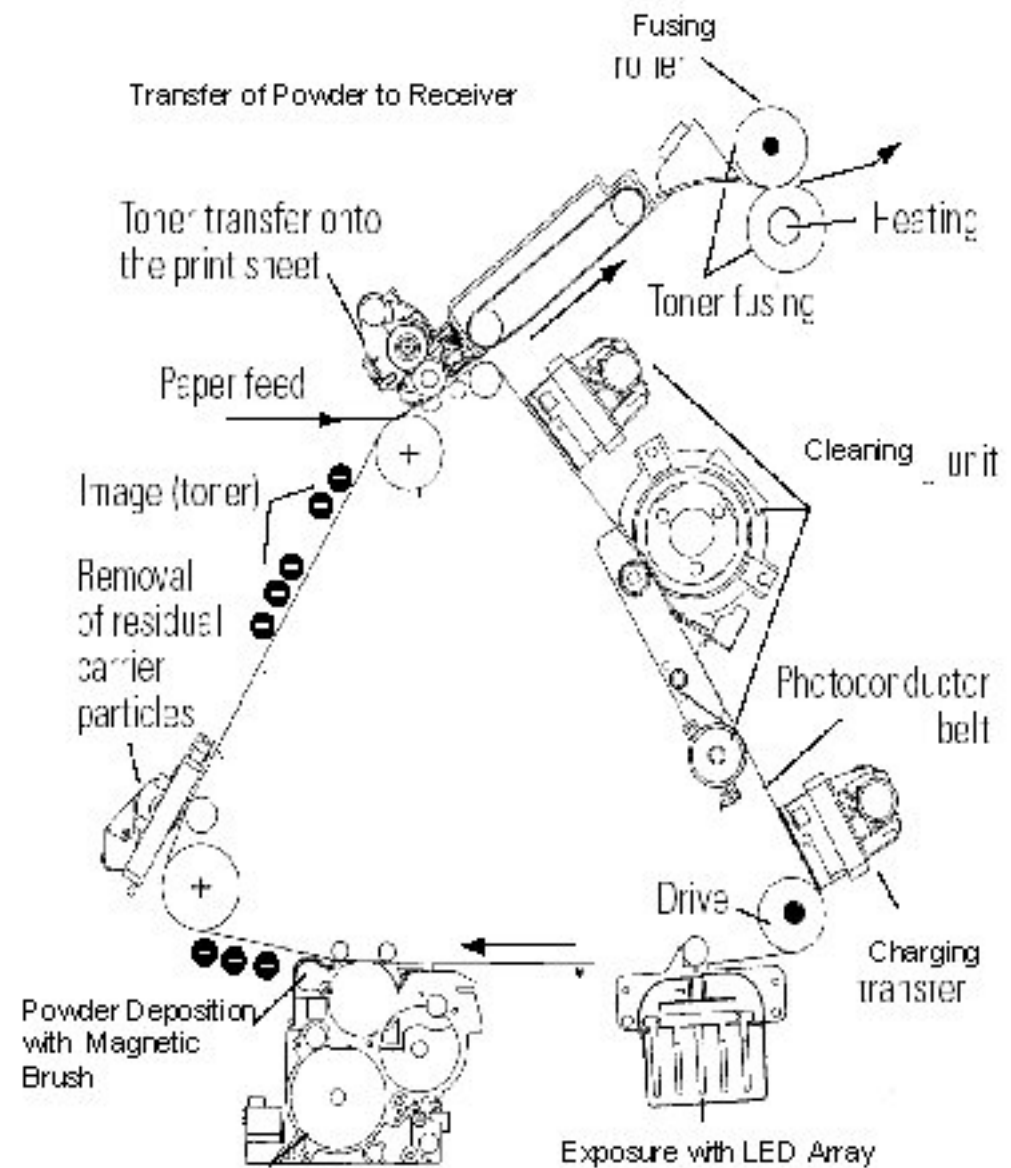
The Toning Process

Digimaster 9110

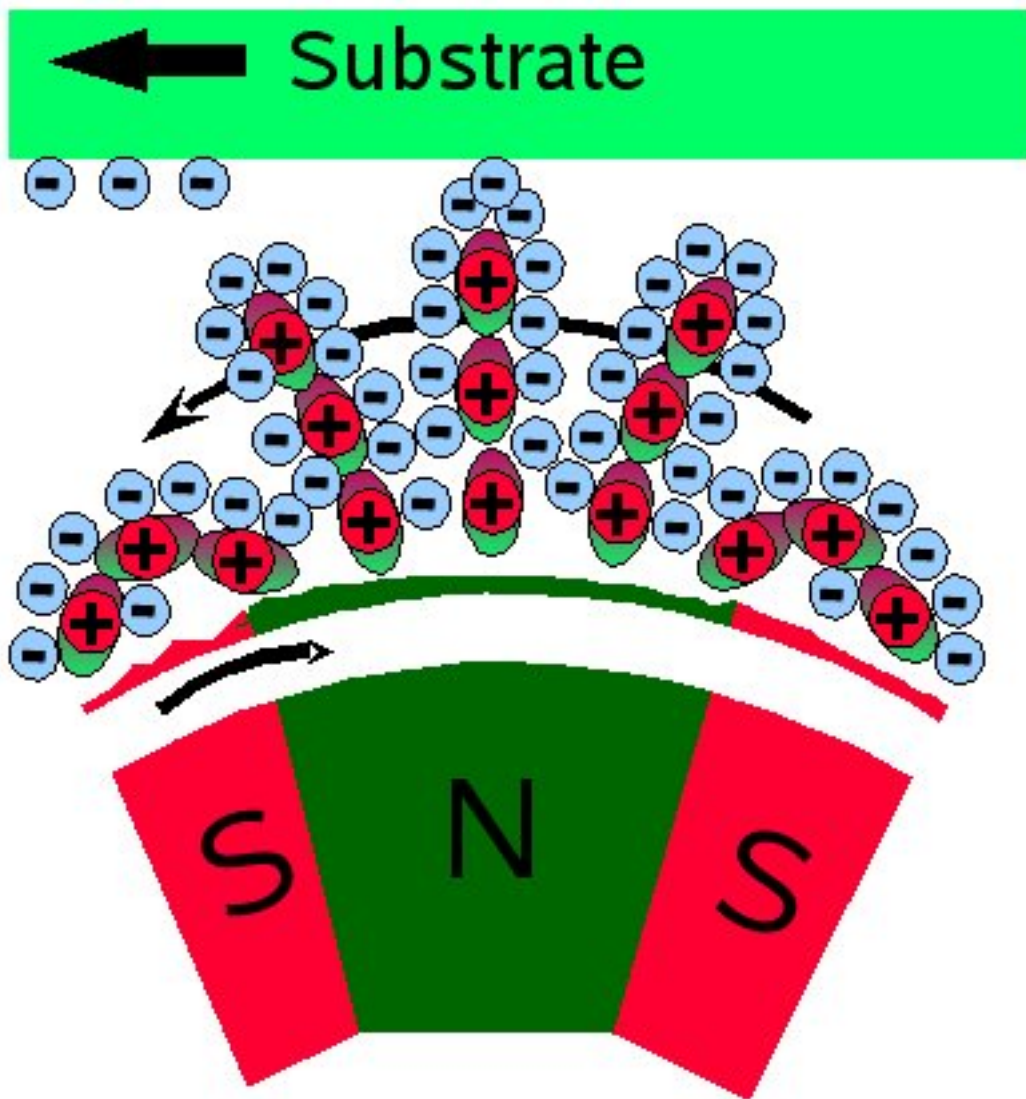


Digital printer for the B/W print on demand market

- Cleaning
- Charging
- Exposure
- **Toning**
- Transfer
- Fusing



Where all this happens



Rotating magnetic brush

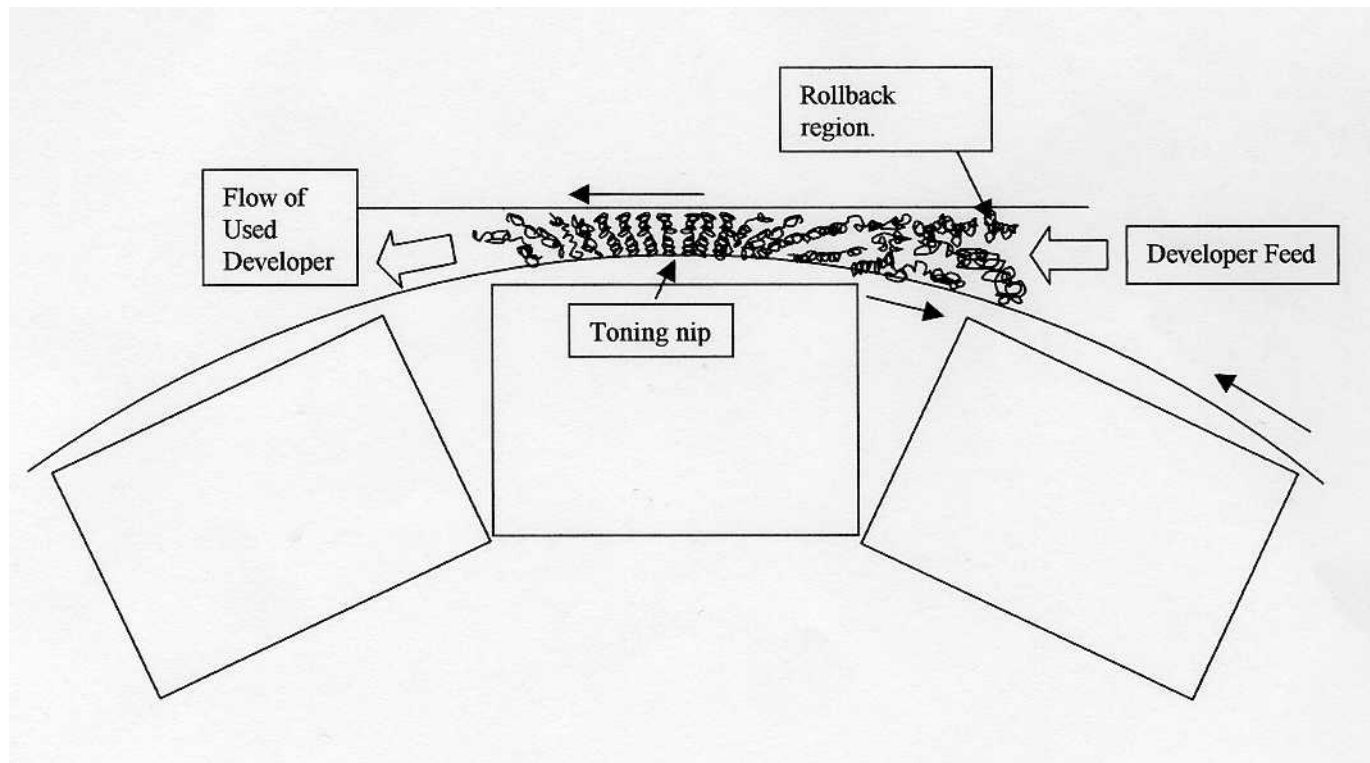
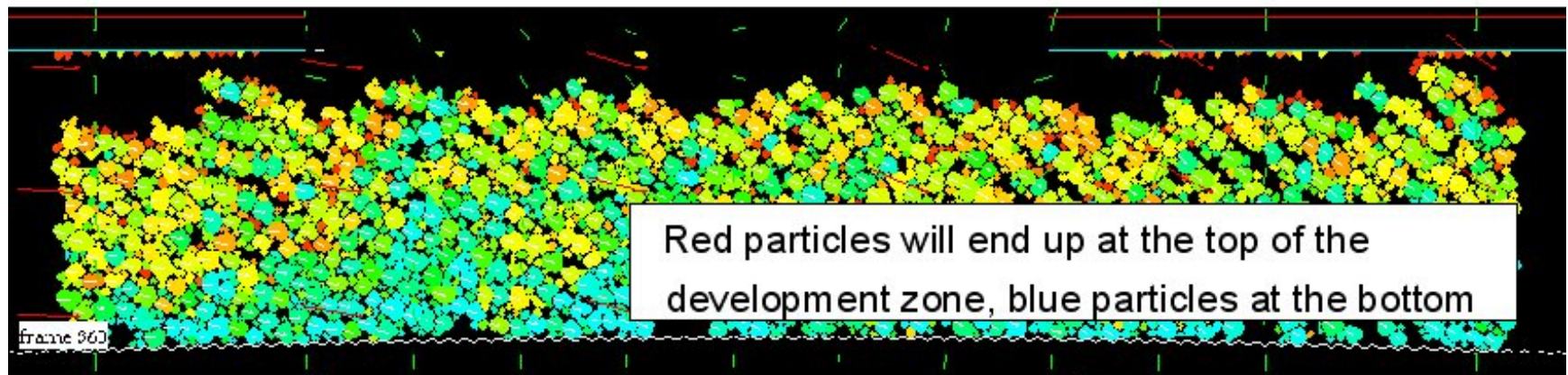
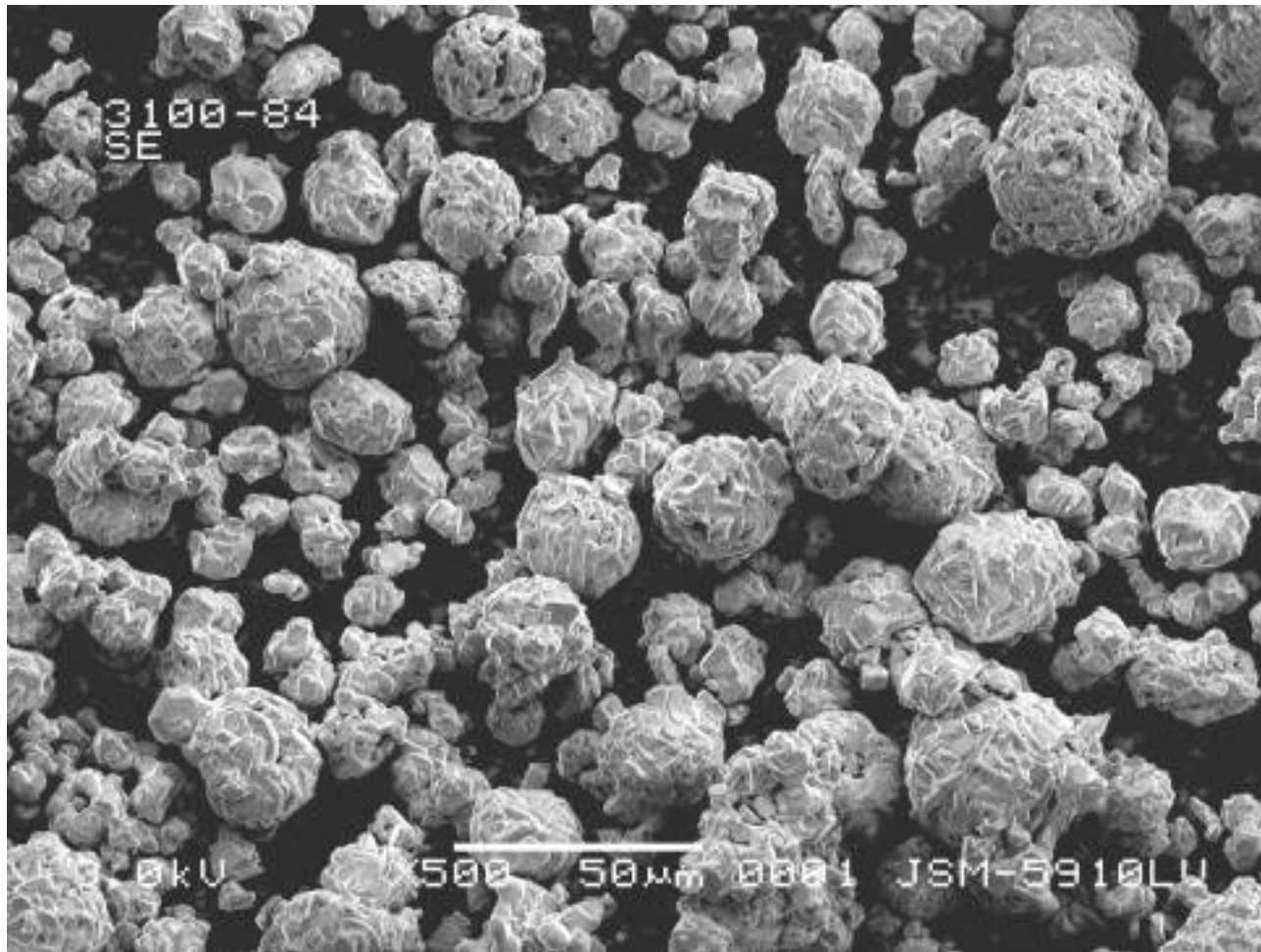


