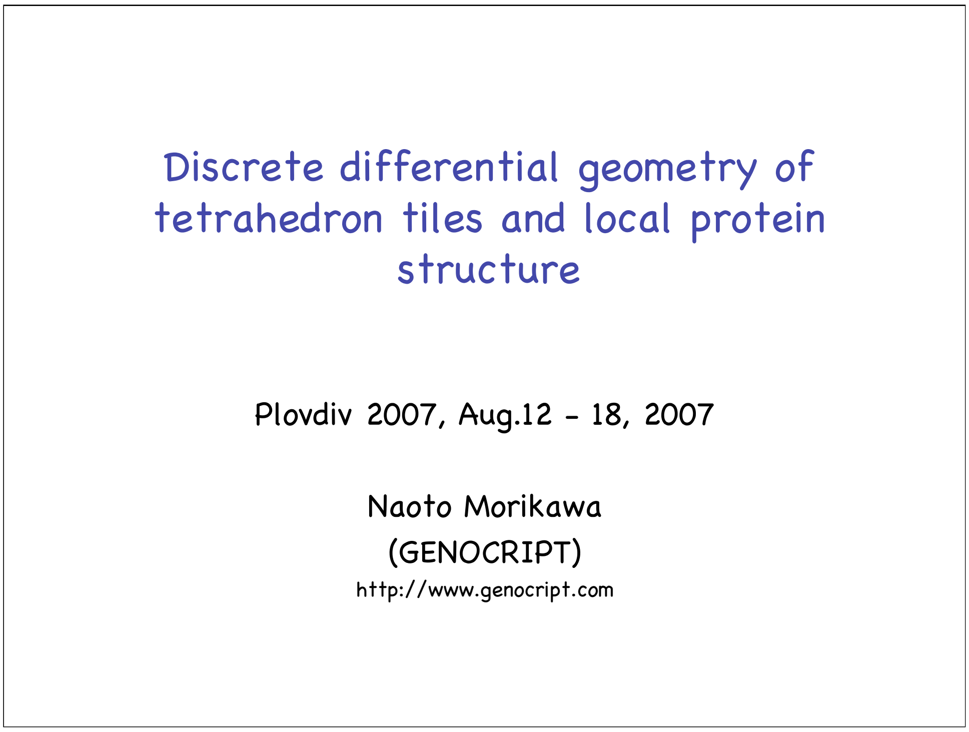
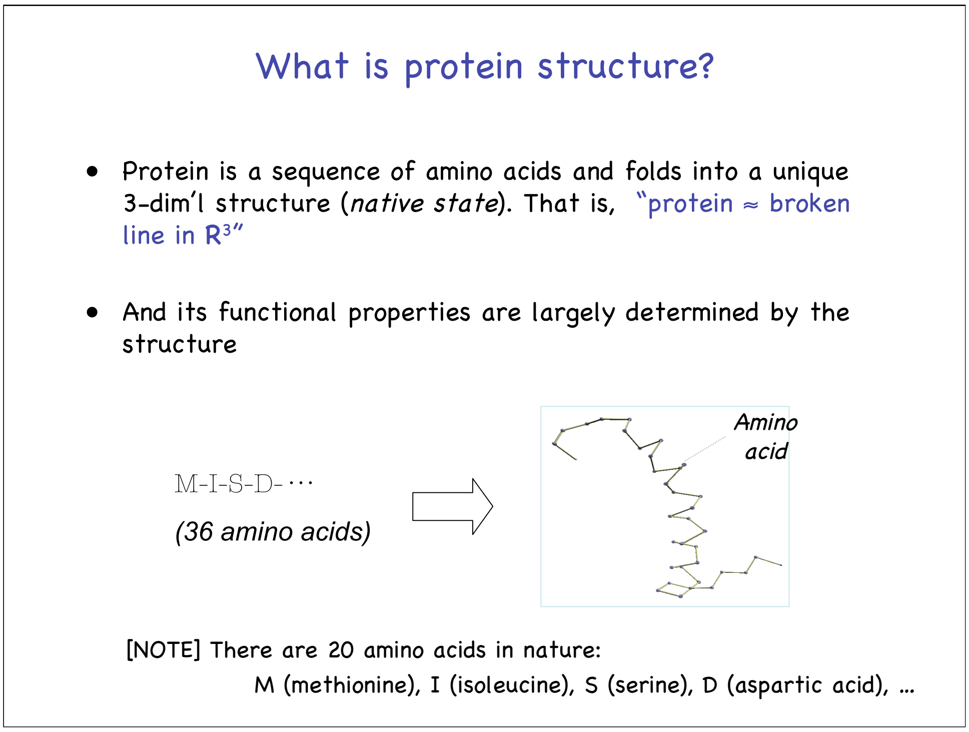


Discrete differential geometry of tetrahedron tiles and local protein structure

Plovdiv 2007, Aug.12 - 18, 2007

Naoto Morikawa
(GENOCRIPT)
http://www.genocript.com

This talk is about “discrete differential geometry of tetrahedron tiles and its application in local protein structure analysis.”



What is protein structure?

- Protein is a sequence of amino acids and folds into a unique 3-dim'l structure (*native state*). That is, "protein ≈ broken line in $\mathbf{R}^3$"
- And its functional properties are largely determined by the structure

M-I-S-D-···

(36 amino acids)

[NOTE] There are 20 amino acids in nature:
M (methionine), I (isoleucine), S (serine), D (aspartic acid), …

Before the talk, let me make a brief summary on protein structure.

Protein is a sequence of ···,

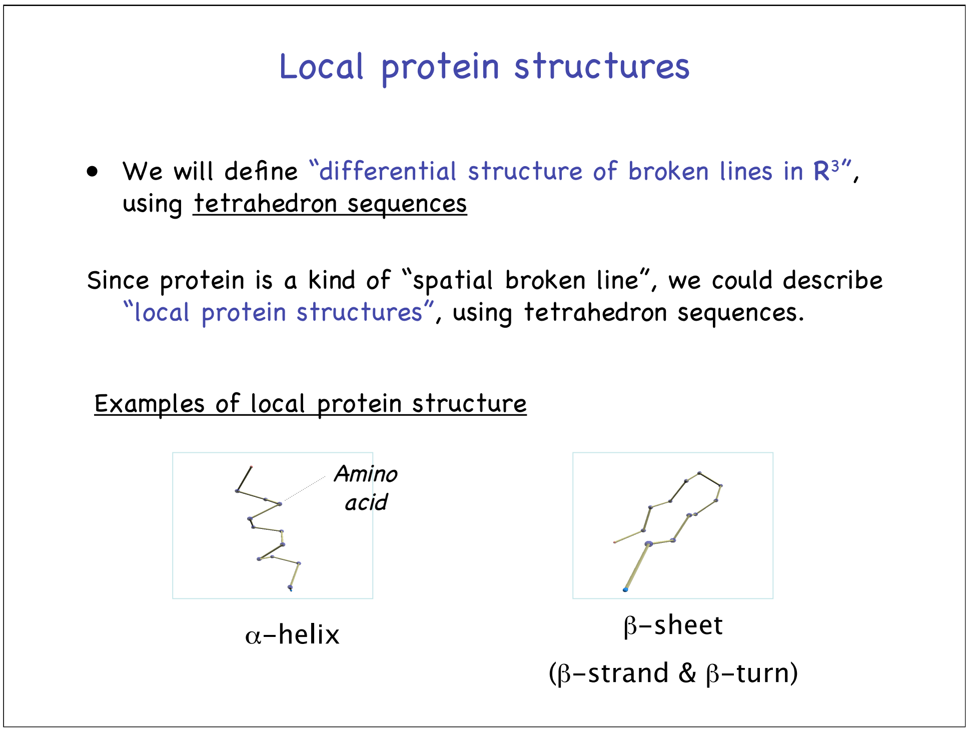
That is, one could identify proteins with broken lines in $\mathbf{R}^3$

And its functional properties are …

Shown below is an amino acid sequence and the native state of the sequence,

where each amino acid is represented by the one–letter code.

Note that ···



Local protein structures

- We will define "differential structure of broken lines in R^3", using <u>tetrahedron sequences</u>

Since protein is a kind of "spatial broken line", we could describe "local protein structures", using tetrahedron sequences.

<u>Examples of local protein structure</u>

α-helix

β-sheet

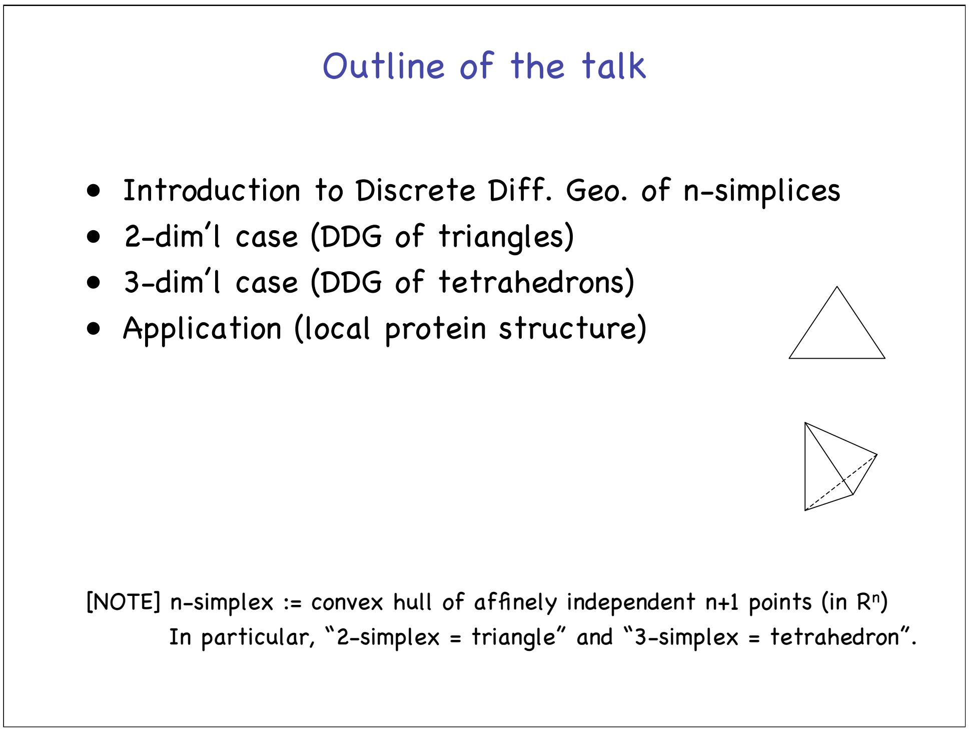
(β-strand & β-turn)

And we will define ...

Since protein is ...

Shown below are examples of local protein structures: *alpha-helix* and *beta-sheet*,

where vertices correspond to amino acids.



Outline of the talk

- Introduction to Discrete Diff. Geo. of n-simplices
- 2-dim'l case (DDG of triangles)
- 3-dim'l case (DDG of tetrahedrons)
- Application (local protein structure)

[NOTE] n-simplex := convex hull of affinely independent n+1 points (in R^n)
In particular, "2-simplex = triangle" and "3-simplex = tetrahedron".

This is the outline of the talk.

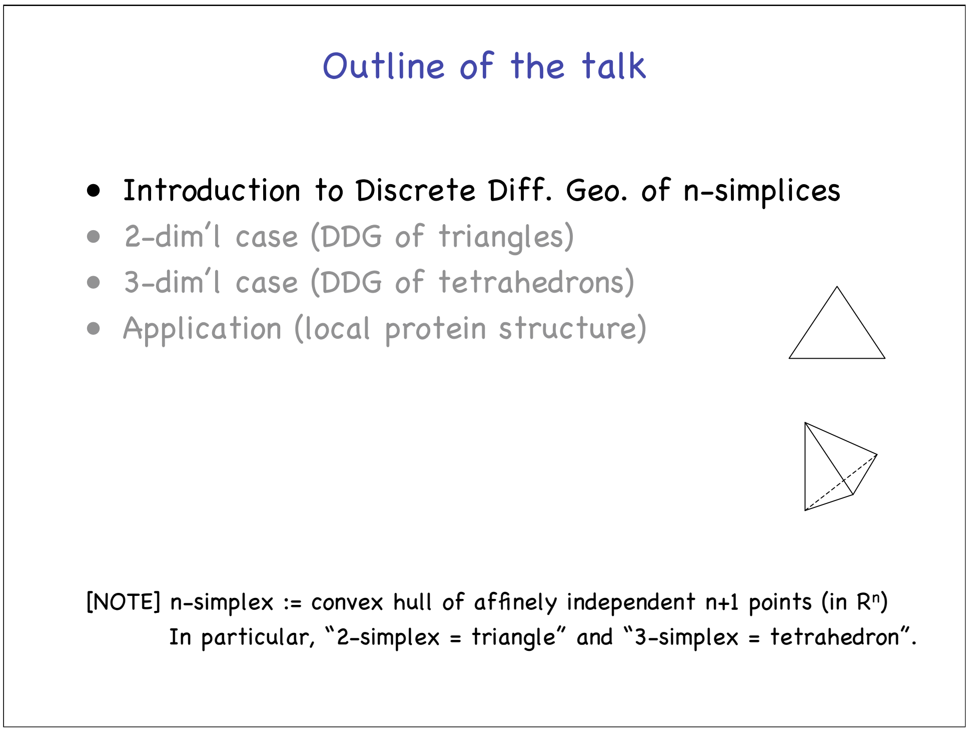
First, I'm going to make a brief introduction to …

Note that n−simplex is …

Next, I'll talk about the 2−dim'l case …

Then, I'll talk about the 3−dim'l case …

Finally, I'll describe its application in …

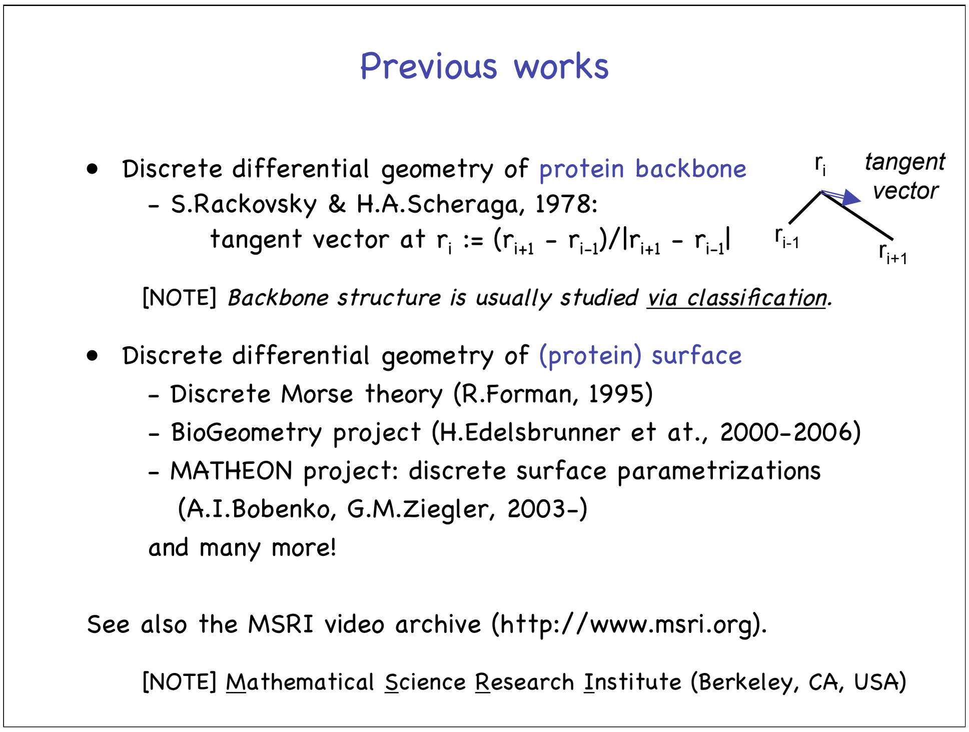


Outline of the talk

- Introduction to Discrete Diff. Geo. of n-simplices
- 2-dim'l case (DDG of triangles)
- 3-dim'l case (DDG of tetrahedrons)
- Application (local protein structure)

[NOTE] n-simplex := convex hull of affinely independent n+1 points (in R^n)
In particular, "2-simplex = triangle" and "3-simplex = tetrahedron".

Now introduction to …



Previous works

- Discrete differential geometry of protein backbone
  - S.Rackovsky & H.A.Scheraga, 1978:
    tangent vector at $r_i := (r_{i+1} - r_{i-1})/|r_{i+1} - r_{i-1}|$

r_i
tangent vector
r_{i-1}
r_{i+1}

[NOTE] *Backbone structure is usually studied* *via classification*.

- Discrete differential geometry of (protein) surface
  - Discrete Morse theory (R.Forman, 1995)
  - BioGeometry project (H.Edelsbrunner et at., 2000-2006)
  - MATHEON project: discrete surface parametrizations (A.I.Bobenko, G.M.Ziegler, 2003-)

  and many more!

See also the MSRI video archive (http://www.msri.org).

[NOTE] Mathematical Science Research Institute (Berkeley, CA, USA)

Firstly, let me review previous works on DDG and protein structure analysis.

As for DDG of protein backbone, the pioneer work of Rackovsky and Scheraga in 1978 is known.

In their method, proteins are represented as spatial broken lines and they defined *curvature* and *torsion* at each vertex.

In particular, they defined tangent vector at vertex r_i as shown on the right.

But, protein backbone structure is usually studied via classification and differential geometrical approach is not taken these days.

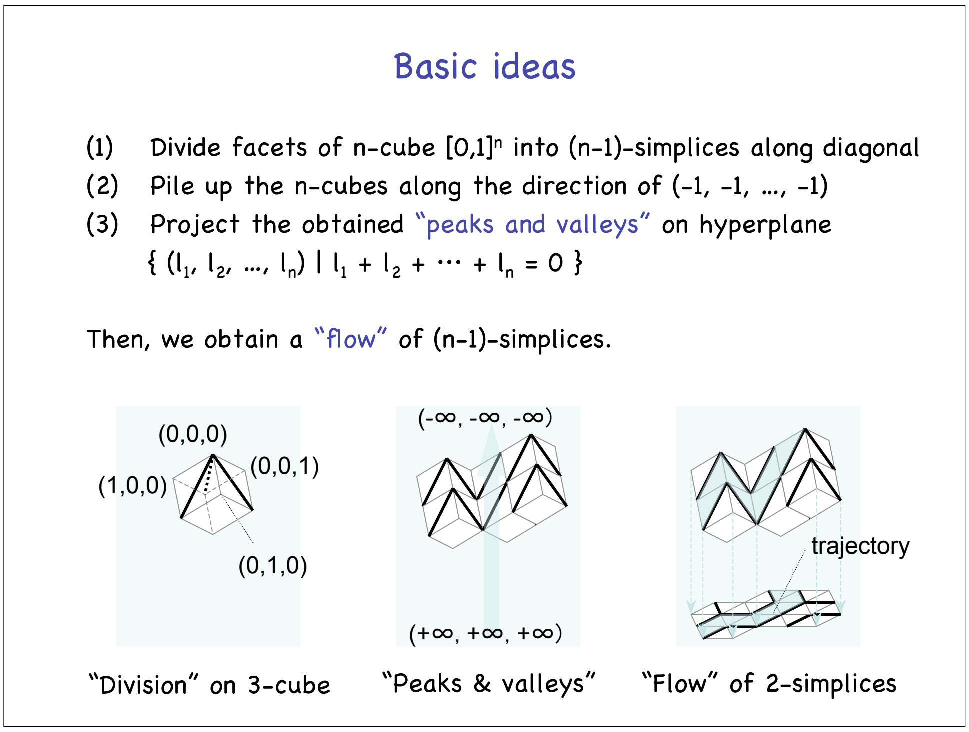
On the other hand, since proteins are a molecule, they have surface. And a bunch of works on DDG of surface are known.

To name a few, Forman proposed a discrete version of Morse theory.

And the BioGeometry project at Duke univ. and others studied geometry of protein in general.

In Germany, there is a MATHEON project on dis. surface parametrizations.

If you visit the homepage of the MSRI, you will find some video lectures on related topics.



