CSC 7700: Scientific Computing

Module D: Scientific Visualization

Lecture 3: Mathematical Concepts II

Dr. Werner Benger

Lecture Overview (2 lectures)

- Mathematical Terminology:
  - "Fiber Bundle": Formal definition, the base and fiber space
  - Describing the Base Space via Topology: Adjacency, Neighborhood, Connectivity, Meshes
  - Describing the Fiber Space via Differential Geometry: Differentiable Manifolds, Coordinate-free formulations, vector calculus
  - Simple fiber bundles
  - Examples of Fiber Bundle visualizations
  - Visualizing tensor fields
- Geometric Algebra
  - n-dimensional coordinate-free formulations,
  - Bi-vectors and rotations
- The Fiber Bundle Data Model
  - Implementation of the Mathematical Principles
  - The F5 File Format
  - Identifying properties of data types

Initial Approach

The proper abstractions for scientific data are known. We just have to use them.

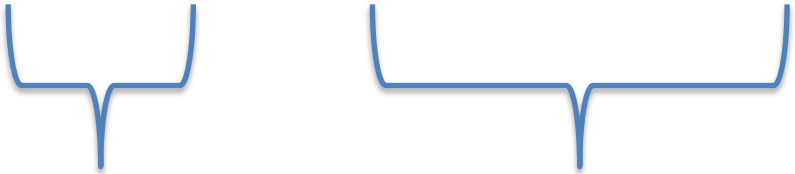
A visualization model based on the mathematics of fiber bundles, David M. Butler and M. H. Pendley, Computers in Physics, Sep/Oct 1989, p 45-51, 3(5)

What is a fiber bundle?

- Given a total space E (e.g. a dataset) it can be written as
  - Base space B
  - Fiber space F

Arrays are simple fiber bundles

- float data[10][20][30];



Fiber space

One number
Per element
➔ 1D

BASE SPACE

THREE INDICES REQUIRED
PER ELEMENT
➔ 3D

Base space: 3D, discrete

Fiber space: 1D, continuous

Arrays are simple fiber bundles

- float data[10][20][30];

Fiber space

BASE SPACE

One number
Per element
➔ 1D

THREE INDICES REQUIRED
PER ELEMENT
➔ 3D

Base space: 3D, discrete

Fiber space: 1D, continuous

Example of a 2D x 1D Fiber Bundle

11.36	7.14	5.74	7.14	11.36
7.14	3	1.73	3	7.14
5.74	1.73	1	1.73	5.74
7.14	3	1.73	3	7.14
11.36	7.14	5.74	7.14	11.36

“Tangential Vectors” live in the fiber space

In 3D: tangential vector is represented by three numbers:

```
struct TangentialVector
{
        float x, y, z;
};
```

These are the components relative to a coordinate system; they tell “how far” to go into a certain coordinate direction.

E.g. “(1.0, 0.4, 0.3)” means:

- $1.0\,\partial_x + 0.4\,\partial_y + 0.3\,\partial_z$

Example of 3D x 3D Fiber Bundle

- TangentialVector data[10][20][30];

Fiber space

Three numbers per element
➔ 3D

BASE SPACE

THREE INDICES REQUIRED PER ELEMENT
➔ 3D

Base space: 3D, discrete

Fiber space: 3D, continuous

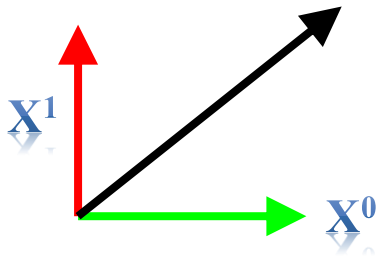
Example of a 2D x 3D Fiber Bundle

(-0.71, 0.71, 1)	(-0.89, 0.45, 1)	(-1, 0, 1)	(-0.89, -0.45, 1)	(-0.71, -0.71, 1)
(-0.45, 0.89, 1)	(-0.71, 0.71, 1)	(-1, 0, 1)	(-0.71, -0.71, 1)	(-0.45, -0.89, 1)
(0, 1, 1)	(0, 1, 1)	(0, 0, 1)	(0, -1, 1)	(0, -1, 1)
(0.45, 0.89, 1)	(0.71, 0.71, 1)	(1, 0, 1)	(0.71, -0.71, 1)	(0.45, -0.89, 1)
(0.71, 0.71, 1)	(0.89, 0.45, 1)	(1, 0, 1)	(0.89, -0.45, 1)	(0.71, -0.71, 1)

Tangential vectors are represented by numbers relative to a chart

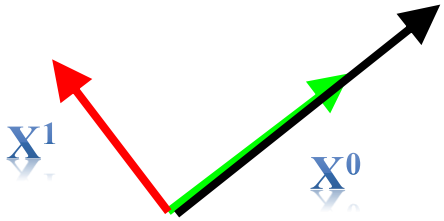
BASIS: {X^0=(1, 0, 0) X^1=(0, 1, 0) }

(-0.71, 0.71, 1)	(-0.89, 0.45, 1)	(-1, 0, 1)	(-0.89, -0.45, 1)	(-0.71, -0.71, 1)
(-0.45, 0.89, 1)	(-0.71, 0.71, 1)	(-1, 0, 1)	(-0.71, -0.71, 1)	(-0.45, -0.89, 1)
(0, 1, 1)	(0, 1, 1)	(0, 0, 1)	(0, -1, 1)	(0, -1, 1)
(0.45, 0.89, 1)	(0.71, 0.71, 1)	(1, 0, 1)	(0.71, -0.71, 1)	(0.45, -0.89, 1)
(0.71, 0.71, 1)	(0.89, 0.45, 1)	(1, 0, 1)	(0.89, -0.45, 1)	(0.71, -0.71, 1)



BASIS: {X^0=(0.71, 0.71, 0) X^1=(-0.71, 0.71, 0) }

(0, 1, 1)	(-0.32, 0.95, 1)	(-0.71, 0.71, 1)	(-0.95, 0.32, 1)	(-1, 0, 1)
(0.32, 0.95, 1)	(0, 1, 1)	(-0.71, 0.71, 1)	(-1, 0, 1)	(-0.95, -0.32, 1)
(0.71, 0.71, 1)	(0.71, 0.71, 1)	(0, 0, 1)	(-0.71, -0.71, 1)	(-0.71, -0.71, 1)
(0.95, 0.32, 1)	(1, 0, 1)	(0.71, -0.71, 1)	(0, -1, 1)	(-0.32, -0.95, 1)
(1, 0, 1)	(0.95, -0.32, 1)	(0.71, -0.71, 1)	(0.32, -0.95, 1)	(0, -1, 1)



Numbers vs. Geometry

Tangential vector components depend on the chosen coordinate system, the tangential vector as object does not.

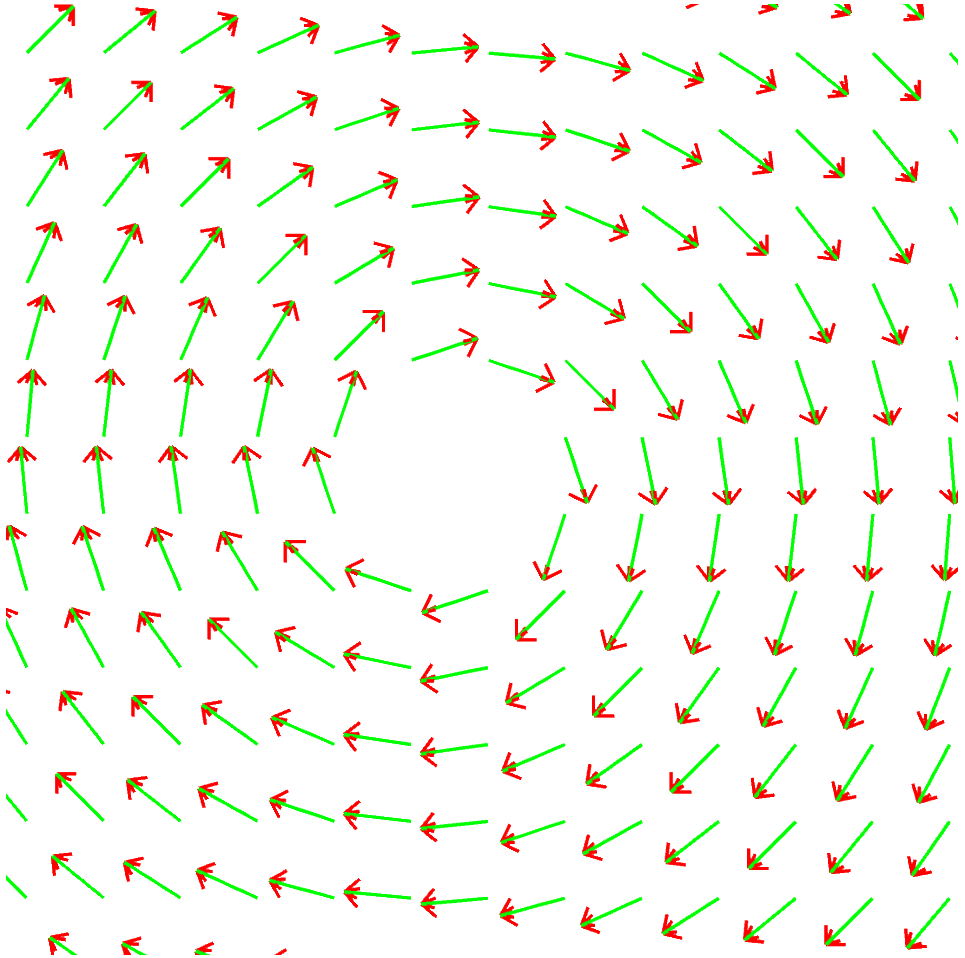
It is easy to construct a numerical expression that depends only on the coordinate system, but does not tell anything about the object.

Danger of Artifacts!

Use coordinate-free expressions!

Accurate Scientific Visualization must take *all* components into account

Graphical Representation is intrinsically coordinate-free



"Visual thinking"

Thinking in visual representations rather than numbers

→ **Geometric Algebra** as a well-founded tool to think about mathematical operations visually