photo conductor belt, toning shell, magnetic roller, developer mass

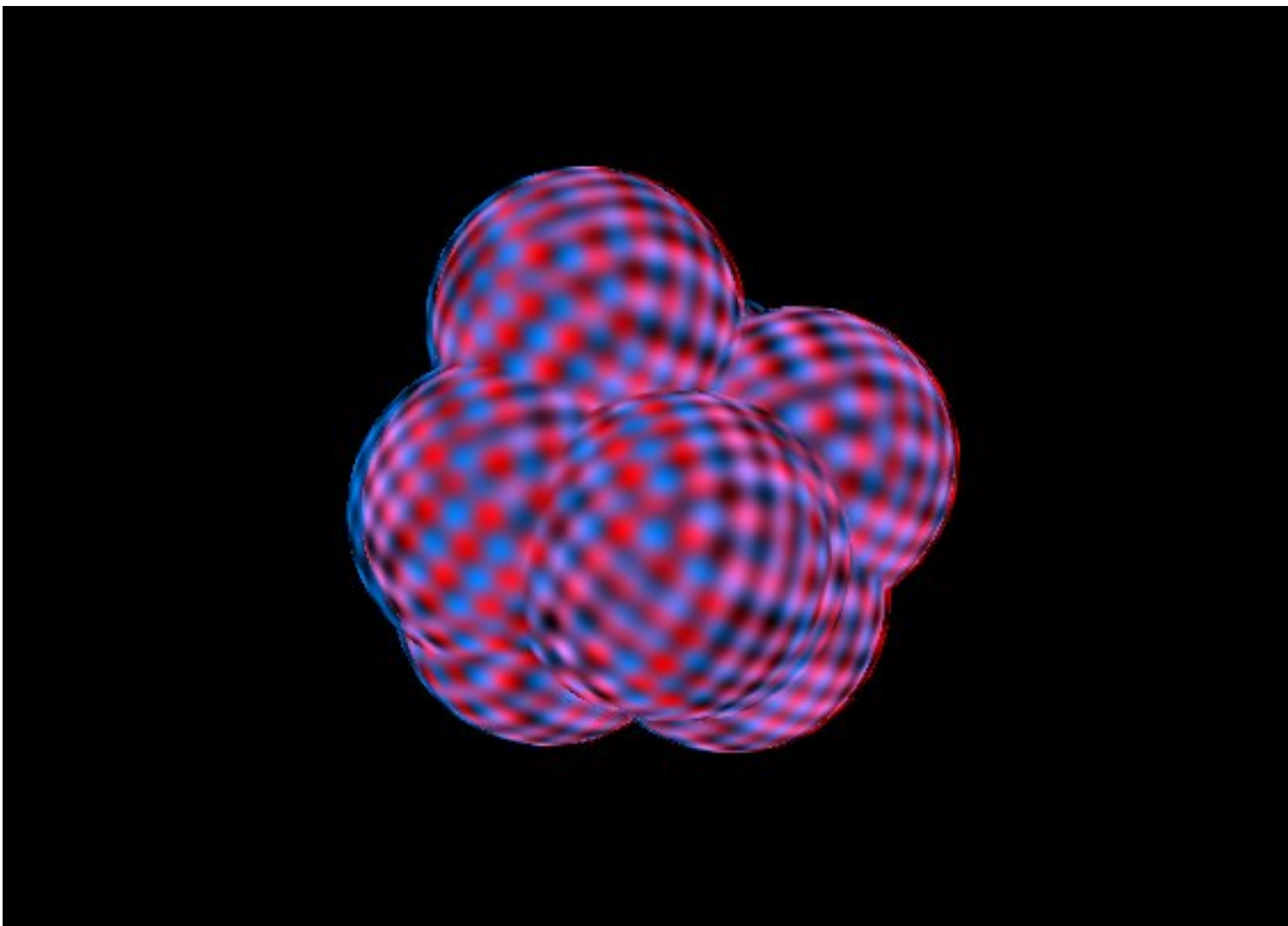


The same in the model



SEM image of typical carrier particles

Modeling the dynamics of irregularly shaped particles



Our particle model

The part of space occupied by a particle is represented as a rigid assembly of strongly overlapping spheres (Idea of Thomas Dera). This gives rise to a list c_1, \dots, c_n of points and a list r_1, \dots, r_n of radii.

