

# Bioinformatics 2 -- lecture 4

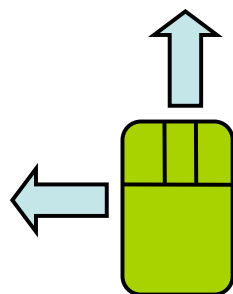
Rotation

Least-squares Superposition

Structure-based alignment algorithms

# 4.1 Rotation

# What happens when you move the mouse to rotate a molecule?



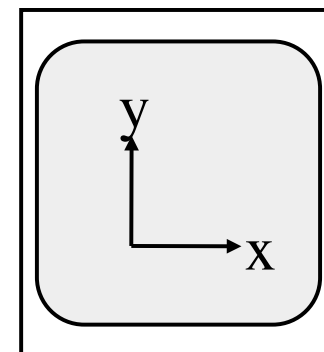
1. Mouse sends mouse coordinates  $(\Delta x, \Delta y)$  to the running program

2. Rotation angles are calculated:  
 $\theta_x = \Delta x * \text{scale}$ ,  $\theta_y = \Delta y * \text{scale}$

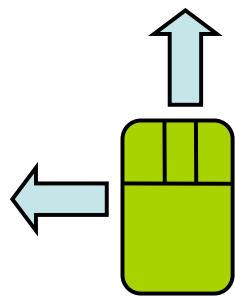
3. Rotation matrices are calculated:

$$\underline{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{pmatrix}$$

$$\underline{R}_y = \begin{pmatrix} \cos \theta_y & 0 & -\sin \theta_y \\ 0 & 1 & 0 \\ \sin \theta_y & 0 & \cos \theta_y \end{pmatrix}$$



# What happens when you move the mouse (cont'd):



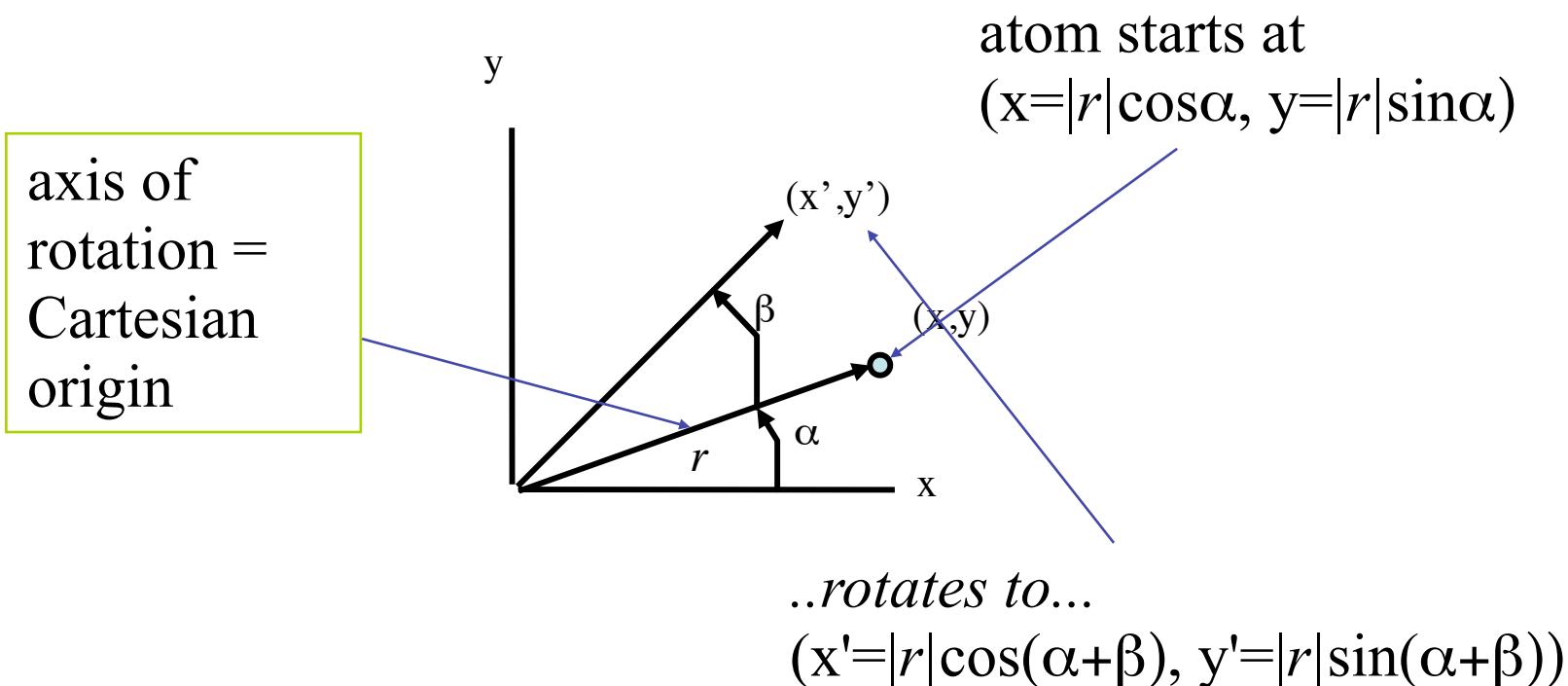
4. New atom coordinates are calculated

$$\vec{r}' = \underline{R}_y \underline{R}_x \vec{r}$$

5. The scene is rendered using the new coordinates.

All of this happens in a fraction of a second.