Basic ideas

(1) Divide facets of n-cube $[0,1]^n$ into (n-1)-simplices along diagonal

(2) Pile up the n-cubes along the direction of (-1, -1, …, -1)

(3) Project the obtained "peaks and valleys" on hyperplane
$\{ (l_1, l_2, \ldots, l_n) \mid l_1 + l_2 + \cdots + l_n = 0 \}$

Then, we obtain a "flow" of (n-1)-simplices.

"Division" on 3-cube "Peaks & valleys" "Flow" of 2-simplices

This slide shows the basic ideas of my approach. Unlike previous works, n−cubes are used to construct DDG of (n−1)−simplices.

First, divide …

As for the division, I will give the definition in a minute.

Shown in the left figure is the division on 3−cube. And each facet is divided into two triangles.

Next, pile …

Shown in the middle figure is an example of 3−cubes, piled up from "+infinity" to "− infinity."

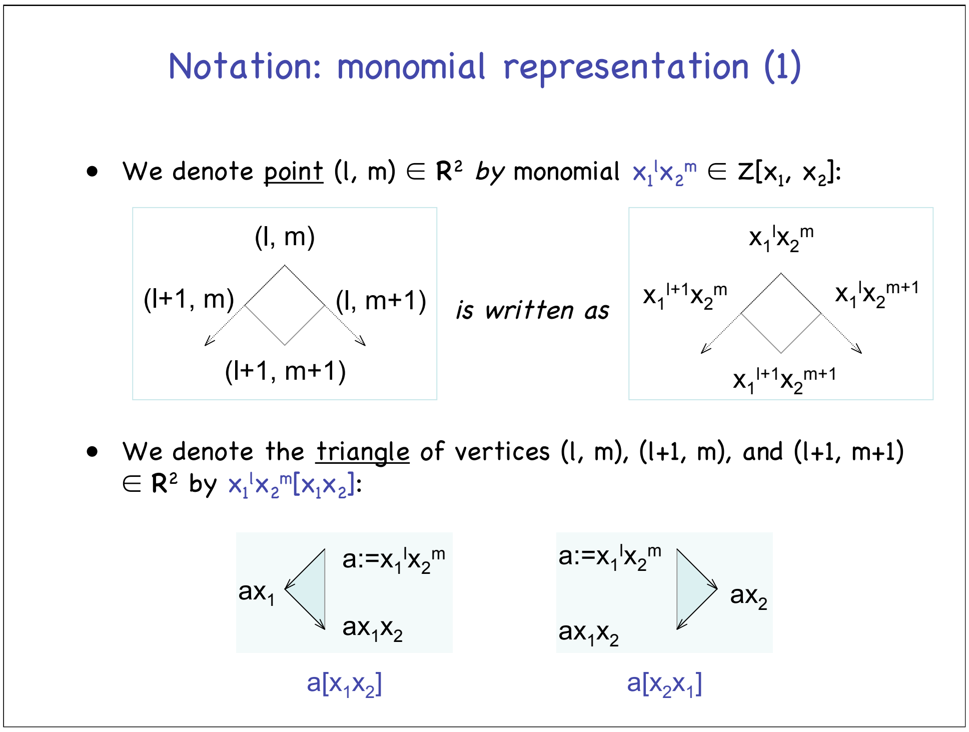
Note that the division of facets makes up a division of the surface of the obtained Ps&Vs.

Finally, project …

Then, by projecting the division of the surface onto the hyperplane, we obtain a "flow" of …

Shown in the right figure is the flow of 2−simplices obtained from the Ps&Vs.

As you see, the division of the surface of the Ps&Vs specifies trajectories of 2−simplices on the hyperplane.



Notation: monomial representation (1)

- We denote <u>point</u> (l, m) ∈ R² *by* monomial $x_1^l x_2^m$ ∈ Z[x_1, x_2]:

- We denote the <u>triangle</u> of vertices (l, m), (l+1, m), and (l+1, m+1) ∈ R² by $x_1^l x_2^m[x_1 x_2]$:

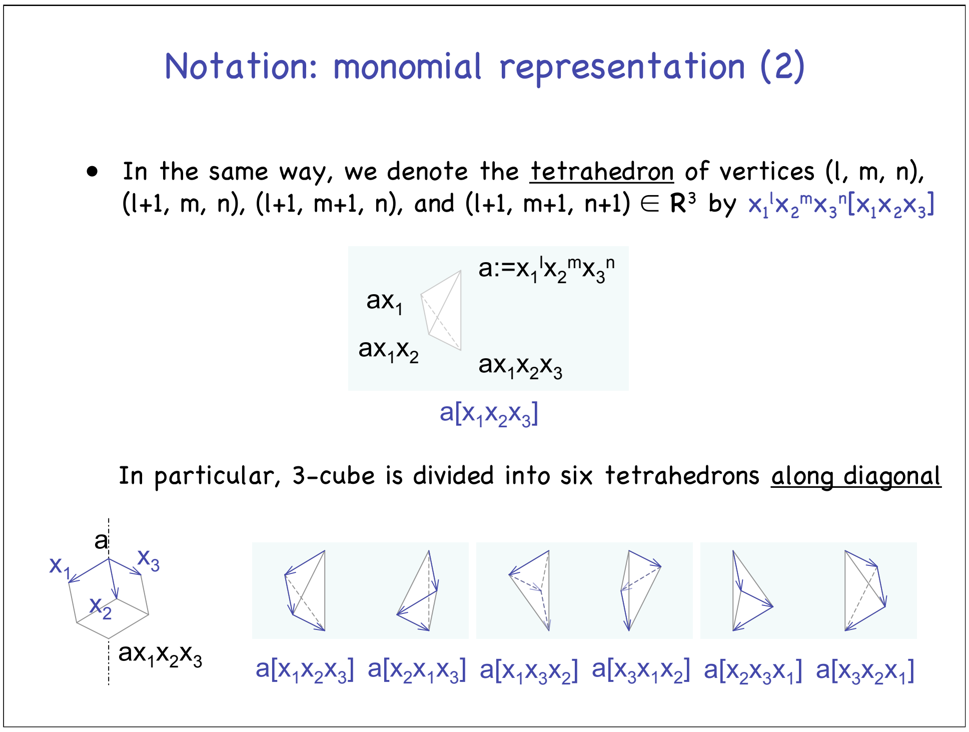
Before giving the definition of the facet division, let me explain the notation used.

In this talk, we denote point …

For example, vertices of the left square are written as shown on the right.

And we denote the triangle of …, as shown on the left.

The mirror image $a[x_2x_1]$ is also defined as shown on the right.



Notation: monomial representation (2)

- In the same way, we denote the tetrahedron of vertices (l, m, n), (l+1, m, n), (l+1, m+1, n), and (l+1, m+1, n+1) ∈ R³ by $x_1^l x_2^m x_3^n [x_1 x_2 x_3]$

$a[x_1x_2x_3]$

In particular, 3-cube is divided into six tetrahedrons along diagonal

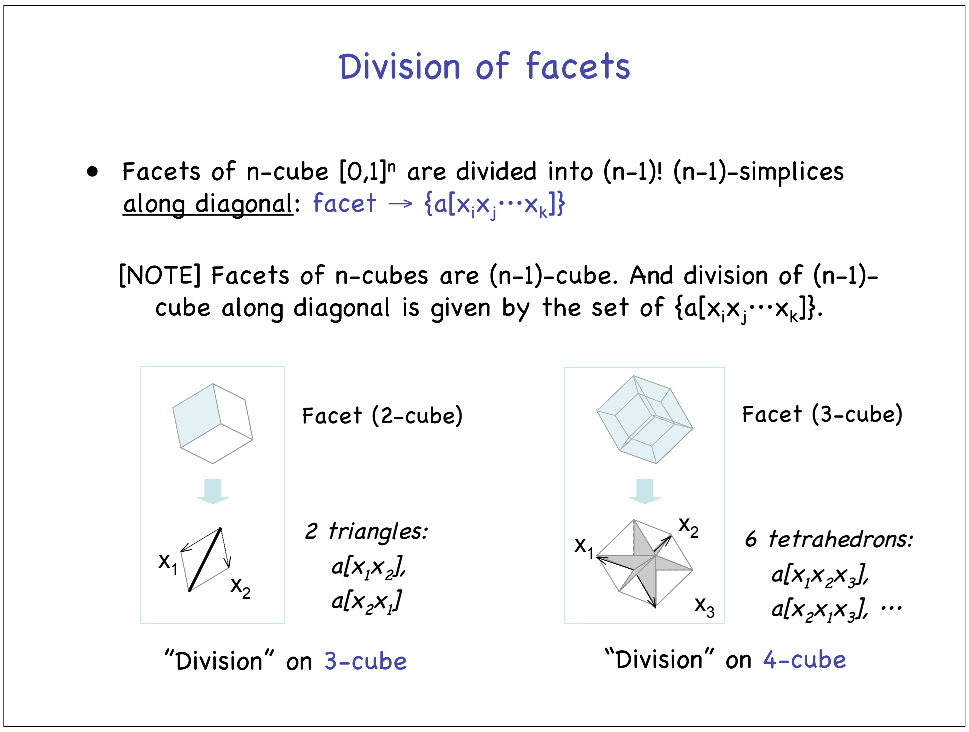
$a[x_1x_2x_3]$ $a[x_2x_1x_3]$ $a[x_1x_3x_2]$ $a[x_3x_1x_2]$ $a[x_2x_3x_1]$ $a[x_3x_2x_1]$

In the same way, we denote …

In particular, …

That is, the 3−cube shown on the left is divided into these six tetrahedrons shown on the right, where the dotted line gives the direction of diagonal.

In the next slide, we use the ″division along diagonal″ to define the division of facets.



Division of facets

- Facets of n-cube $[0,1]^n$ are divided into (n-1)! (n-1)-simplices along diagonal: facet → $\{a[x_i x_j \cdots x_k]\}$

[NOTE] Facets of n-cubes are (n-1)-cube. And division of (n-1)-cube along diagonal is given by the set of $\{a[x_i x_j \cdots x_k]\}$.

Facet (2-cube)

2 triangles:
$a[x_1 x_2]$,
$a[x_2 x_1]$

“Division” on 3-cube

Facet (3-cube)

6 tetrahedrons:
$a[x_1 x_2 x_3]$,
$a[x_2 x_1 x_3]$, …

“Division” on 4-cube

Now we could define division of facets.

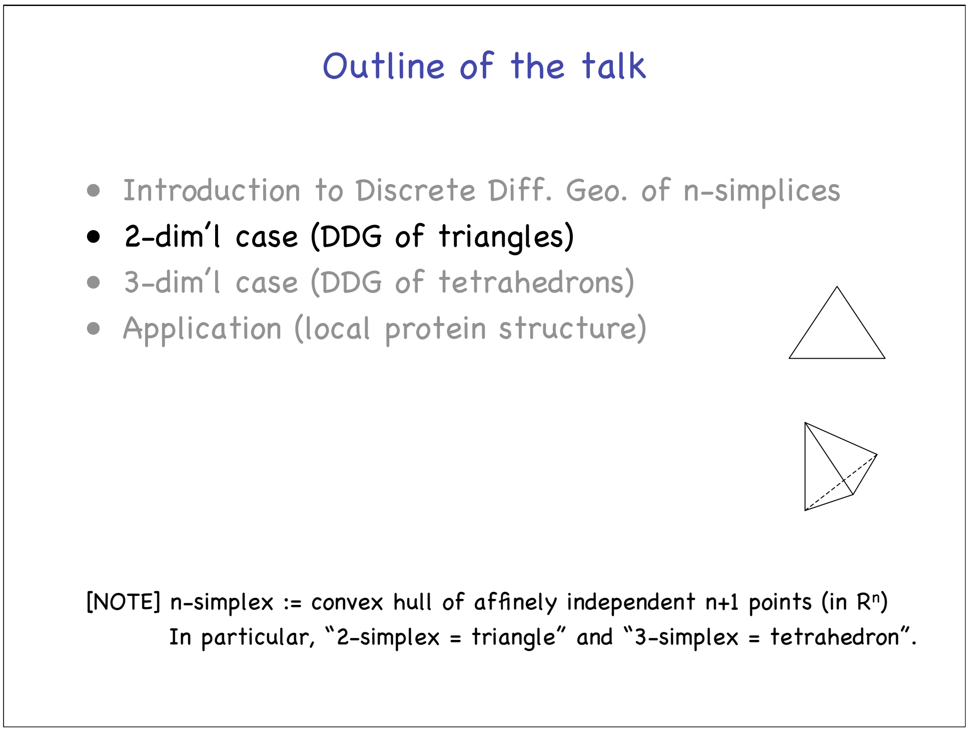
Facets of n-cube …,

where each component is given by $a[x_i x_j \ldots x_k]$ for some i, j, … , k.

Note that facets of …

Shown on the left is the division on 3-cube. As you see, a facet of 3-cube is 2-cube and it is divided into two triangles, $a[x_1 x_2]$ and $a[x_2 x_1]$.

Shown on the right is the division on 4-cube. A facet of 4-cube is 3-cube and it is divided into six tetrahedrons $a[x_1 x_2 x_3]$, …

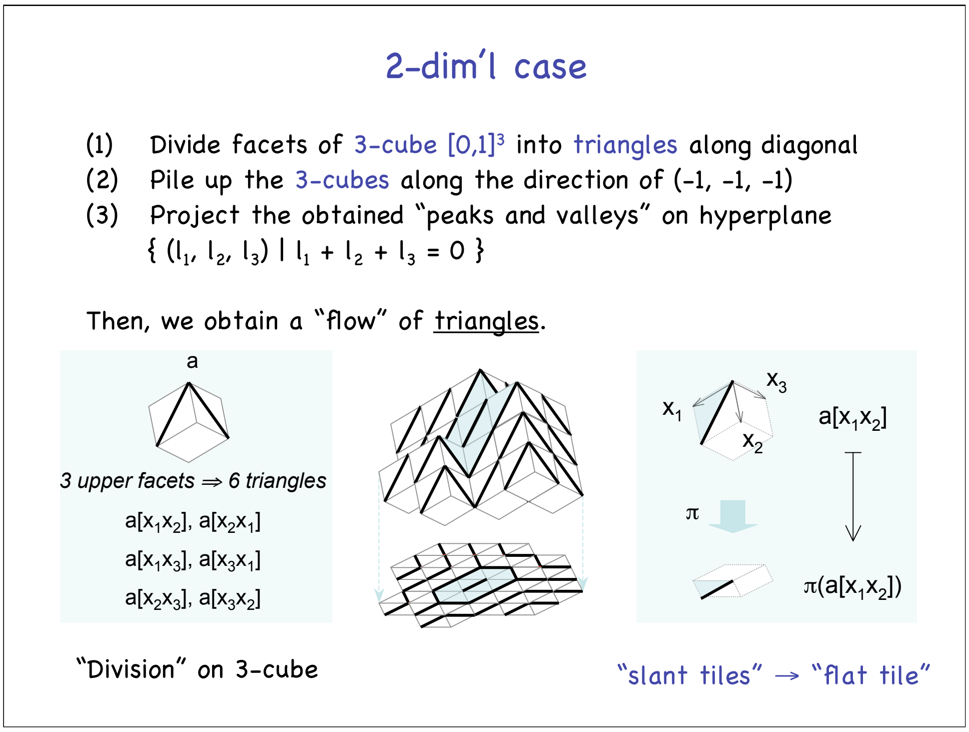


Outline of the talk

- Introduction to Discrete Diff. Geo. of n-simplices
- 2-dim'l case (DDG of triangles)
- 3-dim'l case (DDG of tetrahedrons)
- Application (local protein structure)

[NOTE] n-simplex := convex hull of affinely independent n+1 points (in R^n)
In particular, "2-simplex = triangle" and "3-simplex = tetrahedron".

Next, let me talk about the 2−dim'l case, that is DDG of triangles.



2-dim'l case

(1) Divide facets of 3-cube $[0,1]^3$ into triangles along diagonal
(2) Pile up the 3-cubes along the direction of (-1, -1, -1)
(3) Project the obtained "peaks and valleys" on hyperplane
$\{ (l_1, l_2, l_3) \mid l_1 + l_2 + l_3 = 0 \}$

Then, we obtain a "flow" of triangles.

"Division" on 3-cube

"slant tiles" → "flat tile"

These are the basic ideas of the 2−dim'l case.

First, divide …

Shown in the left figure is the division on 3−cube. Since 3−cubes are to be projected onto a hyperplane, it is enough to consider the "upper" facets. And these three upper facets are divided into 6 triangles $a[x_1x_2]$, ….

Next, pile …

Then, as shown in the middle figure, we obtain "peaks and valleys" of 3−cubes. Note that the division of facets makes up a division of the surface of the Ps&Vs.

Finally, project …

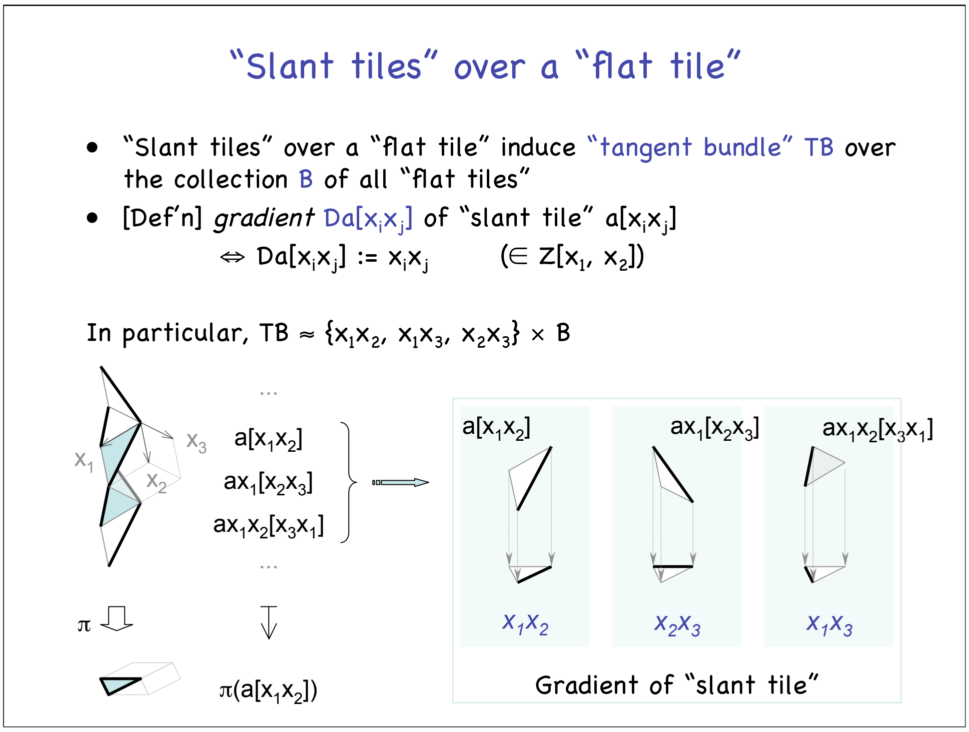
Then, we obtain a "flow" of … as shown in the middle figure.

Note that the division of the surface of the Ps&Vs specifies trajectories

For example, blue triangles in the middle figure form a closed trajectory of length 10.

Shown in the right figure is the projection of triangles.

In the next slide, we use the projection pi to contsruct DDG of triangle tiles.