We fill this space with matter of given density ρ and Young's modulus E .

By stochastic integration we find the volume V , the mass $m = \rho V$, the center of mass x , the *positively oriented* triplet of inertial axes $\vec{e}_1, \vec{e}_2, \vec{e}_3 := \vec{e}_1 \times \vec{e}_2$ and the corresponding moments of inertia $\theta_1, \theta_2, \theta_3$. Further, we place an electric point charge q , and a magnetic dipole \vec{j} at the point x . Finally we determine an approximate value A for the surface of the particle and the radius r of a minimal bounding sphere around x .

Our particle model as a class **Par**

In a *computational* model (as opposed to a *mathematical* one in general) we need to associate computer representable numbers—the type of which we denote by **R**—with the particle descriptors introduced so far. Such number-based (thus storable) representations of descriptors are *attributes* (or *data members*) in OO parlance. From now on we'll encounter OO parlance in many places. In particular, we'll see 'a particle p according to our particle model' promoted to either 'an instance p of class **Par**' or 'an object of type **Par**'. Beware that a OO-class is neither a mathematical set nor a mathematical class. (Having implemented the mathematical class of all hereditarily finite sets as a C++ class, I know this only too well. A pertaining observation is that all mathematical classes and sets define *equality* (by 'extension') whereas many OO classes work well without equality being defined or definable.)

There are two agreements to be made to define attributes:

(1) Choice of units for the physical quantities: I always use SI units and never found any disadvantage in that. (Nature can do this step in an analogous manner since it provides sufficiently many *natural constants* as units.)

(2) Choice of a *frame* (*system of reference*). A frame consists of a reference point (origin) o and a positively oriented list $\vec{u}_1, \vec{u}_2, \vec{u}_3$ (recall that this implies $\vec{u}_3 = \vec{u}_1 \times \vec{u}_2$) of mutually orthogonal unit vectors (reference directions in space). A frame F and a vector \vec{v} determine the list $\vec{u}_1 \cdot \vec{v}, \vec{u}_2 \cdot \vec{v}, \vec{u}_3 \cdot \vec{v} \in \mathbf{R}^3$ which we designate \mathbf{v} . It is natural to define $F(\vec{v}) := \mathbf{v}$ and to define the application of F to a point p as $F(p) := F(p - o)$ where $p - o$ is the vector for which $o + (p - o) = p$.

We select an arbitrary but fixed frame \tilde{F} , the *master frame*. Of course, there will be only one master frame, but each particle gets its *individual initial*

frame F which has the same reference directions as the master frame and the particle's center of mass x as origin. We use F to transform the vectors $\vec{e}_1, \vec{e}_2, \vec{e}_3, \vec{j}$ and points c_1, \dots, c_n into the objects $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{j}, \mathbf{c}_1, \dots, \mathbf{c}_n$ of type \mathbf{R}^3 . We choose these lists, together with the scalar descriptors ρ, E, \dots as attributes of class **Par**. The values of these attributes specify the *internal state* of a **Par** object. This state says nothing about the position in space or the state of motion.

Par describes rigid bodies and their magnetism and charge are permanent. So the internal state is constant in time for any object of this type.

The placement of a particle in space needs to be variable for the model to be useful. Moving a particle around determines a concurrent process of moving the individual initial frame around as a *body-fixed frame*. Therefore, any placement of a particle in space determines its *current individual*

frame F_t , where t is the time-coordinate of the event, as which a placement can be understood.

Let F_1 and F_2 be frames, then there is a uniquely determined element g of the *Euclidean group* \mathcal{E} which moves F_1 into F_2 . Let us write $F_2 = F_1 \cdot g$ to express this (C++ syntax makes it easier to let g be the second factor in a product than the first one). Then it is natural to denote g by $F_1^{-1} \cdot F_2$. So, our descriptor of the placement in space is $g(t) = \tilde{F}^{-1} \cdot F_t \in \mathcal{E}$, where F_t is the current individual frame.

Interlude: Modeling the Euclidean group as a class **E**

First we define descriptors for an arbitrary element $g \in \mathcal{E}$: We decompose g into a minimal positive rotation around the origin of \tilde{F} and a translation. Thereby we get a unit vector \vec{n} in the direction of the rotation axis, a rotation angle φ , $0 \leq \varphi \leq \pi$, and a translation vector \vec{x} . (φ is determined uniquely, \vec{n} only for $\varphi \neq \pi$, otherwise \vec{n} and $-\vec{n}$ correspond to the same rotation.) As descriptors we use the vectors $\vec{r} := \vec{n} \tan(\varphi/2)$ and \vec{x} , and as attributes the objects $\mathbf{r} := \tilde{F}(\vec{r})$ (*rotation vector*) and $\mathbf{a} := \tilde{F}(\vec{x})$ (*translation vector*) of type \mathbf{R}^3 . So it suggests itself to write an arbitrary instance of our class **E** as **E**(**r**,**x**). In the following we define methods of **E** as if we were free to invent them. Actually, they need to be as we define them in order to make **E** a model of \mathcal{E} . First we define a product of rotation vectors \mathbf{r}_1 and \mathbf{r}_1 in steps:

$$s := 1 - \mathbf{r}_1 \cdot \mathbf{r}_2, \quad S := \begin{array}{l} |s| < \varepsilon \quad ? \quad 1/\varepsilon \quad : \quad 1/s, \end{array}$$

$$\mathbf{r}_1 \circ \mathbf{r}_2 := (\mathbf{r}_1 + \mathbf{r}_2 - \mathbf{r}_1 \times \mathbf{r}_2) \cdot S,$$

where ε is a fixed small number for which I use $\varepsilon = 10^{-12}$. Further, we need the action of rotation vectors on translation vectors, written in postfix format:

$$\mathbf{x} * \mathbf{r} := \frac{\mathbf{x}(1 - \mathbf{r} \cdot \mathbf{r}) + 2(\mathbf{x} \cdot \mathbf{r})\mathbf{r} - 2\mathbf{x} \times \mathbf{r}}{1 + \mathbf{r} \cdot \mathbf{r}}$$

which satisfies the consistency relation

$$\mathbf{x} * (\mathbf{r}_1 \circ \mathbf{r}_2) = (\mathbf{x} * \mathbf{r}_1) * \mathbf{r}_2$$

and the approximation $\mathbf{x} * \mathbf{r} = \mathbf{x} - 2\mathbf{x} \times \mathbf{r} + O(|\mathbf{r}|^2)$. Finally the multiplication law in \mathbf{E} is

$$\mathbf{E}(\mathbf{r}_1, \mathbf{x}_1) \circ \mathbf{E}(\mathbf{r}_2, \mathbf{x}_2) := \mathbf{E}(\mathbf{r}_1 \circ \mathbf{r}_2, \mathbf{x}_1 * \mathbf{r}_2 + \mathbf{x}_2).$$

It implies $\mathbf{E}(\mathbf{r}, \mathbf{x}) = \mathbf{E}(\mathbf{r}, \mathbf{0}) \circ \mathbf{E}(\mathbf{0}, \mathbf{x})$ and that the unit element and the inverse are given by

$$\mathbf{E}(\mathbf{0}, \mathbf{0}), \quad \mathbf{E}(\mathbf{r}, \mathbf{x})^{-1} = \mathbf{E}(-\mathbf{r}, -\mathbf{x} * -\mathbf{r}).$$

Notice that computing the product of any huge number of instances of **E** will give a result that represents an Euclidean transformation with the same precision as each factor in the product. The only effect of rounding errors is that the resulting transformation differs from the exact result by just an Euclidean transformation. This property is extremely important for modeling. It guaranties that the shape of particles remains exactly constant under the evolution algorithm to be described later.

The class **E as a nucleus of Euclidean geometry**

The class **E** allows translating Felix Klein's Erlanger Programm into the OO framework: A class **Y** represents a concept of Euclidean geometry iff it defines a product of its instances y with the instances of **E** such that

$$(y \circ \mathbf{E}(\mathbf{r}_1, \mathbf{x}_1)) \circ \mathbf{E}(\mathbf{r}_2, \mathbf{x}_2) = y \circ (\mathbf{E}(\mathbf{r}_1, \mathbf{x}_1) \circ \mathbf{E}(\mathbf{r}_2, \mathbf{x}_2)) .$$

For the classes **Point**, **Vector**, and **AxialVector** with generic elements **Point**(**y**), **Vector**(**y**), and **AxialVector**(**y**), where **y** is of type \mathbf{R}^3 , the appropriate definitions are

$$\mathbf{Point}(\mathbf{y}) \circ \mathbf{E}(\mathbf{r}, \mathbf{x}) := \mathbf{Point}(\mathbf{y} * \mathbf{r} + \mathbf{x})$$

$$\mathbf{Vector}(\mathbf{y}) \circ \mathbf{E}(\mathbf{r}, \mathbf{x}) := \mathbf{Vector}(\mathbf{y} * \mathbf{r})$$

$$\mathbf{AxialVector}(\mathbf{y}) \circ \mathbf{E}(\mathbf{r}, \mathbf{x}) := \mathbf{AxialVector}(\mathbf{y} * \mathbf{r})$$

The first of these actions enforces the particular group multiplication law of **E** given earlier simply as a definition. In order to capture the distinction between classes **Vector** and **AxialVector** one would have to use an enlarged group that includes spacial inversion. The difference becomes striking in the 2D version of geometry, here axial vectors can be faithfully represented by a single \mathbf{R} , whereas a vector needs—of course—two of them.