# Rotation is angular addition



Convention: angles are measured counter-clockwise.

# Sum of angles formuli

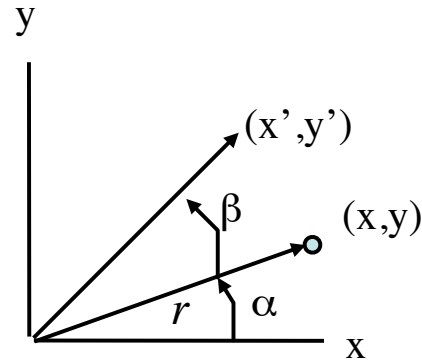
$$\cos (\alpha+\beta) = \cos \alpha \cos \beta - \sin \alpha \sin \beta$$

$$\sin (\alpha+\beta) = \sin \alpha \cos \beta + \sin \beta \cos \alpha$$

# A rotation matrix

$$x = r \cos \alpha$$

$$y = r \sin \alpha$$



$$x' = r \cos (\alpha + \beta)$$

$$= r (\cos \alpha \cos \beta - \sin \alpha \sin \beta)$$

$$= (r \cos \alpha) \cos \beta - (r \sin \alpha) \sin \beta$$

$$= x \cos \beta - y \sin \beta$$

$$y' = r \sin (\alpha + \beta)$$

$$= r (\sin \alpha \cos \beta + \sin \beta \cos \alpha)$$

$$= (r \sin \alpha) \cos \beta + (r \cos \alpha) \sin \beta$$

$$= y \cos \beta + x \sin \beta$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} r \cos \alpha \\ r \sin \alpha \end{pmatrix} = \begin{pmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

rotation matrix is the same for any  $r$ , any  $\alpha$ .

# rotation around a principal axis

The **Z** coordinate stays the same. X and Y change.

$$\underline{\mathbf{R}}_Z = \begin{pmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The **Y** coordinate stays the same. X and Z change.

$$\underline{\mathbf{R}}_Y = \begin{pmatrix} \cos \gamma & 0 & \sin \gamma \\ 0 & 1 & 0 \\ -\sin \gamma & 0 & \cos \gamma \end{pmatrix}$$

The **X** coordinate stays the same. Y and Z change.

$$\underline{\mathbf{R}}_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}$$

# Rotation around two principal axes

Is the product of 2D rotation matrices.

$$\begin{pmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{pmatrix} = \begin{pmatrix} \cos \beta \cos \gamma & -\sin \beta & \cos \beta \\ \sin \beta \cos \gamma & \cos \beta & -\sin \beta \sin \gamma \\ \sin \gamma & 0 & \cos \gamma \end{pmatrix}$$

*Rotation around z*

*Rotation around y*

*3D rotation*

## multiplication order matters.

This is the matrix if the X-rotation is first, then the Y-rotation.

$$\underline{R}_y \underline{R}_x = \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} = \begin{pmatrix} \cos\theta_y & -\sin\theta_x \sin\theta_y & -\sin\theta_y \cos\theta_x \\ 0 & \cos\theta_x & -\sin\theta_x \\ \sin\theta_y & \sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y \end{pmatrix}$$

This is the matrix if the Y-rotation is first, then the X-rotation.

$$\underline{R}_x \underline{R}_y = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix} = \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ -\sin\theta_x \sin\theta_y & \cos\theta_x & -\sin\theta_x \cos\theta_y \\ \sin\theta_y \cos\theta_x & \sin\theta_x & \cos\theta_x \cos\theta_y \end{pmatrix}$$

# Transposing reverses the rotation

For the opposite rotation, flip the matrix.

This is the “transpose”

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^T = \begin{pmatrix} A & C \\ B & D \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

The *inverse* matrix = The *transposed* matrix.

$$\begin{pmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

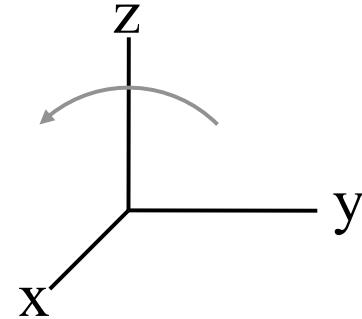
NOTE:  $\cos \beta \cos \beta + \sin \beta \sin \beta = 1$

# Right-handed 90° rotations:

90° rotation around

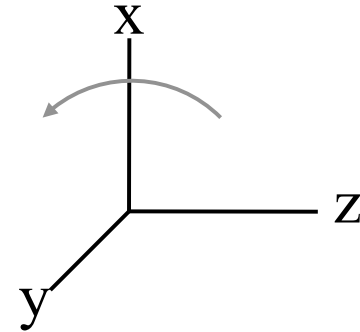
X

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$



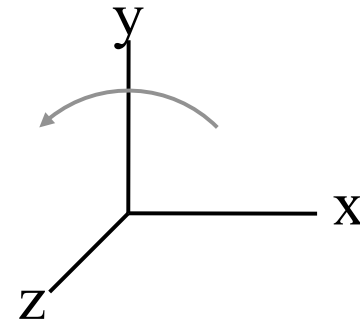
Y

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$



Z

$$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



Helpful hint:

For a R-handed rotation, the *-sine* is up and to the right of the *+sine*.