"Slant tiles" over a "flat tile"

- "Slant tiles" over a "flat tile" induce "tangent bundle" TB over the collection B of all "flat tiles"
- [Def'n] *gradient* $Da[x_i x_j]$ of "slant tile" $a[x_i x_j]$

$$\Leftrightarrow Da[x_i x_j] := x_i x_j \qquad (\in Z[x_1, x_2])$$

In particular, $TB \approx \{x_1x_2, x_1x_3, x_2x_3\} \times B$

Gradient of "slant tile"

Slant tiles over …

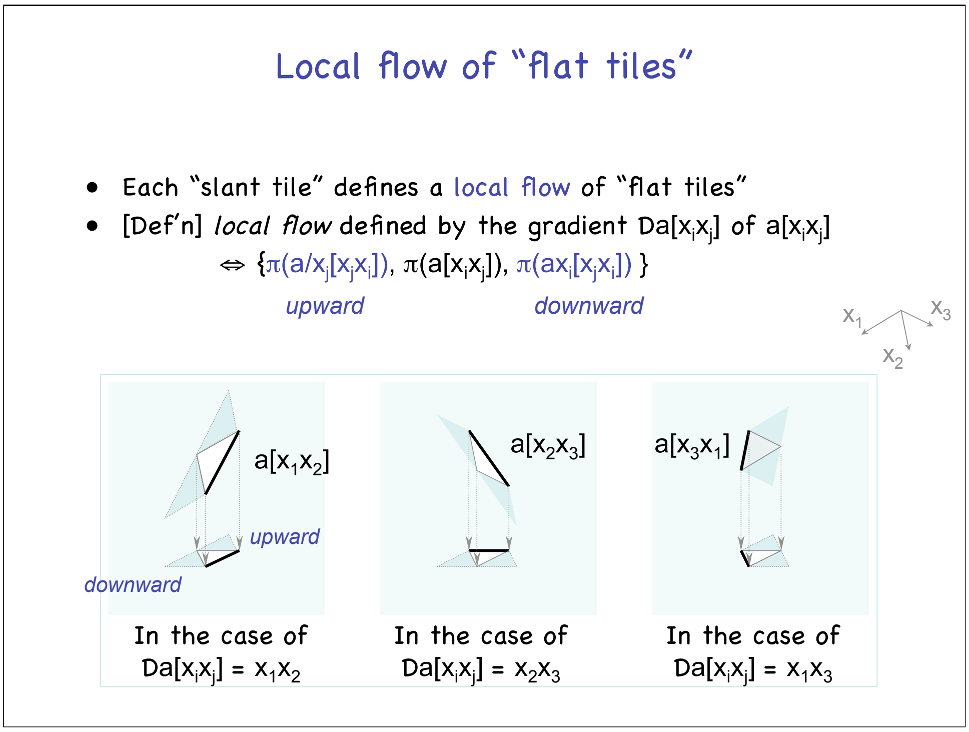
And here is the definition of gradient.

Gradient $Da[x_ix_j]$ of … is monomial x_ix_j.

In particular, tangent bundle TB over B is identified with the cartesian product of …

Shown on the left is the fiber over a flat tile. And all of these slant tiles are projected on the same flat tile.

Each of the slant tiles assumes one of the three gradients shown on the right. For example, the gradient of $a[x_1x_2]$ is x_1x_2 and so on.



Local flow of "flat tiles"

- Each "slant tile" defines a local flow of "flat tiles"
- [Def'n] *local flow* defined by the gradient $Da[x_ix_j]$ of $a[x_ix_j]$

  $\Leftrightarrow \{\pi(a/x_j[x_jx_i]), \pi(a[x_ix_j]), \pi(ax_i[x_jx_i])\ \}$

  upward *downward*

In the case of $Da[x_ix_j] = x_1x_2$

In the case of $Da[x_ix_j] = x_2x_3$

In the case of $Da[x_ix_j] = x_1x_3$

This slide shows the local flow defined by a slant tile.

Each ″slant tile″ defines …

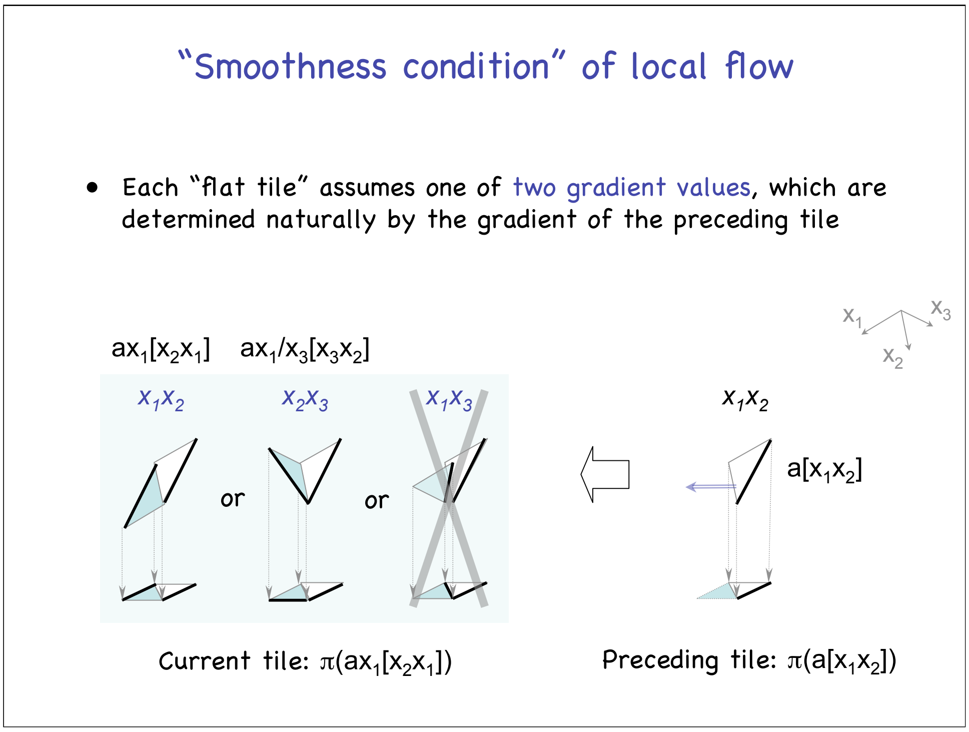
And here is the definition.

Local flow defined by … is the three consecutive flat tiles of { … }.

Shown below are the local flows around a flat tile (colored white).

For example, if the gradient of the flat tile is x_1x_2 as shown in the left figure, then the ″blue″ slant tile on this side of the ″white″ slant tile specifies the downward flat tile and the ″bule″ slant tile on the other side specifies the upward flat tile.

Similarly adjacent blue slant tiles specify the upward and downward flat tiles in the other cases.



"Smoothness condition" of local flow

- Each "flat tile" assumes one of two gradient values, which are determined naturally by the gradient of the preceding tile

Current tile: $\pi(ax_1[x_2x_1])$

Preceding tile: $\pi(a[x_1x_2])$

Next, let's consider smoothness of local flow.

That is, each "flat tile" assumes ...

Shown below is an example of the condition.

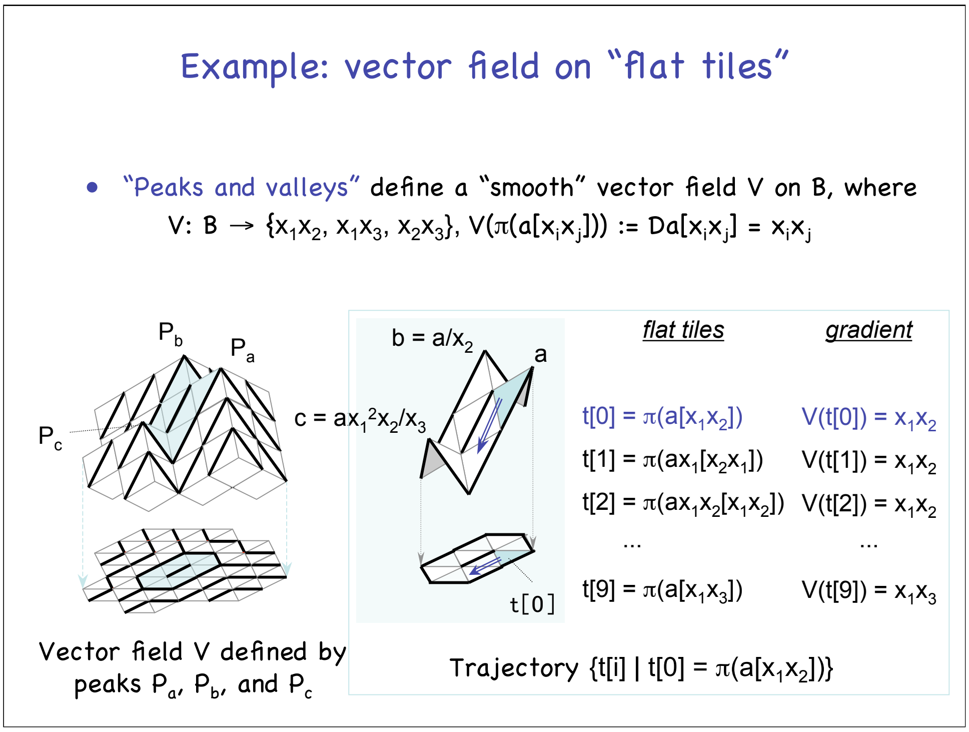
Suppose that the gradient of the preceding flat tile is x_1x_2 and the current flat tile is on the downward side as shown on the right.

Then, with each of the three gradient values, we could associate a slant tile over the current flat tile as shown on the left.

In the case of x_1x_2, the "blue" and "white" slant tiles are connected smoothly.

In the case of x_2x_3, the slant tiles are also connected smoothly.

But in the case of x_1x_3, the slant tiles are not connected smoothly. The smoothness condition means that the current flat tile assumes one of the first two values.



Example: vector field on "flat tiles"

- "Peaks and valleys" define a "smooth" vector field V on B, where $V: B \rightarrow \{x_1x_2, x_1x_3, x_2x_3\}$, $V(\pi(a[x_ix_j])) := Da[x_ix_j] = x_ix_j$

flat tiles	*gradient*
$t[0] = \pi(a[x_1x_2])$	$V(t[0]) = x_1x_2$
$t[1] = \pi(ax_1[x_2x_1])$	$V(t[1]) = x_1x_2$
$t[2] = \pi(ax_1x_2[x_1x_2])$	$V(t[2]) = x_1x_2$
…	…
$t[9] = \pi(a[x_1x_3])$	$V(t[9]) = x_1x_3$

Vector field V defined by peaks P_a, P_b, and P_c

Trajectory $\{t[i] \mid t[0] = \pi(a[x_1x_2])\}$

This is an example of vector field on flat tiles.

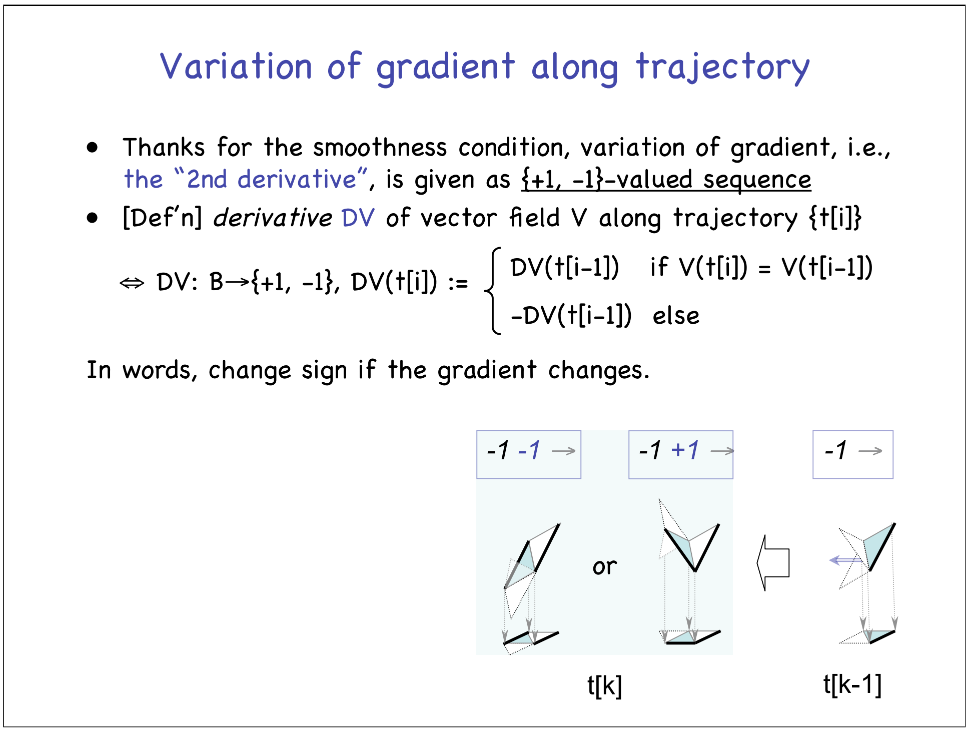
"Peaks and valleys" …, where vector field V is a function from B to the set of gradients and flat tile $pi(a[x_ix_j])$ assumes the gradient $Da[x_ix_j]$ of the corresponding slant tile $a[x_ix_j]$ on the surface.

Shown on the left is an example of vector field V defined by three peaks. The value of V on the closed trajectory of blue tiles is shown on the right.

Let's start from flat tile t[0] colored blue and move downward. t[0] is the image of the blue slant tile $a[x_1x_2]$ by pi. And the value of V on t[0] is x_1x_2 as shown in blue. Then, it specifies a local flow of t[9], t[0], and t[1].

Since we move downward, our next tile is t[1] and it is the image of slant tile $ax_1[x_2x_1]$. Thus, the value of V on t[1] is also x_1x_2.

Continueing the process, we obtain the closed trajectory of length 10. And it satisfies the smoothness condition as you see.



Variation of gradient along trajectory

- Thanks for the smoothness condition, variation of gradient, i.e., the "2nd derivative", is given as <u>{+1, -1}-valued sequence</u>
- [Def'n] *derivative* DV of vector field V along trajectory {t[i]}

$$\Leftrightarrow \text{DV: } B \to \{+1, -1\},\ \text{DV}(t[i]) := \begin{cases} \text{DV}(t[i-1]) & \text{if } V(t[i]) = V(t[i-1]) \\ -\text{DV}(t[i-1]) & \text{else} \end{cases}$$

In words, change sign if the gradient changes.

Next let's consider variation of gradient along trajectory.

Thanks for the smoothness ...

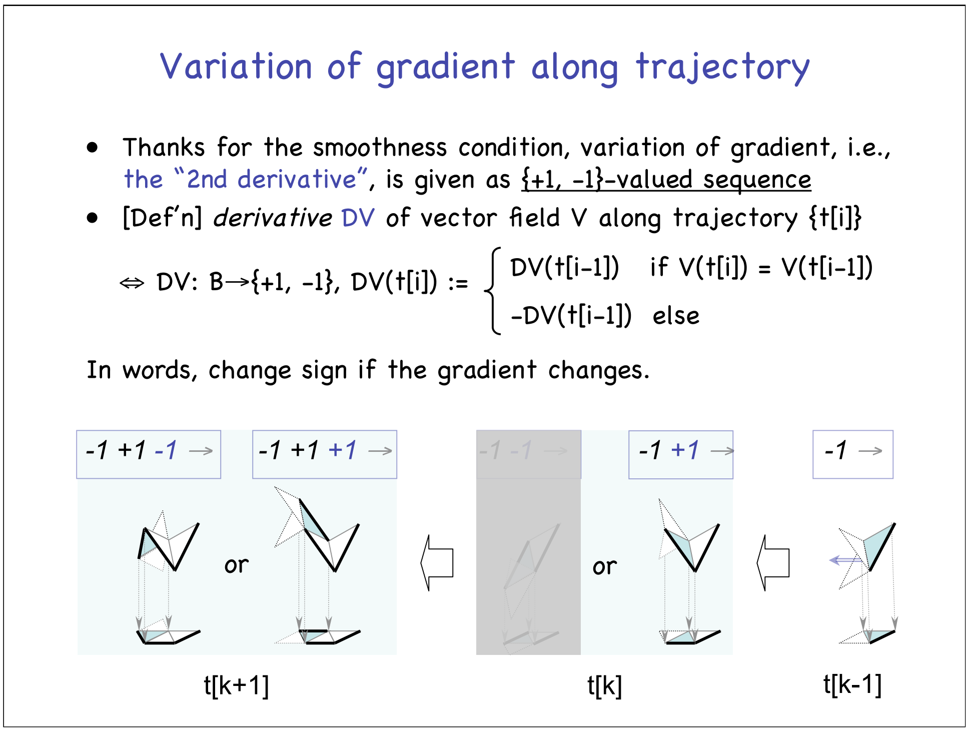
Derivative DV ... is a function from B to {+1, −1} and its value on t[i] is defined as follows. That is, if the gradient V(t[i]) of the curremt tile t[i] is equal to the gradient V(t[i−1]) of the previous tile t[i−1], then the value of DV on t[i] is equal to the value of DV on t[i−1]. Otherwise, the value is multiplied by −1.

In words, ...

Shown below is an example. The left figure shows the previous tile t[k−1] (colored blue) and the right figure shows the current tile t[k] (also colored blue). Because of the smoothness condition, the current tile t[k] could assume one of these two blue slant tiles.

Suppose that the value of DV on t[k−1] is −1. Then, in the left case, the gradient of the "blue" current slant tile is equal to the gradient of the "white" previous slant tile. And the value of DV on t[k] is −1.

On the other hand, in the right case, the gradient of the "blue" slant tile is not equal to the gradient of the "white" slant tile. Thus, the value of DV is changed from −1 to +1.



Variation of gradient along trajectory

- Thanks for the smoothness condition, variation of gradient, i.e., the "2nd derivative", is given as {+1, -1}-valued sequence
- [Def'n] *derivative* DV of vector field V along trajectory {t[i]}

$$\Leftrightarrow \text{DV: } B \rightarrow \{+1, -1\},\ \text{DV}(t[i]) := \begin{cases} \text{DV}(t[i-1]) & \text{if } V(t[i]) = V(t[i-1]) \\ -\text{DV}(t[i-1]) & \text{else} \end{cases}$$

In words, change sign if the gradient changes.

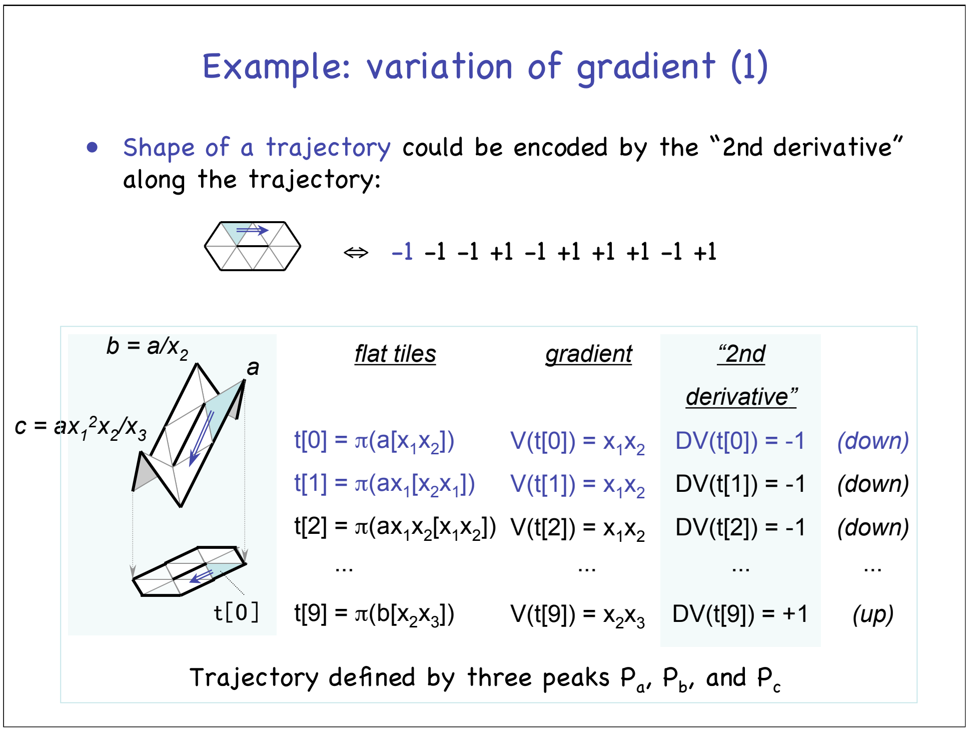
Let's choose the right case and continue the process.

The left figure shows the next tile t[k+1] (colored blue). Again, because of the smoothness condition, t[k+1] could assume one of the two blue slant tiles.

In the left case, the gradient of the "blue" next slant tile is not equal to the gradient of the "white" current slant tile. And the value of DV is changed from +1 to −1.

On the other hand, in the right case, the gradient of the "blue" slant tile is equal to the gradient of the "white" slant tile. Thus, the value of DV on t[k] is +1.

In either case, we obtain a binary valued sequence of length three, which describes variation of gradient along the trajectory.