In my own implementation of geometry there is a common interface for the 2D and 3D versions, so that the setting of a macro decides whether a program becomes 2D or 3D.

Describing velocity

Since in our case position is described by an element of a Lie group \mathbf{E} , it is canonical to describe velocity by an element of the Lie algebra \mathbf{dE} . This class is defined as follows: The general element is $\mathbf{dE}(\omega, \mathbf{v})$ where the arguments, just as in the case \mathbf{E} , are of type \mathbf{R}^3 . The methods of the class are

$$\mathbf{dE}(\omega_1, \mathbf{v}_1) + \mathbf{dE}(\omega_2, \mathbf{v}_2) := \mathbf{dE}(\omega_1 + \omega_2, \mathbf{v}_1 + \mathbf{v}_2)$$

$$\mathbf{dE}(\omega, \mathbf{v}) \cdot \alpha := \mathbf{dE}(\omega \cdot \alpha, \mathbf{v} \cdot \alpha)$$

for α of type \mathbf{R} . And finally the Lie product *

$$\mathbf{dE}(\omega_1, \mathbf{v}_1) \times \mathbf{dE}(\omega_2, \mathbf{v}_2) := \mathbf{dE}(\omega_1 \times \omega_2, \omega_1 \times \mathbf{v}_2 - \omega_2 \times \mathbf{v}_1)$$

*there were omissions in this and the next three formulas shown in the original talk which are corrected here; also some additions have been made

This operation makes the linear space **dE** an algebra. It is the only operation defined here that is not needed to formulate our time stepping algorithm for rigid bodies. The class **dE** becomes related to Euclidean geometry by

$$\mathbf{dE}(\omega, \mathbf{v}) \circ \mathbf{E}(\mathbf{r}, \mathbf{x}) := \mathbf{dE}(\omega * \mathbf{r}, \mathbf{v} * \mathbf{r} + (\omega * \mathbf{r}) \times \mathbf{x})$$

It would be natural for an actual implementation to make use of the hierarchical nature of nominal type systems: It is not required that attributes are directly numerical—they may also belong to a class which was defined before and the attributes of which can be traced probably through a sequence of classes finally down to numerical nature. (This is what I consider ‘des Pudels Kern’ of the OO paradigm). Then one had to use for ω an object of type **AxialVector** and for \mathbf{v} one of type **Vector**. In the following, let for any $\mathbf{x} \in \mathbf{R}^3$ be $\hat{\mathbf{x}}$ the corresponding vector of unit length.

The exponential function from \mathbf{dE} to \mathbf{E} is given by

$$\exp(\mathbf{dE}(\omega, \mathbf{v}) \cdot \tau) := \mathbf{E}(\hat{\omega} \cdot \tan(\frac{\varphi}{2}), \mathbf{v}_{\parallel} \cdot \tau + \mathbf{v}_{\perp} \cdot \tau \cdot \operatorname{Re}(z) + \hat{\omega} \times \mathbf{v}_{\perp} \cdot \tau \cdot \operatorname{Im}(z))$$

where

$$\varphi := |\omega| \cdot \tau, \quad \mathbf{v}_{\parallel} := \hat{\omega}(\hat{\omega} \cdot \mathbf{v}), \quad \mathbf{v}_{\perp} := \mathbf{v} - \mathbf{v}_{\parallel}, \quad z := \frac{\exp(i\varphi) - 1}{i\varphi}.$$

Here τ is introduced so that the function makes sense also if we work with quantities that carry their dimension with them (although I introduced attributes as merely numerical), in this case τ has to carry the dimension ‘time’, of course. The basic properties are:

$$\exp(\mathbf{dE}(\omega, \mathbf{v}) \cdot \tau) \circ \exp(\mathbf{dE}(\omega, \mathbf{v}) \cdot \tau') = \exp(\mathbf{dE}(\omega, \mathbf{v}) \cdot (\tau + \tau')),$$

$$\exp(\mathbf{dE}(\omega, \mathbf{v}) \cdot \tau) = \mathbf{E}(\omega \cdot \frac{\tau}{2}, \mathbf{v} \cdot \tau) + O(\tau^2).$$

The corresponding ‘inverse’ function \log from \mathbf{E} to \mathbf{dE} is given by

$$\log(\mathbf{E}(\mathbf{r}, \mathbf{x})) := \mathbf{E}(\hat{\mathbf{r}} \cdot \boldsymbol{\varphi}, \mathbf{x}_{\parallel} + \mathbf{x}_{\perp} \cdot \operatorname{Re}(z) + \hat{\mathbf{r}} \times \mathbf{x}_{\perp} \cdot \operatorname{Im}(z))$$

where

$$\boldsymbol{\varphi} := 2 \arctan(|\mathbf{r}|), \quad \mathbf{x}_{\parallel} := \hat{\mathbf{r}} (\hat{\mathbf{r}} \cdot \mathbf{x}), \quad \mathbf{x}_{\perp} := \mathbf{x} - \mathbf{x}_{\parallel}, \quad z := \frac{i\boldsymbol{\varphi}}{\exp(i\boldsymbol{\varphi}) - 1}$$

or approximately $\log(\mathbf{E}(\mathbf{r}, \mathbf{x})) = \mathbf{dE}(\mathbf{r} \cdot 2, \mathbf{x}) + O(|\mathbf{r}|^2)$. The basic property is

$$\exp(\log(\mathbf{E}(\mathbf{r}, \mathbf{x}))) = \mathbf{E}(\mathbf{r}, \mathbf{x}).$$

whereas the converse property $\log(\exp(\mathbf{dE}(\boldsymbol{\omega}, \mathbf{v}) \cdot \tau)) = \mathbf{dE}(\boldsymbol{\omega}, \mathbf{v}) \cdot \tau$ holds only for τ not too large.

Es we have seen, the motion of a particle determines a curve $t \mapsto g(t) \in \mathcal{E}$ which in our computational model gives rise to a function object $\mathbf{E}(\mathbf{r}(t), \mathbf{x}(t))$.

The concept of differentiating a \mathbf{R}^n -valued curve by determining the best approximating line has a natural generalization to Lie-Group-valued curves:

$$g'(t) := \lim_{h \rightarrow 0} \log(g(t-h)^{-1} g(t+h)) \cdot \frac{1}{2h}$$

which is a precise version of the heuristic formula

$$g(t+h) = g(t) \exp(g'(t) \cdot h) + O(h^2).$$

concerning the best approximation of g by a one-parameter subgroup. Making use of the first-order approximations of the operations $*$ and \log one easily finds

$$\mathbf{E}(\mathbf{r}(t), \mathbf{x}(t))' = \mathbf{dE}(\boldsymbol{\omega}(t), \mathbf{v}(t) + \mathbf{x}(t) \times \boldsymbol{\omega}(t))$$

where

$$\boldsymbol{\omega}(t) := \mathbf{r}(t)' := \lim_{h \rightarrow 0} \log(\mathbf{r}(t-h)^{-1} \circ \mathbf{r}(t+h)) \cdot \frac{1}{2h},$$

is the *angular velocity* and

$$\mathbf{v}(t) := \mathbf{x}(t)' := \lim_{h \rightarrow 0} (-\mathbf{x}(t-h) + \mathbf{x}(t+h)) \cdot \frac{1}{2h}.$$

is the velocity proper or the *translational velocity*. In all cases the derivation is defined by the same method since a real vector spaces (considered as an abelian group) coincides with its Lie algebra and has the identity function as \log (and \exp). I would have been less surprised if the derivative $\mathbf{E}(\mathbf{r}(t), \mathbf{x}(t))'$ would have turned out to equal $\mathbf{dE}(\omega(t), \mathbf{v}(t))$. But closer inspection shows, that $\mathbf{E}(\mathbf{r}(t), \mathbf{x}(t))'$ is a velocity associated with the transformation of the whole space and is not aware of the position $\mathbf{x}(t)$ (actually the position vector with respect to the master frame) of the particle. So, in order to get the translational velocity at any position \mathbf{y} one has to take $\mathbf{E}(\mathbf{r}(t), \mathbf{x}(t))' \circ \mathbf{y}$ which turns out to be $\mathbf{v}(t) + \omega \times (\mathbf{y} - \mathbf{x}(t))$ and thus $\mathbf{v}(t)$ for $\mathbf{y} = \mathbf{x}(t)$.

if the position after a short time τ is given by

$$\mathbf{E}(\mathbf{r}, \mathbf{0}) \circ \exp(\mathbf{dE}(\omega, \mathbf{v}) \cdot \tau) \circ \mathbf{E}(\mathbf{0}, \mathbf{x})$$

to first order in τ . Here, placing the factor in the middle lets ω act as an angular velocity around the center of mass rather than around the origin. The function \log allows giving a more direct definition:

$$\mathbf{dE}(\omega(t), \mathbf{v}(t)) = \lim_{\tau \rightarrow 0} \frac{\log \left(\mathbf{E}(\mathbf{r}(t), \mathbf{0})^{-1} \circ \mathbf{E}(\mathbf{r}(t + \tau), \mathbf{x}(t + \tau)) \circ \mathbf{E}(\mathbf{0}, \mathbf{x}(t))^{-1} \right)}{\tau}.$$

It may surprise that the seemingly more natural expression

$$\lim_{\tau \rightarrow 0} \frac{\log \left(\mathbf{E}(\mathbf{r}(t + \tau), \mathbf{x}(t + \tau)) \circ \mathbf{E}(\mathbf{r}(t), \mathbf{x}(t))^{-1} \right)}{\tau}$$

does not yield $\mathbf{dE}(\omega(t), \mathbf{v}(t))$ but $\mathbf{dE}(\omega(t), \mathbf{v}(t) + \mathbf{x}(t) \times \omega(t))$. There are two other important physical quantities for which we need \mathbf{dE} as the data type which provides fitting geometrical behavior: (1) pairs of *torque* and *force*, (2) pairs of *angular momentum* and *linear momentum*.