**In class exercise: rotate a  
point**

$$(x,y,z) = (1., 4., 7.)$$

**Rotate this point by 90° around the Z-axis**

**Then...**

**Rotate the new point by 90° around the Y-axis.**

**What are the new coordinates?**

supplementary slides: 3D rotation conventions:

axis of rotation:  $z''$  **Euler angles,  $\alpha \beta \gamma$**   $x$   $z$

Order of rotations:  $3$   $2$   $1$

$$\begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & -\sin \beta \\ 0 & \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Each rotation is around a principle axis.

$z''''$  **Polar angles,  $\phi \psi \kappa$**   $y''''$   $z''$   $-y'$   $-z$

$$\begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & 0 & -\sin \varphi \\ 0 & 1 & 0 \\ \sin \varphi & 0 & \cos \varphi \end{pmatrix} \begin{pmatrix} \cos \kappa & -\sin \kappa & 0 \\ \sin \kappa & \cos \kappa & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{pmatrix} \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

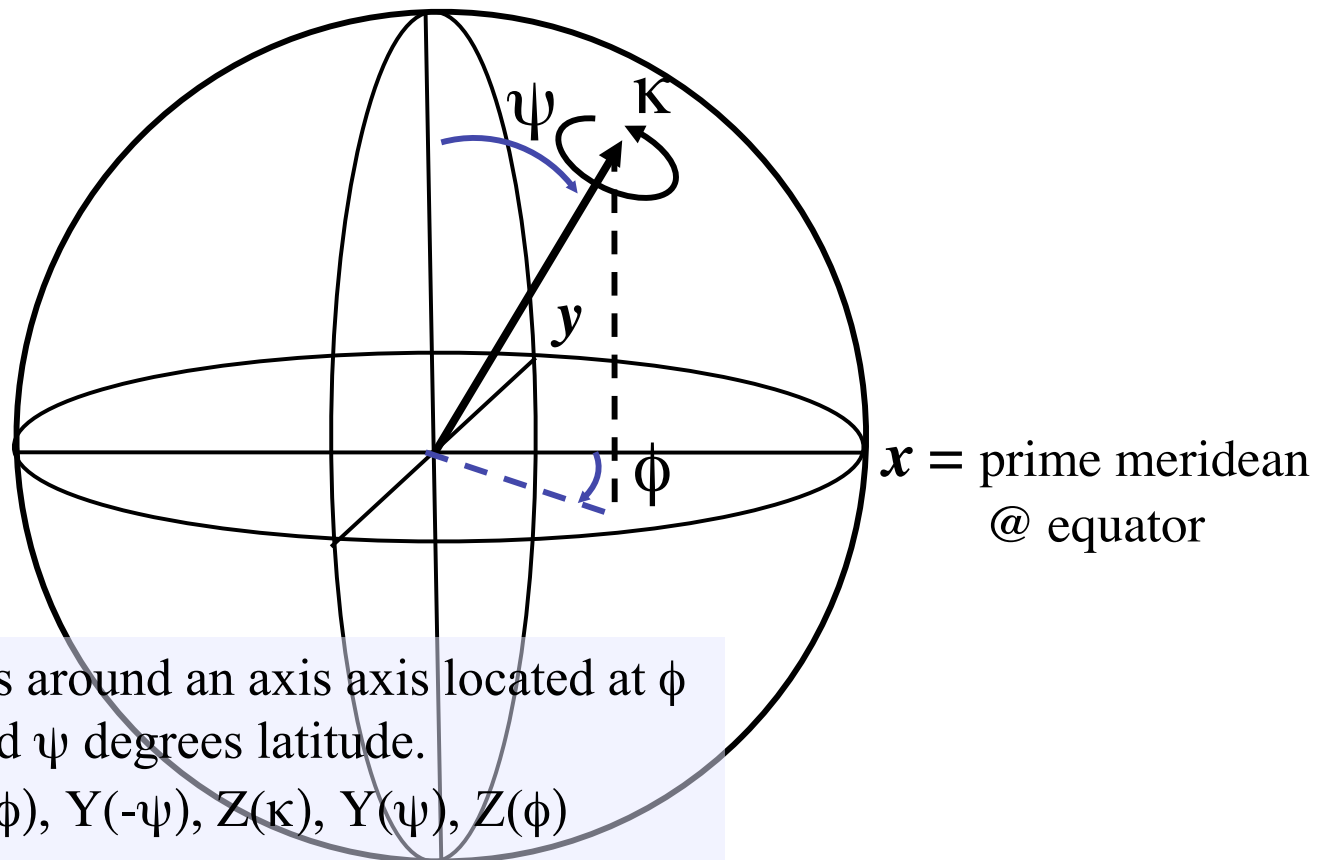
$5$   $4$   $3$   $2$   $1$

Net rotation =  $\kappa$ , around an axis axis defined by  $\phi$  and  $\psi$

supplementary slides: Polar angle convention:

$$\begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & 0 & -\sin \varphi \\ 0 & 1 & 0 \\ \sin \varphi & 0 & \cos \varphi \end{pmatrix} \begin{pmatrix} \cos \kappa & -\sin \kappa & 0 \\ \sin \kappa & \cos \kappa & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{pmatrix} \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$z = \text{north pole}$



Rotation of  $\kappa$  degrees around an axis located at  $\phi$  degrees longitude and  $\psi$  degrees latitude.