Example: variation of gradient (1)

- Shape of a trajectory could be encoded by the “2nd derivative” along the trajectory:

⇔ -1 -1 -1 +1 -1 +1 +1 +1 -1 +1

flat tiles	*gradient*	*“2nd derivative”*	
$t[0] = \pi(a[x_1x_2])$	$V(t[0]) = x_1x_2$	DV(t[0]) = -1	*(down)*
$t[1] = \pi(ax_1[x_2x_1])$	$V(t[1]) = x_1x_2$	DV(t[1]) = -1	*(down)*
$t[2] = \pi(ax_1x_2[x_1x_2])$	$V(t[2]) = x_1x_2$	DV(t[2]) = -1	*(down)*
…	…	…	…
$t[9] = \pi(b[x_2x_3])$	$V(t[9]) = x_2x_3$	DV(t[9]) = +1	*(up)*

Trajectory defined by three peaks P_a, P_b, and P_c

As an example of variation of gradient, let’s consider encoding of shapes.

Shape of a trajectory could …

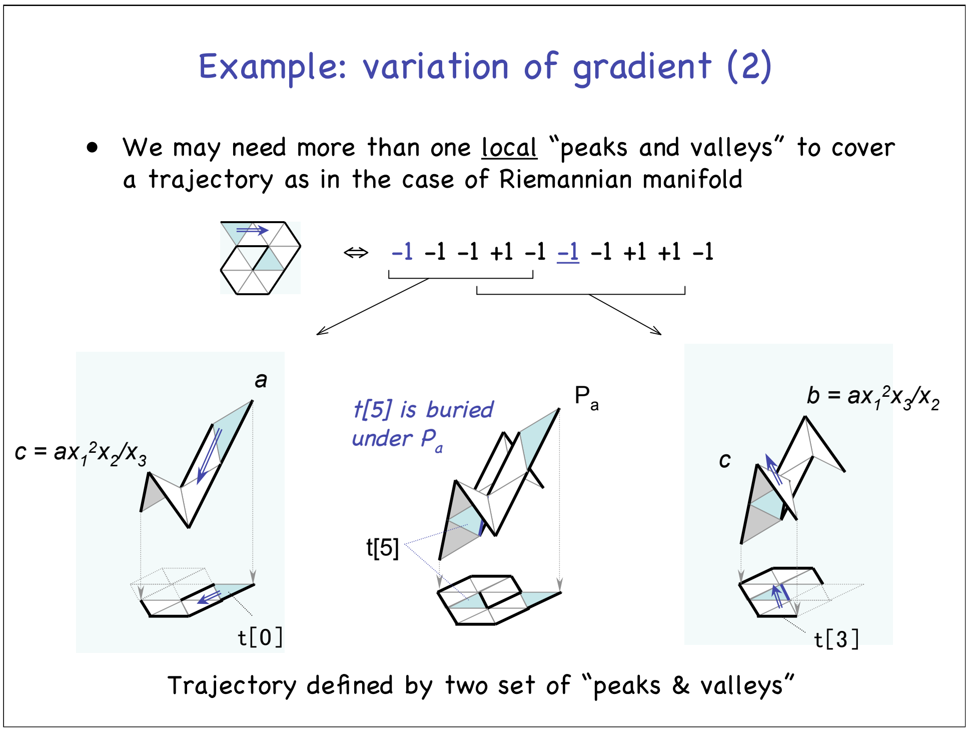
And the hexagon shown on the left is encoded into the binary sequence of length 10 on the right, using the trajectory shown below.

Shown below is the example we considered before: the closed trajectory defined by three peaks.

First, the gradient of the first two flat tiles t[0] and t[1] are given as shown in blue.

Suppose that the ″2nd derivative″ of t[0] is −1. Then, since the first two tiles have the same gradient, the ″2nd derivative″ of t[1] is also −1. When the gradient changes over t[3], the ″2nd derivative″ of t[3] becomes +1 accordingly.

Note that there is a one−to−one correspondence between the ″2nd derivative″ and ″up and down″ along trajectory.



Example: variation of gradient (2)

- We may need more than one <u>local</u> “peaks and valleys” to cover a trajectory as in the case of Riemannian manifold

⇔ $\underline{-1}$ -1 -1 +1 -1 $\underline{-1}$ -1 +1 +1 -1

Trajectory defined by two set of “peaks & valleys”

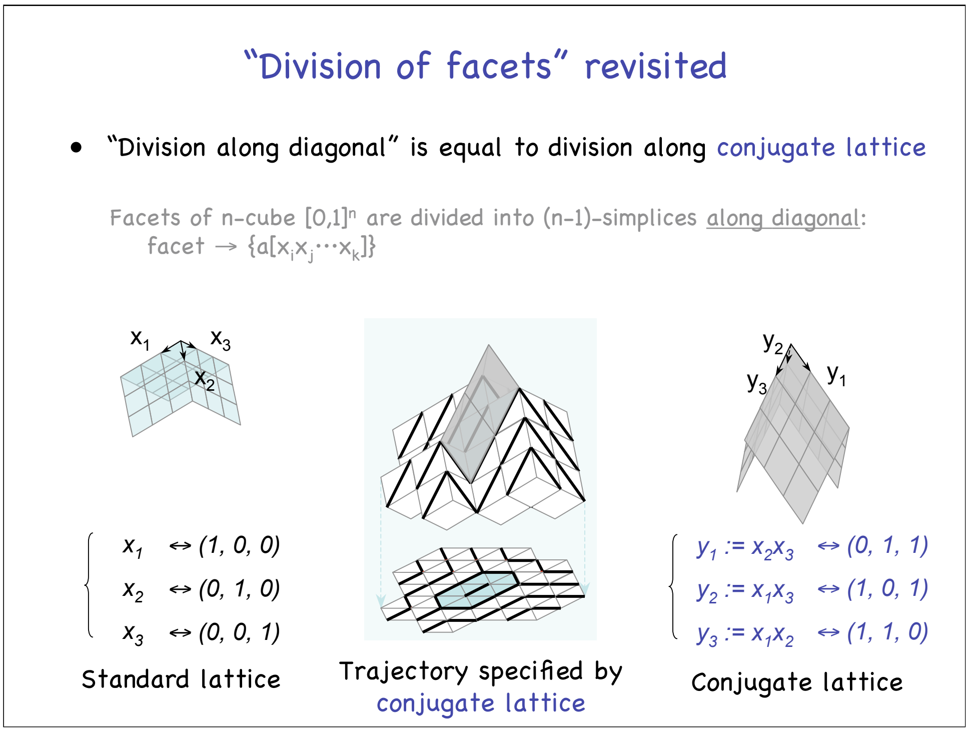
Let’s consider another example.

We may need …

And the trajectory shown on the left is encoded into the binary sequence on the right, using the two sets of ″peaks and valleys″ shown below.

The first five flat tiles t[0], …, t[4] of the trajectory are encoded using the left Ps&Vs. But the 6-th tile t[5] colored blue is buried under peak P_a as shown in the middle figure.

Thus we need another set of Ps&Vs shown on the right to encode the rest of the trajectory.



"Division of facets" revisited

- "Division along diagonal" is equal to division along conjugate lattice

Facets of n-cube $[0,1]^n$ are divided into (n-1)-simplices along diagonal:
facet → $\{a[x_i x_j \cdots x_k]\}$

$$\begin{cases} x_1 \leftrightarrow (1, 0, 0) \\ x_2 \leftrightarrow (0, 1, 0) \\ x_3 \leftrightarrow (0, 0, 1) \end{cases}$$

Standard lattice

Trajectory specified by conjugate lattice

$$\begin{cases} y_1 := x_2 x_3 \leftrightarrow (0, 1, 1) \\ y_2 := x_1 x_3 \leftrightarrow (1, 0, 1) \\ y_3 := x_1 x_2 \leftrightarrow (1, 1, 0) \end{cases}$$

Conjugate lattice

Before moving to the 3-dim'l case, let me show another view point of the "division of facets."

That is, "division along diagonal" …

Recall that facets of n-cube …

Shown on the left is the standard lattce generated by x_1, x_2, and x_3.

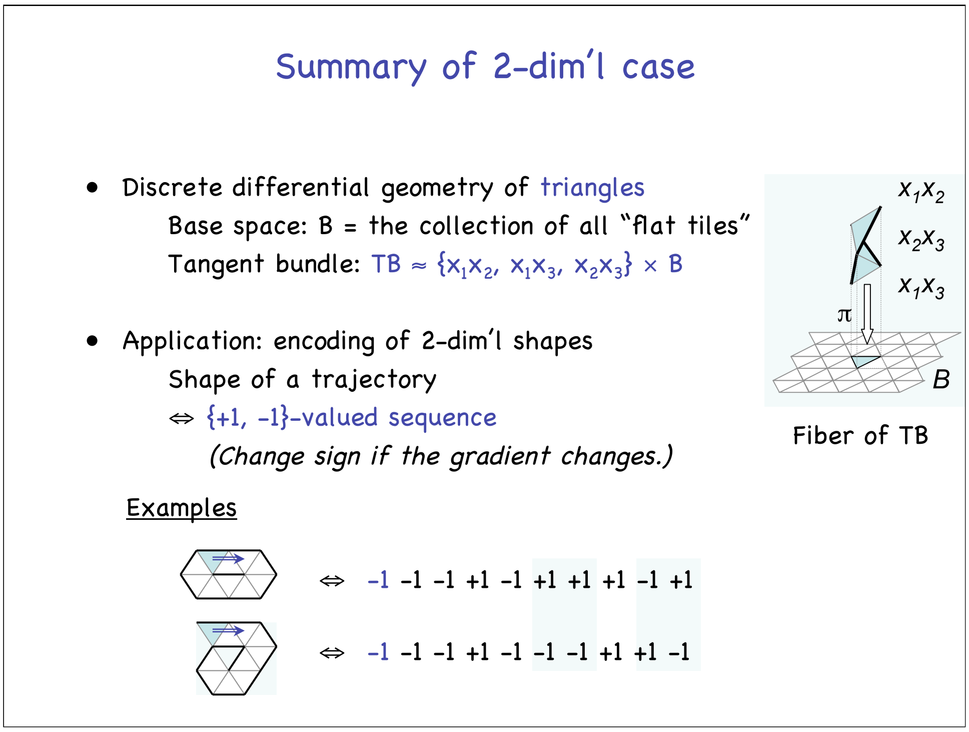
So far, we have used the standard lattice only. And "peaks and valleys" are a cone of the standard lattice generated by its peaks.

On the other hand, the conjugate lattice shown on the right is the lattice geneated by y_1, y_2, and y_3, which are defined as shown below.

Here we used the monomial notation and y_1, y_2, and y_3 correspond to vector (0,1,1), (1,0,1), and (1,1,0) respectively.

Then, as shown in the middle figure, we could use a cone of the conjugate lattice to specify an ("affine") trajectory. In this case, the closed trajectory of blue tiles is specified by the gray cone.

Moreover, one could consider "algebra of closed affine trajectories." But it is not today's subject.



Summary of 2-dim'l case

- Discrete differential geometry of triangles
  Base space: B = the collection of all "flat tiles"
  Tangent bundle: TB ≈ $\{x_1x_2,\ x_1x_3,\ x_2x_3\} \times B$
- Application: encoding of 2-dim'l shapes
  Shape of a trajectory
  ⇔ {+1, -1}-valued sequence
  (Change sign if the gradient changes.)

x_1x_2
x_2x_3
x_1x_3
π
B

Fiber of TB

Examples

⇔ -1 -1 -1 +1 -1 +1 +1 +1 -1 +1

⇔ -1 -1 -1 +1 -1 -1 -1 +1 +1 -1

This is the summary of the 2-dim'l case.

Firstly, DDG of triangles is proposed.

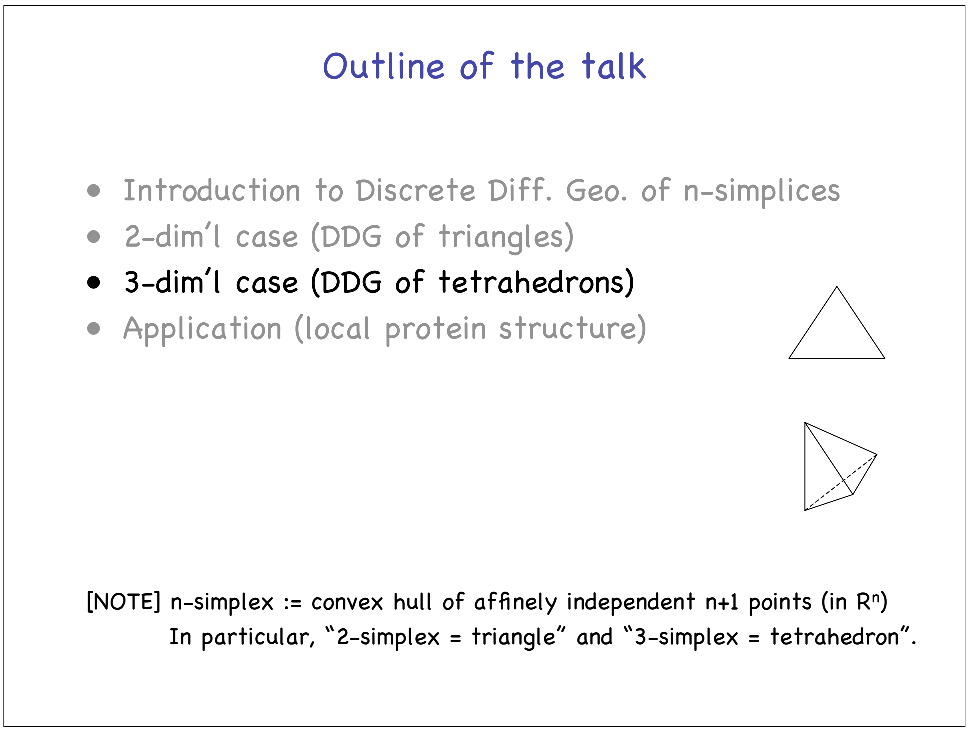
In the construction, base space B is given as the collection of all "flat tiles". And tangent bundle TB over B is identified with the cartesian product of monimials { … } and B.

Shown on the right is fiber of TB, where slant tiles are mapped on a flat tile by projection pi.

Secondly, as an application, encoding of 2-dim'l shapes is proposed.

In the method, the shape of a trajectory is encoded into a {+1, −1}-valued sequence. The encoding rule is, <u>change sign if the gradient changes</u>.

Shown below are examples. These two trajectories on the left are encoded into the binary sequence on the right respectively.

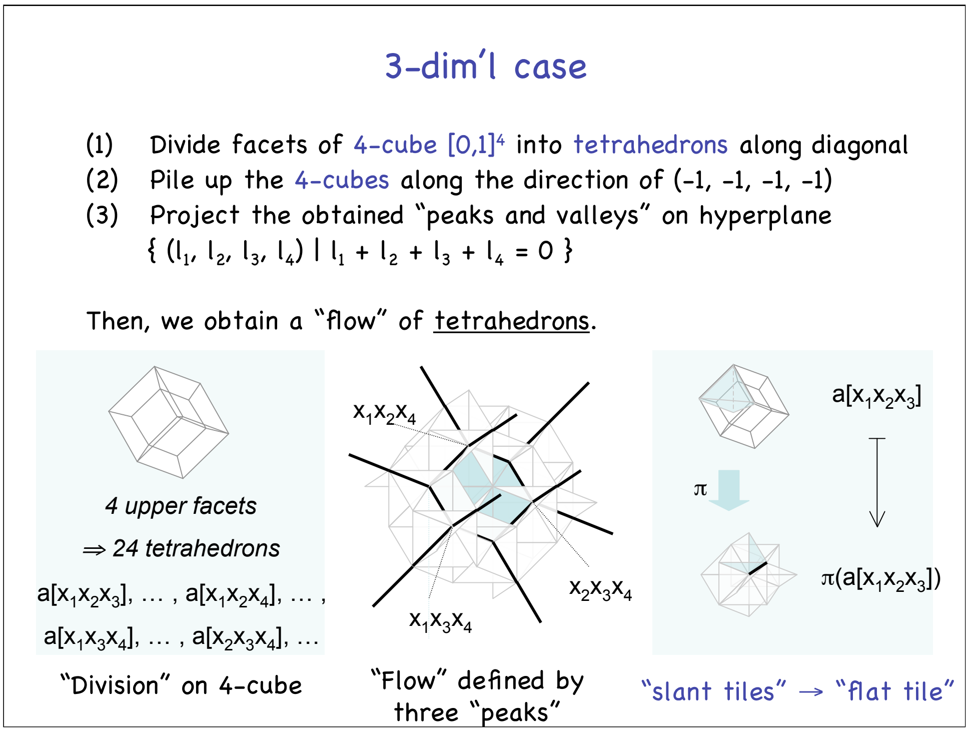


Outline of the talk

- Introduction to Discrete Diff. Geo. of n-simplices
- 2-dim'l case (DDG of triangles)
- 3-dim'l case (DDG of tetrahedrons)
- Application (local protein structure)

[NOTE] n-simplex := convex hull of affinely independent n+1 points (in R^n)
In particular, "2-simplex = triangle" and "3-simplex = tetrahedron".

Next, let me talk about the 3–dim'l case, that is DDG of tetrahedrons.



3-dim'l case

(1) Divide facets of 4-cube $[0,1]^4$ into tetrahedrons along diagonal
(2) Pile up the 4-cubes along the direction of (-1, -1, -1, -1)
(3) Project the obtained "peaks and valleys" on hyperplane
$\{ (l_1, l_2, l_3, l_4) \mid l_1 + l_2 + l_3 + l_4 = 0 \}$

Then, we obtain a "flow" of tetrahedrons.

4 upper facets

⇒ 24 tetrahedrons

$a[x_1x_2x_3]$, … , $a[x_1x_2x_4]$, … ,
$a[x_1x_3x_4]$, … , $a[x_2x_3x_4]$, …

"Division" on 4-cube

"Flow" defined by three "peaks"

"slant tiles" → "flat tile"

These are the basic ideas of the 3-dim'l case.

First, divide …

Shown in the left figure is the division on 4-cube. Since 4-cubes are to be projected onto a hyperplane, it is enough to consider the "upper" facets. And these four upper facets are divided into 24 tetrahedrons $a[x_1x_2x_3]$, ….

Next, pile …

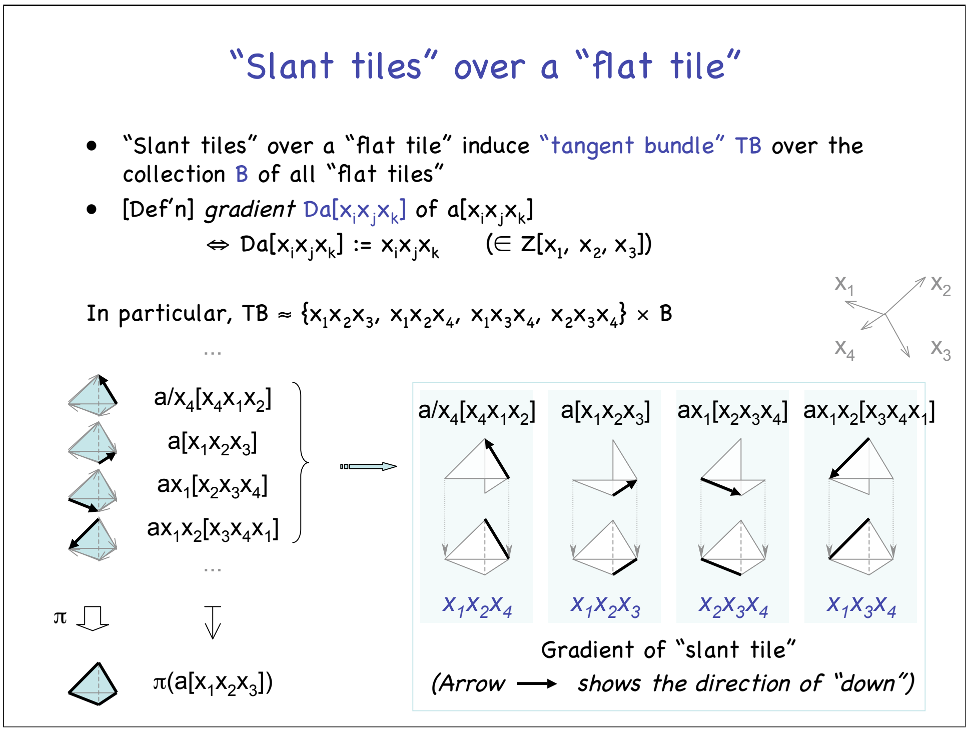
And we obtain "peaks and valleys" of 4-cubes.