Describing inertia

The last class we need is the class **I** which describes the inertia in reacting to forces and torques. We write the general instance as $\mathbf{I}(I_1, I_2, I_3, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mu)$ where μ and the I_i are of type \mathbf{R} and the \mathbf{a}_i of type \mathbf{R}^3 . We define the method of inversion

$$\mathbf{I}(I_1, I_2, I_3, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mu)^{-1} := \mathbf{I}(1/I_1, 1/I_2, 1/I_3, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, 1/\mu)$$

and the action of **E** by

$$\mathbf{I}(I_1, I_2, I_3, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mu) \circ \mathbf{E}(\mathbf{r}, \mathbf{x}) := \mathbf{I}(I_1, I_2, I_3, \mathbf{a}_1 * \mathbf{r}, \mathbf{a}_2 * \mathbf{r}, \mathbf{a}_3 * \mathbf{r}, \mu)$$

and the rule for forming momentum from velocity

$$\begin{aligned} &\mathbf{I}(I_1, I_2, I_3, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mu) * \mathbf{dE}(\omega, \mathbf{v}) := \\ &\mathbf{dE}(\mathbf{a}_1 \cdot I_1 \cdot (\omega \cdot \mathbf{a}_1) + \mathbf{a}_2 \cdot I_2 \cdot (\omega \cdot \mathbf{a}_2) + \mathbf{a}_3 \cdot I_3 \cdot (\omega \cdot \mathbf{a}_3), \mathbf{v} \cdot \mu) \end{aligned}$$

Finalizing class **Par**

We denote the general instance of a **Par** as $\mathbf{Par}(\sigma, g, h)$, where σ is a collective name for all attributes describing the internal state and the position g is of type **E** and the velocity h is of type **dE**. The action of **E** is then

$$\mathbf{Par}(\sigma, g, h) \circ \mathbf{E}(\mathbf{r}, \mathbf{x}) = \mathbf{Par}(\sigma, g \circ \mathbf{E}(\mathbf{r}, \mathbf{x}), h \circ \mathbf{E}(\mathbf{r}, \mathbf{x}))$$

The point to notice here is that σ is not only constant in time but also invariant with respect to re-placement in space, just what action of **E** is to mean here.

Time stepping algorithm for **Par**

First we give an algorithm for the free evolution step (no force and no torque acts on the particle during the time Δt of the step). The values of the attributes at the end of the step will be stored in the same variables which carried the values at the beginning of the step. We thus describe the evolution step as a *mutating algorithm*.

freeStep(Δt): From σ get $I_0 := \mathbf{I}(\theta_1, \theta_2, \theta_3, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, m)$ and from $g = \mathbf{E}(\mathbf{r}, \mathbf{x})$ get \mathbf{r} and \mathbf{x} . Then $\tau := \Delta t/2$, $g_c := \exp(h \cdot \tau)$, $I := I_0 \circ g$, $I_c := I \circ g_c$, $\Delta h := I^{-1} * (I * h - I_c * h)$, $h_c := h + \Delta h$, $g = \mathbf{E}(\mathbf{r}, \mathbf{0}) \circ \exp(h_c \cdot \Delta t) \circ \mathbf{E}(\mathbf{0}, \mathbf{x})$, $h = h_c + \Delta h$.

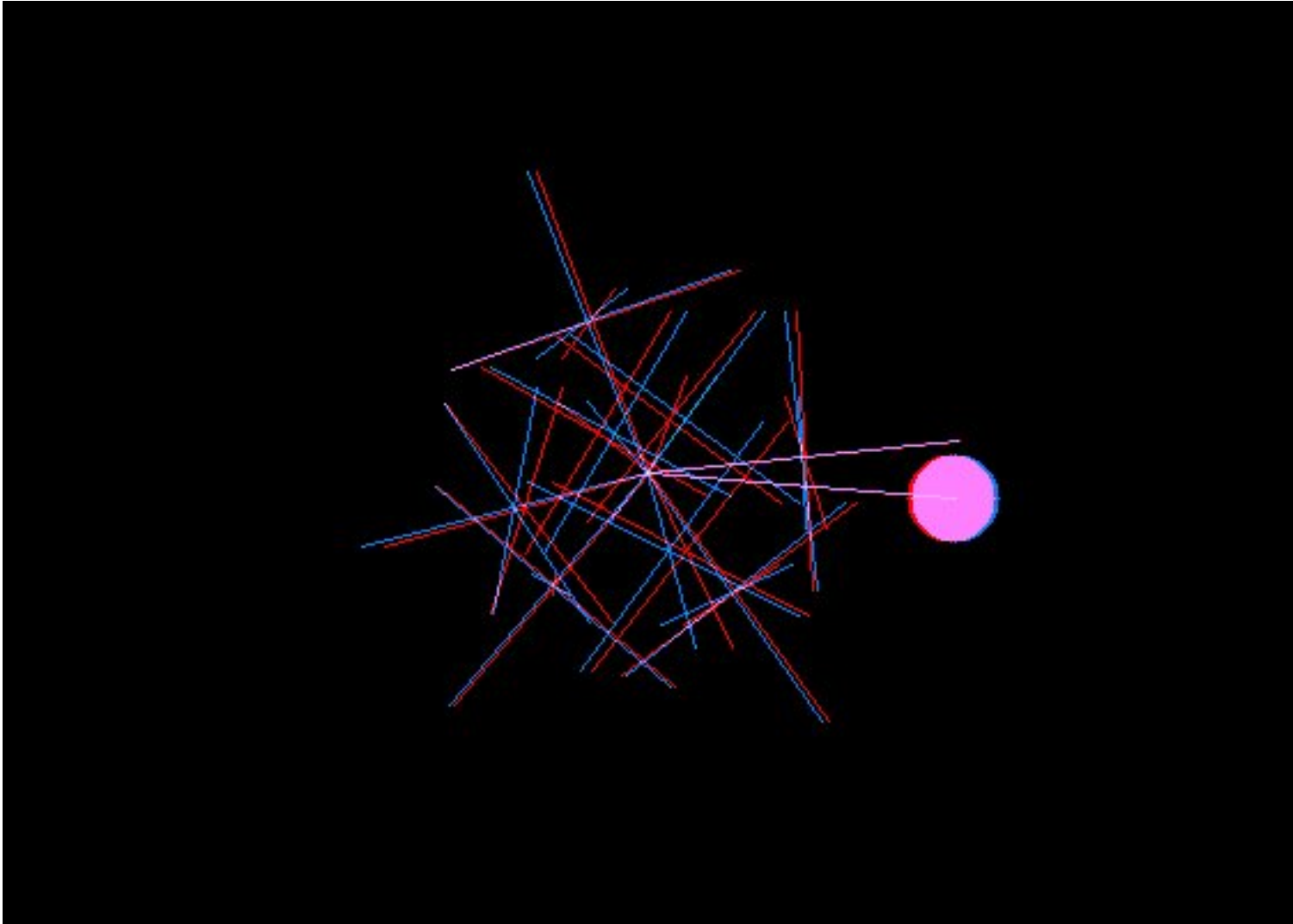
The next function describes how the velocity of a particle changes if during the time span Δt it is acted upon by the torque \vec{T} relative to the center of mass and the force \vec{F} on the center of mass. As a descriptor of this situation

we use the object $\mathbf{dE}(\mathbf{T} \cdot \Delta t, \mathbf{F} \cdot \Delta t)$ which is an increment in momentum (which here means a pair of angular momentum and linear momentum). So that we have the proper argument for the following function, for which we use explicit typing of the argument

$\text{react}(\mathbf{dE} \tilde{h}) : \Delta h := I^{-1} \tilde{h}, h = h + \Delta h$

The full time step is formed from these two building blocks in a manner that not only superficially resembles a Feynman graph of first order

$\text{step}(\mathbf{dE} \tilde{h}) : \text{freeStep}(\Delta t / 2), \text{react}(\tilde{h}), \text{freeStep}(\Delta t / 2)$