Rotation order is  $Z(-\phi)$ ,  $Y(-\psi)$ ,  $Z(\kappa)$ ,  $Y(\psi)$ ,  $Z(\phi)$

Notice the nested rotations.

## supplementary slides: Special properties of rotation matrices

- They are square, 2x2 or 3x3 (higher dimensions in principle)
- The product of any two rotation matrices is a rotation matrix.
- The inverse equals the transpose,  $R^{-1} = R^T$
- Every row/column is a *unit vector*.
- Any two rows/columns are *orthogonal vectors*.
- The *cross-product* of any two rows equals the third.
- $|x| = |Rx|$ , where  $R$  is a rotation matrix.

Read more about rotation matrices at: <http://mathworld.wolfram.com/RotationMatrix.html>

supplementary slides: Principal axes unit vectors are matrix columns.

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a & d & g \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} b & e & h \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c & f & i \end{pmatrix}$$

You can create a rotation matrix by defining three mutually orthogonal unit<sup>17</sup> vectors, then lining them up side-by-side in a 3x3 matrix.

# 4.2 Least Squares Superposition

# RMSD

Root Mean Square Deviation in superimposed coordinates is the standard measure of **structural difference**.

$$\sqrt{\frac{\sum_{i=1, N} (\vec{x}_i - \vec{y}_i)^2}{N}}$$

Where  $x_i$  and  $y_i$  are the *equivalent\* coordinates* from molecules 1 and 2, respectively.

\*Defined by an **alignment**.

# Least squares superposition

*Problem:* find the rotation matrix,  $\underline{M}$ , and a vector,  $v$ , that minimize the following quantity:

$$\sum_i \left| \underline{M} \vec{x}_i + \vec{v} - \vec{y}_i \right|^2$$

Where  $x_i$  and  $y_i$  are the *equivalent coordinates* from molecules 1 and 2, respectively.

In comparative modeling

## Alignment defines structural equivalence.

Any position that is aligned is included in the sum of squares.

4DFR:A ISLIAALAVDRVIGMENAMPWNLPA DLAWFKRNTLDKPVIMGRHTWESIG-RPLPGRKNI  
1DFR:\_ TAFLWAQNRNGLIGKDGHLPWHL PDDLHYFRAQT V GKIMVVGRRTYESFPKRPLPERTNV

4DFR:A ILSSQ-PGTDDRVTWVKSVD EAIAC--GDVPEIMVIGGGRVYE QFLPKAQKLYLTHIDA  
1DFR:\_ VLTHQEDYQAQGAVVVHDVA AVFAYAKQHLDQELVIAGGAQIF TAFKDDVD TLLVTRLAG

4DFR:A EVEGDTHFPDYEPDDWESVFSEFHDADAQNS--HSYCFKILERR  
1DFR:\_ SFEGDTKMIPLNWDDFTKVSSRTVEDT---NPALHTHTYEVWQKK

Unaligned positions are not.

# Finding structural equivalences by linear Least squares

(1) At the position of best superposition, we have an approximate equality:

$$\underline{M}\vec{x}_i + \vec{v} \cong \vec{y}_i$$

(2) We can eliminate  $v$  by translating the center of mass of both molecules to the origin. The equation simplifies to:

$$\underline{M}\vec{x}'_i \cong \vec{y}'_i$$

We have one equation ( $i$ ) for each atom,  $M$  has 9 unknowns.

If there are more equations than unknowns, there is a unique solution.

# Least squares

Least squares solves a set of a linear equations in the form :