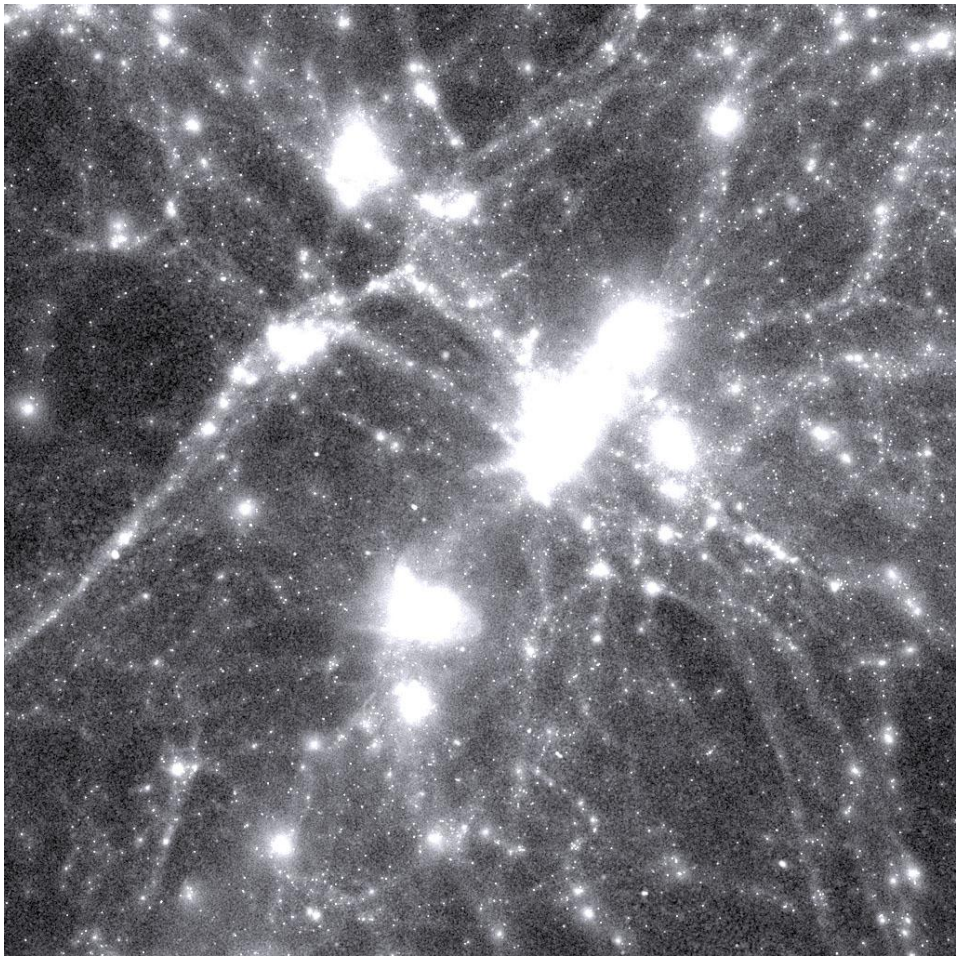
→ Intrinsically coordinate free

Base space dimensionality

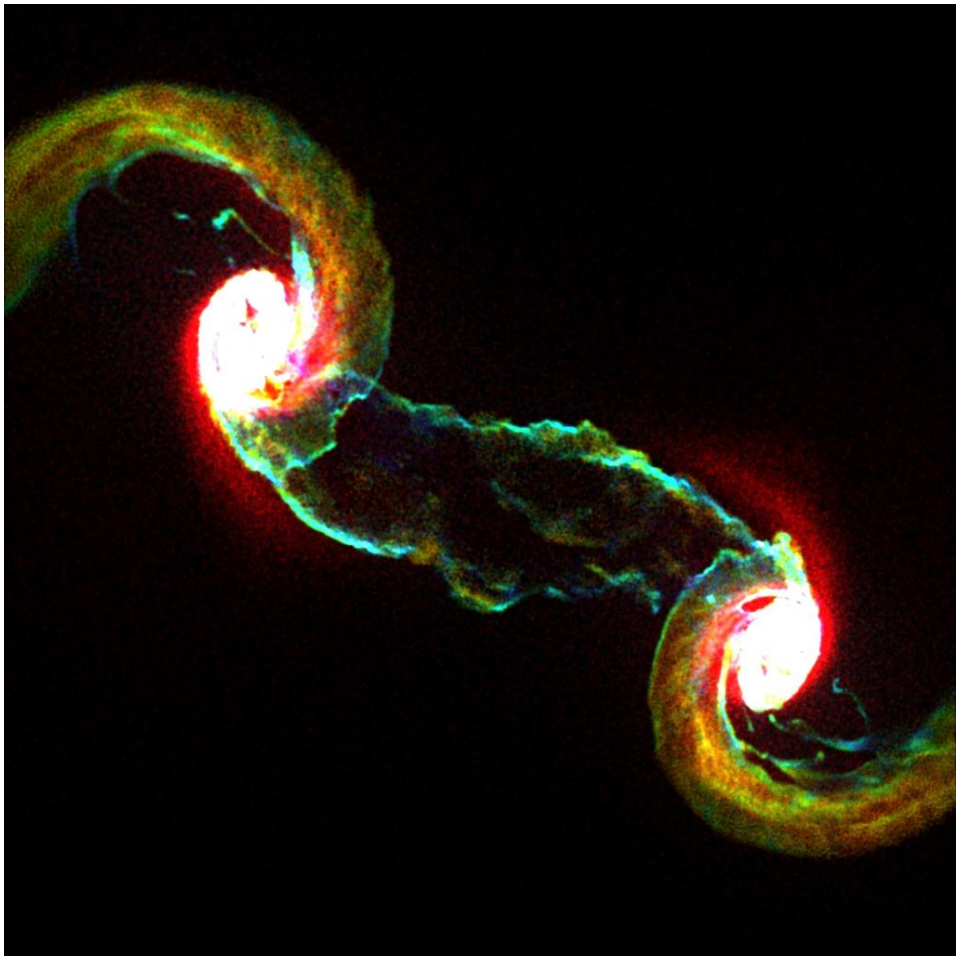
Fiber space dimensionality

VIZ EXAMPLES OF FIBER BUNDLES

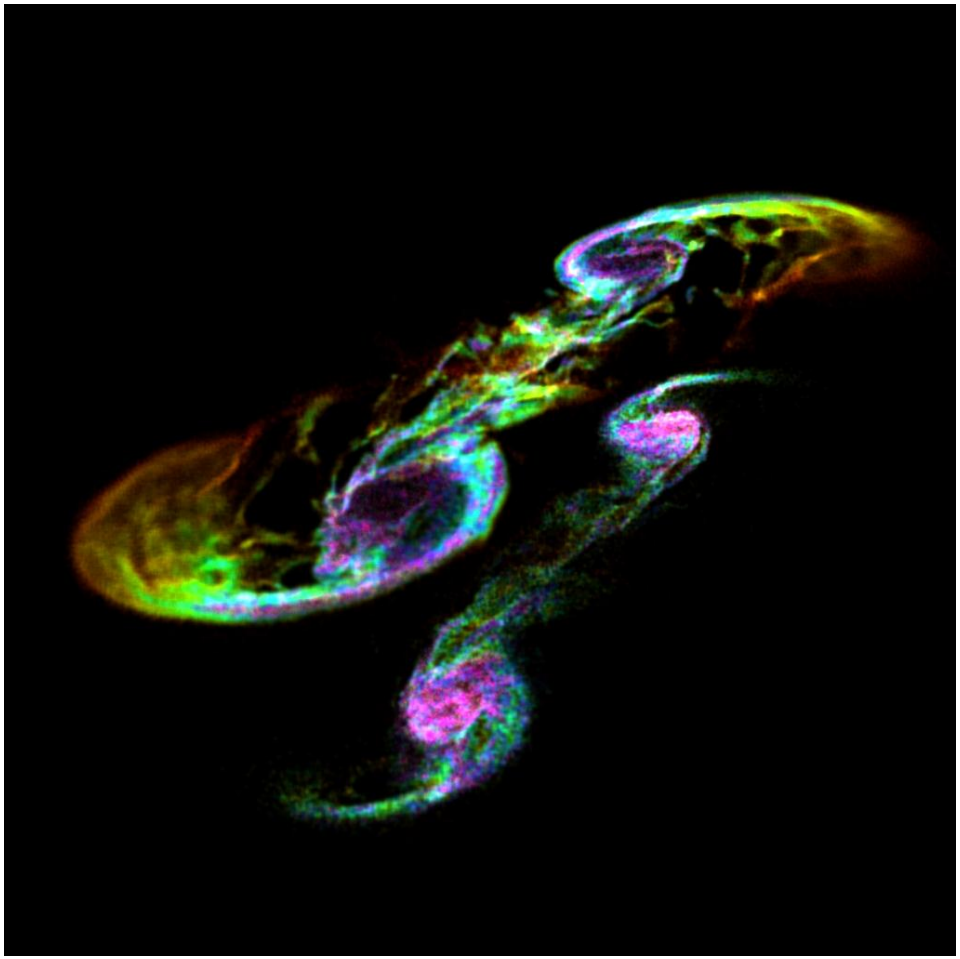
0D – 0D



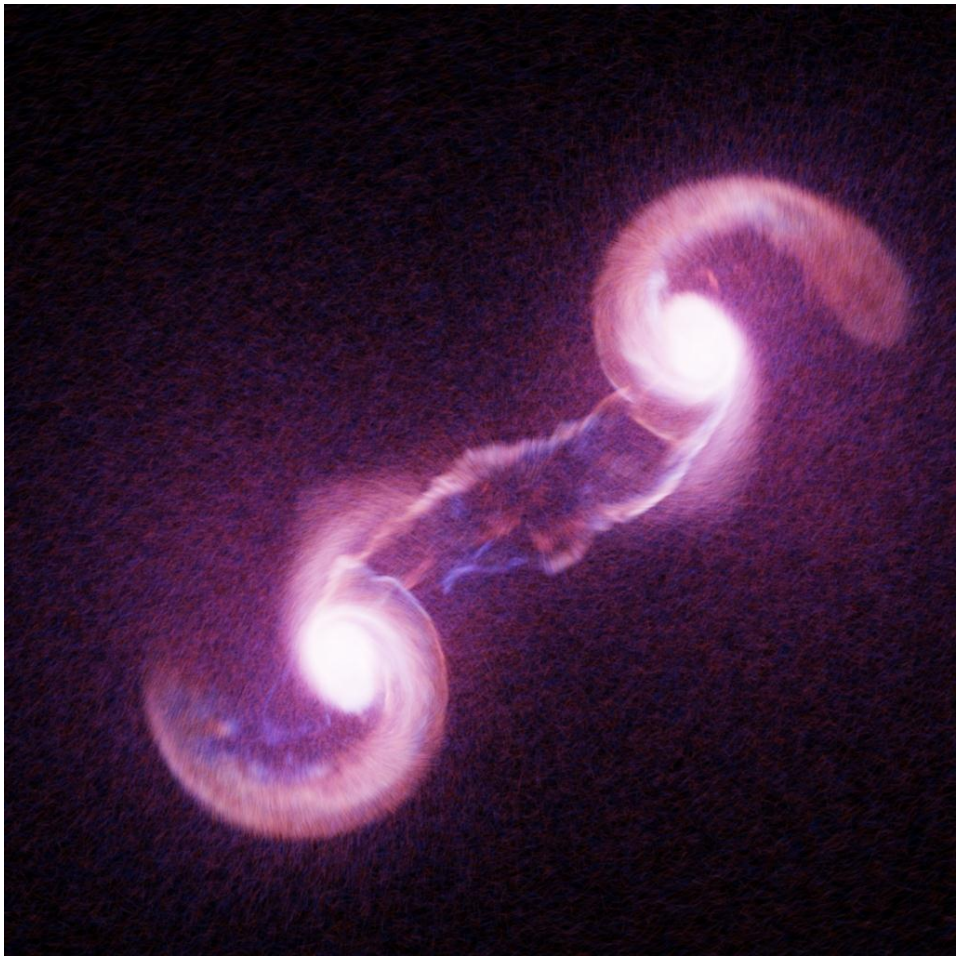
0D – 1D



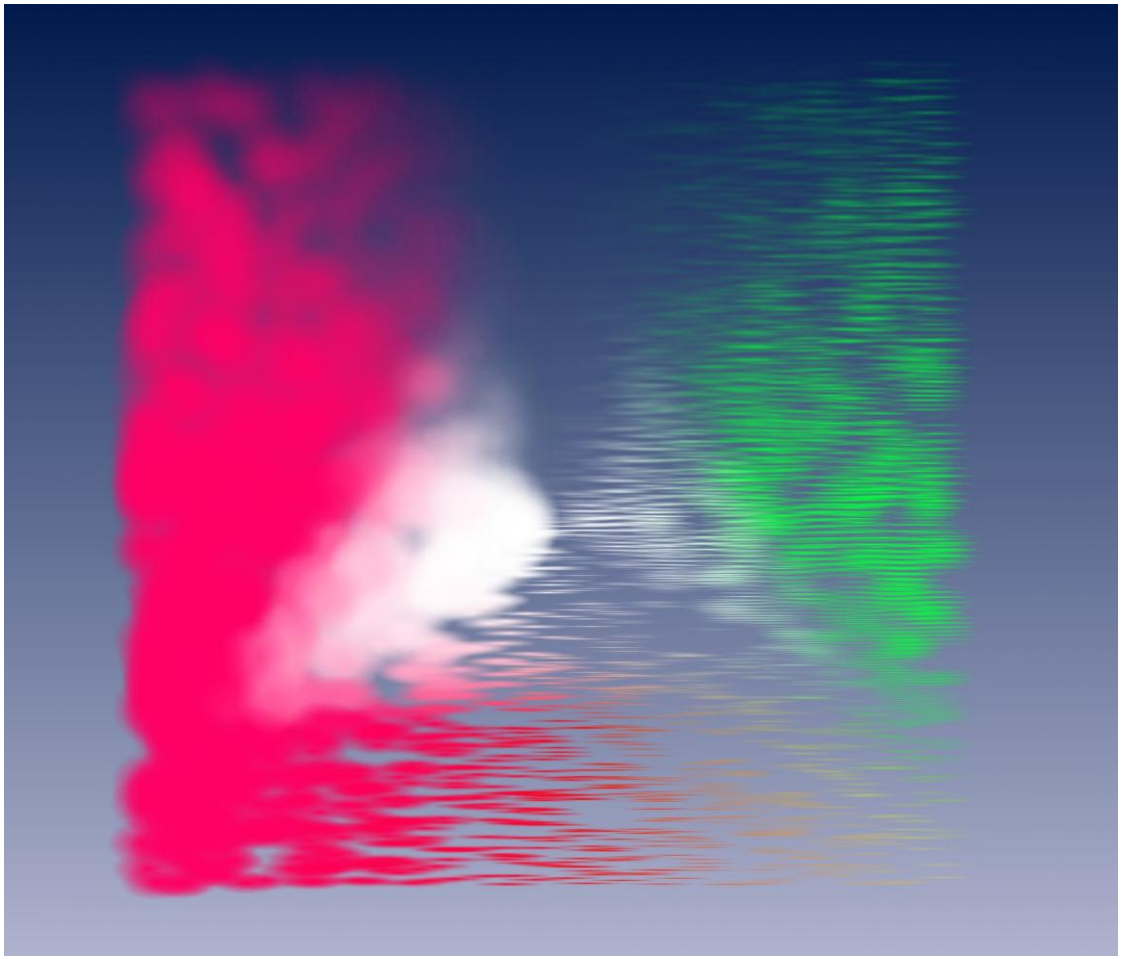
0D – 2D



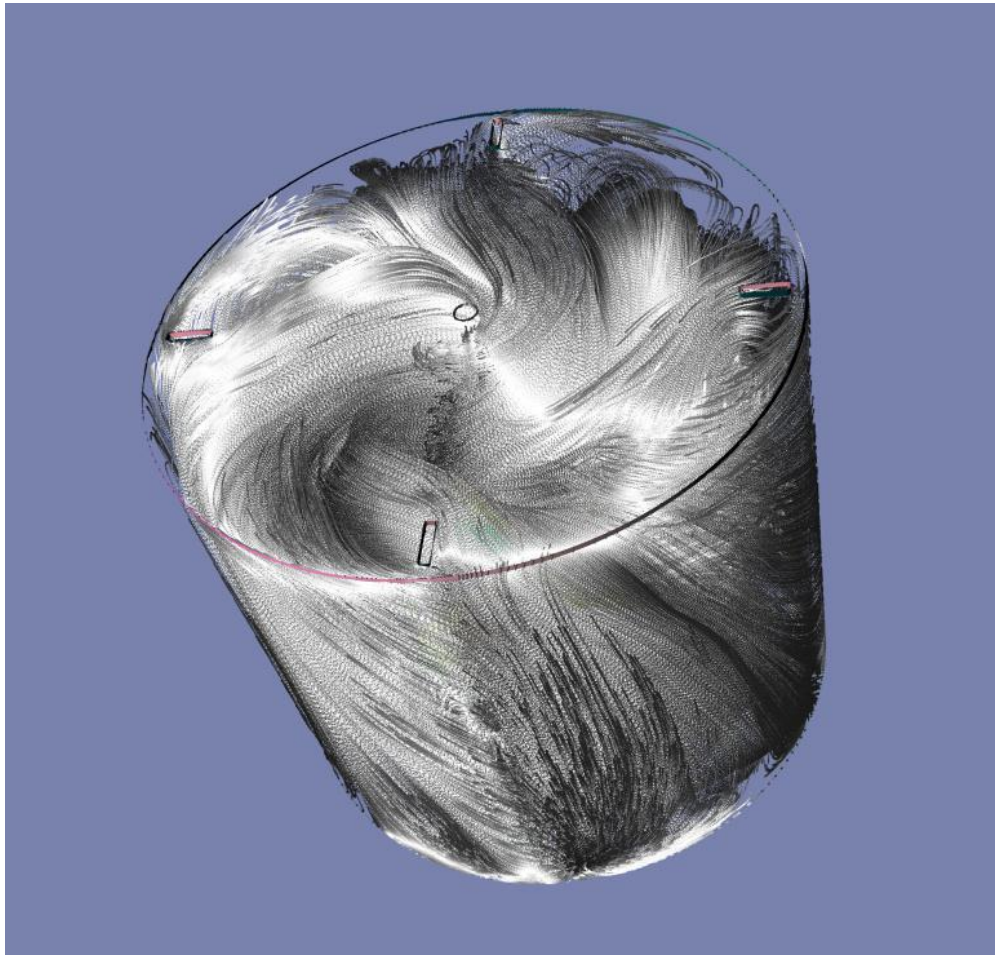
0D – 3D



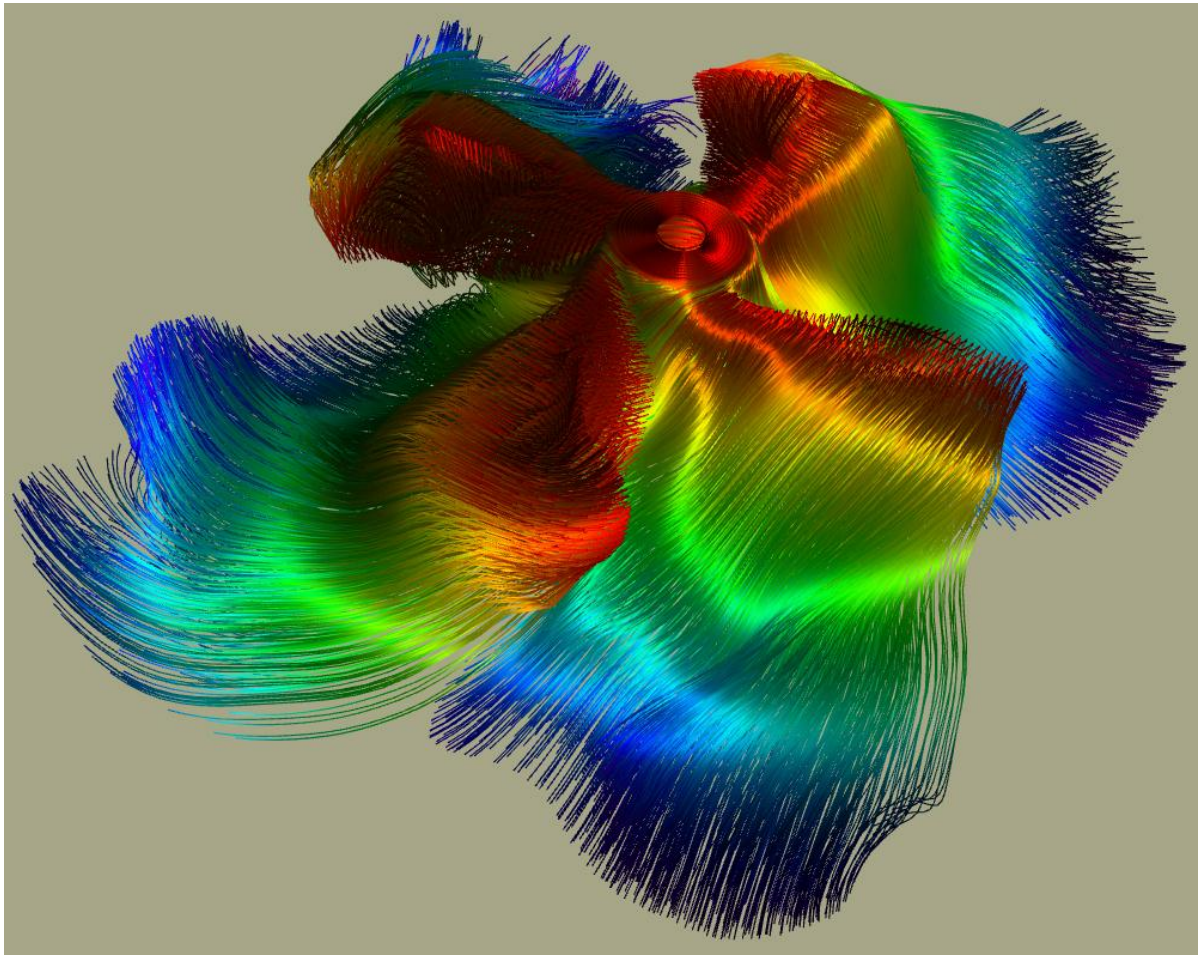
0D – 6D



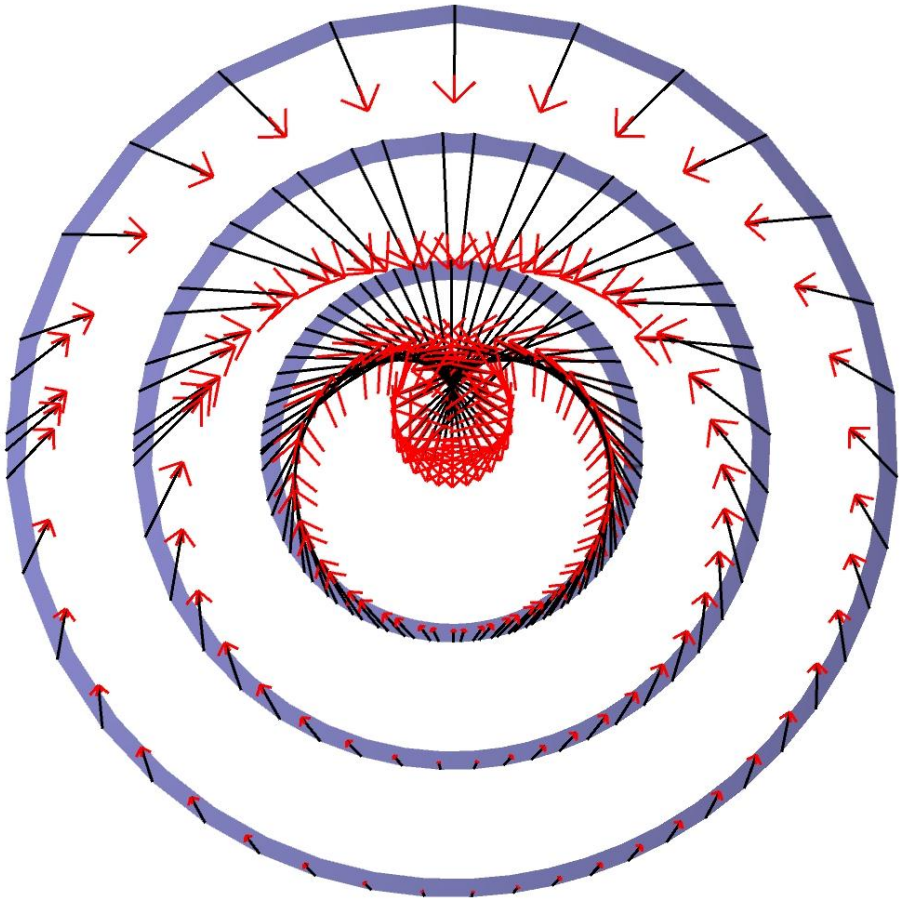
1D – 0D



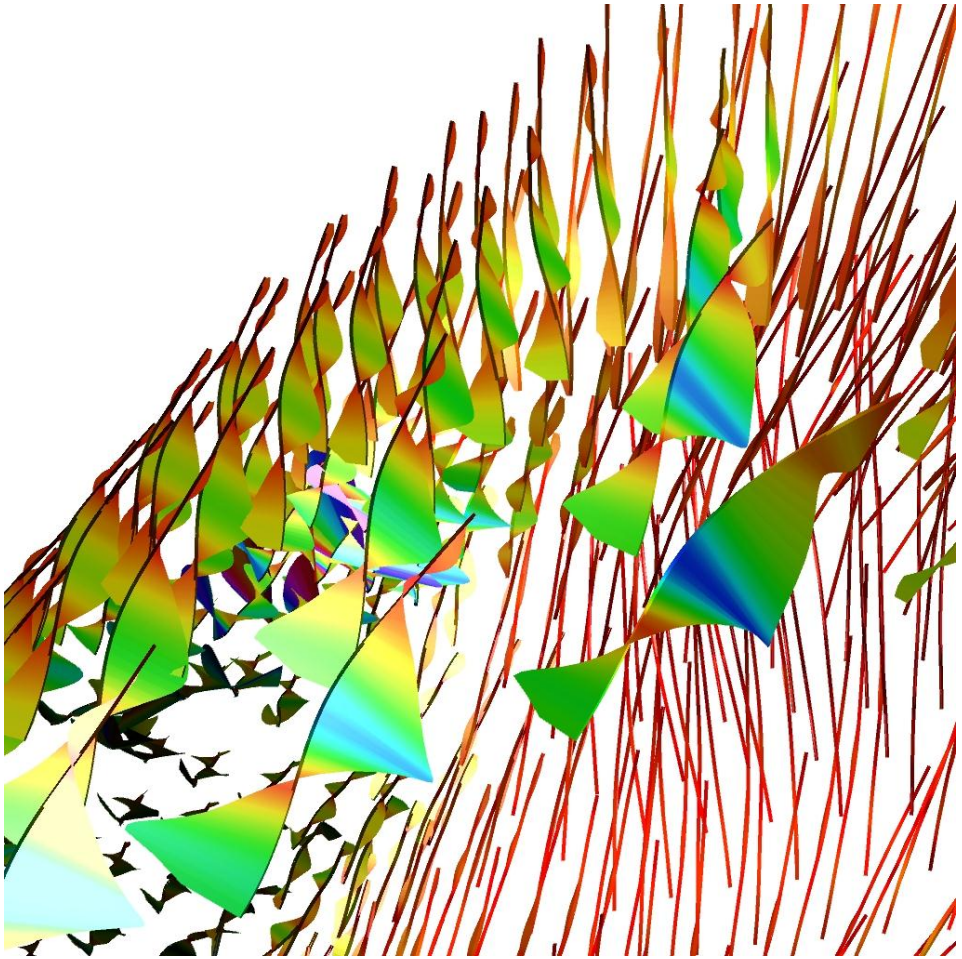
1D – 1D



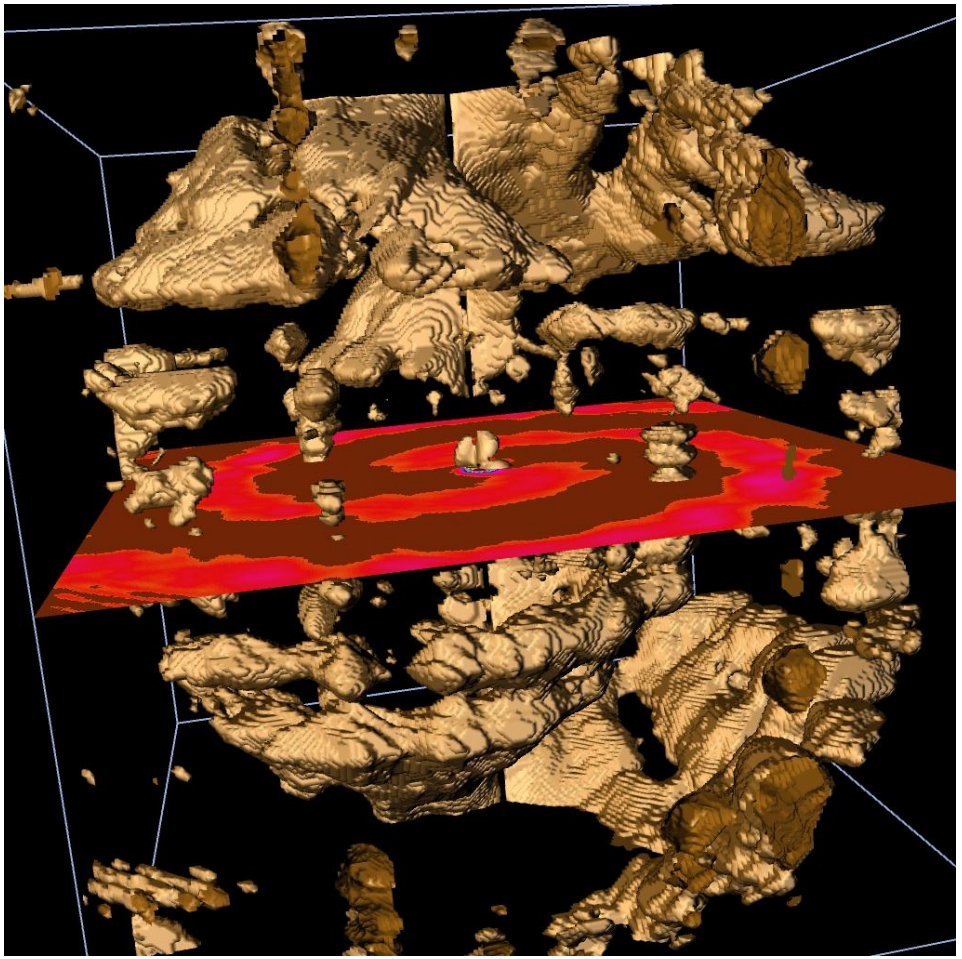
1D – 3D



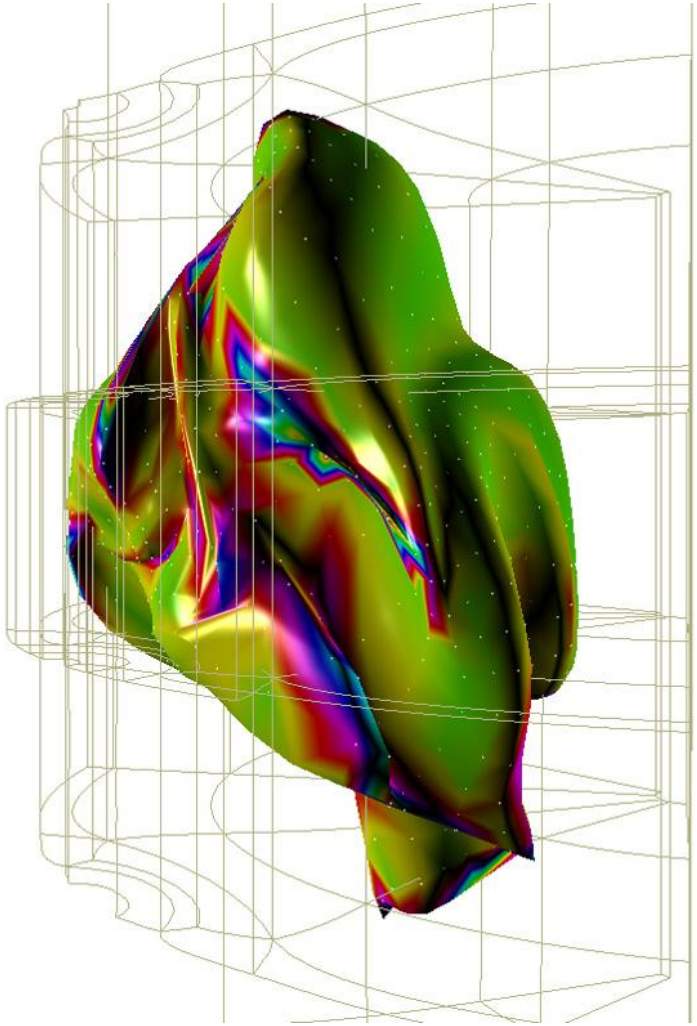
1D – nD



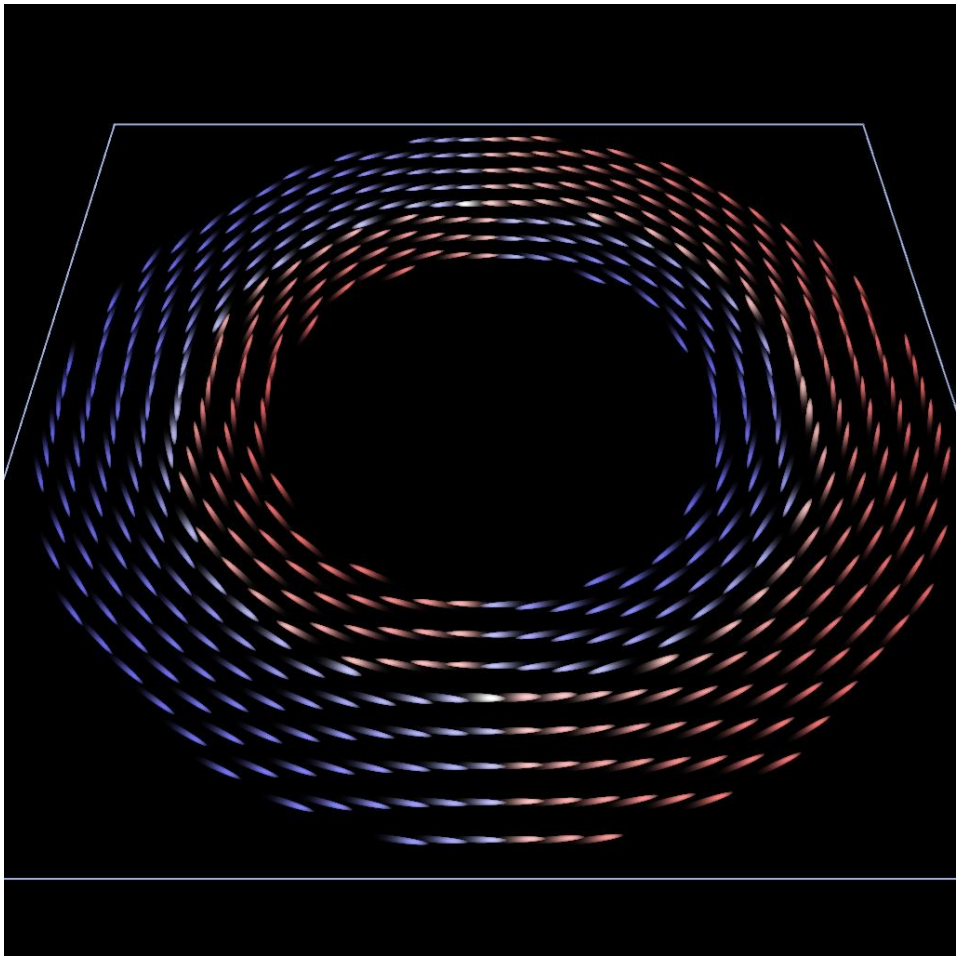
2D – 0D



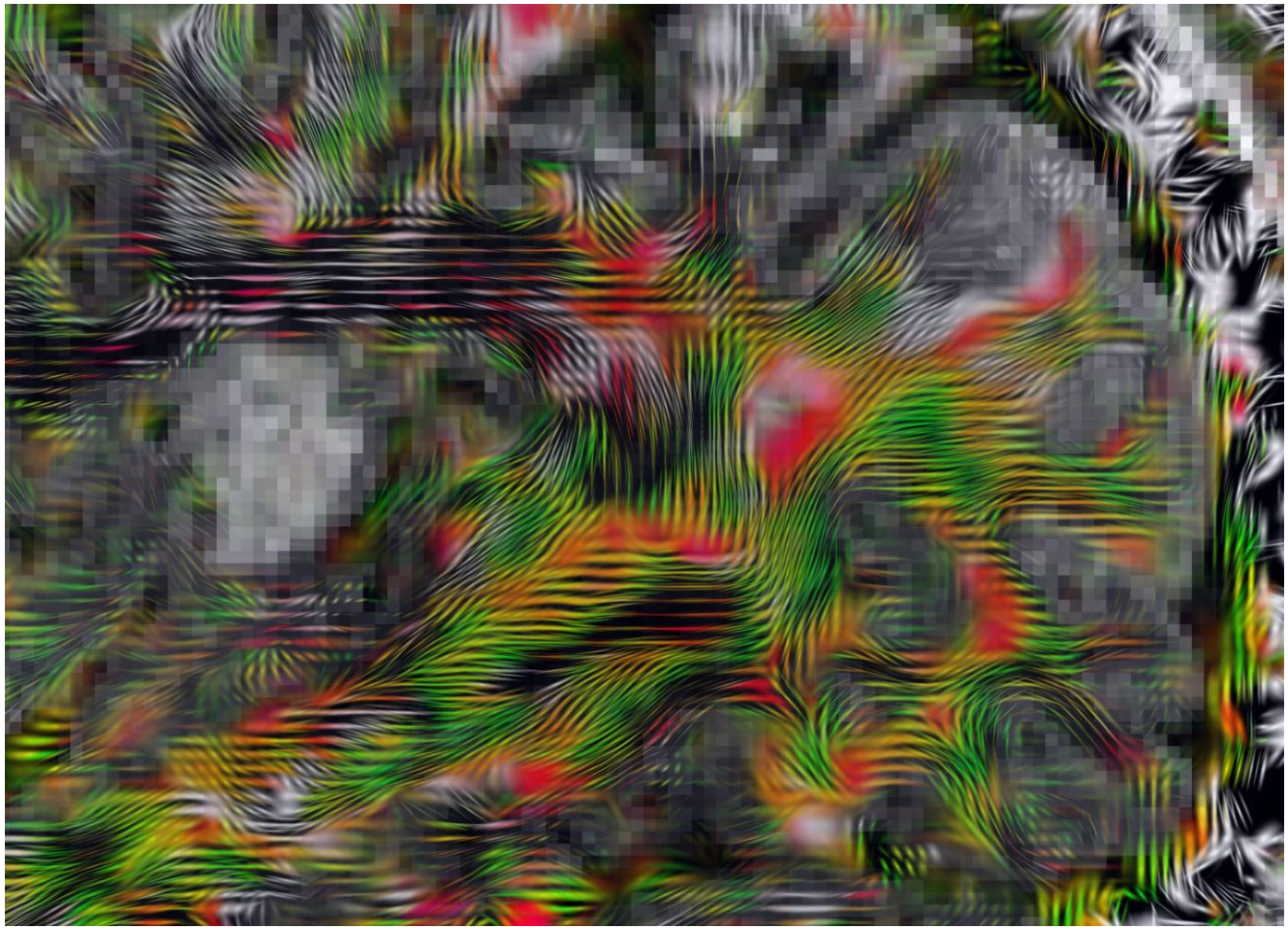
2D – 1D



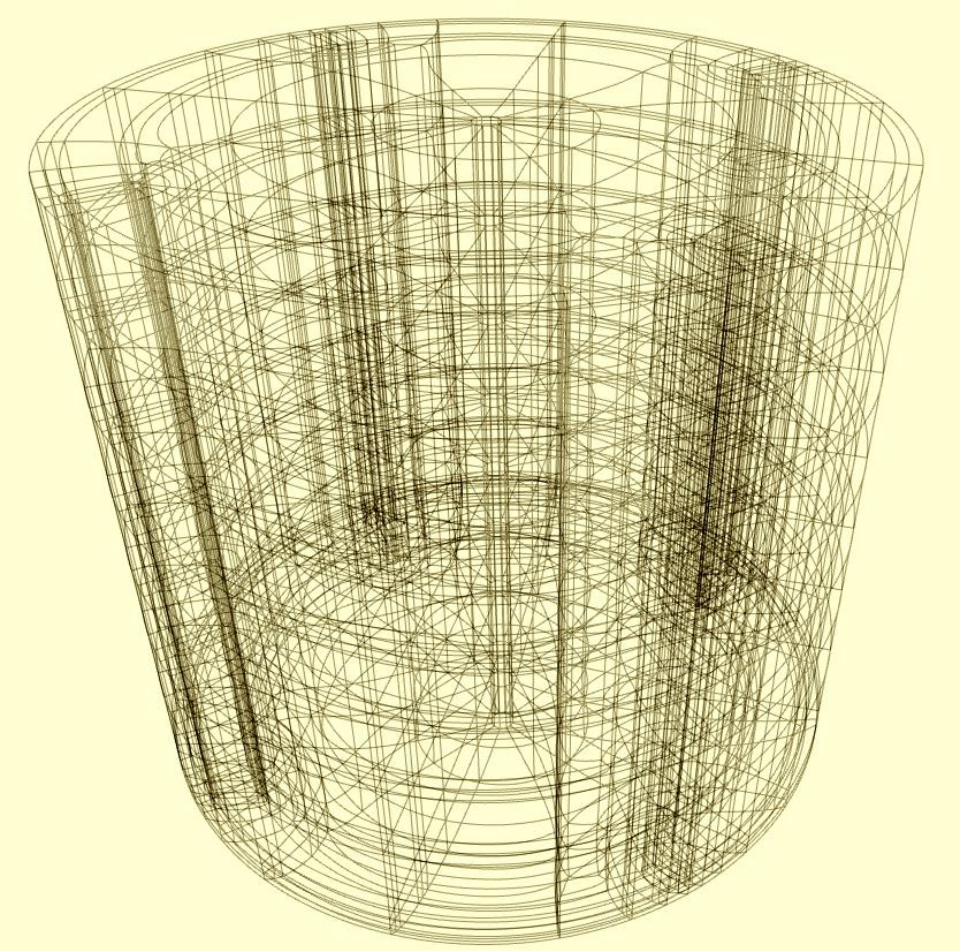