Finally, project …

Then, we obtain a "flow" of ... as shown in the middle figure, which is the "flow" defined by three peaks x1x2x4, x2x3x4, and x1x3x4.

As you see, the division of the surface of the Ps&Vs specifies trajectories of tetrahedrons on the hyperplane. For example, the blue tiles form a closed trajectory of length six.

Shown in the right figure is the projection of tetrahedrons. In the next slide, we use the projection pi to contsruct DDG of tetrahedron tiles.



"Slant tiles" over a "flat tile"

- "Slant tiles" over a "flat tile" induce "tangent bundle" TB over the collection B of all "flat tiles"
- [Def'n] *gradient* Da[$x_ix_jx_k$] of a[$x_ix_jx_k$]

  ⇔ Da[$x_ix_jx_k$] := $x_ix_jx_k$ (∈ Z[x_1, x_2, x_3])

In particular, TB ≈ {$x_1x_2x_3$, $x_1x_2x_4$, $x_1x_3x_4$, $x_2x_3x_4$} × B

Gradient of "slant tile"

(Arrow ⟶ shows the direction of "down")

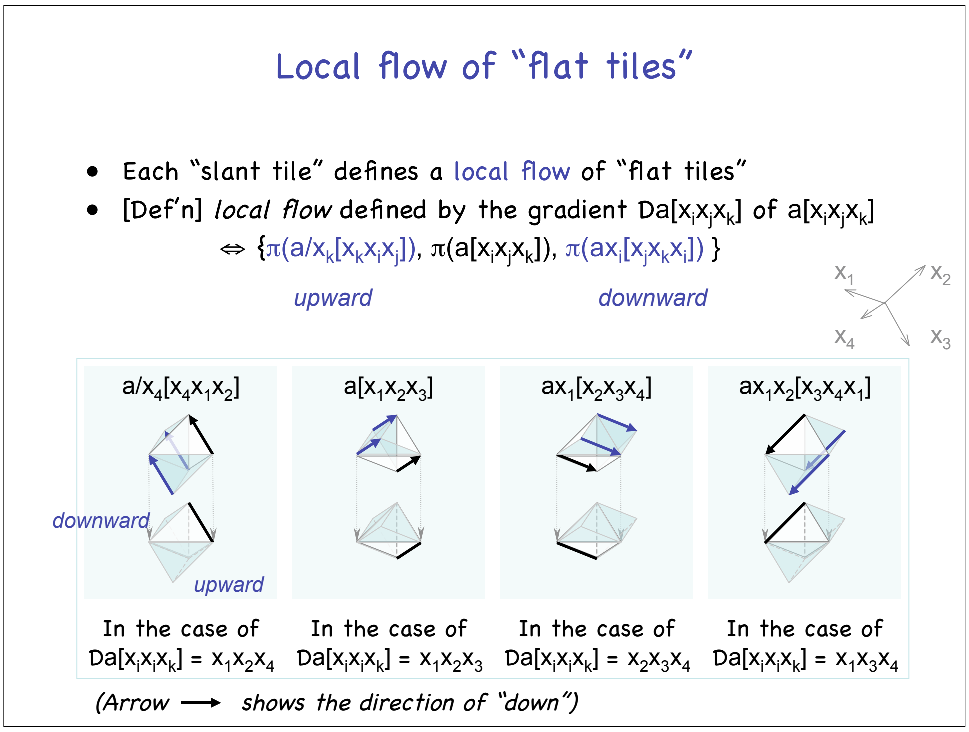
As in the case of 2-dim, slant tiles over …

Gradient Da[$x_ix_jx_k$] of a[$x_ix_jx_k$] is monomial $x_ix_jx_k$.

In particular, tangent bundle TB over B is identified with the cartesian product of …

Shown on the left is the fiber over a flat tile. And all of these slant tiles are projected on the same flat tile.

Each of the slant tiles assumes one of the four gradients shown on the right, where the arrow shows the direction of "down." For example, gradient of a/x_4[$x_4x_1x_2$] is $x_1x_2x_4$ and so on.



Local flow of "flat tiles"

- Each "slant tile" defines a local flow of "flat tiles"
- [Def'n] *local flow* defined by the gradient $Da[x_ix_jx_k]$ of $a[x_ix_jx_k]$

⇔ {$\pi(a/x_k[x_kx_ix_j])$, $\pi(a[x_ix_jx_k])$, $\pi(ax_i[x_jx_kx_i])$ }

upward *downward*

In the case of $Da[x_ix_jx_k] = x_1x_2x_4$

In the case of $Da[x_ix_jx_k] = x_1x_2x_3$

In the case of $Da[x_ix_jx_k] = x_2x_3x_4$

In the case of $Da[x_ix_jx_k] = x_1x_3x_4$

(Arrow ⟶ shows the direction of "down")

This slide shows the local flow defined by a slant tile.

Each "slant tile" defines ...

And here is the definition.

Local flow defined by ... is the three consecutive flat tiles of { ... }.

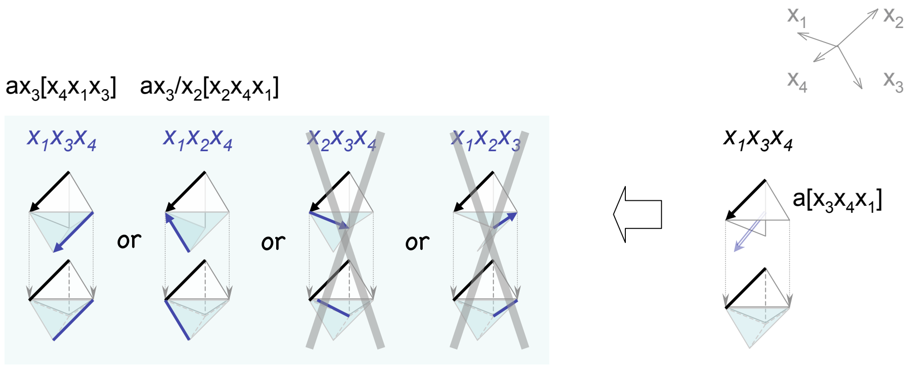
Shown below are the local flows around a flat tile (colored white).

For example, if the gradient of the flat tile is $x_1x_2x_4$ as shown in the left figure, then the "blue" slant tile on this side of the "white" slant tile specifies the upward flat tile and the "bule" slant tile on the other side specifies the downward flat tile.

Similarly adjacent blue slant tiles specify the upward and downward flat tiles in the other cases.

“Smoothness condition” of local flow

- Each “flat tile” assumes one of two gradient values, which are determined by the gradient of the preceding tile



Current tile: $\pi(ax_3[x_4x_1x_3])$ Preceding tile: $\pi(a[x_3x_4x_1])$

Next, let’s consider smoothness of local flow.

Each ″flat tile″ assumes …

Shown below is an example of the condition. Suppose that the gradient of the preceding flat tile is $x_1x_3x_4$ and the current flat tile is on the downward side as shown on the right.

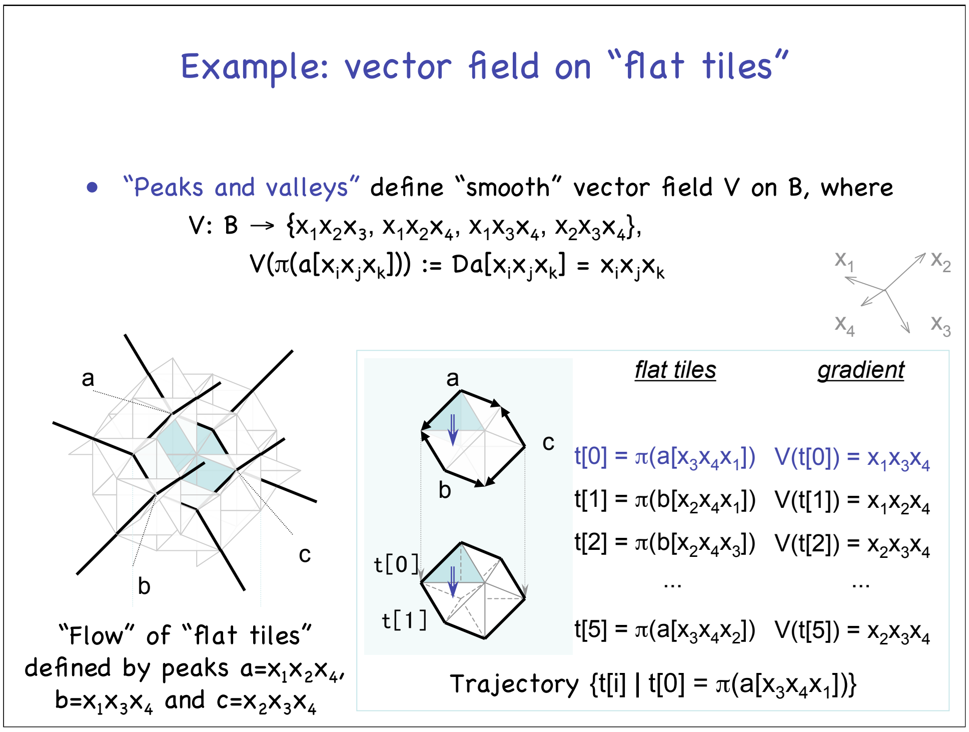
Then, with each of the four gradient values, we could associate a slant tile over the current flat tile as shown on the left.

In the case of $x_1x_3x_4$, the ″blue″ and ″white″ slant tiles are connected smoothly. In the case of $x_1x_2x_4$, the slant tiles are also connected smoothly.

But in the case of $x_2x_3x_4$, the slant tiles are not connected smoothly. In the case of $x_1x_2x_3$, the slant tiles are not connected smoothly neither.

The smoothness condition means that the current flat tile assumes one of the first two values.

As you see, regardless of the dimension, there are only two possible values for each flat tile.



Example: vector field on "flat tiles"

- "Peaks and valleys" define "smooth" vector field V on B, where

  $V: B \rightarrow \{x_1x_2x_3, x_1x_2x_4, x_1x_3x_4, x_2x_3x_4\}$,

  $V(\pi(a[x_ix_jx_k])) := Da[x_ix_jx_k] = x_ix_jx_k$

"Flow" of "flat tiles" defined by peaks $a=x_1x_2x_4$, $b=x_1x_3x_4$ and $c=x_2x_3x_4$

flat tiles	*gradient*
$t[0] = \pi(a[x_3x_4x_1])$	$V(t[0]) = x_1x_3x_4$
$t[1] = \pi(b[x_2x_4x_1])$	$V(t[1]) = x_1x_2x_4$
$t[2] = \pi(b[x_2x_4x_3])$	$V(t[2]) = x_2x_3x_4$
…	…
$t[5] = \pi(a[x_3x_4x_2])$	$V(t[5]) = x_2x_3x_4$

Trajectory $\{t[i] \mid t[0] = \pi(a[x_3x_4x_1])\}$

This is an example of vector field on flat tiles.

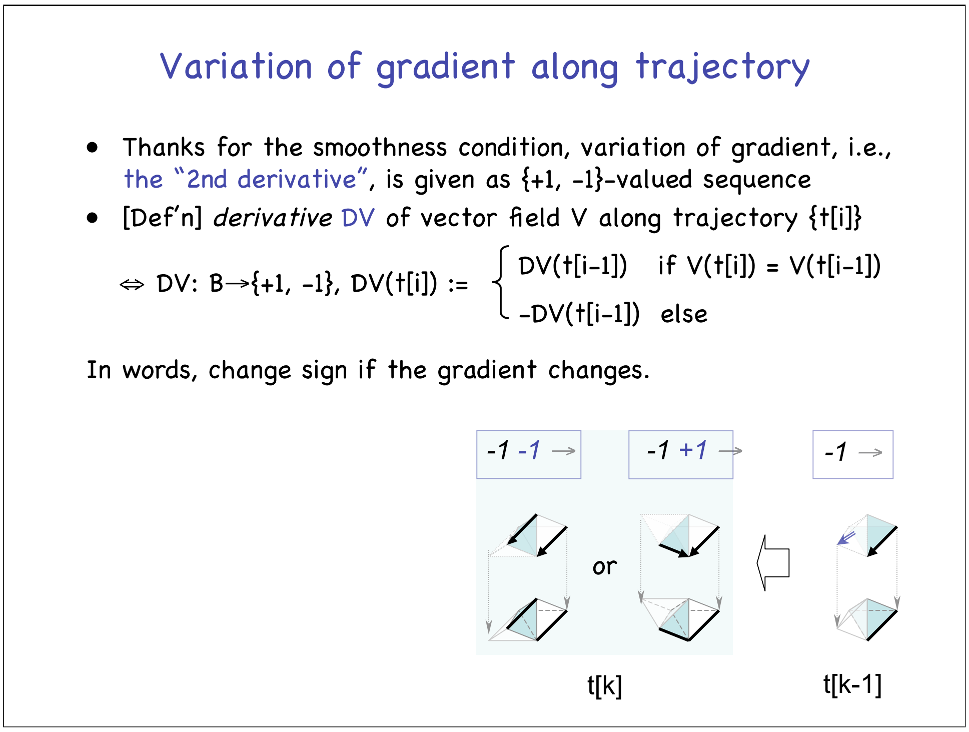
″Peaks and valleys″ ...,

where vector field V is a function from B to the set of gradients and flat tile pi(a[$x_ix_jx_k$]) assumes the gradient of the corresponding slant tile a[$x_ix_jx_k$] on the surface.

Shown on the left is an example of flow defined by three peaks. The value of V on the closed trajectory of blue tiles is shown on the right.

Let's start from flat tile t[0] colored blue and move downward. t[0] is the image of the blue slant tile a[$x_3x_4x_1$] by pi. And the value of V on t[0] is $x_1x_3x_4$ as shown in blue. Then, it specifies a local flow of t[5], t[0], and t[1]. Since we move downward, our next tile is t[1] and it is the image of slant tile b[$x_2x_4x_1$]. Thus, the value of V on t[1] is also $x_1x_2x_4$.

Continueing the process, we obtain the closed trajectory of length six. And it satisfies the smoothness condition as you see.



Variation of gradient along trajectory

- Thanks for the smoothness condition, variation of gradient, i.e., the "2nd derivative", is given as {+1, -1}-valued sequence
- [Def'n] *derivative* DV of vector field V along trajectory {t[i]}

$$\Leftrightarrow \text{DV: B} \to \{+1, -1\},\ \text{DV(t[i])} := \begin{cases} \text{DV(t[i-1])} & \text{if V(t[i]) = V(t[i-1])} \\ \text{-DV(t[i-1])} & \text{else} \end{cases}$$

In words, change sign if the gradient changes.

Next let's consider variation of gradient along trajectory.

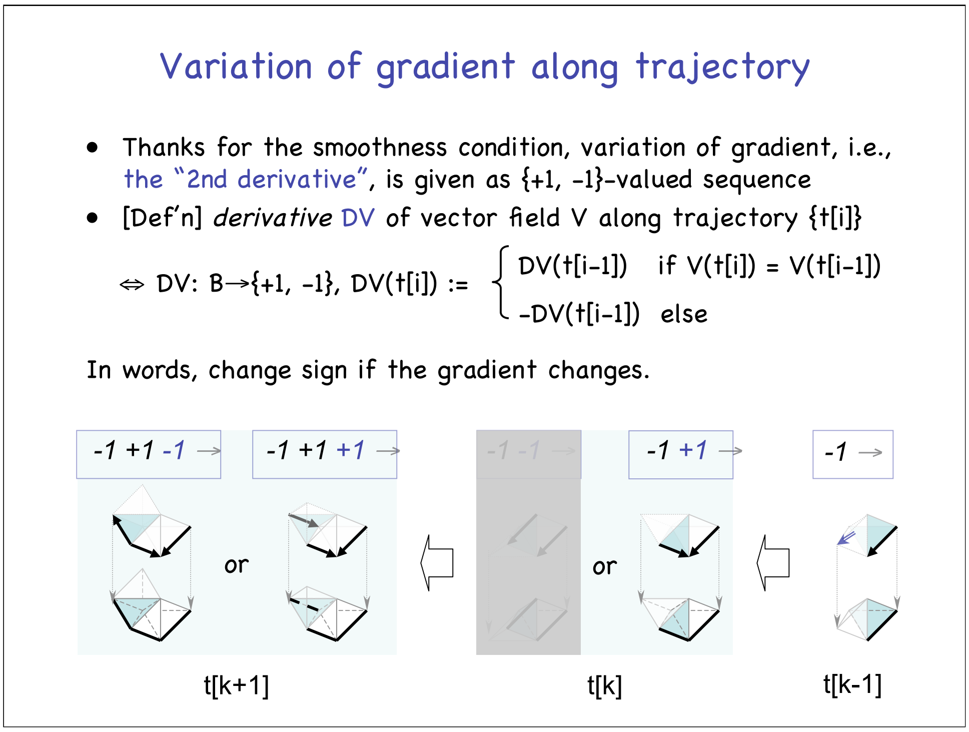
Thanks for the smoothness ...

Derivative DV ... is a function from B to {+1, −1} and its value on t[i] is defined as follows. That is, if the gradient V(t[i]) of the curremt tile t[i] is equal to the gradient V(t[i−1]) of the previous tile t[i−1], then the value of DV on t[i] is equal to the value of DV on t[i−1]. Otherwise, the value is multiplied by −1.

In words, ...

Shown below is an example. The left figure shows the previous tile t[k−1] (colored blue) and the right figure shows the current tile t[k] (also colored blue). Because of the smoothness condition, the current tile t[k] could assume one of these two blue slant tiles.

Suppose that the value of DV on t[k−1] is −1. Then, in the left case, the gradient of the "blue" current slant tile is equal to the gradient of the "white" previous slant tile. And the value of DV on t[k] is −1. On the other hand, in the right case, the gradient of the "blue" slant tile is not equal to the gradient of the "white" slant tile. Thus, the value of DV is changed from −1 to +1.



Variation of gradient along trajectory

- Thanks for the smoothness condition, variation of gradient, i.e., the "2nd derivative", is given as {+1, -1}-valued sequence
- [Def'n] *derivative* DV of vector field V along trajectory {t[i]}

$$\Leftrightarrow \text{DV: } B \rightarrow \{+1, -1\},\ \text{DV}(t[i]) := \begin{cases} \text{DV}(t[i-1]) & \text{if } V(t[i]) = V(t[i-1]) \\ -\text{DV}(t[i-1]) & \text{else} \end{cases}$$

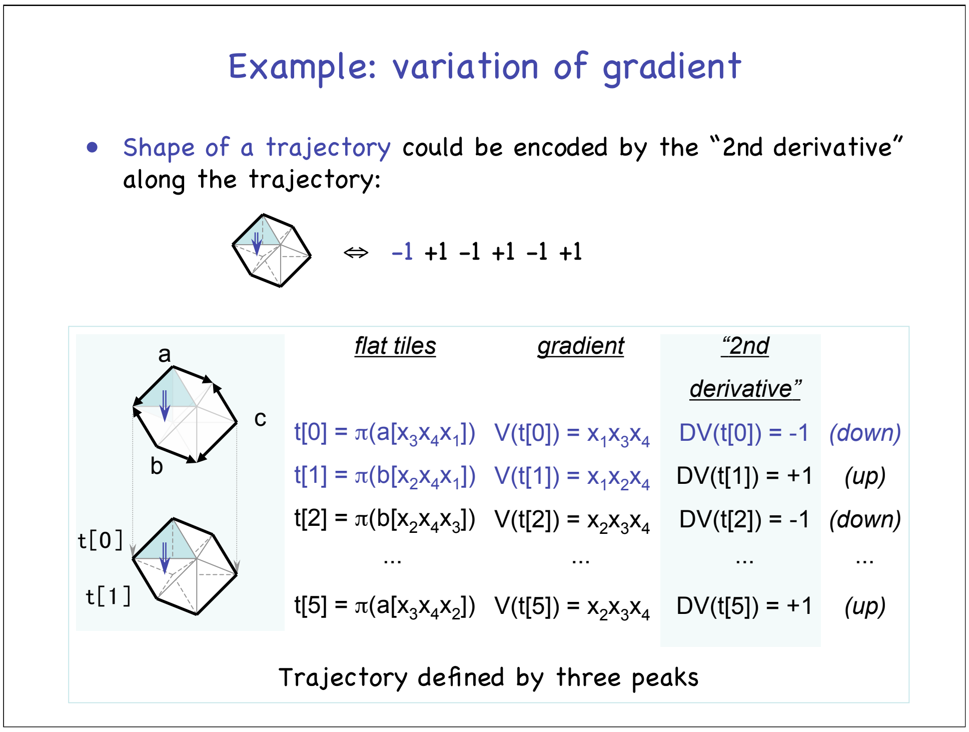
In words, change sign if the gradient changes.