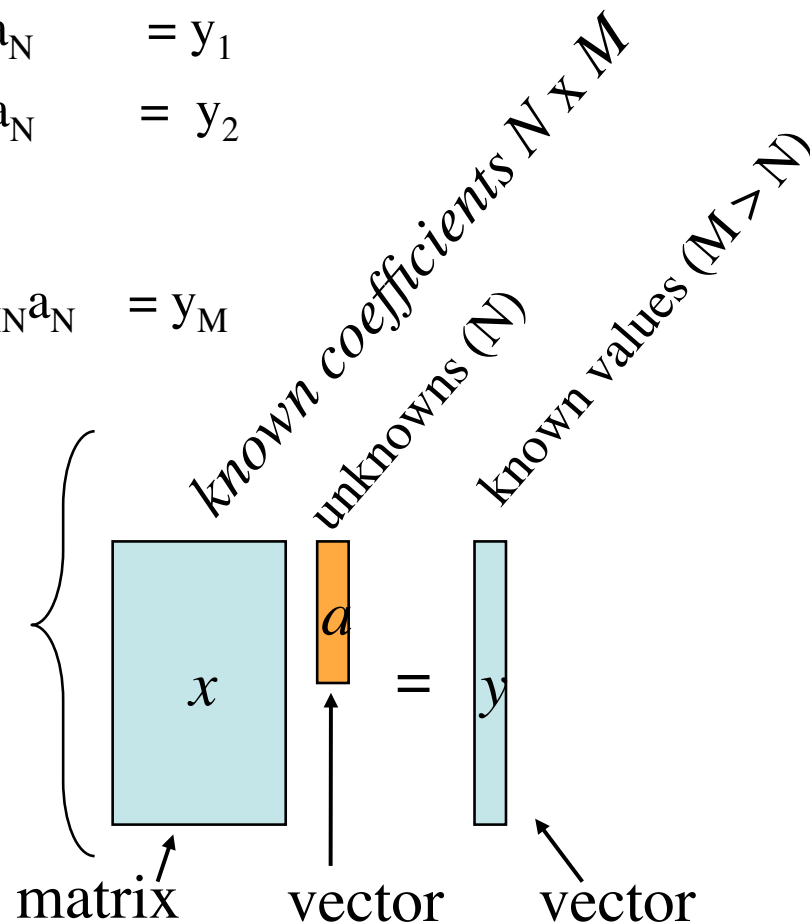
$$x_{11}a_1 + x_{12}a_2 + \dots + x_{1N}a_N = y_1$$

$$x_{21}a_1 + x_{22}a_2 + \dots + x_{2N}a_N = y_2$$

...

$$x_{M1}a_1 + x_{M2}a_2 + \dots + x_{MN}a_N = y_M$$

This is 'shorthand'  
notation for the  
equations.



# Least squares, continued

$$\begin{array}{c} \text{Fat Green} \\ x \end{array} \begin{array}{c} \text{Thin Orange} \\ a \end{array} = \begin{array}{c} \text{Thin Green} \\ y \end{array}$$

Green elements are known.

Orange are unknown.

Fat Rectangles are matrices.

Thin rectangles are vectors.

$$\begin{array}{c} \text{Fat Green} \\ x^T \end{array} \begin{array}{c} \text{Fat Green} \\ x \end{array} \begin{array}{c} \text{Thin Orange} \\ a \end{array} = \begin{array}{c} \text{Fat Green} \\ x^T \end{array} \begin{array}{c} \text{Thin Green} \\ y \end{array}$$

Multiply both sides by transpose of  $x$ . "Squaring"

$$\begin{array}{c} \text{Fat Green} \\ x^T x \end{array} \begin{array}{c} \text{Thin Orange} \\ a \end{array} = \begin{array}{c} \text{Thin Green} \\ x^T y \end{array}$$

"Squared" matrix can be inverted. (We can use the "LU decomposition".)

$$\begin{array}{c} \text{Fat Green} \\ (x^T x)^{-1} \end{array} \begin{array}{c} \text{Fat Green} \\ x^T x \end{array} \begin{array}{c} \text{Thin Orange} \\ a \end{array} = \begin{array}{c} \text{Fat Green} \\ (x^T x)^{-1} \end{array} \begin{array}{c} \text{Thin Green} \\ x^T y \end{array}$$

Multiplying both sides by the inverse of "squared" matrix solves for  $a$ .

$$\begin{array}{c} \text{Thin Orange} \\ a \end{array} = \begin{array}{c} \text{Fat Green} \\ (x^T x)^{-1} \end{array} \begin{array}{c} \text{Thin Green} \\ x^T y \end{array}$$

Summary:  $a = (x^T x)^{-1} x^T y$

# least-squares superimposed molecules



## 4.3 Structure-based alignment algorithms

# Chicken/Egg circular logic in structure-based alignment?

- Least squares superposition defines the alignment.
- Least squares superposition depends on the alignment.

# Structural alignment algorithm types

Alignment algorithms create a one-to-one mapping of subset(s) of one sequence to subset(s) of another sequence.

## ***Structure-based alignment types:***

### **Geometric--intermolecular**

Algorithms may do this by minimizing the intermolecular *distances* or root-mean-square deviation (*rmsd*) in superimposed alpha-carbon positions.

### **Geometric--intramolecular**

Algorithms minimize the difference between aligned contact maps or distance matrices. Intramolecular distances are used

### **Non-Geometric**

Algorithms align structural properties, such as %buried, or secondary structure type, usually using dynamics programming (DP)

# Structure-based alignment tools

## Programs

DALI

VAST

CE

\*KENOBI

MAMMOTH

PRiSM

\*SCALI

\*SARF

## Databases

FSSP

HOMSTRAD

COMPASS

PALI

\*SARF, SCALI and KENOBI do *non-sequential* alignment.