2D – 3D



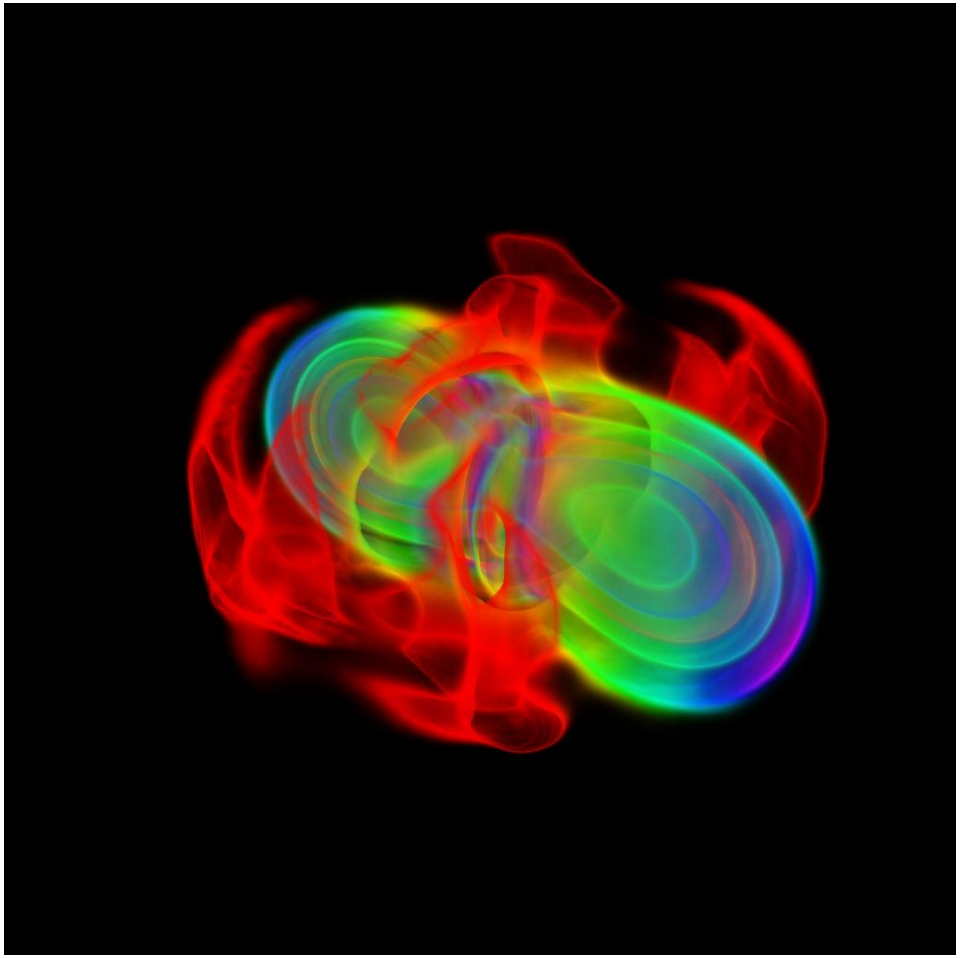
2D – 6D



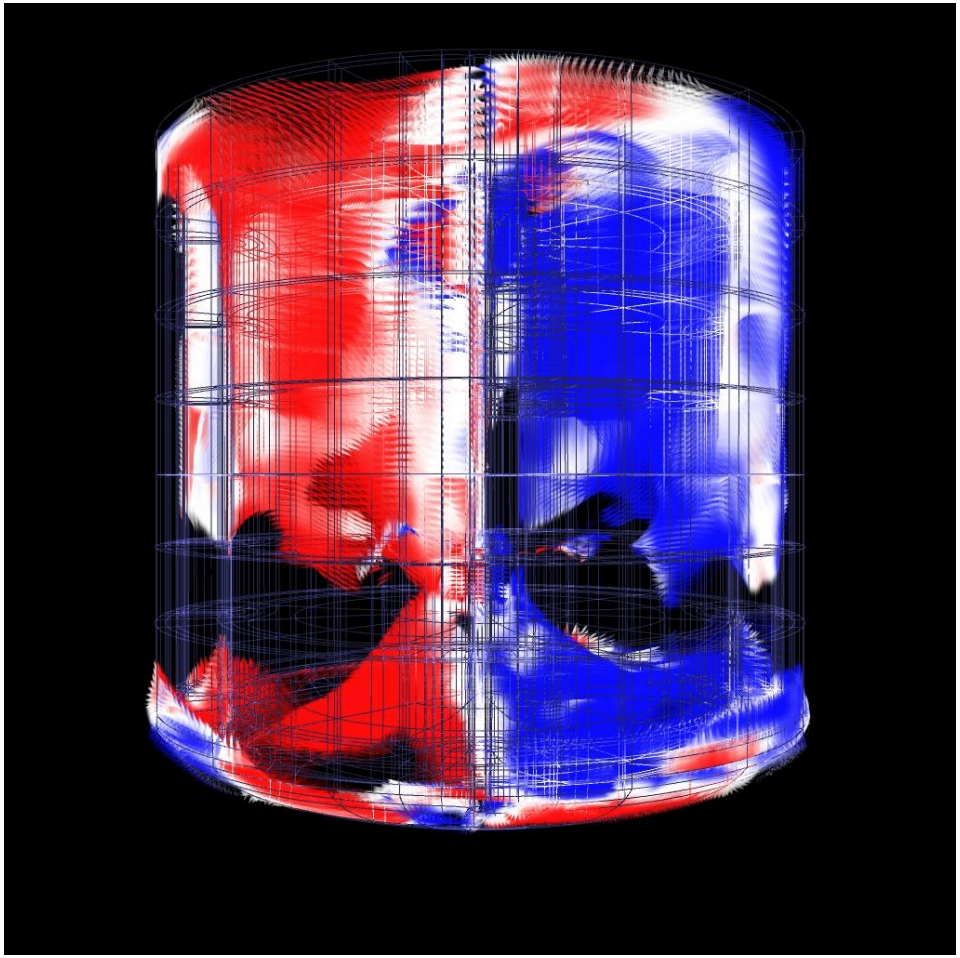
3D – 0D



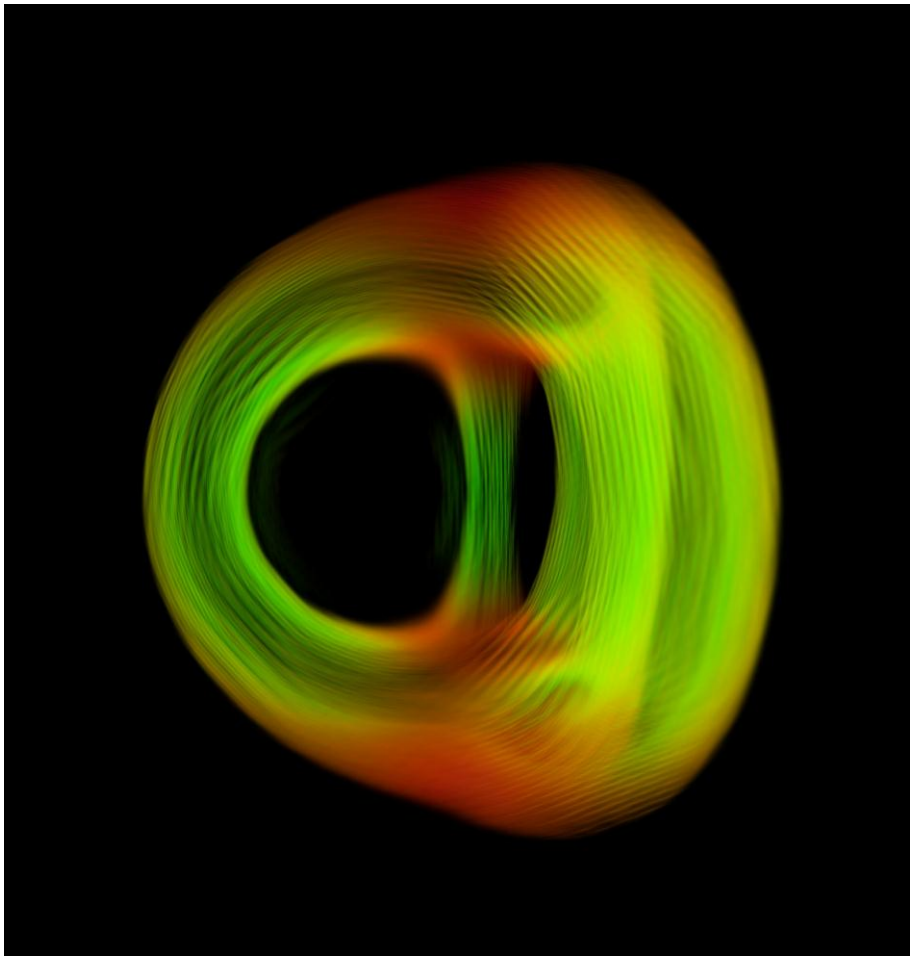
3D – 1D

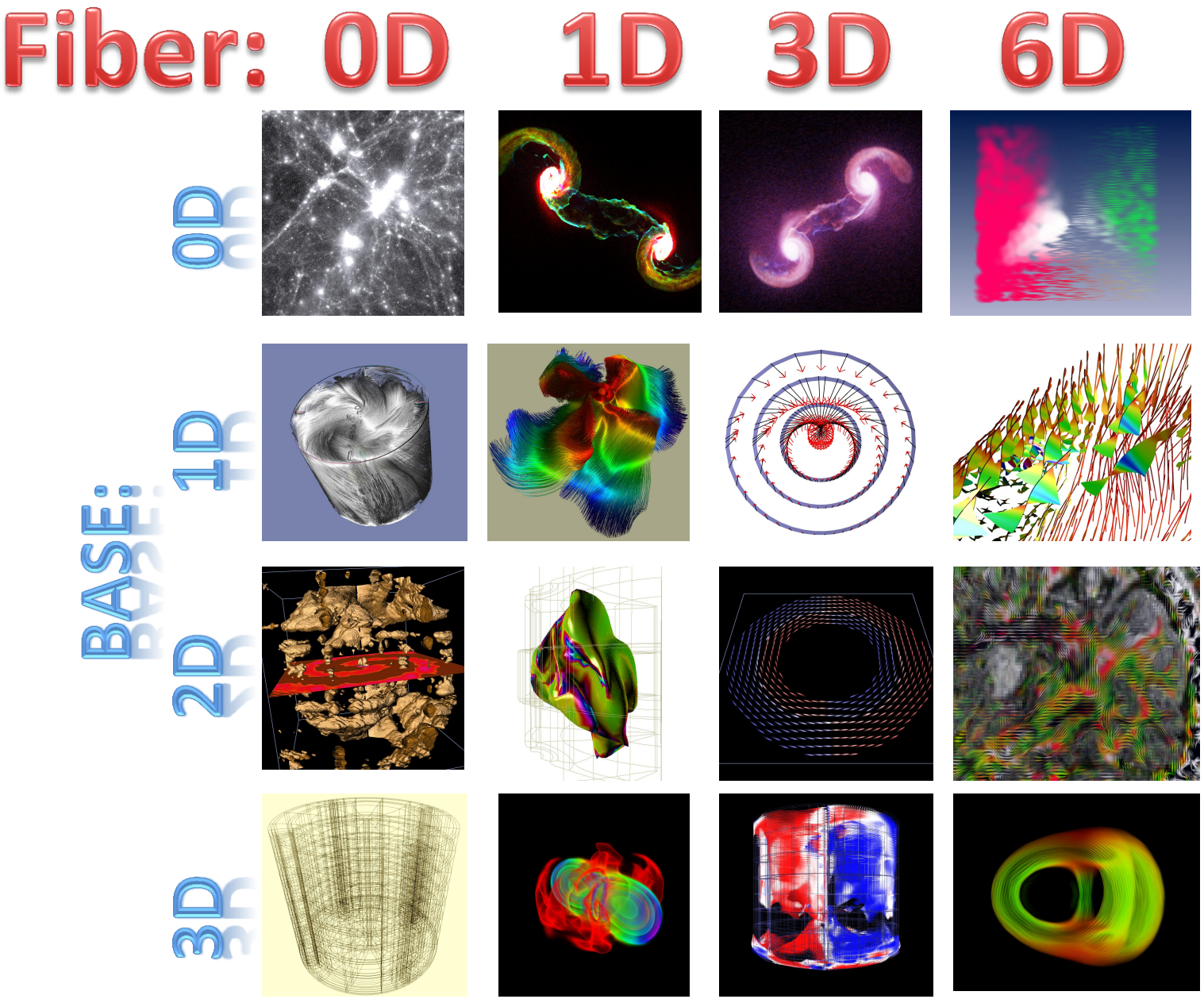


3D – 3D



3D – 6D




Fiber: 0D 1D 3D 6D
BASE: 0D 1D 2D 3D

Questions on a Data Set

What is the dimensionality of the base space?

What is the dimensionality of the fiber space?

What is the discretization scheme of the base space?

- Simplex, n-cubes, hierarchical, …

What are the properties of the fiber space?

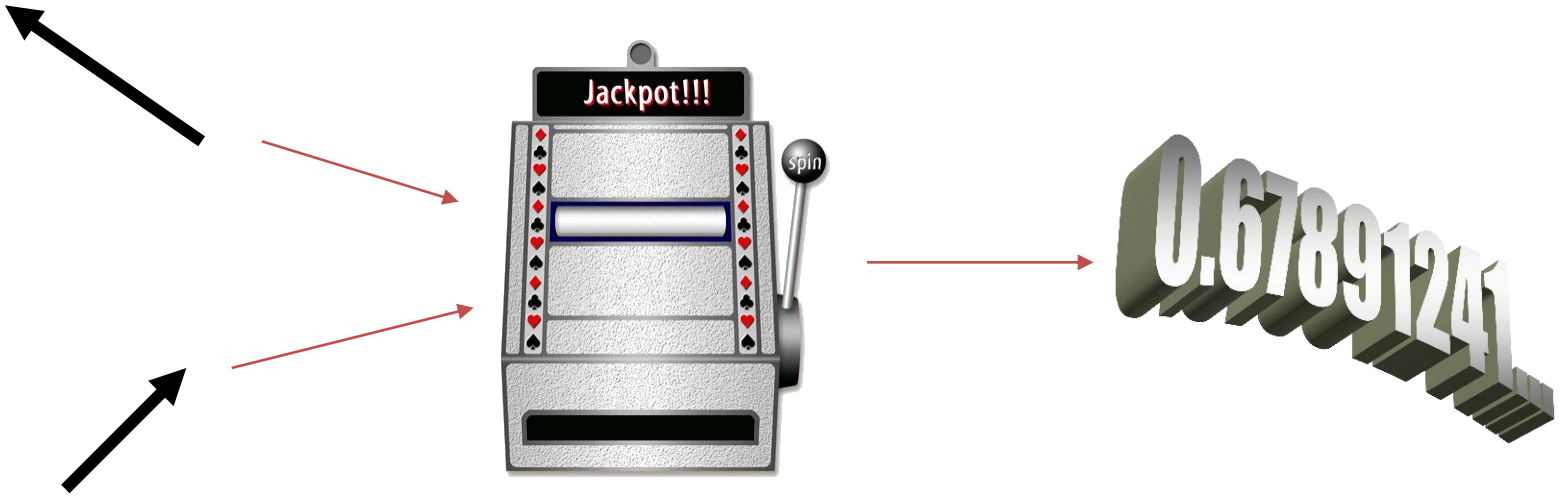
- Scalar, vector, tensor, …

Visualizing high dimensional fiber spaces

TENSOR FIELDS

What is a “tensor”?

- “two-slot machine”: put in two vectors, out pops a number (C. Leubner):



Coordinate expression