Let's choose the right case and continue the process.

The left figure shows the next tile t[k+1] (colored blue). Again, because of the smoothness condition, t[k+1] could assume one of the two blue slant tiles.

In the left case, the gradient of the "blue" next slant tile is not equal to the gradient of the "white" current slant tile. And the value of DV is changed from +1 to −1. On the other hand, in the right case, the gradient of the "blue" slant tile is equal to the gradient of the "white" slant tile. Thus, the value of DV on t[k+1] is +1.

In either case, we obtain a binary valued sequence of length three, which describes variation of gradient along the trajectory.



Example: variation of gradient

- Shape of a trajectory could be encoded by the "2nd derivative" along the trajectory:

⇔ -1 +1 -1 +1 -1 +1

flat tiles	*gradient*	*"2nd derivative"*	
$t[0] = \pi(a[x_3x_4x_1])$	$V(t[0]) = x_1x_3x_4$	$DV(t[0]) = -1$	*(down)*
$t[1] = \pi(b[x_2x_4x_1])$	$V(t[1]) = x_1x_2x_4$	$DV(t[1]) = +1$	*(up)*
$t[2] = \pi(b[x_2x_4x_3])$	$V(t[2]) = x_2x_3x_4$	$DV(t[2]) = -1$	*(down)*
…	…	…	…
$t[5] = \pi(a[x_3x_4x_2])$	$V(t[5]) = x_2x_3x_4$	$DV(t[5]) = +1$	*(up)*

Trajectory defined by three peaks

As an example of variation of gradient, let's consider encoding of shapes.

Shape of a trajectory could …

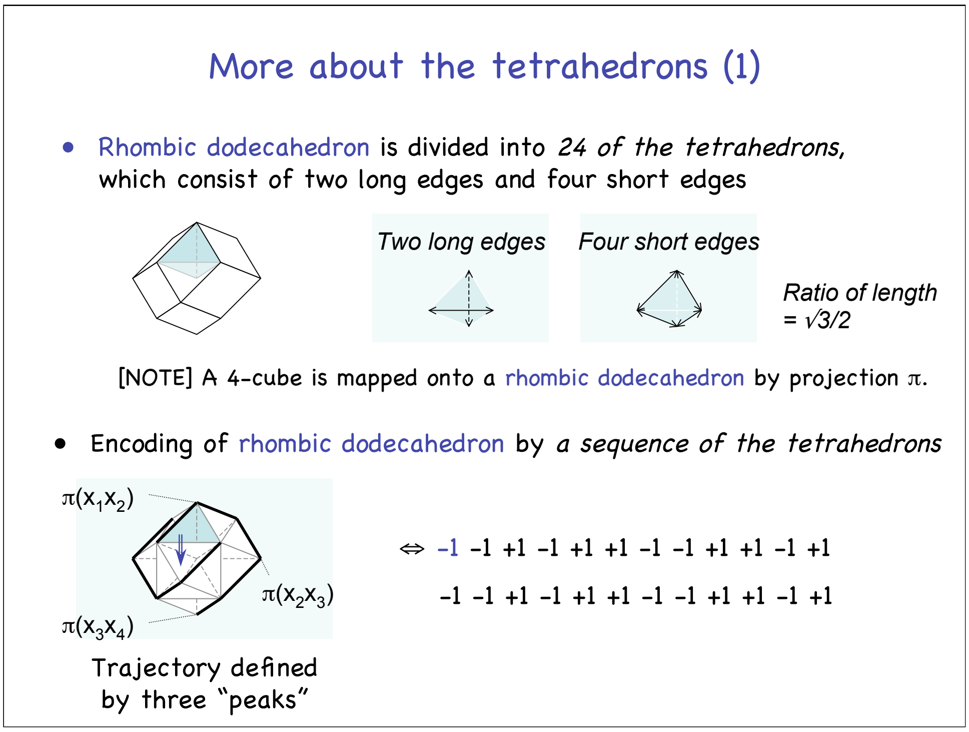
And the shape shown above is encoded into the binary sequence of length six on the right, using the trajectory shown below.

Shown below is the example we considered before: the closed trajectory defined by three peaks.

First, the gradient of the first two flat tiles t[0] and t[1] are given as shown in blue.

Suppose that the "2nd derivative" of t[0] is −1. Then, since the first two tiles t[0] and t[1] assume different gradient values, the "2nd derivative" of t[1] becomes +1. The third tile t[2] assumes yet another gradient and the "2nd derivative" of t[2] becomes −1 accordingly.

Note that there is a one-to-one correspondence between the "2nd derivative" and "up and down" along trajectory.



More about the tetrahedrons (1)

- Rhombic dodecahedron is divided into *24 of the tetrahedrons,* which consist of two long edges and four short edges

[NOTE] A 4-cube is mapped onto a rhombic dodecahedron by projection π.

- Encoding of rhombic dodecahedron by *a sequence of the tetrahedrons*

⇔ -1 -1 +1 -1 +1 +1 -1 -1 +1 +1 -1 +1

-1 -1 +1 -1 +1 +1 -1 -1 +1 +1 -1 +1

Trajectory defined by three "peaks"

Now let me talk more about the tetrahedron used.

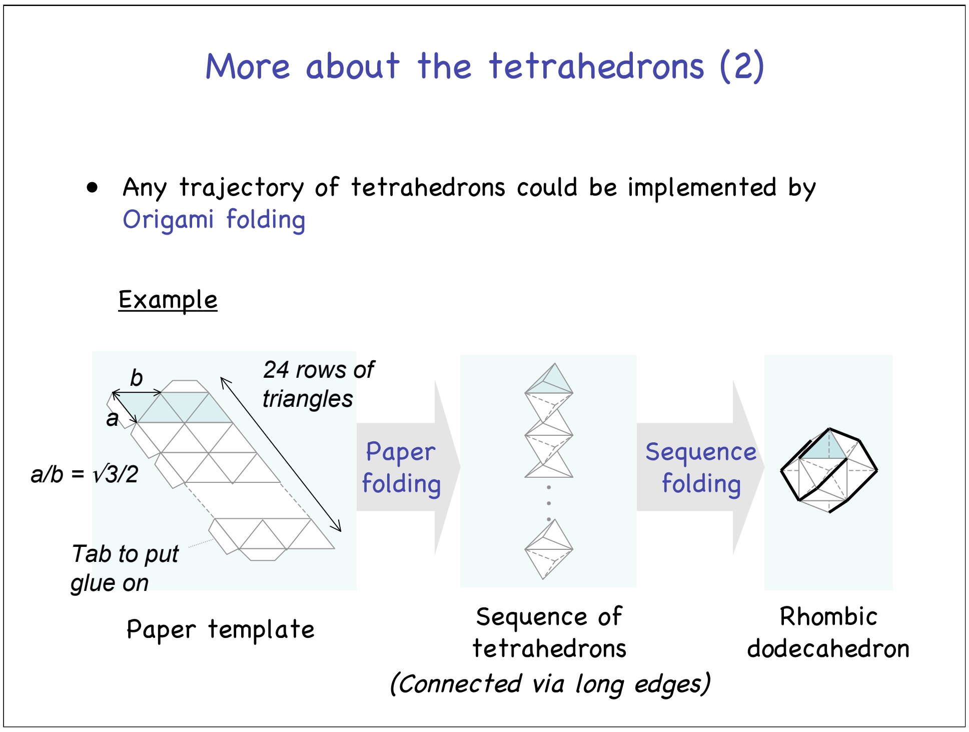
Firstly, a rhombic dedecahedron is …

As shown below, the tetrahedron has two long edges and four short edges, whose ratio of length is sqrt(3)/2.

Note that a 4-cube is …

Secondly, encoding of … is obtained as follows:

Shown on the left is a trajectory defined by three "peaks" x_1x_2, x_2x_3, and x_3x_4. Starting from the blue flat tile, we obtain a binary sequence of length 24 along the trajectory, which is a code of rhombic dedecahedron. (The code is not unique.)



More about the tetrahedrons (2)

- Any trajectory of tetrahedrons could be implemented by Origami folding

Example

b

a

24 rows of triangles

a/b = √3/2

Tab to put glue on

Paper folding

Sequence folding

Paper template

Sequence of tetrahedrons

(Connected via long edges)

Rhombic dodecahedron

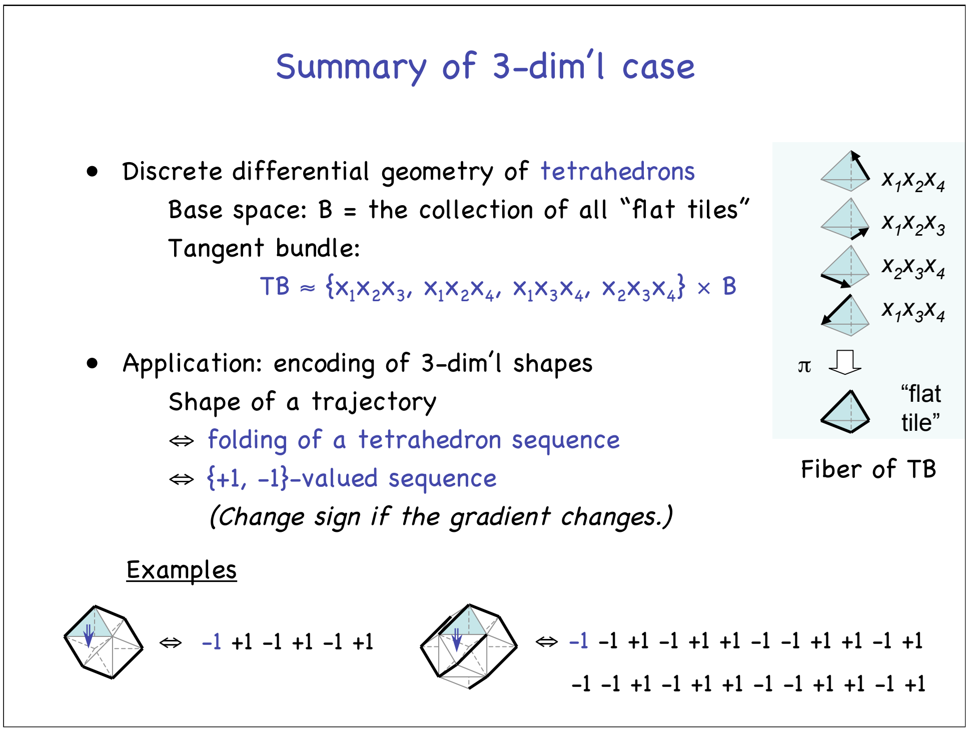
Finally, any trajectory of …

As an example, let's consider rhombic dodecahedron again.

Shown on the left is the paper template used. ″a″ is the length of the shorter edge and ″b″ is that of the longer edge. (Don't forget tabs to put glue on!)

Folding the paper along the lines somehow, we obtain a sequence of 24 tetrahedrons. Successive tetrahedrons are connected via long edges and the sequence has a rotational freedom around the long edges.

Then, folding the sequence somehow again, we obtain a rhombic dodecahedron. (In this case, there are more than one pattern of sequence folding.)



Summary of 3-dim'l case

- Discrete differential geometry of tetrahedrons

  Base space: B = the collection of all "flat tiles"

  Tangent bundle:

  $TB \approx \{x_1x_2x_3,\ x_1x_2x_4,\ x_1x_3x_4,\ x_2x_3x_4\} \times B$

- Application: encoding of 3-dim'l shapes

  Shape of a trajectory

  ⇔ folding of a tetrahedron sequence

  ⇔ {+1, -1}-valued sequence

  (Change sign if the gradient changes.)

$x_1x_2x_4$

$x_1x_2x_3$

$x_2x_3x_4$

$x_1x_3x_4$

π ⇩

"flat tile"

Fiber of TB

<u>Examples</u>

⇔ -1 +1 -1 +1 -1 +1

⇔ -1 -1 +1 -1 +1 +1 -1 -1 +1 +1 -1 +1
-1 -1 +1 -1 +1 +1 -1 -1 +1 +1 -1 +1

This is the summary of the 3-dim'l case.

Firstly, DDG of tetrahedrons are proposed.

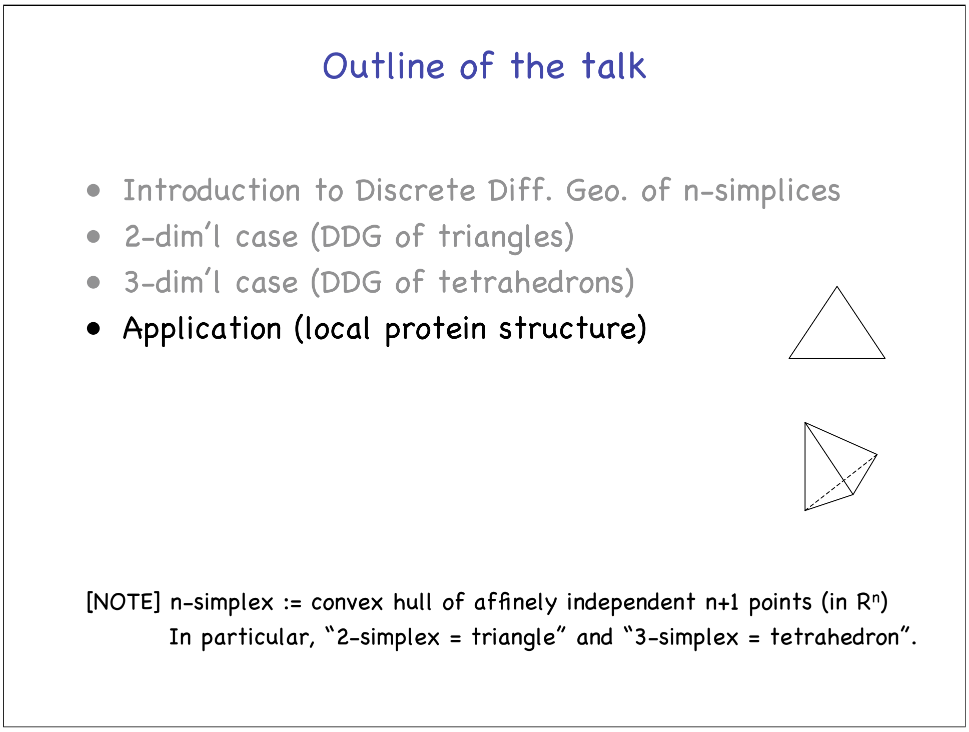
In the construction, base space B is given as the collection of all "flat tiles". And tangent bundle TB over B is identified with the cartesian product of monimials { … } and B.

Show on the right is fiber of TB, where slant tiles are mapped on a flat tile by projection pi.

Secondly, as an application, encoding of 3-dim'l shapes are proposed.

In the method, the shape of a trajectory is obtained by folding a tetrahedron sequence, and encoded into a {+1, −1}-valued sequence. The encoding rule is, change sign if the gradient changes.

Shown below are examples. These two trajectories are encoded into the binary sequence on the right respectively.

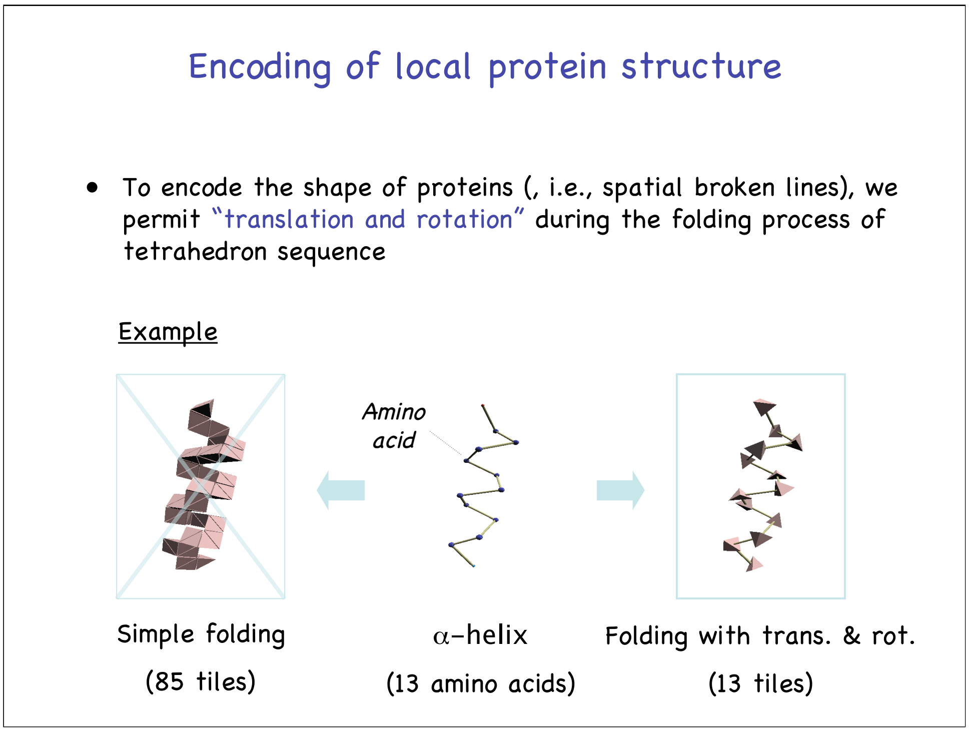


Outline of the talk

- Introduction to Discrete Diff. Geo. of n-simplices
- 2-dim'l case (DDG of triangles)
- 3-dim'l case (DDG of tetrahedrons)
- **Application (local protein structure)**

[NOTE] n-simplex := convex hull of affinely independent n+1 points (in R^n)
In particular, "2-simplex = triangle" and "3-simplex = tetrahedron".

Finally, let me talk about application in local protein structure analysis.



Encoding of local protein structure

- To encode the shape of proteins (, i.e., spatial broken lines), we permit "translation and rotation" during the folding process of tetrahedron sequence

Example

Simple folding (85 tiles)

α-helix (13 amino acids)

Folding with trans. & rot. (13 tiles)

Our objective is encoding of local protein structure.

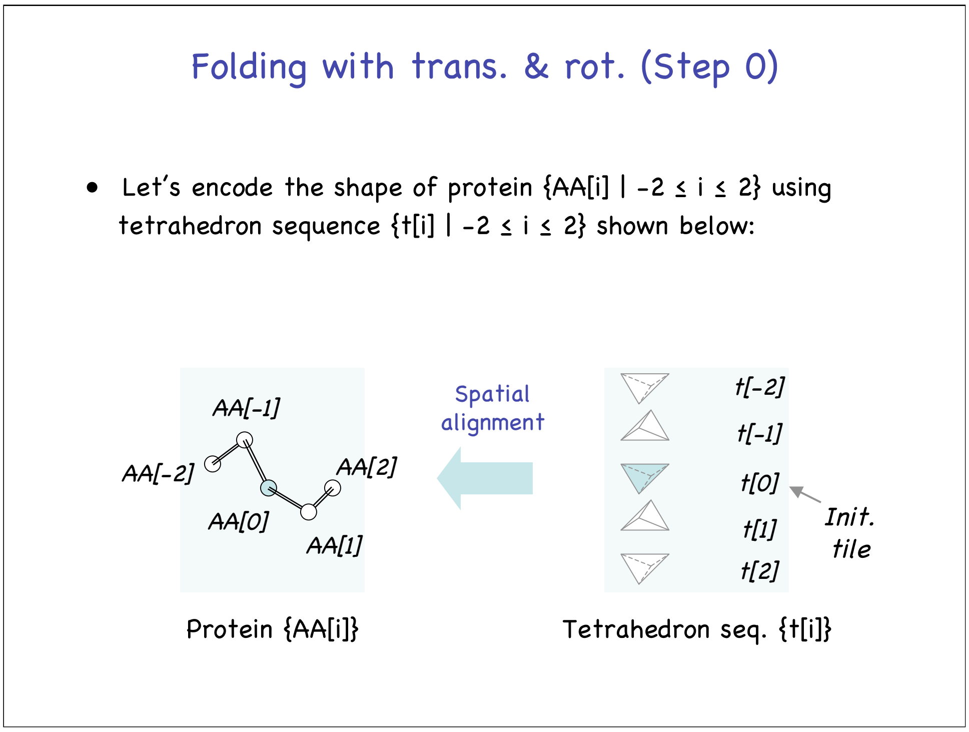
And to encode the shape of …

As an example, let's consider the alpha-helix of length 13 shown in the middle figure.