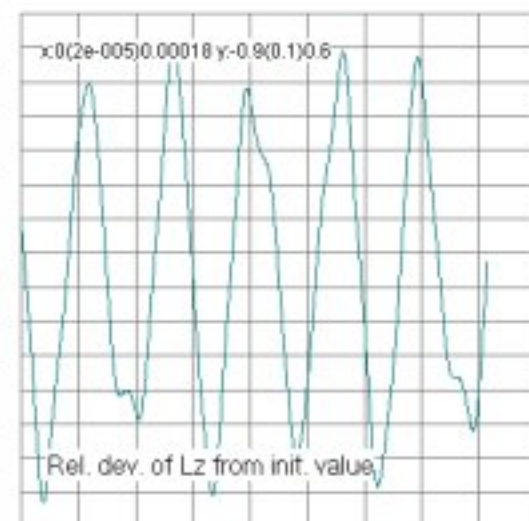
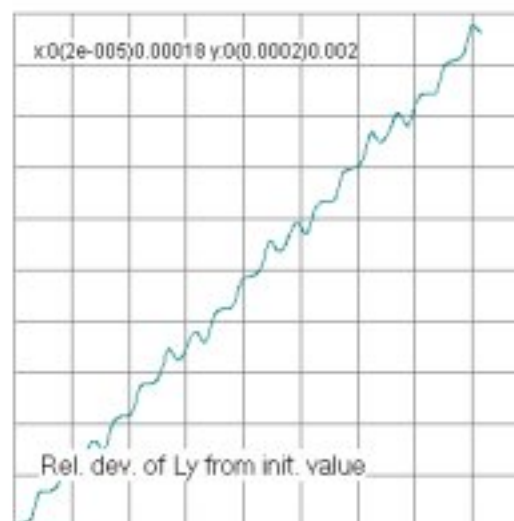
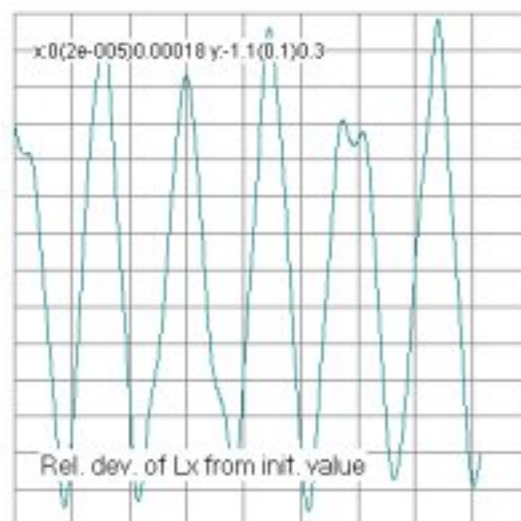
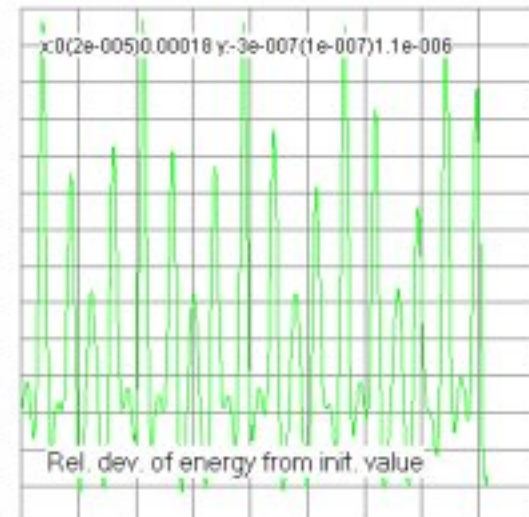
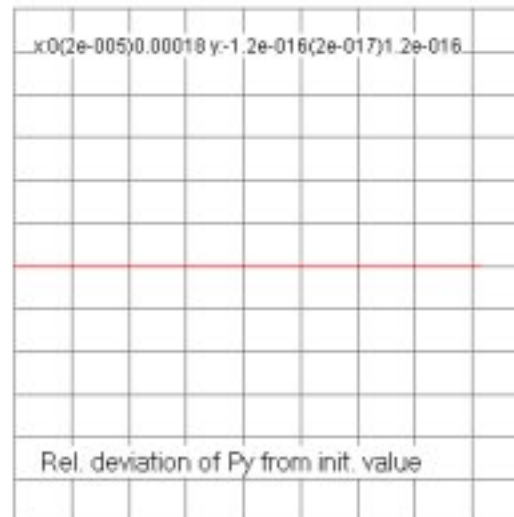
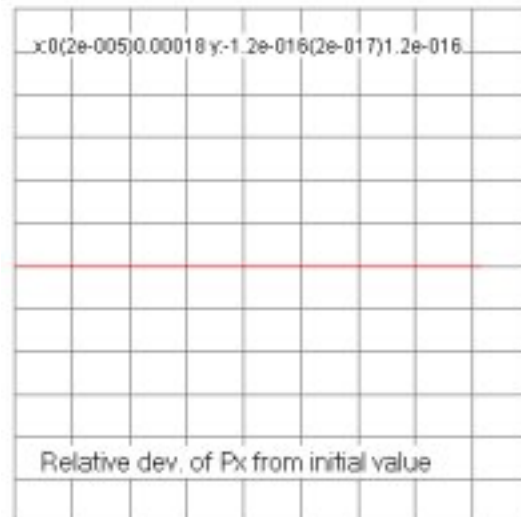


Behavior of linear momentum, energy, and angular momentum of a magnetized particle in a homogeneous magnetic field.

Linear momentum, energy and L_y are conserved quantities in this situation. In the model the energy fluctuates with unnoticeable trend and with an amplitude that is proportional to dt^2 .

L_y fluctuates with an obvious constant trend and an amplitude that is proportional to dt .

Linear momentum is conserved also in the model.



Fast Rigid Bodies Integrator (*FBI*) for the many-body system

1. Input: A list p_1, \dots, p_d of objects of type **Par**, time t , and time step Δt .
2. First step: For each particle p of the list do: $\text{freeStep}(\Delta t/2)$.
3. Interaction step: Let $\mathbf{Par}(\sigma, g, h)$ be the generic element of the list. Define objects \mathbf{r} and \mathbf{x} of type \mathbf{R}^3 such that $g = \mathbf{E}(\mathbf{r}, \mathbf{x})$. From σ get the data $\mathbf{c}_1, \dots, \mathbf{c}_n$ and r_1, \dots, r_n and \mathbf{j} . From these compute the list $\mathbf{x} + \mathbf{c}_1 * \mathbf{r}, \dots, \mathbf{x} + \mathbf{c}_n * \mathbf{r}$ of centers of spheres with radii r_1, \dots, r_n , and the magnetic moment $\mathbf{j} * \mathbf{r}$ placed at \mathbf{x} , together with the electric point charge q . To define interaction between particles we single out a second generic particle \tilde{p} . Particle p feels a force (on its center of mass)

and a torque (around its center of mass) due to the presence of \tilde{p} . This force and torque arises as a sum of electric, magnetic, and elastic contributions. For the first two, physics gives definitive prescriptions; for the elastic part we use Heinrich Hertz's formulas for elastic spheres in forced contact. An important point is that the presence of further particles simply adds their forces and torques to the contribution from \tilde{p} . Thus we know how to find for p a total force and total torque which p feels due to the presence of all other particles. If there are external electric and magnetic fields, these add to the force and torque on p in a straightforward manner. If the external fields depend on time, their values at time $t + \Delta t/2$ have to be taken for this computation. The final values \mathbf{T} and \mathbf{F} of torque and force on particle p determine $\tilde{h}_p := \mathbf{dE}(\mathbf{T} \cdot \Delta t, \mathbf{F} \cdot \Delta t)$. So the following prescription is well defined: For each particle p of the list do: $\text{react}(\tilde{h}_p)$.

4. Last step: For each particle p of the list do: $\text{freeStep}(\Delta t/2)$.

This algorithm is modified from the *explicit midpoint method* for ordinary differential equations. It shares the salient feature of this method that forces and torques have to be computed only once in a time step, and that the time step may change from step to step.

In my toning application there are, in addition, adhesive forces, forces between induced electrical dipoles, forces generated by electrostatic mirror charges from conducting enclosing walls, elastic and adhesive forces from enclosing walls, friction forces between particles, particles and walls, and particles to air. Lorentz forces are ignored so far.

A detour on graphical representation

Bodies as graphical objects are described here by *metrical indicator functions* which are defined everywhere in space, take negative values inside the body, positive values outside the body, and an approximate value for the distance from the surface for points near to the surface.

The metrical indicator function of a sphere is simple:

$$f_{Sphere}(p) = |p - center| - radius$$

Metical indicator functions of complex bodies from simple ones are built from *

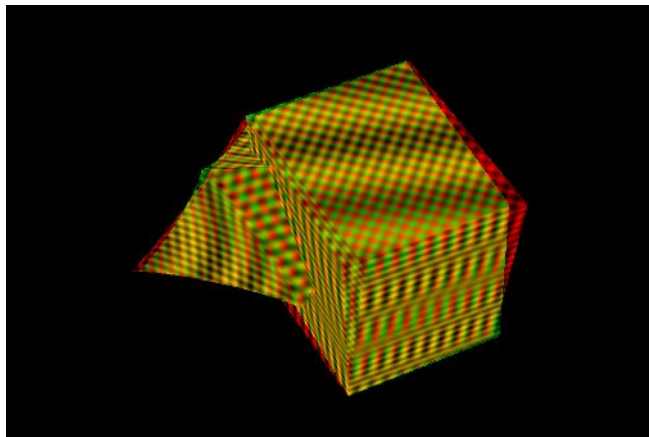
$$f_{Body_1 \cap Body_2} = Max(f_{Body_1}, f_{Body_2})$$

$$f_{Body_1 \cup Body_2} = Min(f_{Body_1}, f_{Body_2})$$

It is not difficult to develop efficient algorithms for finding the point where a given ray hits the surface of a body first.