$v=v^x\,\partial_x + v^y\,\partial_y + v^z\,\partial_z$

$u=u^x\,\partial_x + u^y\,\partial_y + u^z\,\partial_z$

$$\begin{bmatrix} u^x & u^y & u^z \end{bmatrix} \begin{bmatrix} g_{xx} & g_{xy} & g_{xz} \\ g_{yx} & g_{yy} & g_{yz} \\ g_{zx} & g_{zy} & g_{zz} \end{bmatrix} \begin{bmatrix} v^x \\ v^y \\ v^z \end{bmatrix}$$

$g(u,v)=$

$g(v^x\,\partial_x + v^y\,\partial_y + v^z\,\partial_z, u)=$

$v^x g(\partial_x, u) + v^y\, g(\partial_y, u) + v^z\, g(\partial_z, u)=$

$v^x u^x\, g_{xx} + v^x\, u^y\, g_{xy} + v^x u^z\, g_{xz} +$

$v^y u^x\, g_{yx} + v^y\, u^y\, g_{yy} + v^y u^z\, g_{yz} +$

$v^z u^x\, g_{zx} + v^z\, u^y\, g_{zy} + v^z u^z\, g_{zz}$

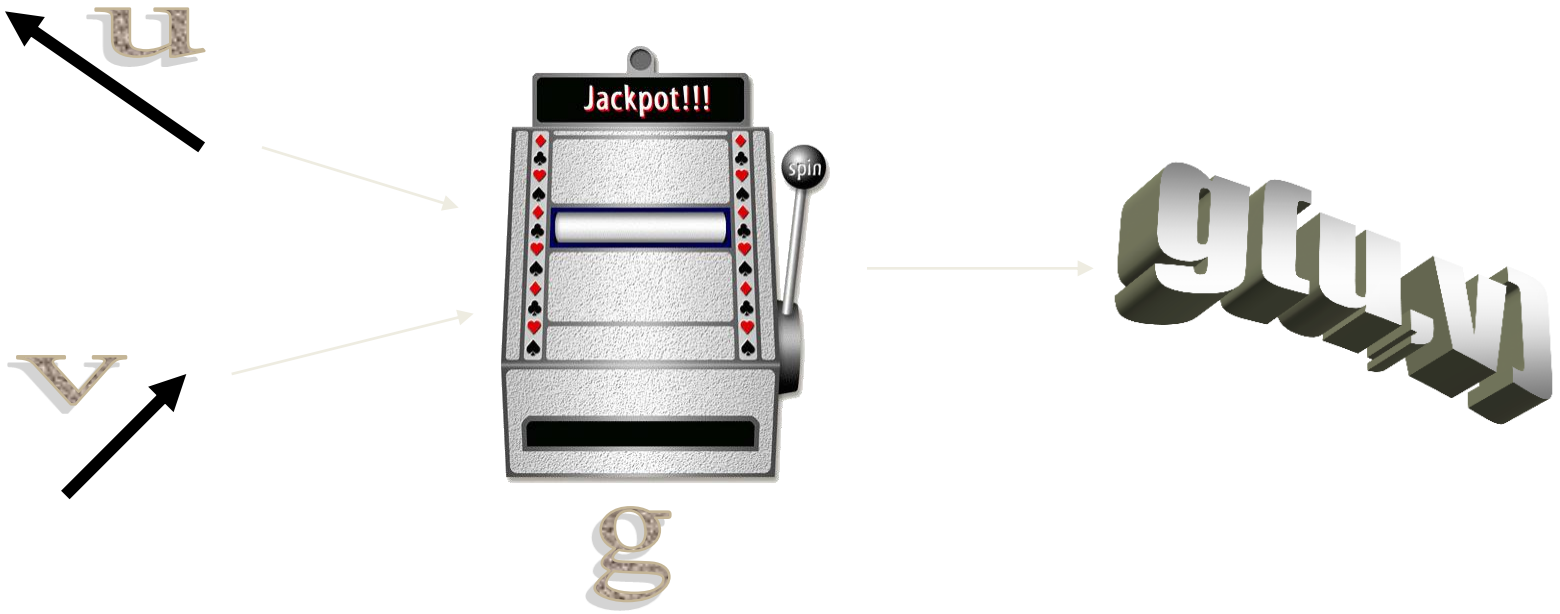
$g(\partial_a, \partial_b) =: g_{ab}$

What is a “tensor field”?

- Lot of slot machines at each point in space

Visualizing a tensor?