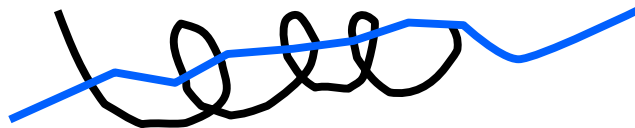
# What a structure-based alignment *should* mean

**Aligned residues are structurally similar** (i.e. same secondary structure)

**Pairs of aligned residues have the same contact property** (either both *in contact* or both *not in contact*)

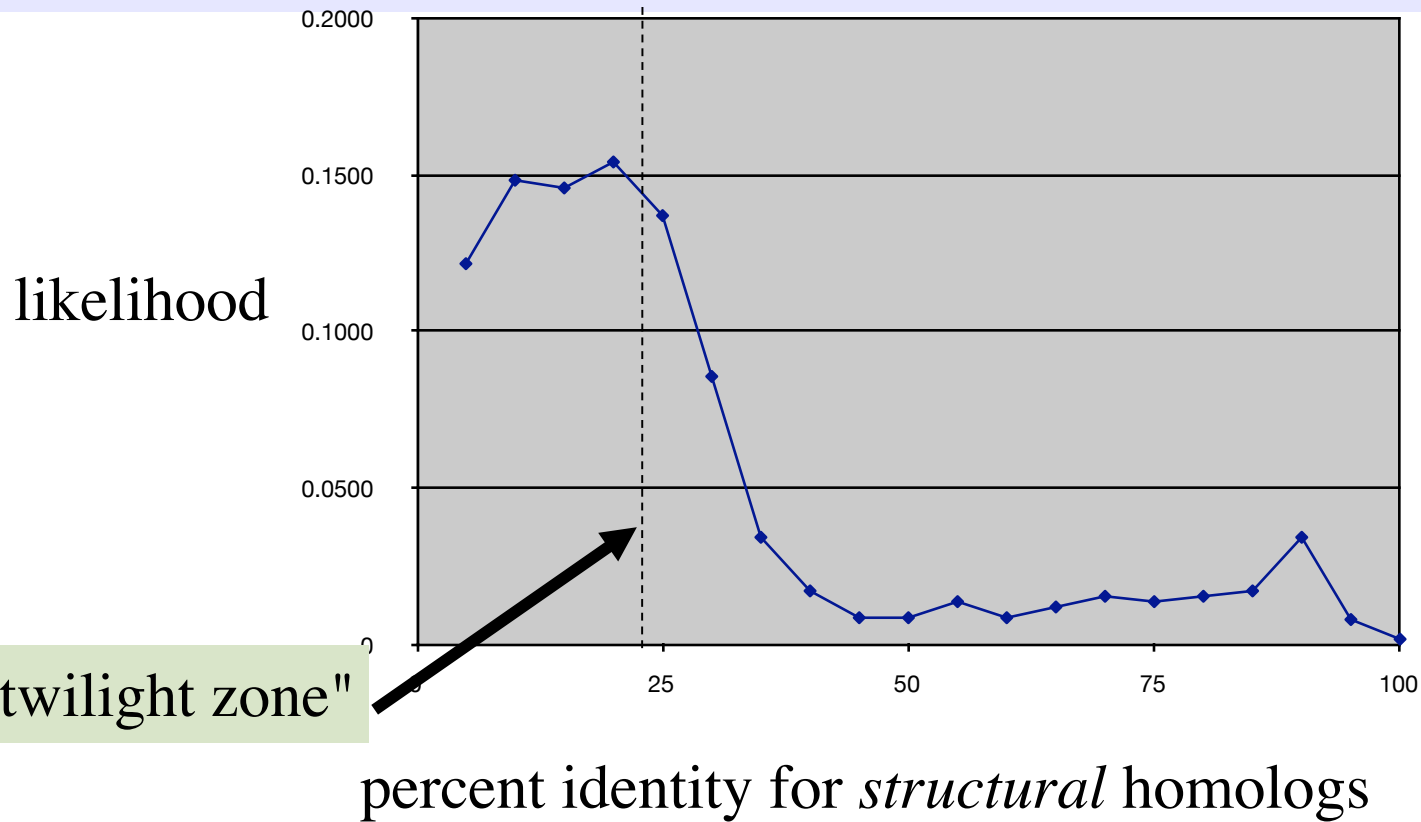
## What it often does mean:

Aligned residues *superimpose in space*.



# Remote homologs are more likely than close homologs

The existence of large numbers of remote homologs shows us that true structural similarity is *hard to see* in the amino acid sequence. Structural conservation is stronger than sequence conservation.