By simple folding, we obtain the red object of 85 tetrahedron tiles shown on the left. But the object is more complicated than the original alpha-helix.

On the other hand, if we permit translation and rotation, we would obtain the object of 13 tetrahedron tiles shown on the right. And it will turn out that we could capture the local features of proteins by the discontinuous tetrahedron sequence.

The following slides illustrate the coding algorithm briefly.



Folding with trans. & rot. (Step 0)

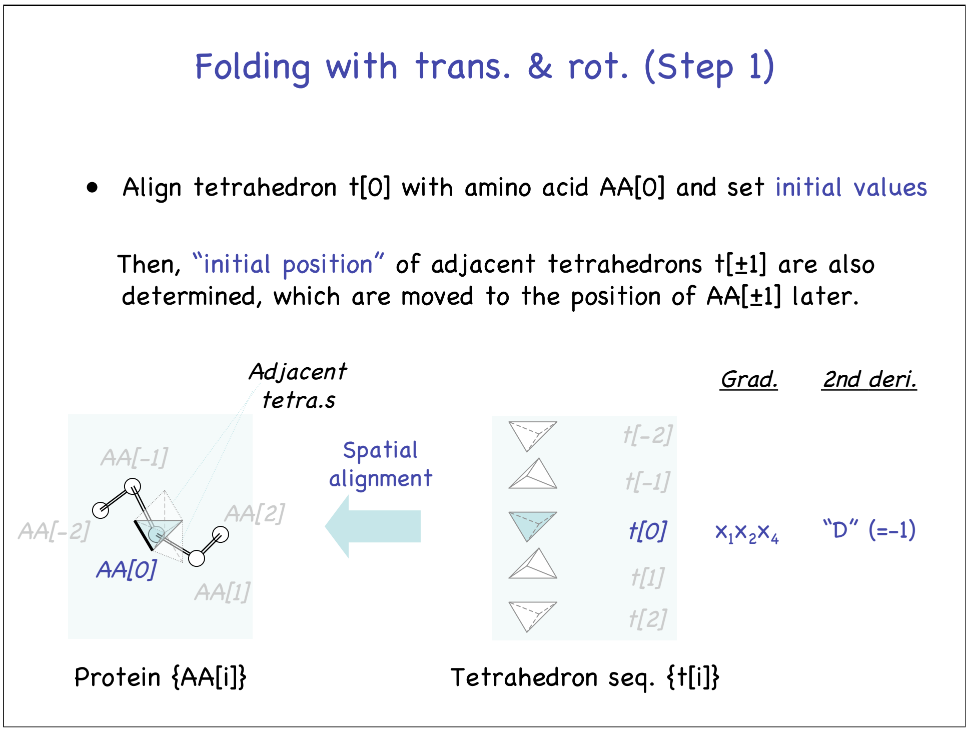
- Let's encode the shape of protein {AA[i] | -2 ≤ i ≤ 2} using tetrahedron sequence {t[i] | -2 ≤ i ≤ 2} shown below:

Protein {AA[i]}

Tetrahedron seq. {t[i]}

Let's encode ...

Shown on the left is the protein of length five. And we encode the protein using the five tetrahedrons shown on the right. We start encoding from the middle amino acid AA[0] (colored blue).



Folding with trans. & rot. (Step 1)

- Align tetrahedron t[0] with amino acid AA[0] and set initial values

Then, “initial position” of adjacent tetrahedrons t[±1] are also determined, which are moved to the position of AA[±1] later.

Protein {AA[i]}

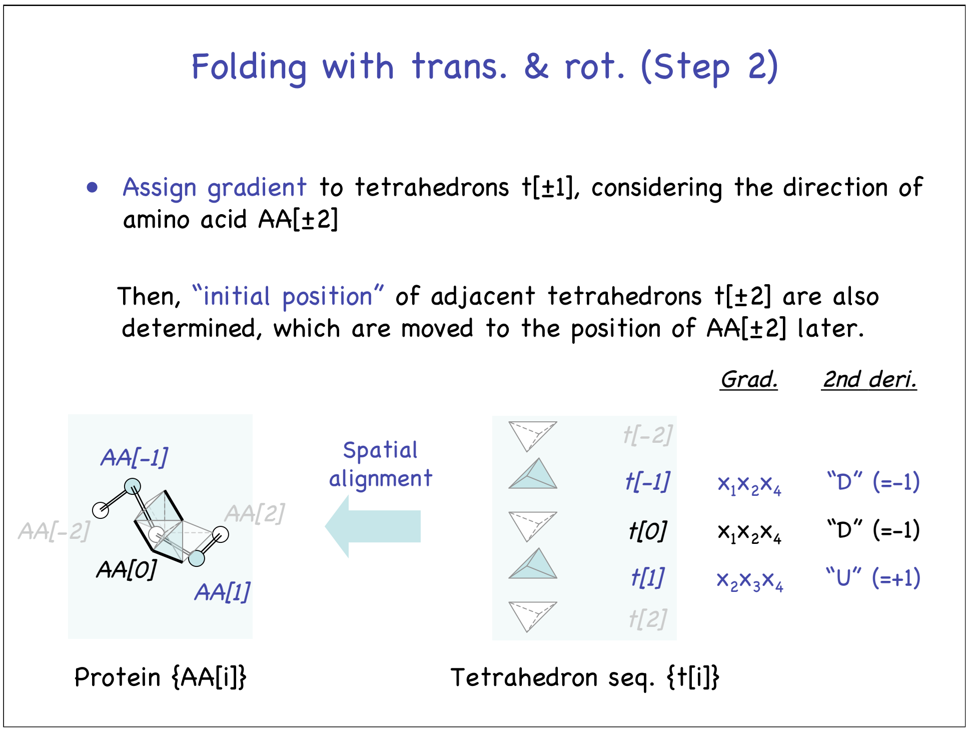
Tetrahedron seq. {t[i]}

First, align tetrahedron t[0] with …

Then, ″initial position″ …

The ″blue″ tetrahedron is aligned with AA[0] as shown on the left. And assign any values of gradient and 2nd derivative to the initial tile t[0] as shown on the right: in this example, $x_1x_2x_4$ and ″D″.

The ″white″ tetrahedrons shown in the left figure are the adjacent tetrahedrons t[1] and t[−1]. They are moved to the position of AA[1] and AA[−1] respectively later.



Folding with trans. & rot. (Step 2)

- Assign gradient to tetrahedrons t[±1], considering the direction of amino acid AA[±2]

Then, "initial position" of adjacent tetrahedrons t[±2] are also determined, which are moved to the position of AA[±2] later.

	Grad.	2nd deri.
t[-2]		
t[-1]	$x_1x_2x_4$	"D" (=-1)
t[0]	$x_1x_2x_4$	"D" (=-1)
t[1]	$x_2x_3x_4$	"U" (=+1)
t[2]		

Protein {AA[i]}

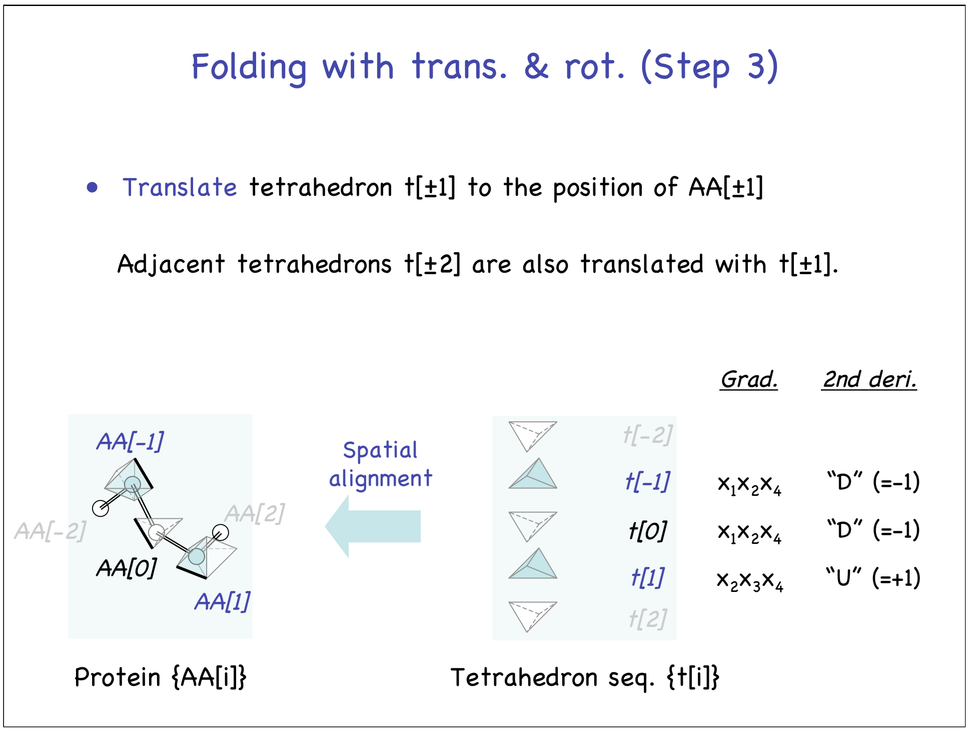
Tetrahedron seq. {t[i]}

Next, assign gradient to tetrahedron …

Then, "initial position" …

The "blue" tetrahedrons are assigned gradient (bold edge) as shown on the left. And their values are shown on the right. For example, t[1] is assigned gradient $x_2x_3x_4$ and the 2nd derivative becomes "U" since the gradient value changes.

The "white" tetrahedrons at ends shown in the left figure are the adjacent tetrahedrons t[2] and t[−2]. They are moved to the position of AA[2] and AA[−2] respectively later.



Folding with trans. & rot. (Step 3)

- Translate tetrahedron t[±1] to the position of AA[±1]

Adjacent tetrahedrons t[±2] are also translated with t[±1].

	Grad.	2nd deri.
t[-2]		
t[-1]	$x_1x_2x_4$	"D" (=-1)
t[0]	$x_1x_2x_4$	"D" (=-1)
t[1]	$x_2x_3x_4$	"U" (=+1)
t[2]		

Protein {AA[i]}

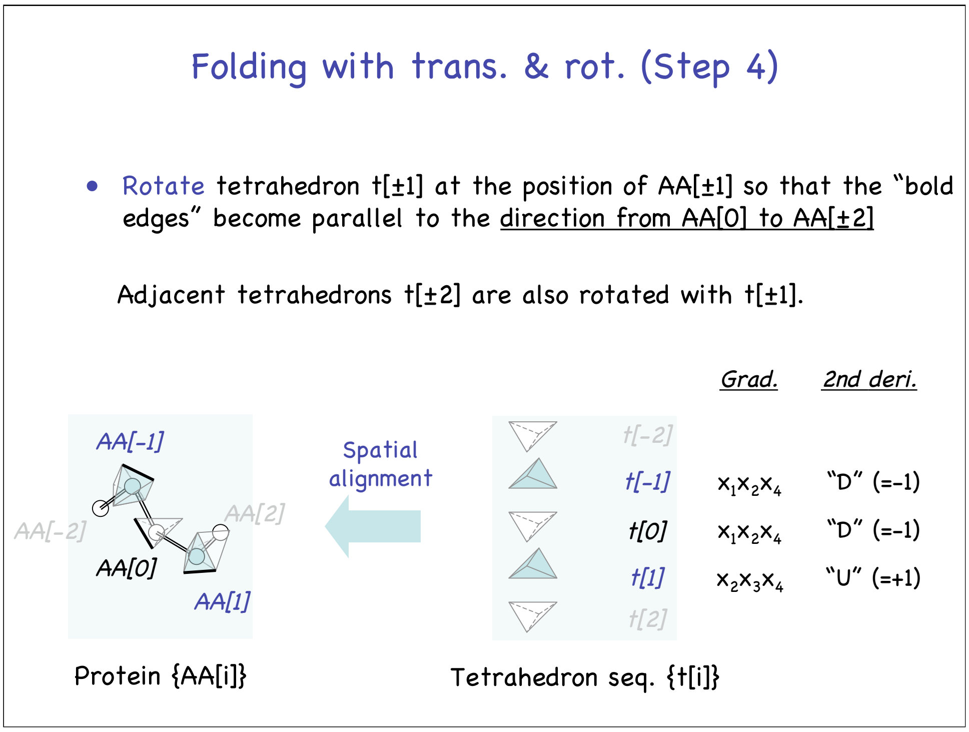
Tetrahedron seq. {t[i]}

Then, translate tetrahedron ...

Adjacent tetrahedrons ...

The ″blue″ tetrahedrons are now at the position of AA[1] and AA[−1] as shown on the left.

The ″white″ tetrahedrons at ends are also translated with the ″blue″ ones.



Folding with trans. & rot. (Step 4)

- Rotate tetrahedron t[±1] at the position of AA[±1] so that the "bold edges" become parallel to the direction from AA[0] to AA[±2]

Adjacent tetrahedrons t[±2] are also rotated with t[±1].

	Grad.	*2nd deri.*
t[-2]		
t[-1]	$x_1x_2x_4$	"D" (=-1)
t[0]	$x_1x_2x_4$	"D" (=-1)
t[1]	$x_2x_3x_4$	"U" (=+1)
t[2]		

Protein {AA[i]}

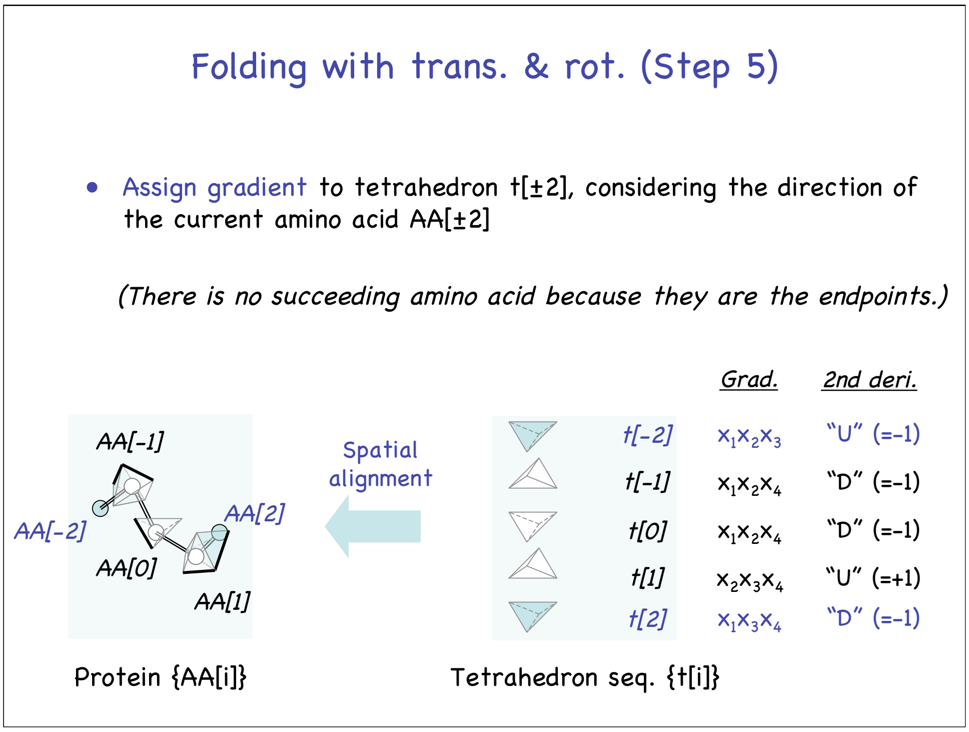
Tetrahedron seq. {t[i]}

Next, rotate tetrahedron …

Adjacent tetrahedrons …

The "blue" tetrahedrons are rotated at the position of AA[1] and AA[−1] as shown on the left.

The "white" tetrahedrons at ends are also moved with the "blue" ones.



Folding with trans. & rot. (Step 5)

- Assign gradient to tetrahedron t[±2], considering the direction of the current amino acid AA[±2]

(There is no succeeding amino acid because they are the endpoints.)

	Grad.	2nd deri.
t[-2]	$x_1x_2x_3$	"U" (=-1)
t[-1]	$x_1x_2x_4$	"D" (=-1)
t[0]	$x_1x_2x_4$	"D" (=-1)
t[1]	$x_2x_3x_4$	"U" (=+1)
t[2]	$x_1x_3x_4$	"D" (=-1)

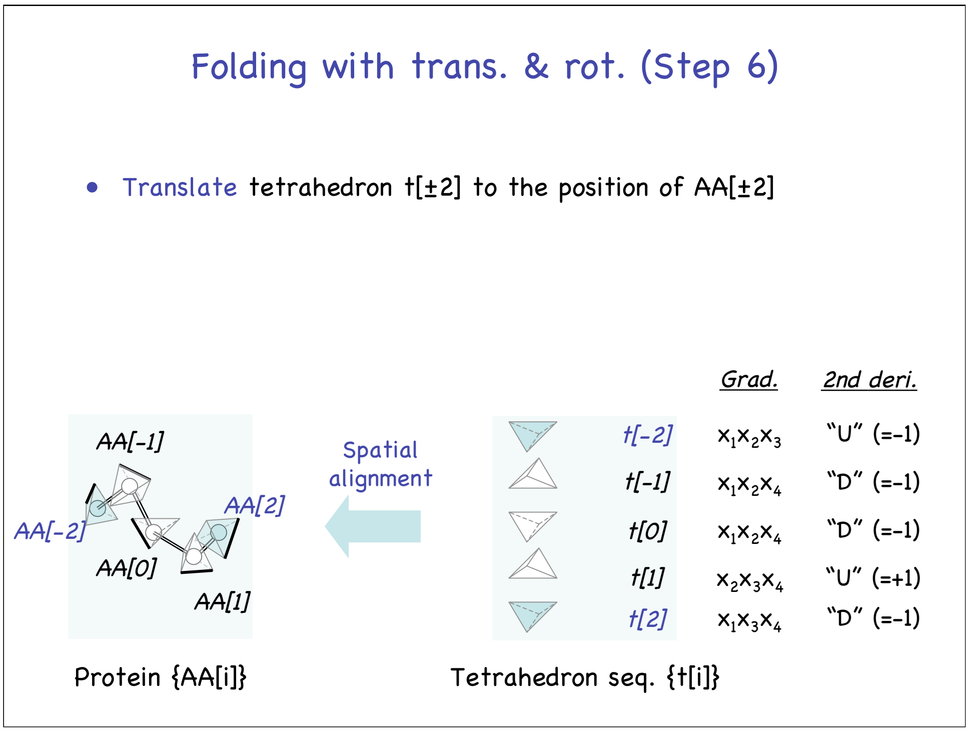
Protein {AA[i]}

Tetrahedron seq. {t[i]}

Next, assign gradient to tetrahedron …

Note that there is no succeeding …

The "blue" tetrahedrons are assigned gradient (, I.e. bold edge,) as shown on the left. And their values are shown on the right. For example, t[2] is assigned gradient $x_1x_3x_4$ and the 2nd derivative becomes "D" since the gradient value changes.