- Linear in each argument → multilinear
  - Vectors u,v, Tensor g: g(u,v),
  - special case: symmetric: g(u,v)=g(v,u)



Coordinate expression

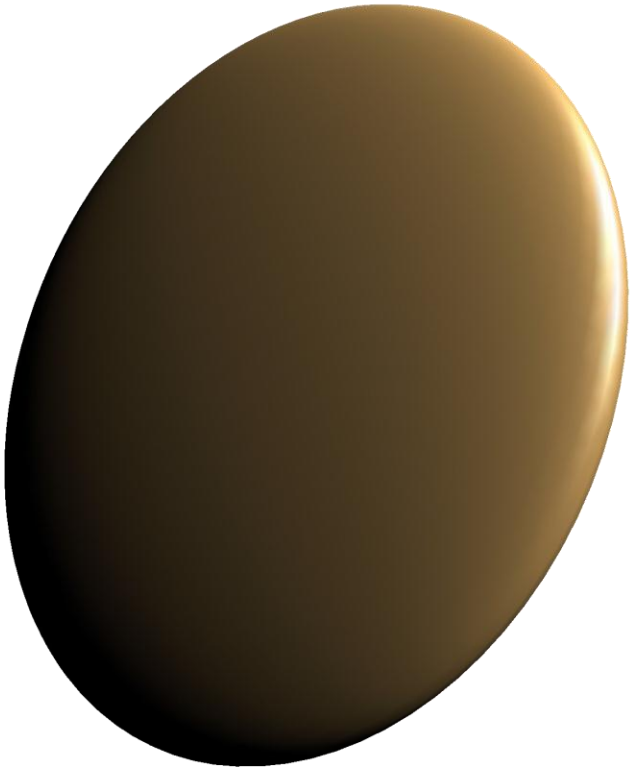
$v=v^x \partial_x + v^y \partial_y + v^z \partial_z$

$u=u^x \partial_x + u^y \partial_y + u^z \partial_z$

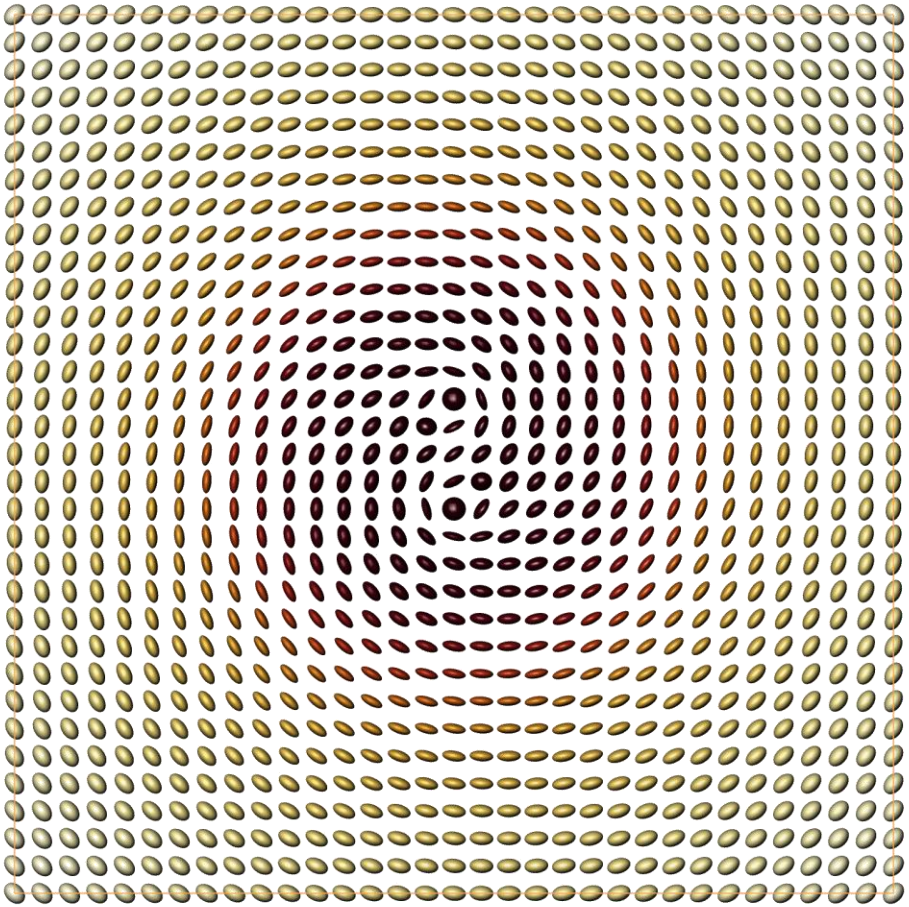
$g(v,u)=v^x u^x g_{xx} + 2\, v^x u^y g_{xy} + v^y u^y g_{yy} + 2\, v^y u^z g_{yz} + v^z u^z g_{zz} + 2\, v^x u^z g_{xz}$

Quadratic expression, inspect *g(v,v)=const.*

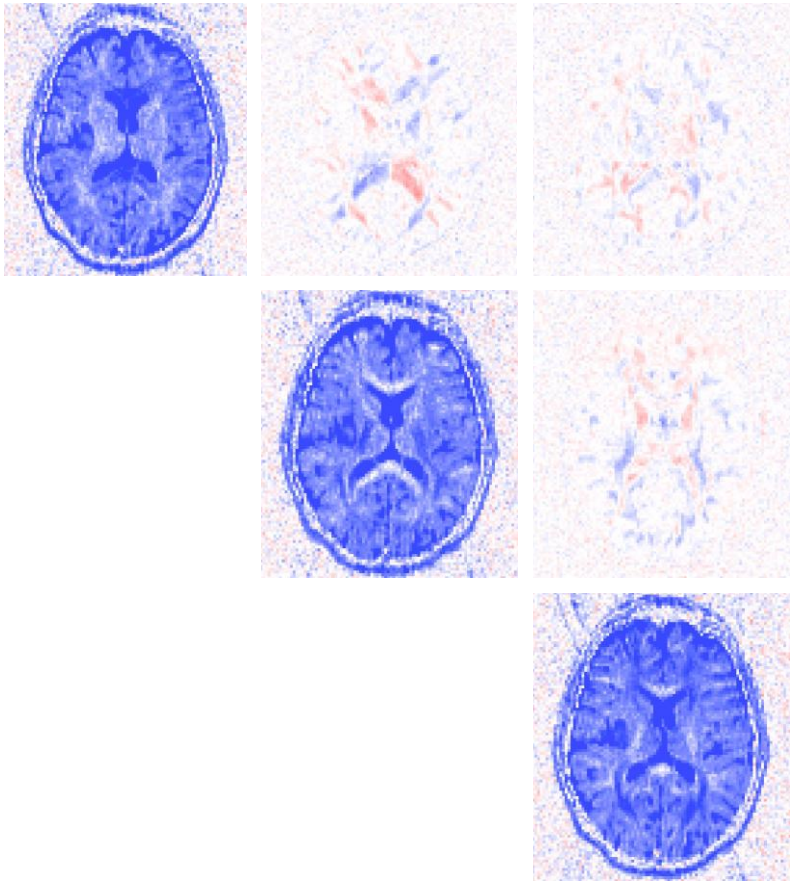
If g(u,v)>0 for all u,v, then positive definite → equation of an ellipsoid



Rendering a Tensor Field



Diffusion Matrix (2^{nd} order flux)



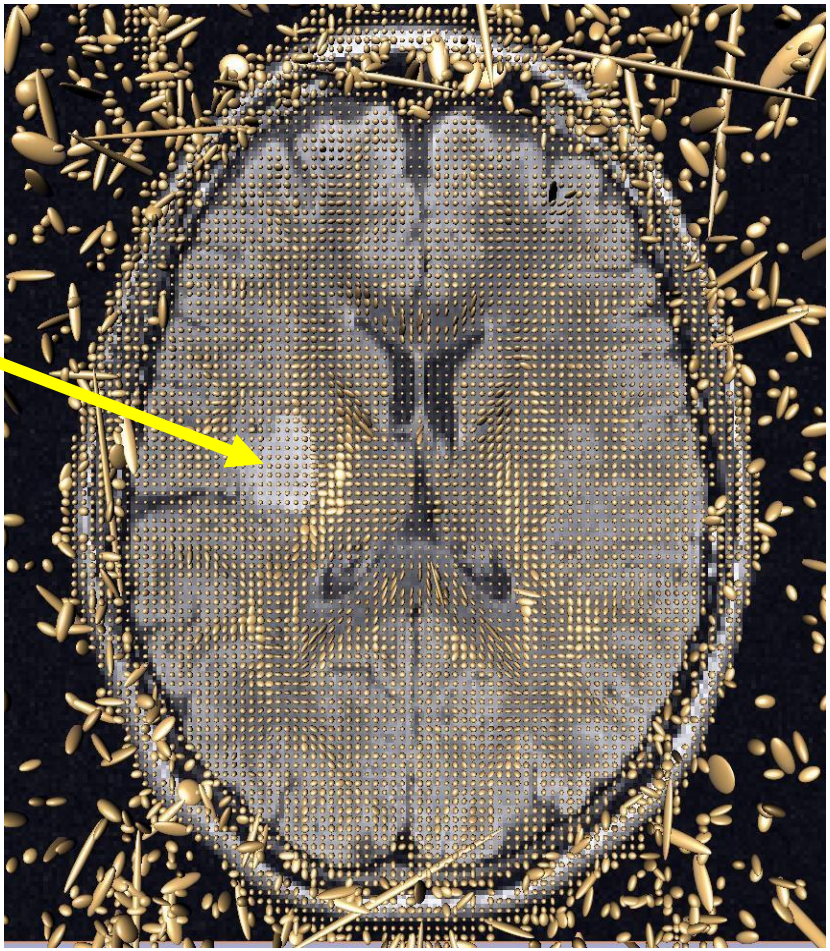
$$D(v) = D_{ij}\, v^i\, e^j + D_{ijk}\, v^i\, e^j\, e^k + \ldots$$

$$(Dij) = \begin{pmatrix} D_{xx} & D_{xy} & D_{zz} \\ & D_{yy} & D_{yz} \\ & & D_{zz} \end{pmatrix}$$

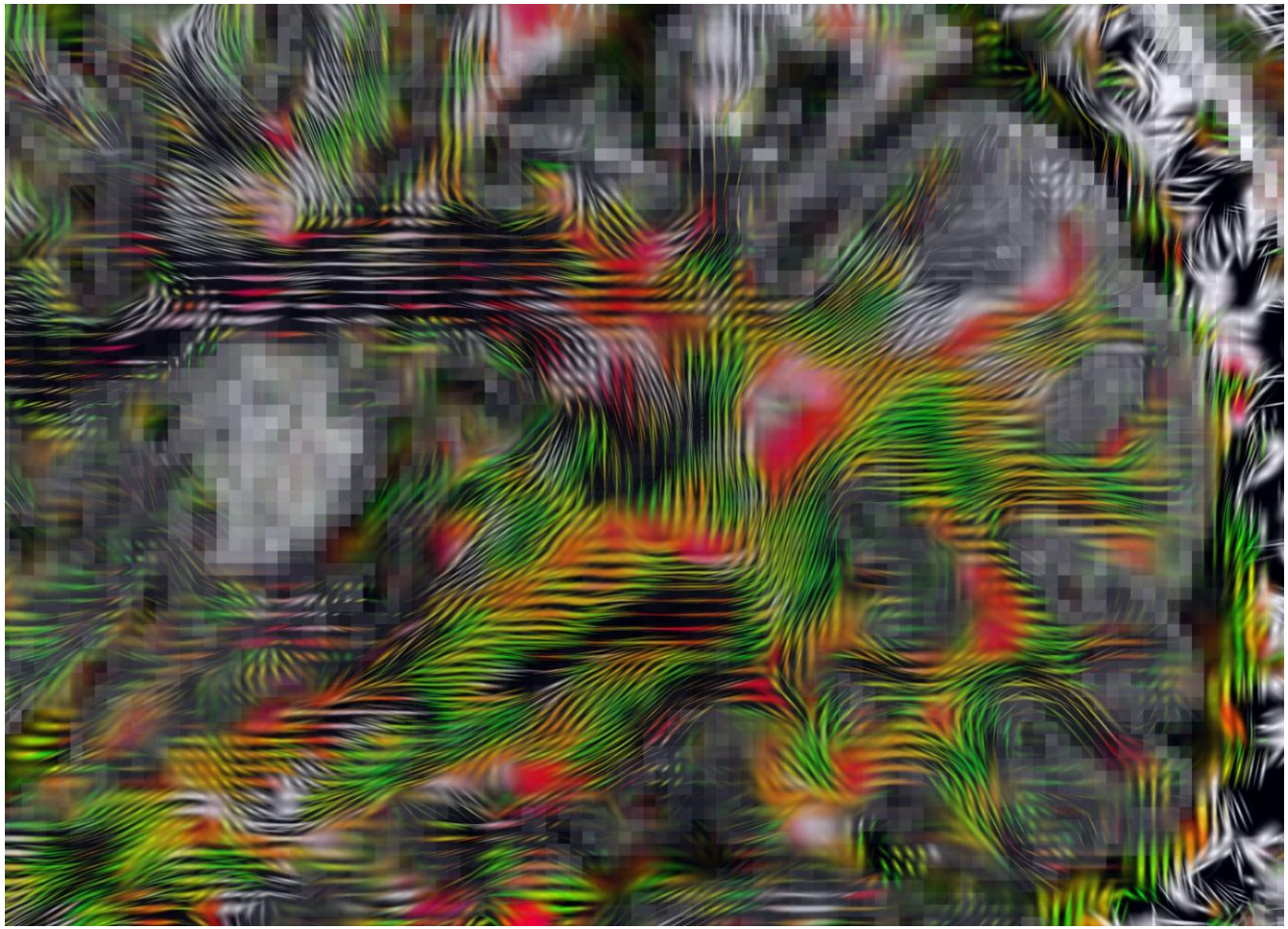
Six independent images
Each of them coordinate-specific information
Need holistic visualization

Rendering a Tensor Field

Tumor
(this one clearly visible in T2 image)

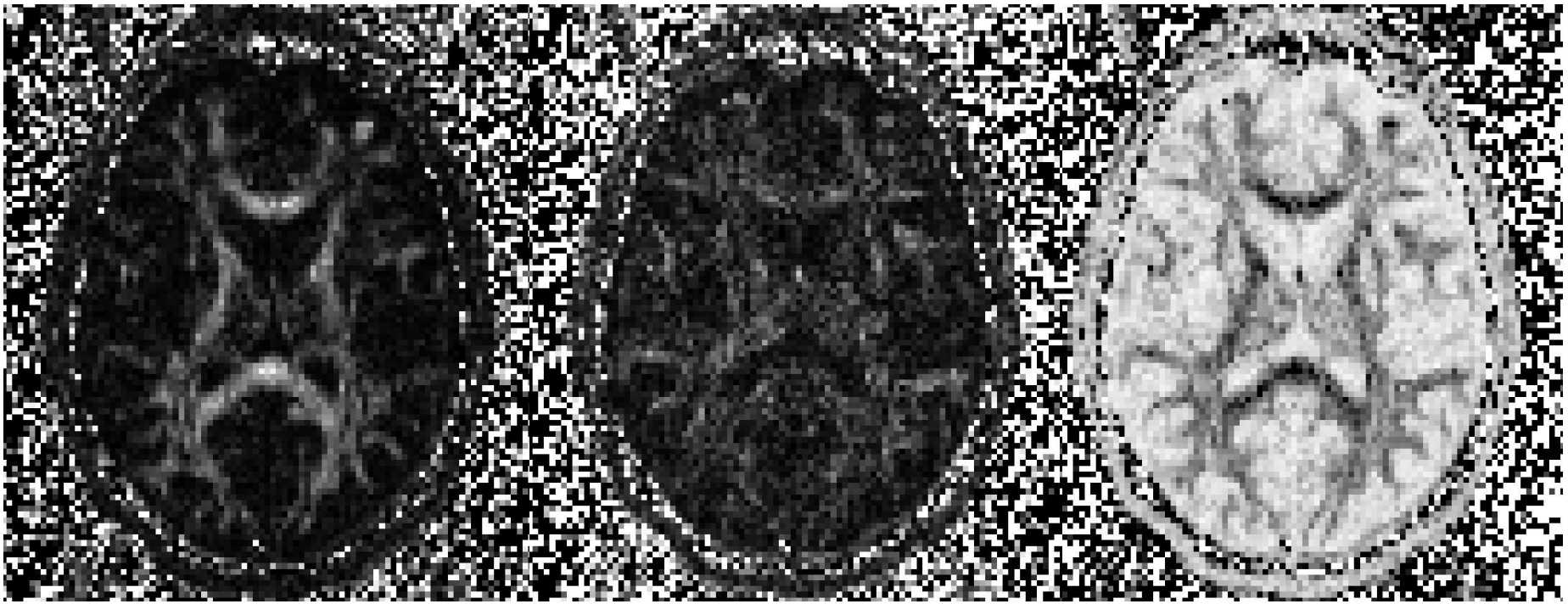


2D – 6D



Scalar Invariants: Shape Factors as Scalar Fields

Not all scalars are “bad”- but they need to be define *covariant* (invariant) and only reveal an aspect of the data set, not its entire properties.



Structure of the Fiber Space

Unifying Data – Unify Mathematics?

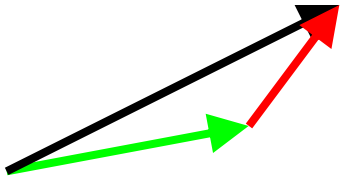
Fiber Space Operations

GEOMETRIC ALGEBRA

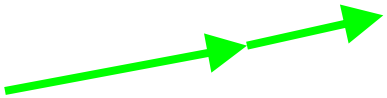
Geometry and Vectors

- Geometric interpretation of a vector
  - Directed line segment or *tangent*

- Vector-algebra in Euclidean Geometry or $T_p(M)$



  - Addition / subtraction of vectors **a**+**b**
  - Multiplication / division by scalars α **a**



  - **Multiplication / Division of vectors??**

Is the vector calculus algebraically closed?

- Have: vector addition and scalar multiplication
- But: Could there be an *invertible product* of vectors?
- What would it mean to divide vectors "**a**/**b**" ??
  - **ab**=C → **b**=$\mathbf{a}^{-1}$C
  - Note: We do not require C to be a vector as well!