# Example of structural homologs (analog)

4DFR: Dihydrofolate reductase

1YAC: Octameric Hydrolase Of Unknown Specificity

**5.9% sequence identity (best alignment)**

**1YAC structure solved without knowing function.**

**Alignment to 4DFR and others implies it is a hydrolase of some sort, probably uses NAD cofactors.**

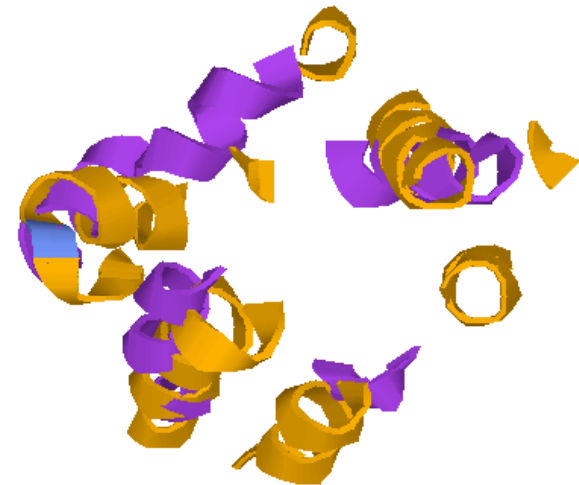
## Viewing structural homologs (analogs)



**DHFR in yellow and orange. YAC in green and purple**



**sheets only**



**helices only**

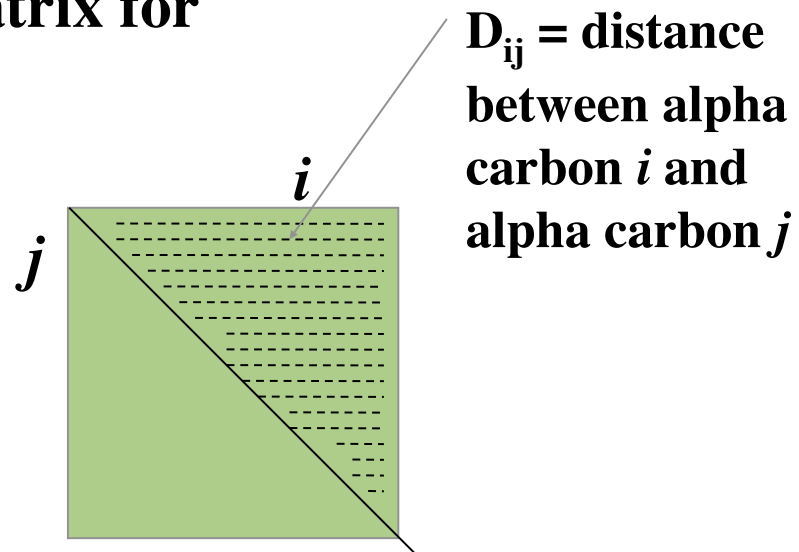
# DALI: a intramolecular geometric structural alignment algorithm

DALI: (Distance matrix-based ALIgnment)

Liisa Holm & Chris Sander

**(1) Generate a distance matrix for each protein**

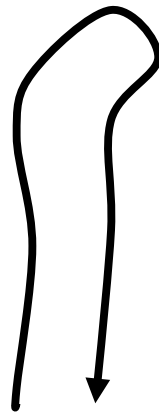
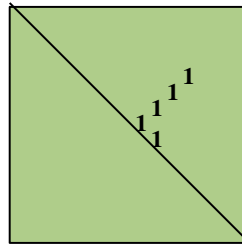
**The distance matrix contains all pairwise distances.(symmetrical)**



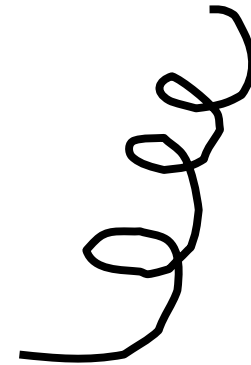
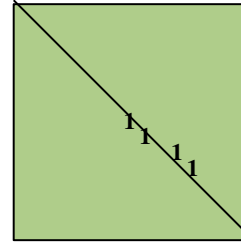
**Geometric--intramolecular**

# Shapes in distance matrices

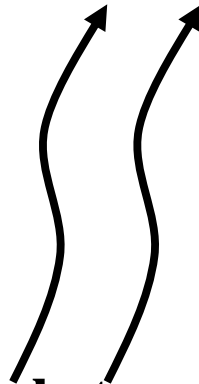
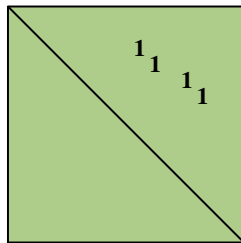
In a Contact Map “1” means close in space, typically  $D_{ij} < 8\text{\AA}$



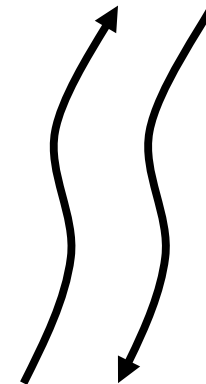
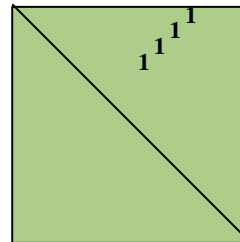
**hairpin**