*Here, Max and Min were erroneously interchanged until 2017-05-18

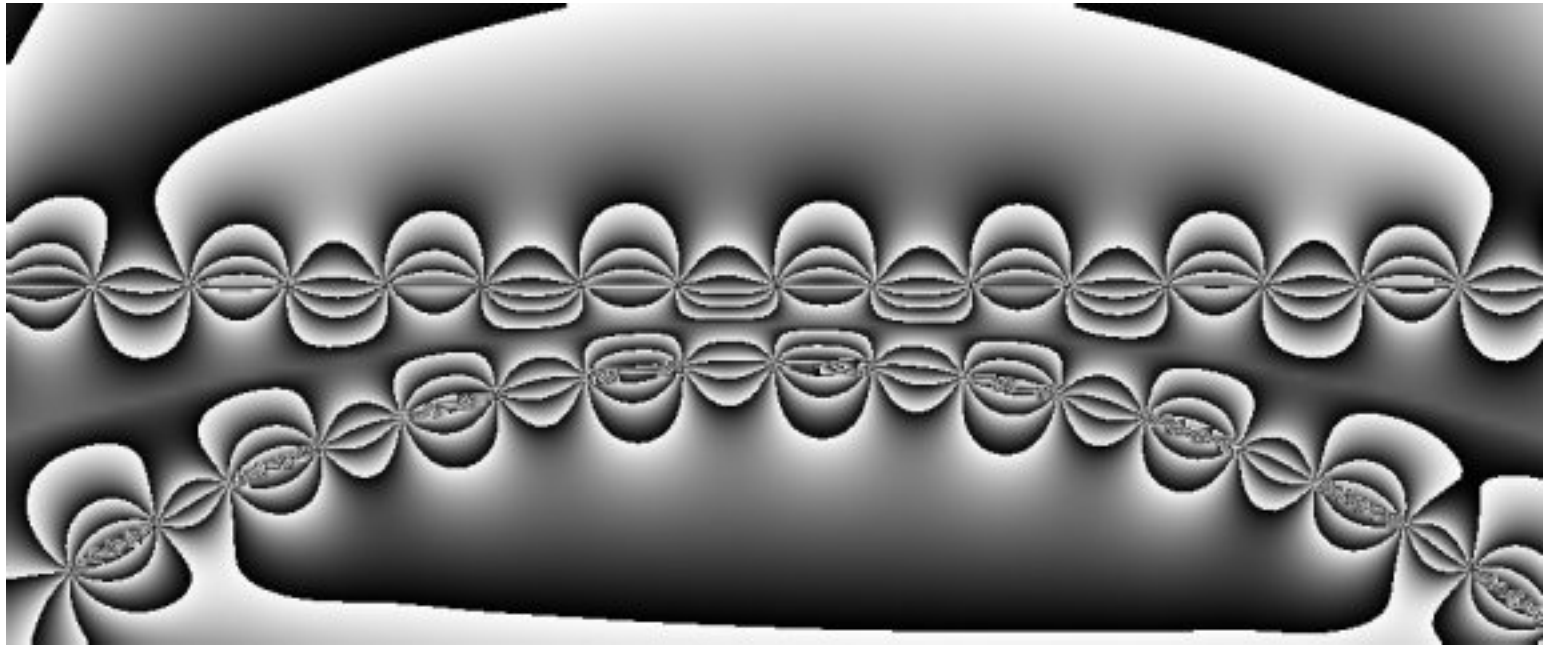
The set union of two non-spherical bodies



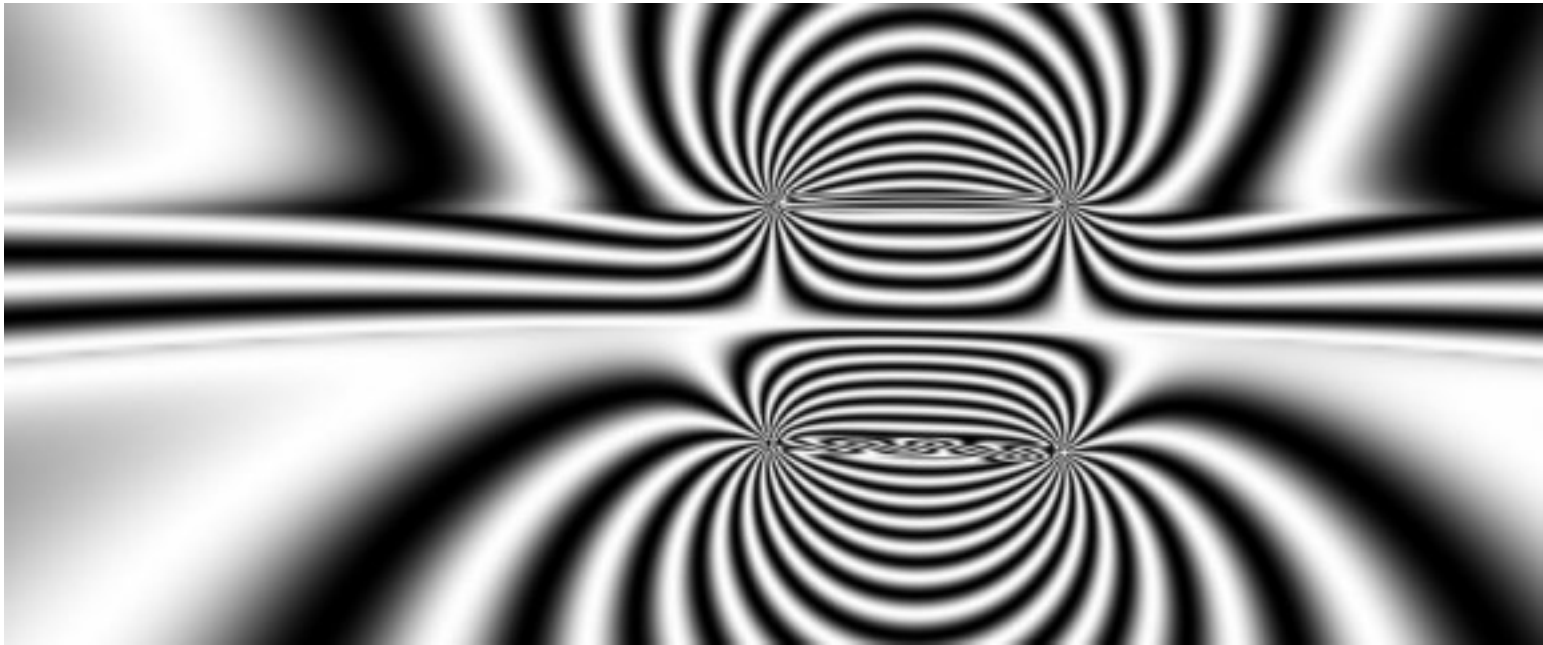
An acceptable ray tracing algorithm has to cope with sharp edges. Patterns (that are needed for the anaglyph method to work) are here not defined only on the surface but in space. Here the periodic reflectance pattern forms a cubic lattice in space.

Some aspects of the toning model

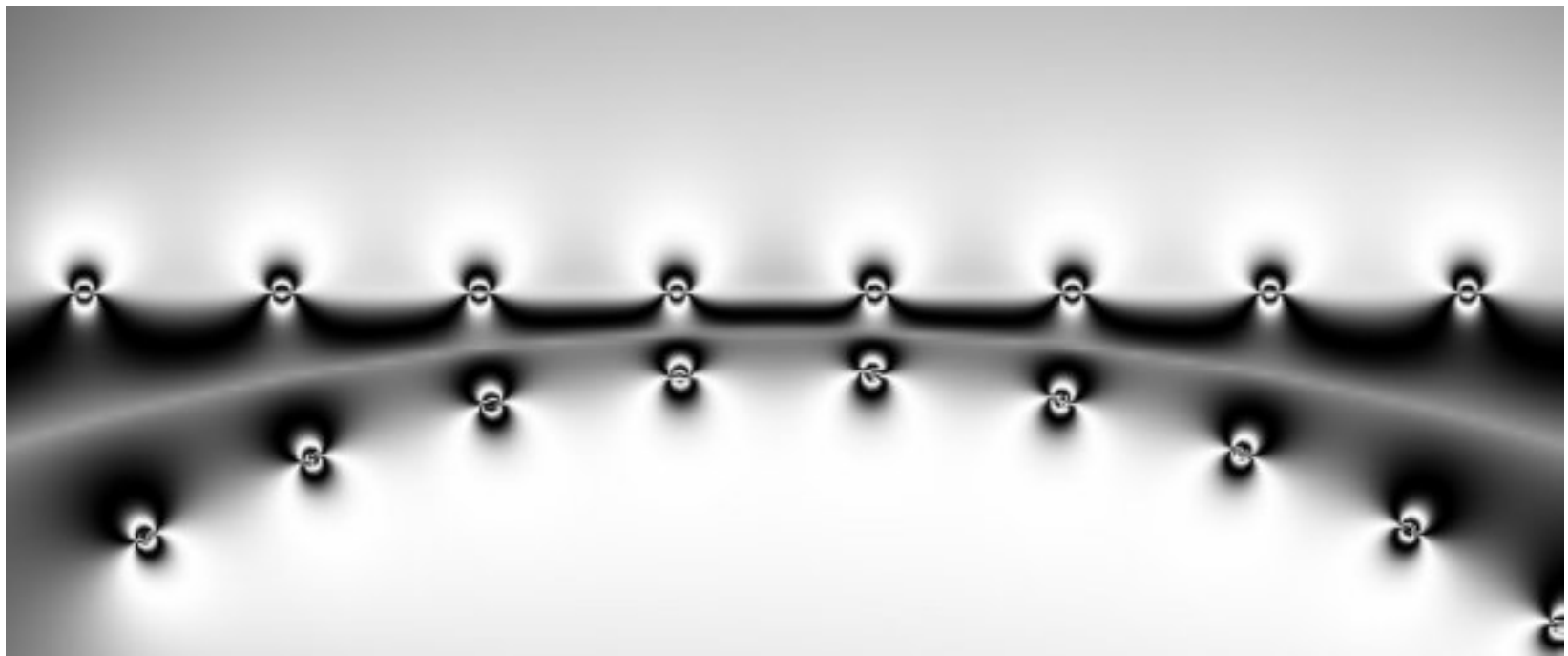
Holding the electrical potential constant on the toning shell