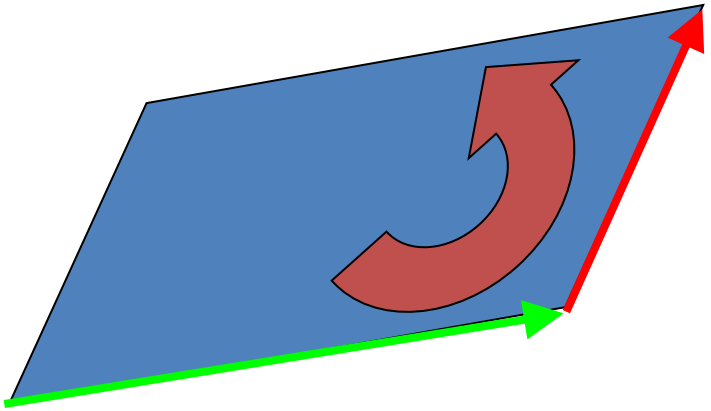
  - Inner product (not associative): **a**·**b** → skalar
    - *Is not invertible*

      e.g. **a**·**b** =0 where **a**≠**0**, **b**≠**0** but orthogonal
  - Exterior product (associative): **a**∧**b** → bivector
    - Generalization of the cross-product in 3D: **a**×**b**
    - *Not invertible*

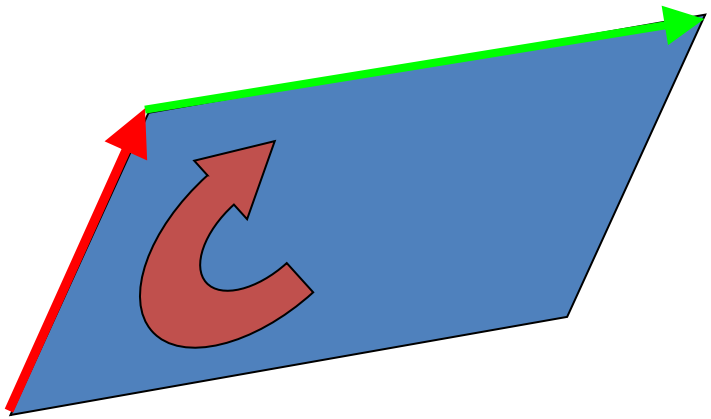
      e.g. **a**×**b** =0 when **a**≠**0**, **b**≠**0** but parallel

Grassmann-Product
Bivector a∧b

Describes the area that is defined by two vectors **a** and **b**, the sign is the *orientation* of the area



a∧b

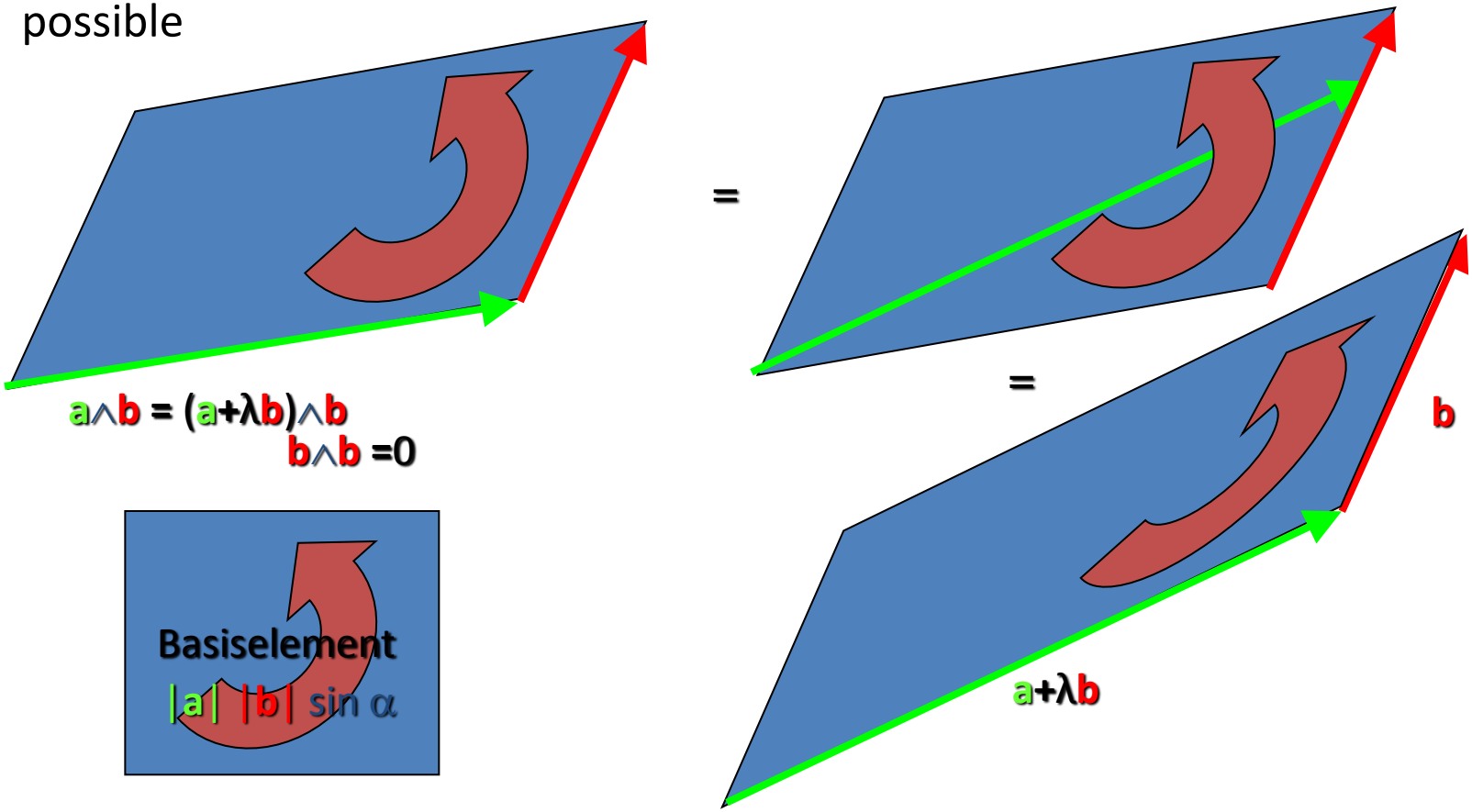


b∧a = -a∧b

Defined in arbitrary dimensions, anti-symmetric (→ not commutative), associative, distributive, does not require additional structures, a∧a=0

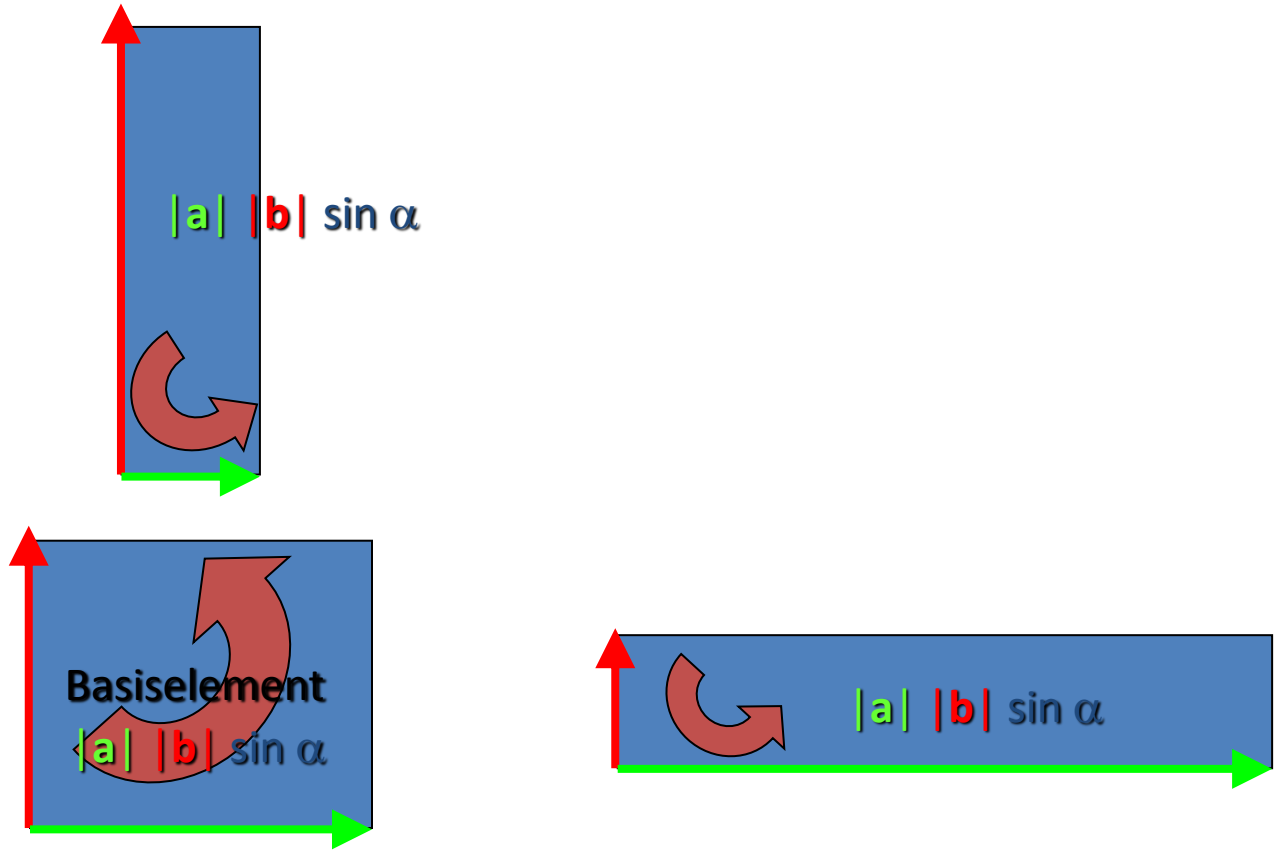
Constructing Bivectors

Equivalence relations → No unique reconstruction of generating vectors possible



Equivalent Bivectors

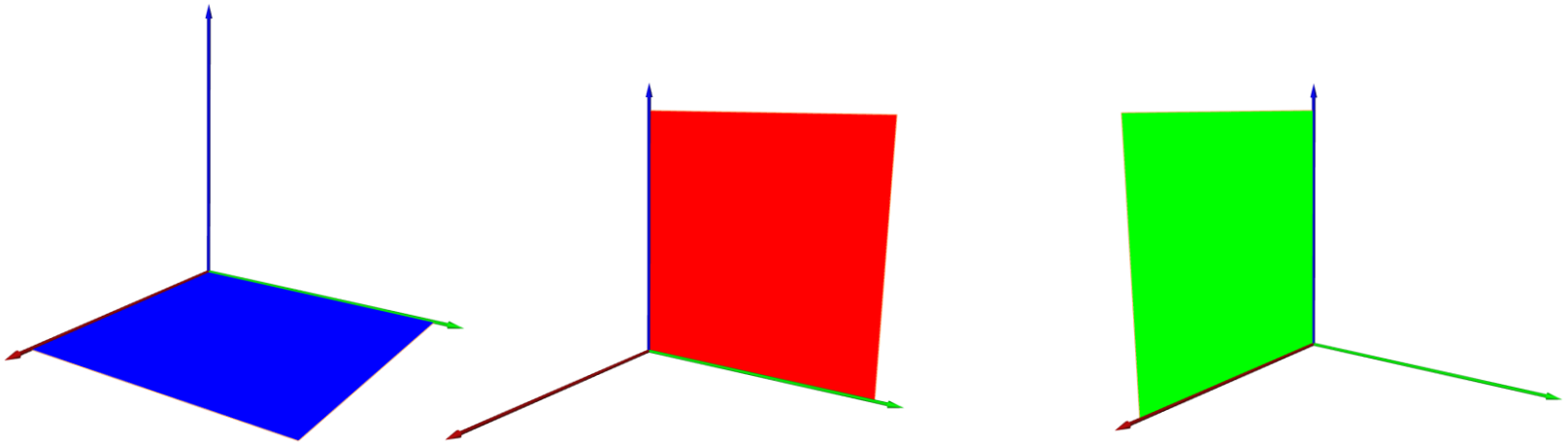
All bivectors of equal area $|a|\ |b| \sin \alpha$ and orientation are equivalent



Basis Bivectors in R^3

- 3 basis-elements ($e_\mu \approx \partial_\mu$)

$e_x \wedge e_y$, $e_y \wedge e_z$, $e_z \wedge e_x$



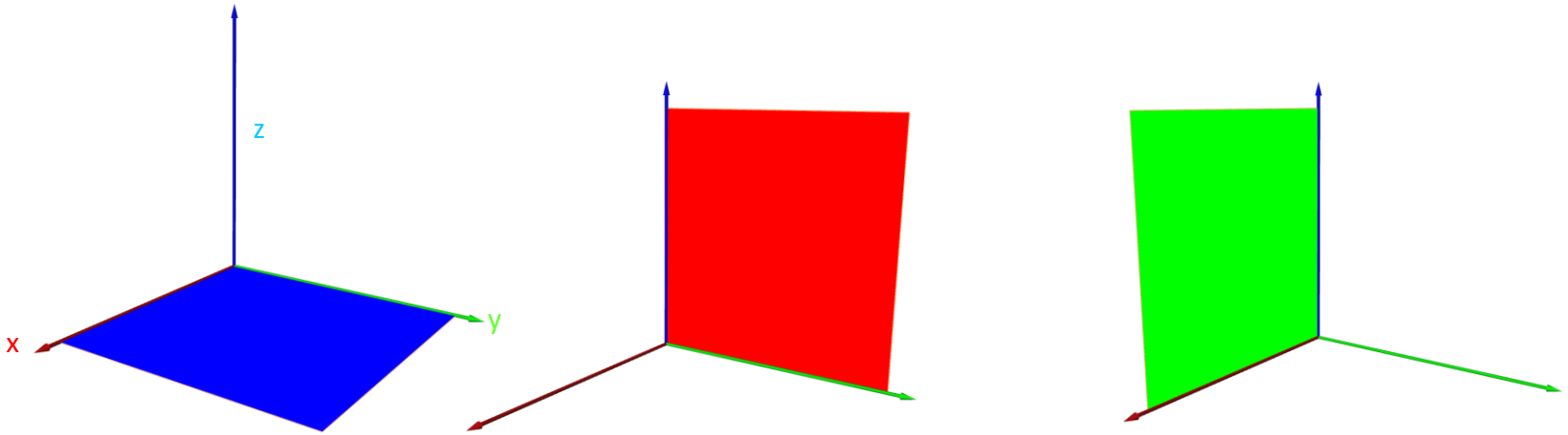
- Generalization: $e_x \wedge e_y \wedge e_z$ is a (oriented) *volume*

Basis Bivectors in R^3

- 3 basis-elements ($e_\mu \approx \partial_\mu$)

$e_x \wedge e_y$, $e_y \wedge e_z$, $e_z \wedge e_x$

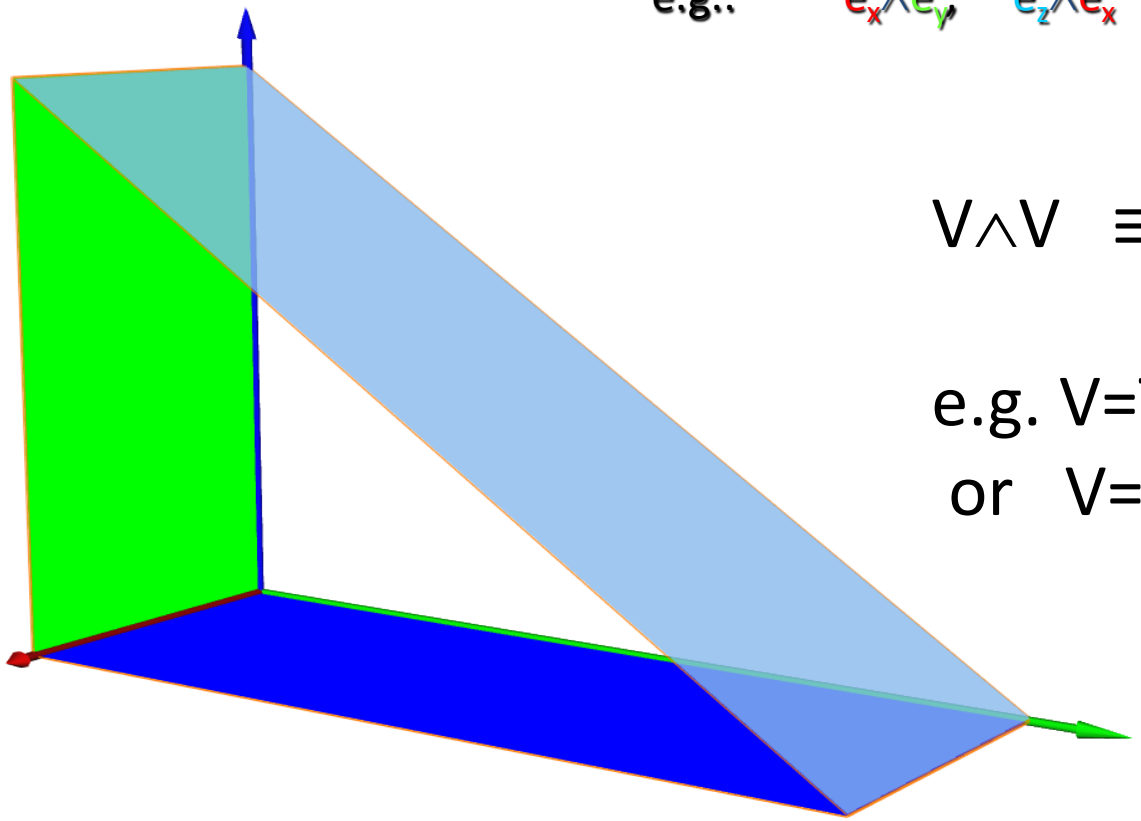
```
struct Bivector
{
        float xy, yz, zx;
};
```



Vectorspace of bi-vectors

Linear combinations possible

e.g.: $e_x \wedge e_y$, $e_z \wedge e_x$



$V \wedge V \equiv \Lambda^2(V)$

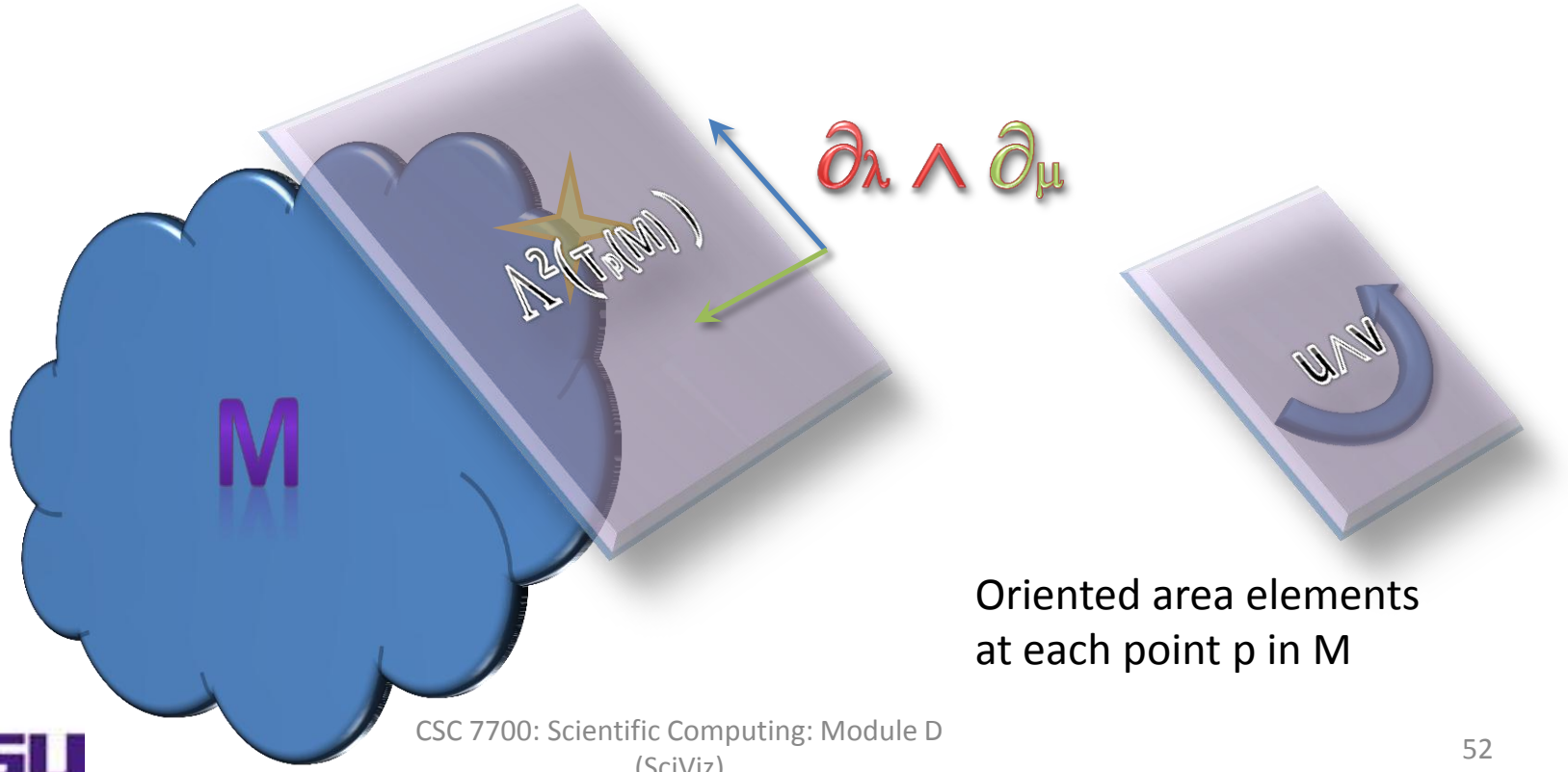
e.g. $V=T_p$

or $V=T^*_p$

Grassmann Algebra on $T_p(M)$

- $\wedge: T_p(M) \times T_p(M) \rightarrow \Lambda^2(T_p(M))$

  $u, v \quad \Rightarrow u \wedge v \quad$ where $\quad u \wedge v = -v \wedge u$



Oriented area elements at each point p in M

Coordinate representation of "$\wedge$"-product in R^3

Generic Bivector $a \wedge b$:

- $A = A_{xy}\, e_x \wedge e_y + A_{yz}\, e_y \wedge e_z + A_{zx}\, e_z \wedge e_x$

$$(a_x e_x + a_y e_y + a_z e_z) \wedge (b_x e_x + b_y e_y + b_z e_z) =$$

- $a_x e_x \wedge b_x e_x + a_x e_x \wedge b_y e_y + a_x e_x \wedge b_z e_z + a_y e_y \wedge b_x e_x + a_y e_y \wedge b_y e_y + a_y e_y \wedge b_z e_z + a_z e_z \wedge b_x e_x + a_z e_z \wedge b_y e_y + a_z e_z \wedge b_z e_z =$

$$(a_x b_y - a_y b_x) e_{xy} + (a_y b_z - a_z b_y) e_{yz} + (a_x b_z - a_z b_x) e_{xz}$$

GA in Javascript

Vectors

```
function Vector(x,y,z)
{
        this.x = x;
        this.y = y;
        this.z = z;
}
```

Omitted: vector space operations “+”, “•”

```
function wedge(VectorA,VectorB)
{
        return new Bivector(
        VectorA.x*VectorB.y - VectorA.y*VectorB.x,
        VectorA.y*VectorB.z - VectorA.z*VectorB.y,
        VectorA.z*VectorB.x - VectorA.x*VectorB.z
                        );
}
```

Implements $a \wedge b$

Bivectors

```
function Bivector(xy,yz,zx)
{
        this.xy = xy;
        this.yz = yz;
        this.zx = zx;
}
```

Omitted: bi-vector space operations “+”, “•”

```
function BivectorFromVector(v)
{
        return new Bivector( v.z, v.x, v.y);
}

function VectorFromBivector(v)
{
        return new Vector( v.yz, v.zx, v.xy);
}
```

Implements the “dual” or “Hodge-star-operator” *v (is a vector only in 3D)

3D cross product: $a \times b = *(a \wedge b)$

Grassmann-Product "$\wedge$" in $\mathbb{R}^n$

Works in arbitrary dimensions

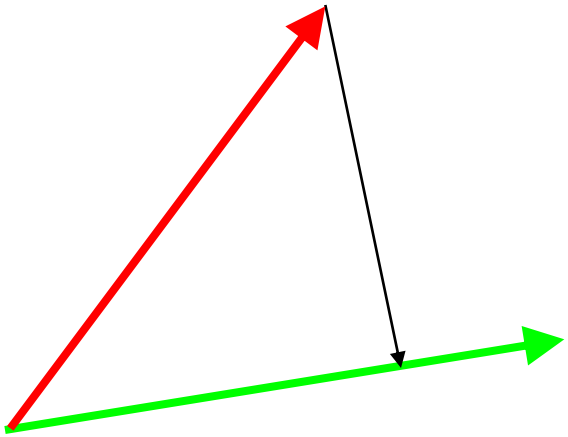
Always yields an area element defined by two directions

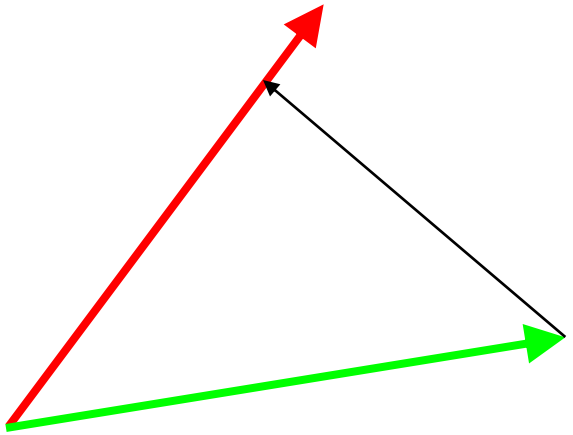
Simple rules to remember

$a \wedge b = -b \wedge a$	linear in each component	associative $(a \wedge b) \wedge c = a \wedge (b \wedge c)$

Inner product a·b

- Describes projections





$$a \cdot b = |a|\,|b| \cos\alpha = b \cdot a$$

Symmetric (commutative), requires quadratic form (Metric g:TxT→R) as additional structure, not associative (a·b)·c ≠ a·(b·c)

Comparing the products

Inner product	Exterior product
Not associative	Associative
• $(a \cdot b) \cdot c \neq a \cdot (b \cdot c)$	• $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
Commutative	Not commutative
• $a \cdot b = b \cdot a$	• $a \wedge b \neq b \wedge a$
Not invertible	*Not invertible*
Yields a scalar	*Yields a bivector*
$\lvert a \cdot b \rvert = \lvert a \rvert \lvert b \rvert \cos \alpha$	$\lvert a \wedge b \rvert = \lvert a \rvert \lvert b \rvert \sin \alpha$

1. Requirements and definition
2. Structure of the operands
3. Calculus using GP
4. Rotations using GP

GEOMETRIC PRODUCT

Requirements to GP

- For elements **A**,**B**,**C** of a vector space V with metric g(A,A):VxV→R (i.e. a symmetric bi-linear function defining the inner product) we require:

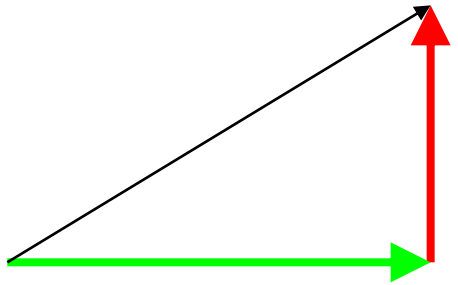
```
function VectorNorm(v)
{
        return Math.sqrt( v.x*v.x + v.y*v.y + v.z*v.z );
}
```

1. Associative: (**AB**)**C** = **A**(**BC**)
2. Left-distributive: **A**(**B**+**C**) = **AB**+**AC**
3. Right-distributive: (**B**+**C**)**A**= **BA**+**CA**
4. Scalar product: $\mathbf{A}^2 = g(\mathbf{A},\mathbf{A}) = |\mathbf{A}|^2$

Study Properties of the GP

- Right-angled triangle

$$|\mathbf{a}+\mathbf{b}|^2 = |\mathbf{a}|^2+|\mathbf{b}|^2$$



$$(A+B)(A+B) = AA+BA+AB+BB = A^2 + B^2$$

$\rightarrow AB = -BA$ for $A \cdot B = 0$ anti-symm if orthogonal

- However: not purely anti-symmetric

$$|AB|^2 = |A|^2 |B|^2 \text{ for } A \wedge B = 0 \text{ (i.e. } A,B \text{ parallel: } B=\alpha A)$$

Geometric Product

- William Kingdon Clifford (1845-79):
  - Combine inner and outer product to defined the geometric product AB (1878):

$$AB := A \cdot B + A \wedge B$$

  - Result is not a vector, but the sum of a scalar + bivector!
  - Operates on “multivectors”
  - **Geometric Product is invertible!**

Summing Scalars and Vectors

- Can't do that in vector algebra
- Need to treat scalar and vector component separately
- Leads to higher-dimensional objects
- Scalar + 3D Vector yields four-dimensional objects in 3D
- Forms a four-dimensional vector space
- Components are added and scalar-multiplied separatedly
- Space of "Multivectors"

JAVA SCRIPT

```
function MultiVectorScalarPlusVector(scalar, vector)
{
        this.scalar   = scalar;
        this.vector = vector;
}
```

C++

```
struct MultiVectorScalarPlusVector
{
        float                scalar;
        TangentialVector   vector;
};
```

Constructing Multivectors

- Can also add vectors + bivectors
  - Adding two three-dimensional objects
  - Result is six-dimensional

C++

```
struct MultiVectorVectorPlusBivector
{
        TangentialVector    vector;
        Bivector            bivector;
};
```

Represented by three numbers

Represented by three numbers

- Most general Multivector: add scalar, vector, bivector, trivector

Multi-vector components

- R^2: $A = A_0 + A_1\, e_0 + A_2\, e_1 + A_3\, e_0 \wedge e_1$

- R^3: $A =$

$$A_0$$

$$+$$

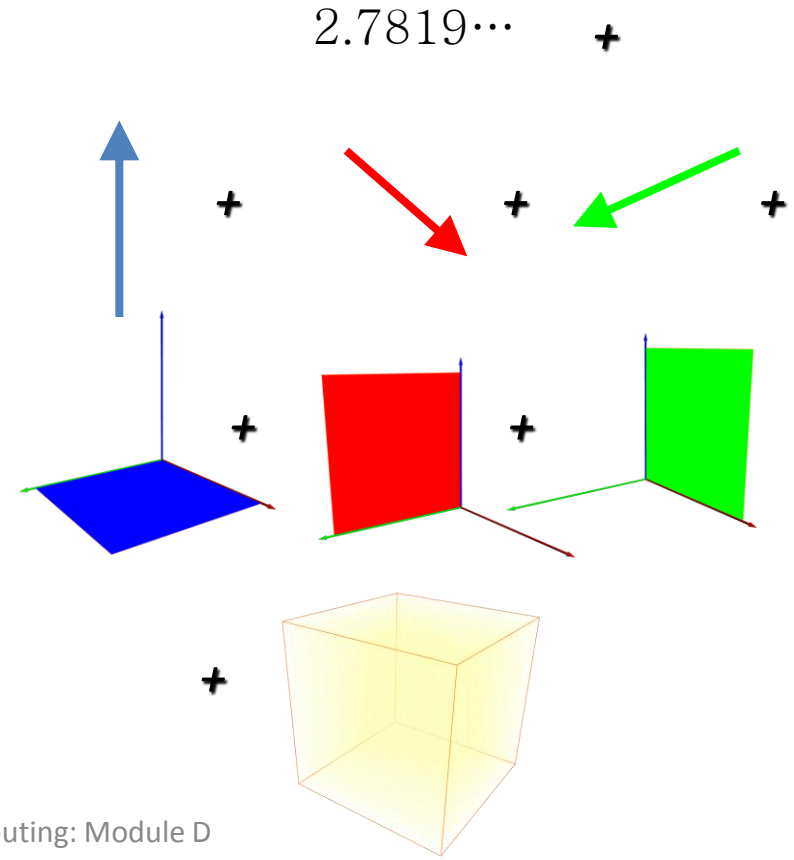
$$A_1\, e_0 + A_2\, e_1 + A_3\, e_2$$

$$+$$

$$A_4\, e_0 \wedge e_1 + A_5\, e_1 \wedge e_2 + A_6\, e_0 \wedge e_2$$

$$+$$

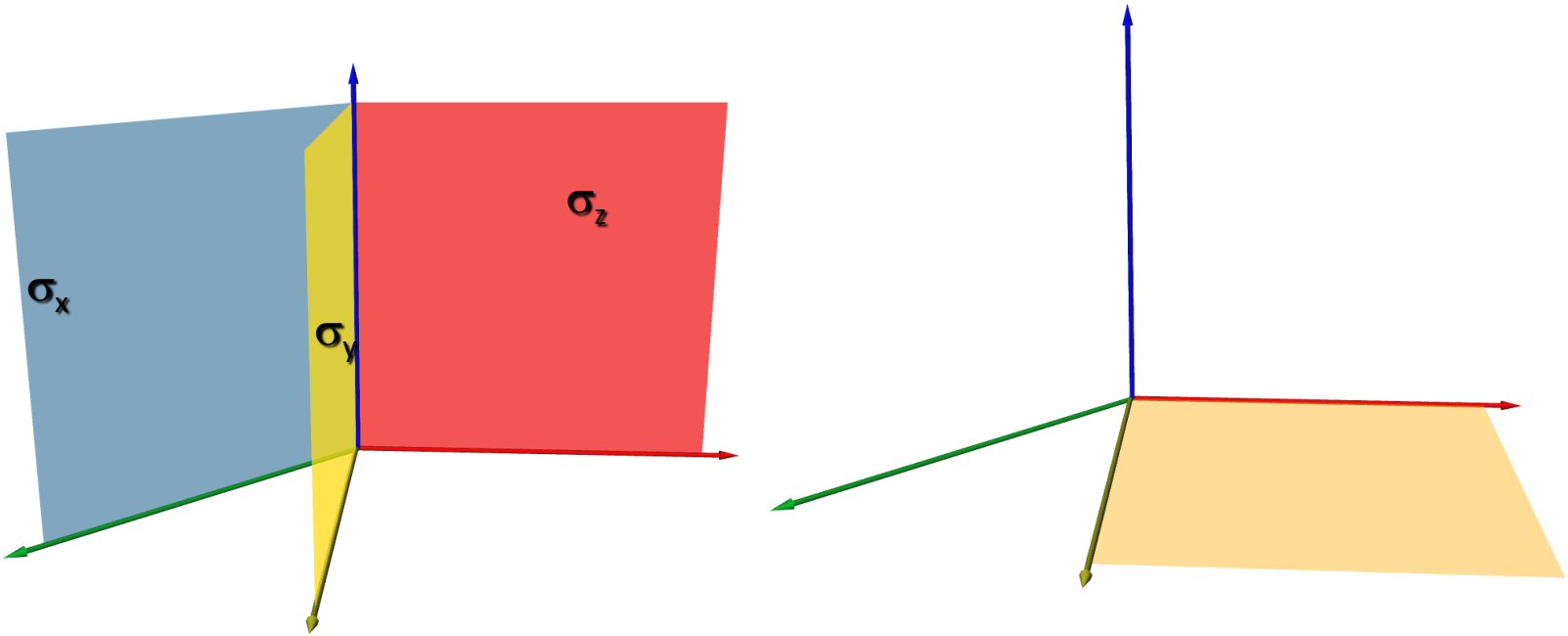
$$A_7\, e_0 \wedge e_1 \wedge e_2$$



4D: six-basis bi-vectors ("spacetime-algebra", STA)

Three bi-vectors with time-component: xt, yt, zt

Three bi-vectors without time-component: xy, yz, zx



Structure of Multi-vectors

Linear combination of anti-symmetric basis elements

2^n components

0D	1 Scalar
1D	1 Scalar, 1 Vector
2D	1 Scalar, 2 Vectors, 1 Bivector
3D	1 Scalar, 3 Vectors, 3 Bivectors, 1 Volume
4D	1 Scalar, 4 Vectors, 6 Bivectors, 4 Volume, 1 Hyper-volume
5D	…

Inner/Exterior/Cross Product via GP

- Given vectors a,b:

$a \cdot b = \frac{1}{2}(ab + ba)$ symmetric part

$a \wedge b = \frac{1}{2}(ab - ba)$ anti-symmetric part

$a \times b = -(a \wedge b) \cdot (e_x \wedge e_y \wedge e_z)$ Dual/Hodge-star in 3D

→We don't really need anything else than the GP

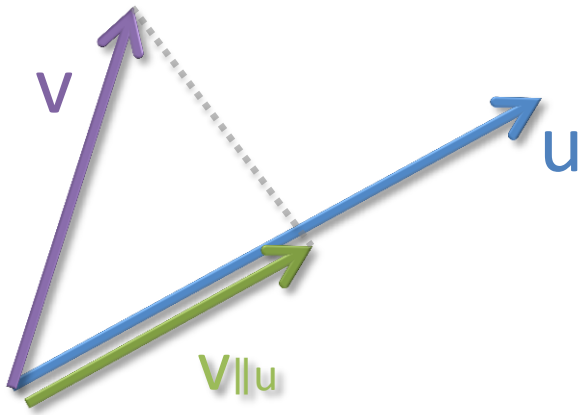
→GP has nice properties like associativity and inversibility

What is the inverse of a vector?

- Given a vector v, what multiplied with v gives the scalar 1?
- Simple answer: $v/|v|^2 =: v^{-1}$
- Then: $v\, v^{-1} = 1$
- With arbitrary vector u: "u/v" = $uv^{-1} = uv/|v|^2$
- Where $(uv^{-1})v = u(v^{-1}v) = u\ 1 = u$ (NOT possible with just inner product)
- BUT: uv^{-1} is not a vector! It is a scalar + bivector!
- BUT: $uv^{-1} \neq v^{-1}u$ in general (not commutative!)