Folding with trans. & rot. (Step 6)

- Translate tetrahedron t[±2] to the position of AA[±2]

Spatial alignment

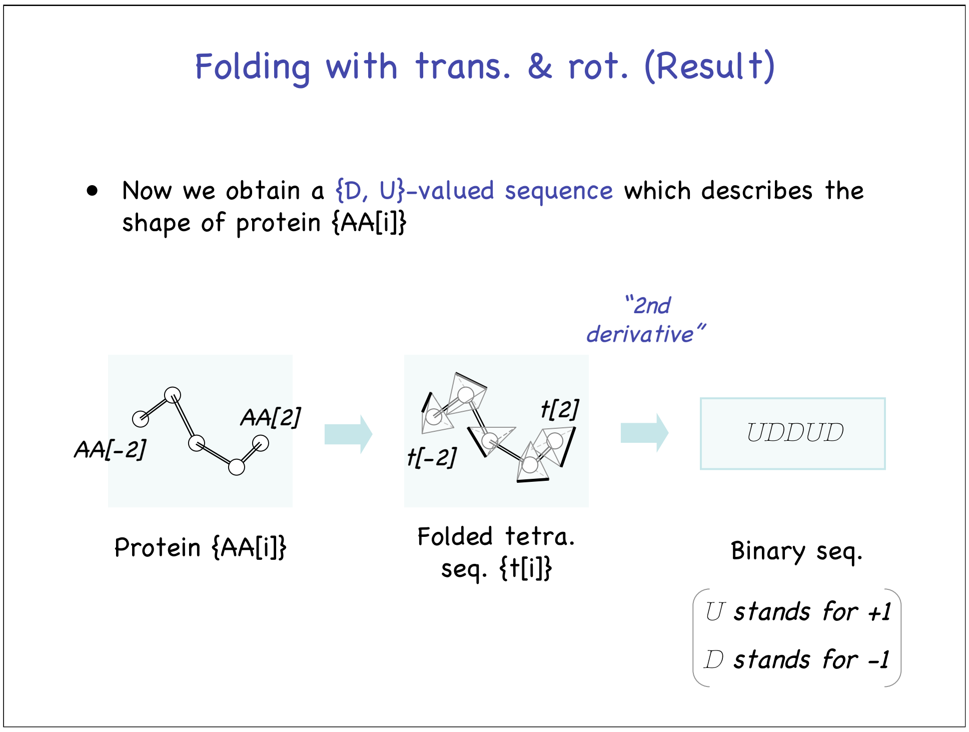
	Grad.	2nd deri.
t[-2]	$x_1x_2x_3$	“U” (=-1)
t[-1]	$x_1x_2x_4$	“D” (=-1)
t[0]	$x_1x_2x_4$	“D” (=-1)
t[1]	$x_2x_3x_4$	“U” (=+1)
t[2]	$x_1x_3x_4$	“D” (=-1)

Protein {AA[i]}

Tetrahedron seq. {t[i]}

Then, translate tetrahedron …

The ″blue″ tetrahedrons are now at the position of AA[2] and AA[−2] as shown on the left.



Folding with trans. & rot. (Result)

- Now we obtain a {D, U}-valued sequence which describes the shape of protein {AA[i]}

Protein {AA[i]}

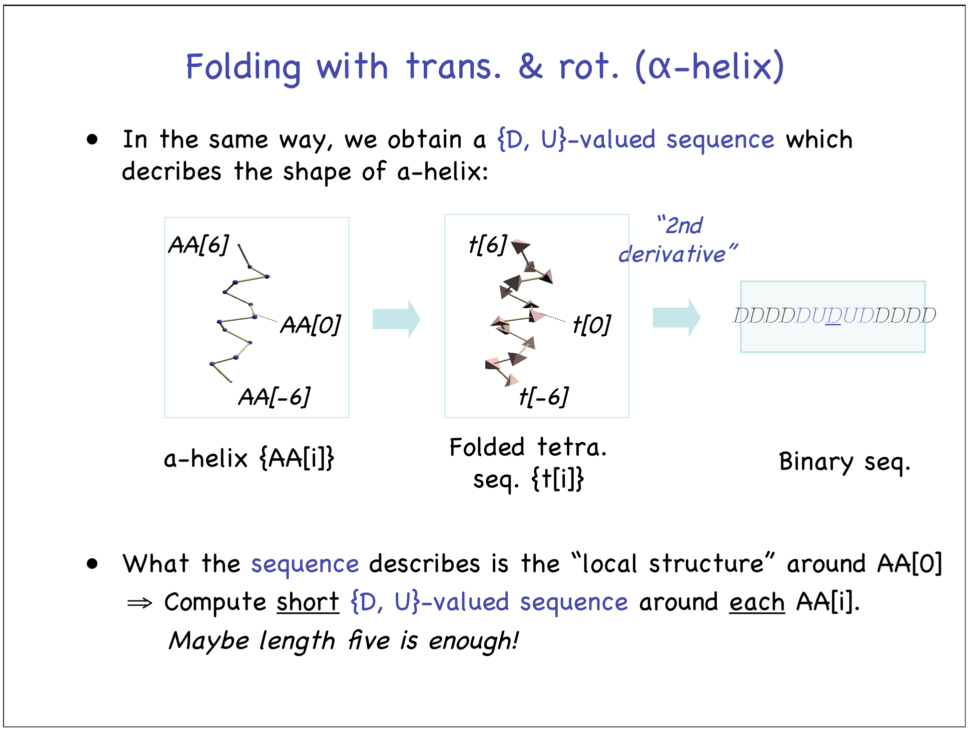
Folded tetra. seq. {t[i]}

Binary seq.

U stands for +1
D stands for -1

And now we obtain ...

That is, the protein on the left is approximated by a folded tetrahedron sequence as shown in the middle. Then, the 2nd derivative of the tetrahedron sequence gives the binary sequence on the right.



Folding with trans. & rot. (α-helix)

- In the same way, we obtain a {D, U}-valued sequence which decribes the shape of a-helix:

a-helix {AA[i]}

Folded tetra. seq. {t[i]}

Binary seq.

- What the sequence describes is the "local structure" around AA[0]
  ⇒ Compute short {D, U}-valued sequence around each AA[i].
  Maybe length five is enough!

In the same way, …

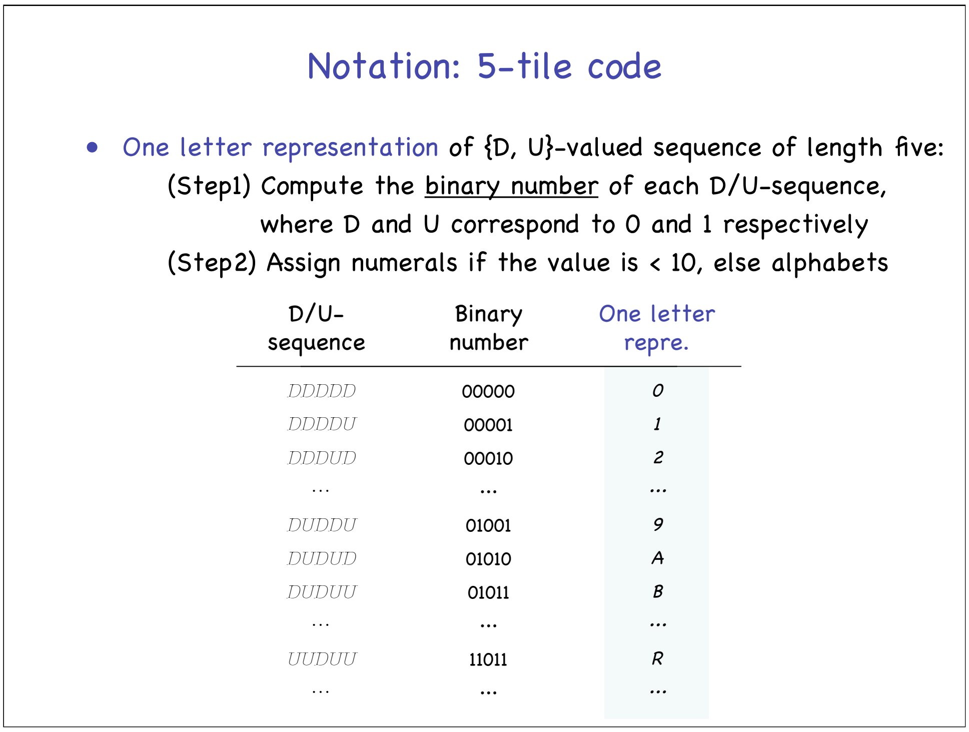
The alpha-helix on the left is approximated by a folded tetrahedron sequence as shown in the middle. Then, the 2nd derivative of the tetrahedron sequence gives the binary sequence on the right.

Note that the four letters at both ends are all D.

That is, what the sequence describe …

And it seems reasonable to compute short …,

where maybe length five is enough, considering the above example.



Notation: 5-tile code

- One letter representation of {D, U}-valued sequence of length five:

  (Step1) Compute the binary number of each D/U-sequence,

  where D and U correspond to 0 and 1 respectively

  (Step2) Assign numerals if the value is < 10, else alphabets

D/U-sequence	Binary number	One letter repre.
DDDDD	00000	*0*
DDDDU	00001	*1*
DDDUD	00010	*2*
…	…	…
DUDDU	01001	*9*
DUDUD	01010	*A*
DUDUU	01011	*B*
…	…	…
UUDUU	11011	*R*
…	…	…

Before computing {D, U}–valued sequence of length five around each amino acid, let me explain the notation used.

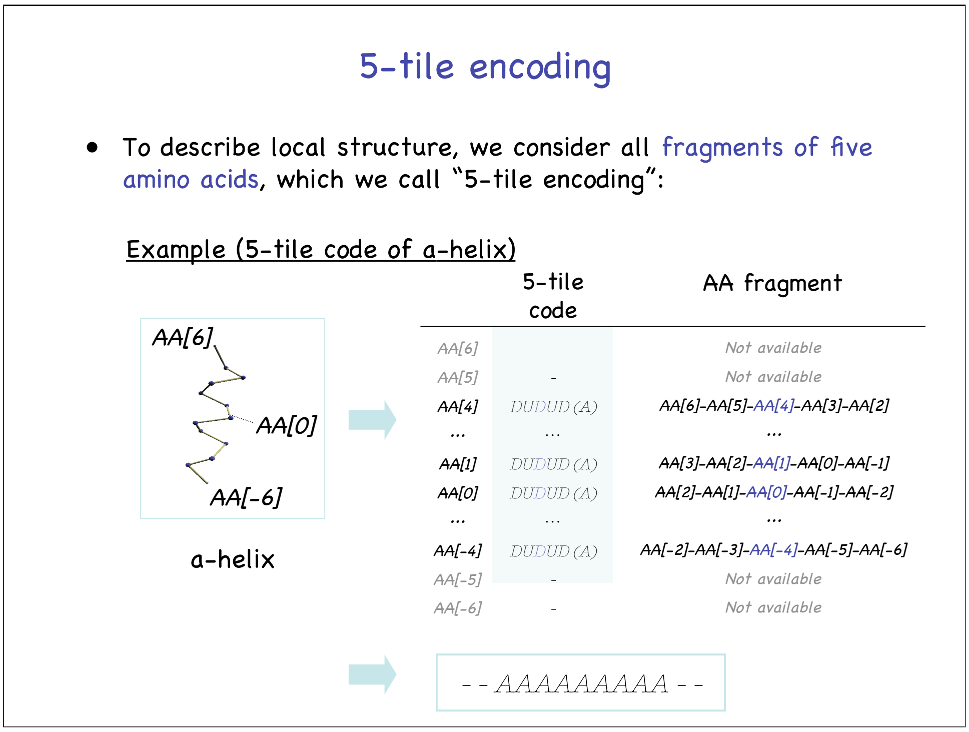
One letter representation of … is defined as follows.

(Step 1) …

For example, the sequence of all D corresponds to binary number 00000 and its value is 0. DDDDU corresponds to binary number 00001 and its value is 1. And so on.

(Step 2) …

For example, DUDDU corresponds to binary number 01001, whose value is 9. Thus numeral ″9″ is assigned. On the other hand, DUDUD corresponds to binary number 01010, whose value is 10. Then, alphabet ″A″ is assigned.



5-tile encoding

- To describe local structure, we consider all fragments of five amino acids, which we call "5-tile encoding":

Example (5-tile code of a-helix)

AA[6]
AA[0]
AA[-6]
a-helix

	5-tile code	AA fragment
AA[6]	-	Not available
AA[5]	-	Not available
AA[4]	DUDUD (A)	AA[6]-AA[5]-AA[4]-AA[3]-AA[2]
…	…	…
AA[1]	DUDUD (A)	AA[3]-AA[2]-AA[1]-AA[0]-AA[-1]
AA[0]	DUDUD (A)	AA[2]-AA[1]-AA[0]-AA[-1]-AA[-2]
…	…	…
AA[-4]	DUDUD (A)	AA[-2]-AA[-3]-AA[-4]-AA[-5]-AA[-6]
AA[-5]	-	Not available
AA[-6]	-	Not available

- - AAAAAAAAA - -

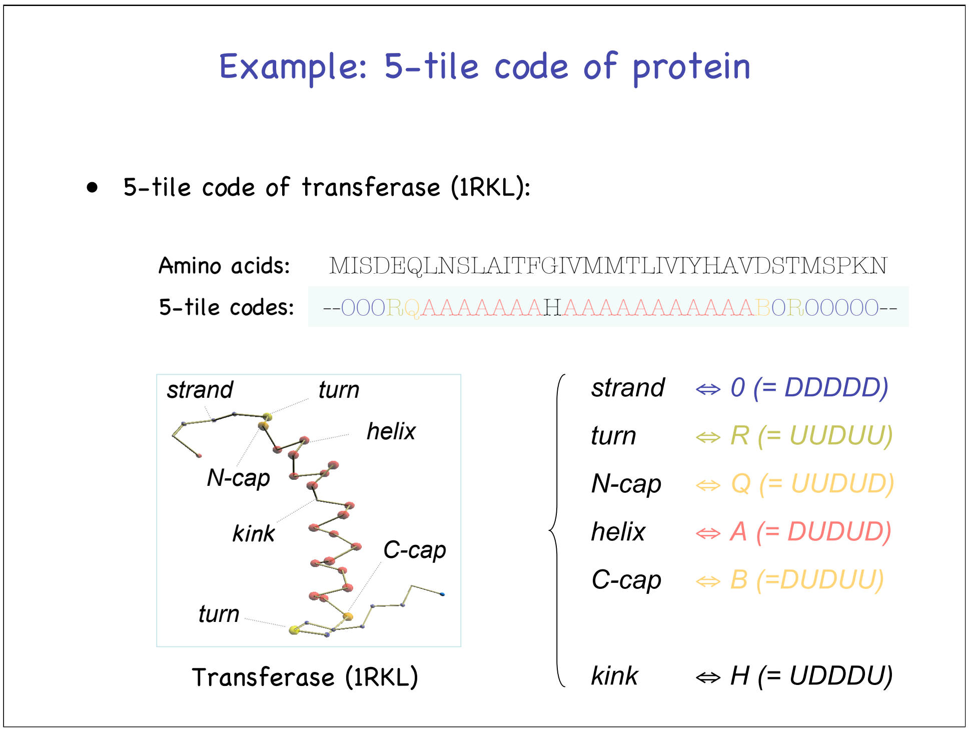
Now to describe …

For example, the ″5−tile code″ of the alpha−helix considered before is computed as follows.

First of all, the two amino acids at both ends have no corresponding amino acid fragment. And ″5−tile code″ is computed for amino acids from AA[4] thorough AA[−4].

As shown on thr right, amino acid AA[4] corresponds to the fragment from AA[6] through AA[2] and its 5−tile code is DUDUD, which is denoted by ″A″. In the same way, amino acid AA[1] corresponds to the fragment from AA[3] through AA[−1] and its 5−tile code is also DUDUD, which is denoted by ″A″. And so on.

Finally, we obtain the sequence of straight ″A″ of length nine as 5−tile code of the alpha−helix, where hyphens stand for the two amino acids at both ends.



Example: 5-tile code of protein

- 5-tile code of transferase (1RKL):

Amino acids: MISDEQLNSLAITFGIVMMTLIVIYHAVDSTMSPKN

5-tile codes: --000RQAAAAAAAHAAAAAAAAAAAB0R00000--

Transferase (1RKL)

strand	⇔ 0 (= DDDDD)
turn	⇔ R (= UUDUU)
N-cap	⇔ Q (= UUDUD)
helix	⇔ A (= DUDUD)
C-cap	⇔ B (=DUDUU)
kink	⇔ H (= UDDDU)

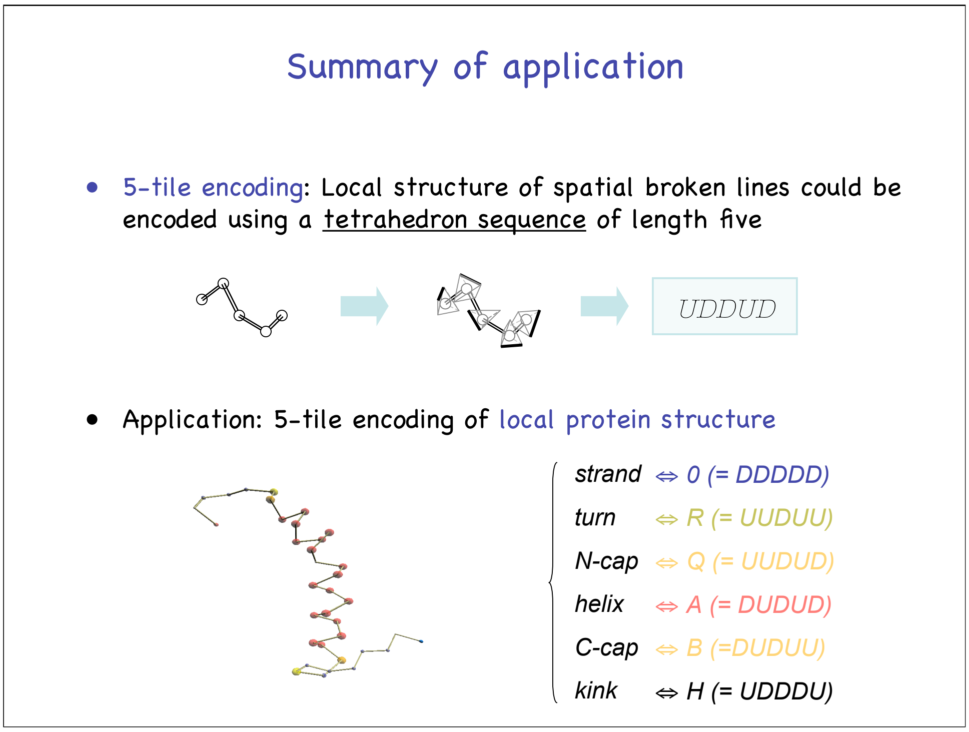
This slide shows an example of the "5–tile code" of an actual protein, transferase (1RKL).

Its amino acid sequence is given above and the 5–tile code is shown below.

The figure shows the structural features of the protein and spatial arrangement of the 5–tile codes, where small blue balls stand for 5–tile code "0", large yellow balls for "R", …

As you see, "strand" corresponds to "0", "turn" corresponds to "R", …

That is, "5–tile code" could capture local features of the protein precisely.



Summary of application

- 5-tile encoding: Local structure of spatial broken lines could be encoded using a tetrahedron sequence of length five

UDDUD

- Application: 5-tile encoding of local protein structure

strand	⇔	0 (= DDDDD)
turn	⇔	R (= UUDUU)
N-cap	⇔	Q (= UUDUD)
helix	⇔	A (= DUDUD)
C-cap	⇔	B (=DUDUU)
kink	⇔	H (= UDDDU)

This is the summary of application.

Firstly, 5–tile encoding is proposed. Local structure …

For example, the spatial broken line shown on the left is approximated by a folded tetrahedron sequence as shown in the middle. And we obain the sequence of UDDUD as its code.

Secondly, as an application, 5–tile encoding of … is proposed.

For example, the local structure of a protein is encoded as shown below. In particular, ″5–tile code″ could distinguish local features, such as strand, turn, …, from each other.

Summary of application

- 5-tile encoding: Local structure of spatial broken lines could be encoded using a tetrahedron sequence of length five

UDDUD

Thank you!

- Application: 5-tile encoding of local protein structure

strand ⇔ 0 (= DDDDD)
turn ⇔ R (= UUDUU)
N-cap ⇔ Q (= UUDUD)
helix ⇔ A (= DUDUD)
C-cap ⇔ B (=DUDUU)
kink ⇔ H (= UDDDU)

Slides & manuscript of the talk are available from
http://www.genocript.com

Thank you for your attention.