Holding the electrical potential constant on the toning shell



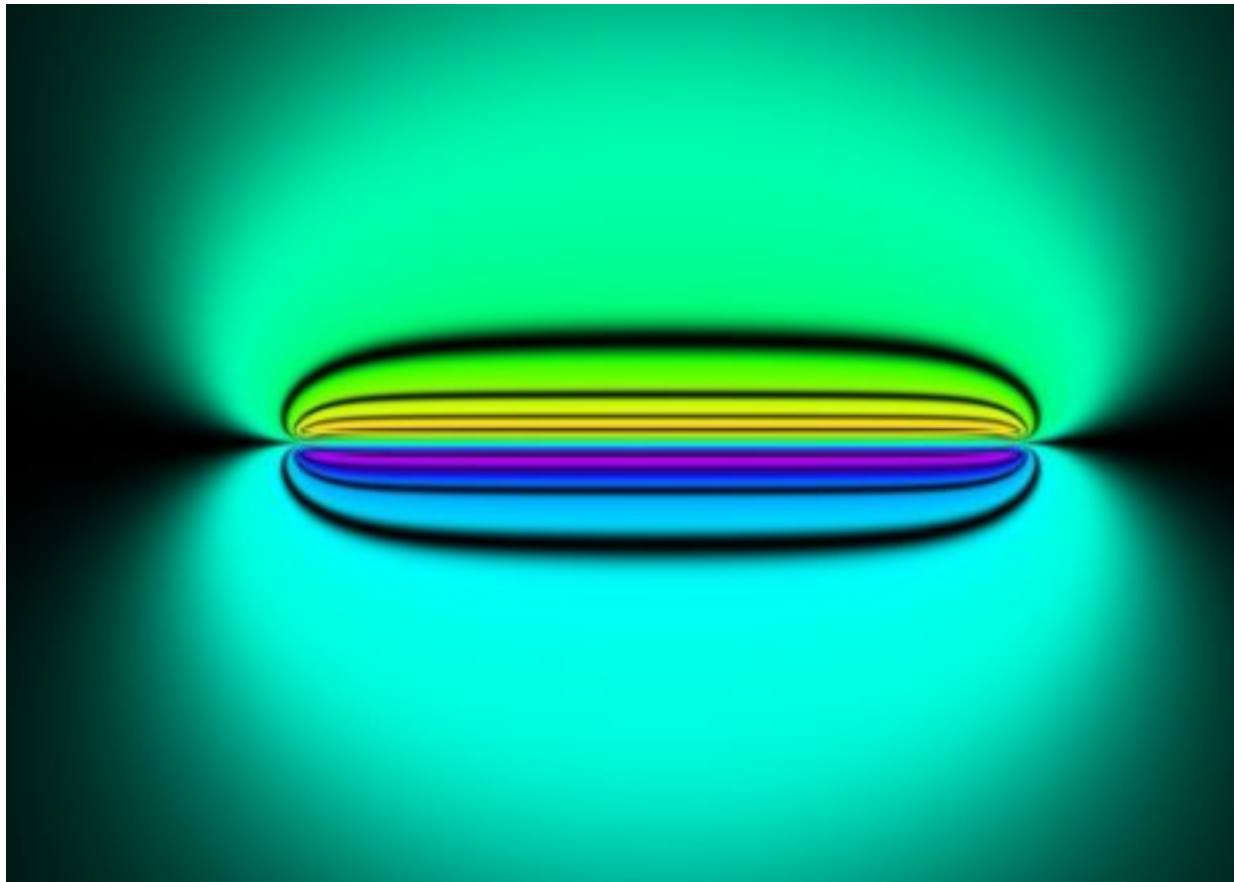
Holding the electrical potential constant on the toning shell



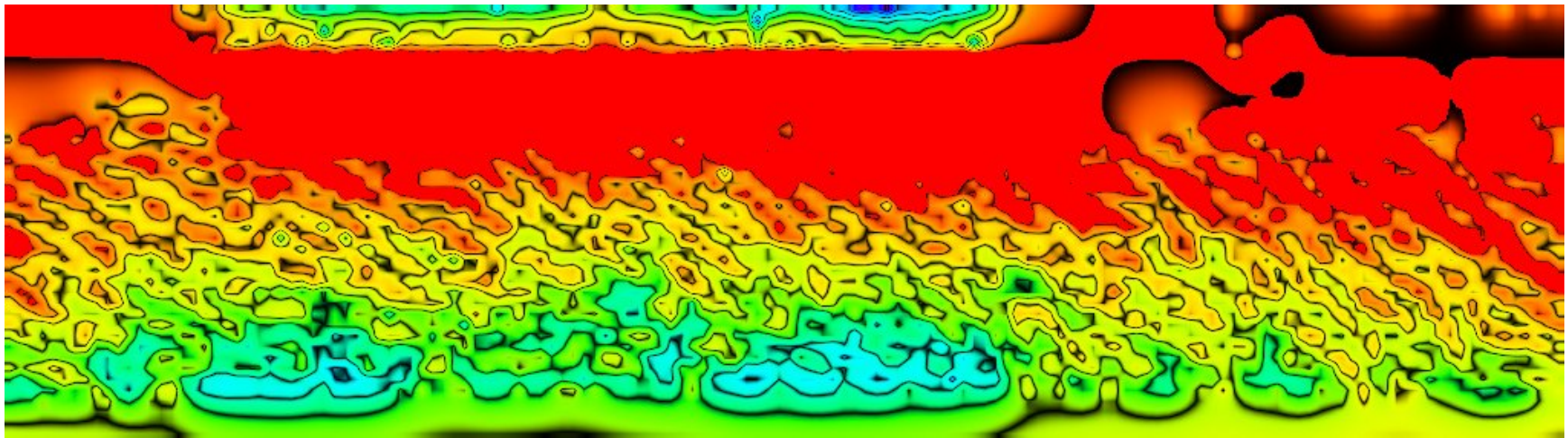
The potential of a rectangular charge patch spoiled by a programing error




Now correct



Potential created by the particles alone



Going parallel



Ulrich Mutze

has successfully completed the
Cornell Theory Center Virtual Workshop
**Parallel Programming with
Message-Passing Interface**

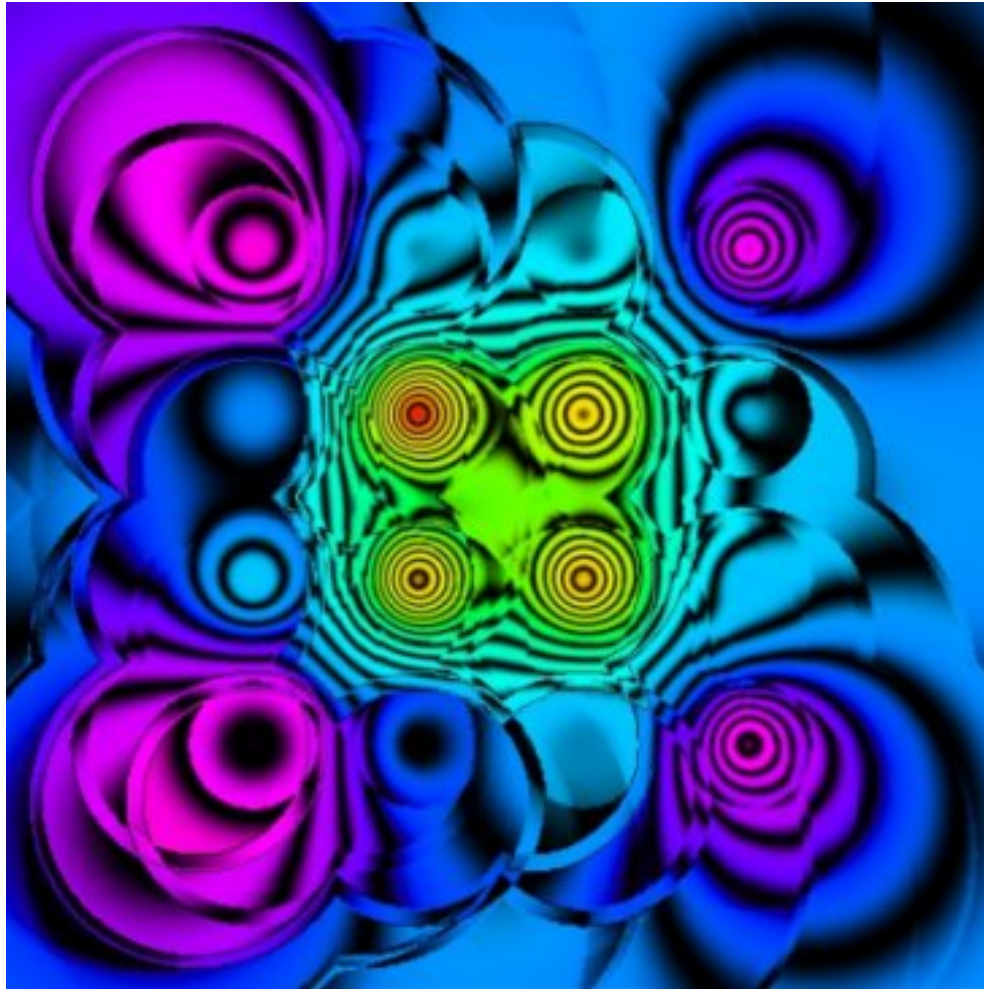
February 26 - March 30, 2001

Background Graphic: Allele Distribution on Patchy Environmental Gradients, computed on Velocity

Partitioning of the nip space



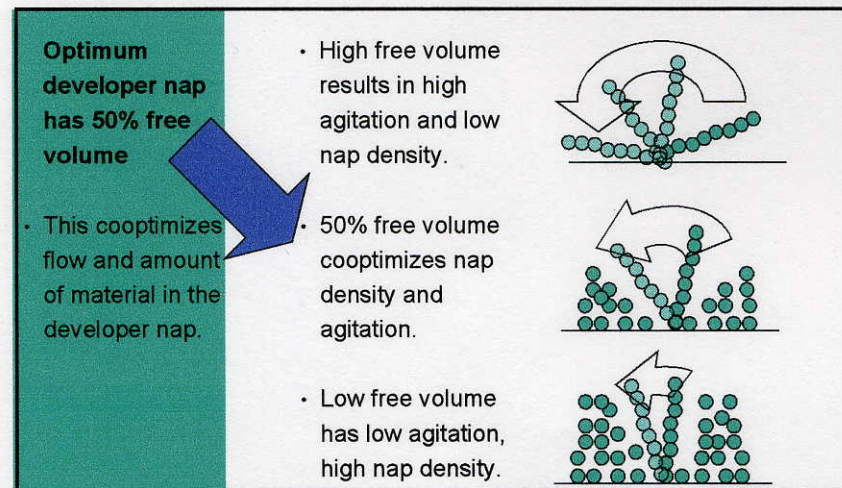
Defining a coarse grained mean field by a wide lattice of sources



Results

The basic insight after having passed the patenting process

Figure 3. Free volume of approximately 50% in the toning nip is optimum.



On Simplicity

... the tentative postulation of the simplest formal solution of a problem is a conventional and frequently successful mode of procedure in theoretical physics.

From: Herbert B. Callen: Thermodynamics, Wiley 1960, p. 24

When consistency is a must, simplicity is the result of hard work

Anonymus

Last modification: 2017-05-18