Vector calculus in GA: Projections

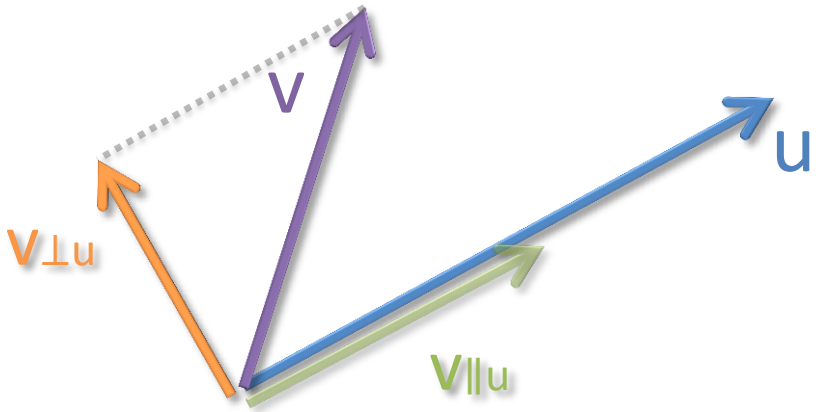
- Given a vector v and a unit vector u:
- Vector v projected onto u given by $v_{\parallel u} = (v \cdot u)\, u$
- If u not unit vector, then $v_{\parallel u} = (v \cdot u)\, u/u^2 = (v \cdot u)\, u^{-1}$
- Write as GP: $v_{\parallel u} = \frac{1}{2}(vu + uv)\, u/u^2 = \frac{1}{2}(vu + uv)\, u^{-1}$



Vector calculus in GA: Rejection

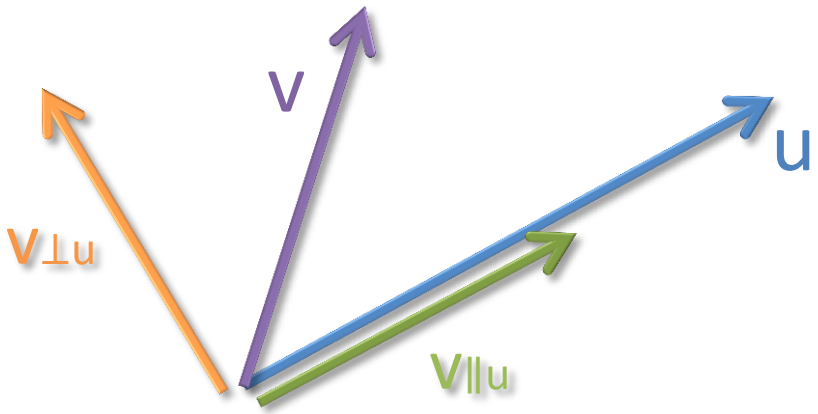
- Orthogonal component: $v_{\perp u}$
- Have $v = v_{\perp u} + v_{\parallel u}$, with $v_{\parallel} = \frac{1}{2}(vu + uv)\, u^{-1}$
- → $v_{\perp u} = v - v_{\parallel u} = v(uu^{-1}) - \frac{1}{2}(vu + uv)\, u^{-1} =$

  $(vu - \frac{1}{2}vu - \frac{1}{2}uv)\, u^{-1} = \frac{1}{2}(vu - uv)\, u^{-1} = (v \wedge u)\, u^{-1}$
- In 3D vector calculus: $v_{\perp u} = v \times (u \times v) / v^2$



Projection & Rejection

- Projection with inner product
- $u_{\parallel v} = \frac{1}{2}(uv+vu)v/v^2$
  $= (u \cdot v)v^{-1}$
- Rejection with outer product
- $u_{\perp v} = \frac{1}{2}(uv-vu)v/v^2$
  $= (u \wedge v)v^{-1}$



Reflection at a Vector

- Unit vector u, arbitrary vector v

  projection&rejection: $v_{\parallel}=(v \cdot u)\, u$ & $v_{\perp u}=v - v_{\parallel u}$

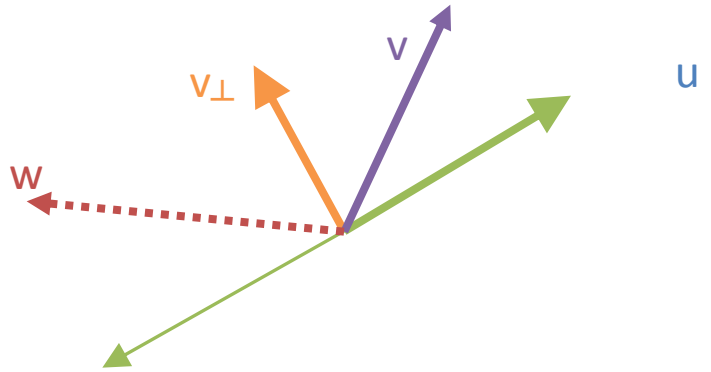
  reflected vector $w = v_{\perp} - v_{\parallel} = v - 2v_{\parallel}$

  thus $w = v - 2(v \cdot u)\, u$

  with GP $w = v - 2[\frac{1}{2}(vu+uv)]\, u = v - vuu - uvu$

  →

$$w = -uvu$$



Rotations

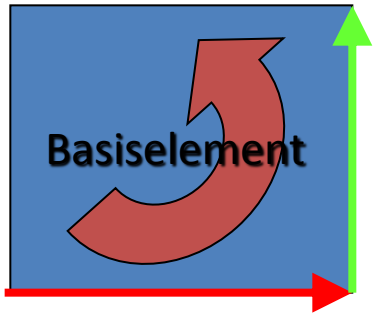
1. Identification with Quaternions
2. Rotation in 2D
3. Rotation in nD
4. Rotation of arbitrary Multivectors in nD

Geometrical Quadrate

Consider $(AB)^2$ of Bivector-basis element where

$|A|=1,\ |B|=1,\ A \cdot B = 0$

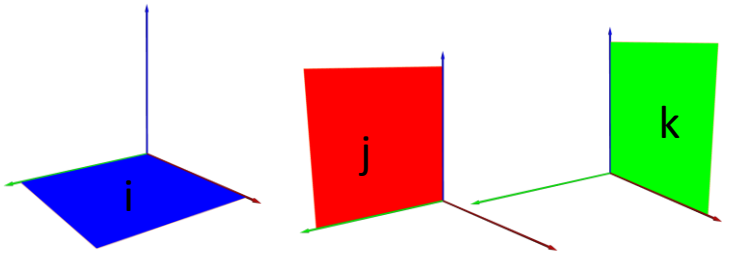
$\rightarrow AB = A \wedge B = -BA$



$(AB)^2 = (AB)(AB) = -(AB)(BA) = -A(BB)A = \underline{\mathbf{-1}}$

Equivalent to "Quaternions" in 3D

- 2D: complex numbers
  - $i := e_x e_y$, $i^2 = -1$
- 3D: quaternions
  - $i := e_x \wedge e_y = e_x e_y$, $j := e_y \wedge e_z = e_y e_z$, $k := e_x \wedge e_z = e_x e_z$
  - $i^2 = -1$, $j^2 = -1$, $k^2 = -1$
  - $ijk = (e_x e_y)(e_y e_z)(e_x e_z) = -1$

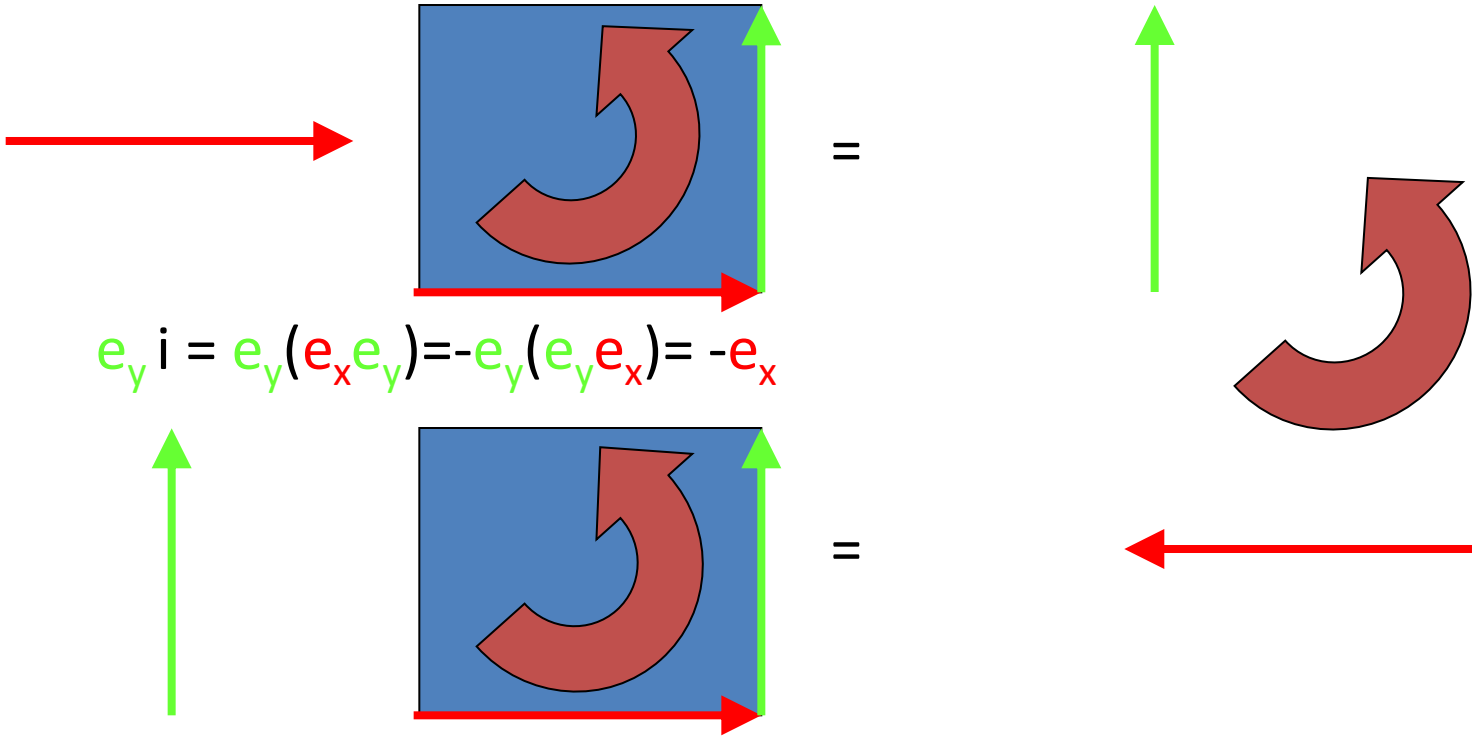


- 4D: Biquaternions (complex quaternions, spacetime algebra)

Rotation and GA

Right-multiplication of Vectors by Bivectors

$e_x i = e_x (e_x e_y) = (e_x e_x) e_y = e_y$



$e_y i = e_y (e_x e_y) = -e_y (e_y e_x) = -e_x$

Meaning of a Bi-Vector

Bivector can be understood as the operation that rotates a vector by 90 degrees in the plane defined by the bi-vector!

Interpretation works in arbitrary dimensions!

Generic Rotation in 2D

- Multiple Rotation

  $e_x\, i\, i = (e_x\, i)\, i = e_y\, i = -e_x = -1\, e_x$

- Arbitrary vector

  $A = A_x\, e_x + A_y\, e_y$

  $A\, i = A_x\, e_x\, i + A_y\, e_y\, i = A_x\, e_y - A_y\, e_x$

- Rotation by arbitrary angle:

  $\rightarrow A \cos\varphi + A\, i \sin\varphi \equiv$ "$A\, e^{i\varphi}$"

  rotates vector A by angle φ in plane i

  Inverse rotation: $Ai = -iA$: $\varphi \rightarrow -\varphi$

  $\rightarrow A\, e^{i\varphi} = e^{-i\varphi}\, A$

Rotor in 2D

- Rotor: sum of scalar + bivector

  $R := e^{\varphi i} = \cos\varphi + i \sin\varphi$ where $i^2 = -1$

  $A\, e^{i\varphi} = e^{-i\varphi} A = e^{-i\varphi/2} A\, e^{\varphi i/2} = R A R^{-1}$

  With $R=e^{-i\varphi/2}$ "Rotor"

  $R^{-1}=e^{i\varphi/2}$ "inverse Rotor"

  $A R^{-2} = R^2 A = R A R^{-1}$

- Product of rotors is multiple rotation R=ABCD, R^{-1}=DCBA is "reverse" R

Rotor in nD

- Rotor in plane U, vector v:

  $R = \cos\varphi + \sin\varphi\ U \qquad U^2 = -1$

  Expect: Rv or vR^{-1} or $R\,v\,R^{-1}$

- Problem: With arbitrary vector v there would be a tri-vector component if we use one-side multiplication

$$Rv = v\cos\varphi + \sin\varphi\ (U \cdot v + U \wedge v)$$

iff $U \wedge v \neq 0$ (v not coplanar with U)

Rotation in nD

Consider: $R\, v\, R^{-1}$ where $v = v_\perp + v_\parallel$:

– We have: $U \cdot v_\perp = 0$ thus $U v_\perp = U \wedge v_\perp$

$= u_1 \wedge u_2 \wedge v_\perp = - u_1 \wedge v_\perp \wedge u_2 = v_\perp \wedge u_1 \wedge u_2 = v_\perp \wedge U = v_\perp U$

i.e. $v_\perp$ commutes with U, thus also R

$R\, v\, R^{-1} = R\, v_\perp\, R^{-1} + R\, v_\parallel\, R^{-1}$

$R\, v_\perp\, R^{-1} = (\cos\varphi + \sin\varphi\ U)\, v_\perp\, (\cos\varphi - \sin\varphi\ U)$

$= v_\perp(\cos^2\varphi - \sin^2\varphi\ U^2) = v_\perp$

$$R\, v\, R^{-1} = v_\perp + e^{\varphi U}\, v_\parallel\, e^{-\varphi U} = v_\perp + v_\parallel\, e^{-2\varphi U}$$

Summary: Rotor

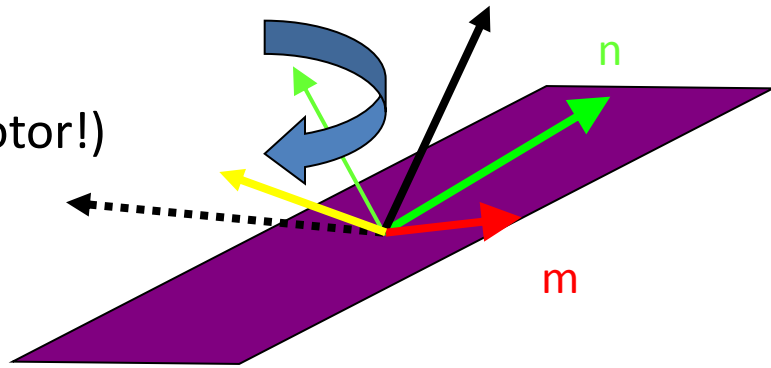
- Given a unit bivector U and an angle φ
- Rotor $R := \cos\varphi + \sin\varphi\ U = e^{-\varphi U}$
- The operation

$$RvR^{-1}$$

- Rotates a vector v by 2φ by in the plane U
- In arbitrary dimensions
- Using four numbers in 3D (rotation matrix: 9)

Rotation as multiple reflection

- Alternative Interpretation:
  - Reflect vector v by vector n, then by vector m:
    - v → - nvn → m nvn m = mn v nm
    - Operation mn is Scalar+Bivector (Rotor!)
    - Rotor: R = mn
    - Inverse Rotor: $R^{-1} = nm$



- Theorem:

Rotation is consecutive reflection on two corresponding vectors with the rotation angle equal to twice the angle between these vectors.

GA in JavaScript

https://svn.cct.lsu.edu/repos/sci-comp/public/Module-D

https://svn.cct.lsu.edu/repos/sci-comp/public/Module-D/GeometricAlgebra.js (182 lines)

```
<HEAD><TITLE>GA in Java ScriptL</TITLE>
<script src="GeometricAlgebra.js"></script>


<script type="text/javascript">
function main()
{
var        X = new Vector(1,0,0);
var        Y = new Vector(0,1,0);

var        PlaneOfRotation = wedge(X,Y);

var        v = new Vector(1,0,1);

document.write( "Plane of rotation = " + PlaneOfRotation.xy + "(XY)  " +
PlaneOfRotation.zx + "(ZX)  " + PlaneOfRotation.yz  + "(YZ)<BR>" );
document.write( "Test vector = " + v.x + " x  " + v.y + " y " + v.z  + " z<P><HR>" );
```

$$R = e^{\varphi(x \wedge y)}$$

$$v \rightarrow RvR^{-1}$$

```
var        phi;
           for(phi = 0.0; phi<= 360; phi += 45)
           {
           var        MyRotor = GAexp(phi*3.141952/180.0, PlaneOfRotation);
           var        rotV = rotate(v, MyRotor);

                      document.write( "phi=" +  phi + "   ==> (x="
                                 + rotV.x + ", y=" + rotV.y + ", z=" + rotV.z +
                                            ") , |v|="+
                      Math.sqrt( rotV.x*rotV.x + rotV.y*rotV.y + rotV.z*rotV.z ) +
                                 " <HR>" );
           }
}
</script>
</HEAD>

<BODY>

<H2> Geometric Algebra in JavaScript </H2>
<script type="text/javascript">
main();
</script>

</BODY>
</HTML>
```

Javascript GA in Action

http://sciviz.cct.lsu.edu/projects/vish/CSC7700/ga.html

Geometric Algebra in JavaScript

Plane of rotation = 1(XY) 0(ZX) 0(YZ)

Test vector = 1 x 0 y 1 z

phi=0 ==> (x=1, y=0, z=1) , |v|=1.4142135623730951

phi=45 ==> (x=-0.00017967320413619925, y=-0.9999999838587696, z=1) , |v|=1.414213562373095

phi=90 ==> (x=-0.9999999354350795, y=0.00035934640247244933, z=1) , |v|=1.4142135623730951

phi=135 ==> (x=0.0005390195892085448, y=0.9999998547289306, z=1) , |v|=1.4142135623730951

phi=180 ==> (x=0.9999997417403262, y=-0.0007186927585425548, z=1.0000000000000002) , |v|=1.4142135623730954

phi=225 ==> (x=-0.0008983659046762904, y=-0.9999995964692693, z=1) , |v|=1.4142135623730951

phi=270 ==> (x=-0.9999994189157649, y=0.001078039021809311, z=1) , |v|=1.4142135623730951

phi=315 ==> (x=0.0012577121041397188, y=0.9999992090798189, z=1) , |v|=1.4142135623730951

phi=360 ==> (x=0.9999989669614375, y=-0.0014373851458663945, z=1) , |v|=1.414213562373095

Literature

http://modelingnts.la.asu.edu/
http://www.mrao.cam.ac.uk/~clifford

- David Hestenes: *New Foundations for Classical Mechanics (Second Edition)*. ISBN 0792355148, Kluwer Academic Publishers (1999)
- *Oersted Medal Lecture 2002: Reforming the Mathematical Language of Physics* (David Hestenes)
- *Geometric (Clifford) Algebra: a practical tool for efficient geometrical representation* (Leo Dorst, University of Amsterdam)
- *An Introduction to the Mathematics of the Space-Time Algebra* (Richard E. Harke, University of Texas)
- EUROGRAPHICS 2004 Tutorial: *Geometric Algebra and its Application to Computer Graphics* (D. Hildenbrand, D. Fontijne, C. Perwass and L. Dorst)
- *Rotating Astrophysical Systems and a Gauge Theory Approach to Gravity* (A.N. Lasenby, C.J.L. Doran, Y. Dabrowski, A.D. Challinor, Cavendish Laboratory, Cambridge), astro-ph/9707165

Optional Homework

- Implement a C++ class that follows three-dimensional Geometric Algebra. Use C++ operator overloading.