**helix**



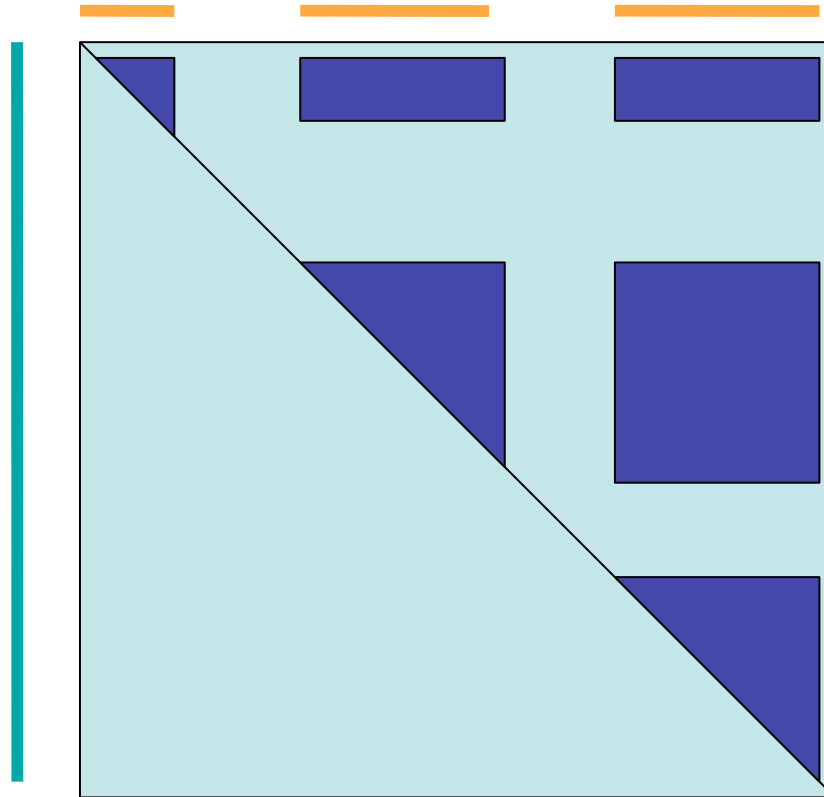
**parallel strands**



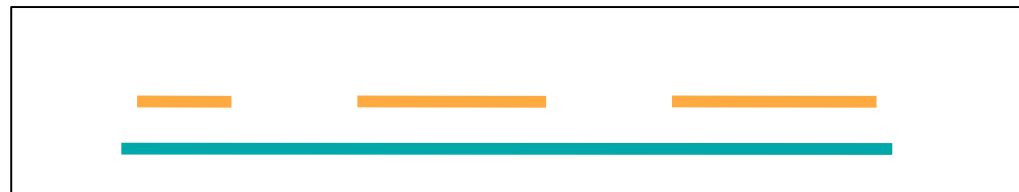
**anti-parallel strands**

# Aligning two distance matrices

**Cut-and-paste  
alignment  
of distance  
matrices**



**Resulting  
sequence  
alignment**



# DALI algorithm

DALI optimizes the S score.

$$S = \sum_{i=1}^L \sum_{j=1}^L \phi^R(i, j)$$

where,

$$\phi^R(i, j) = \theta^R - |d_{ij}^A - d_{ij}^B|$$

**Phi is a constant (theta) minus the absolute difference of the aligned distances.**

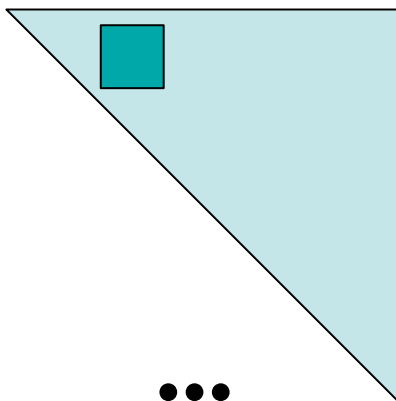
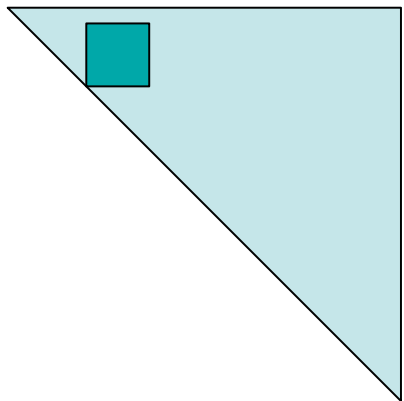
**S is the highest when  $d_{ij}^A$  is equal to  $d_{i',j'}^B$ .**

...where  $i$  in A is aligned to  $i'$  in B and  $j$  in A is aligned to  $j'$  in B

DALI algorithm:

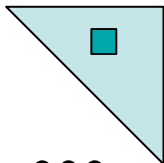
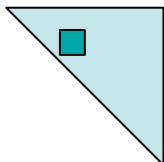
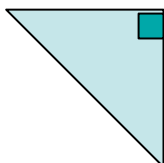
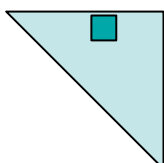
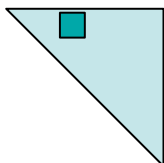
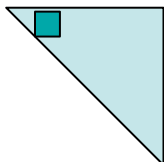
**Making the pairs list**

**Structure A**



...

**Structure B**



...

**S (2 6x6's)**

45.3

13.2

56.2

33.3

20.2

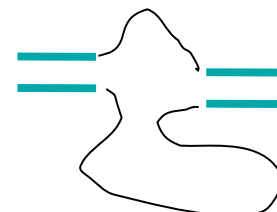
12.5

...

**VS**

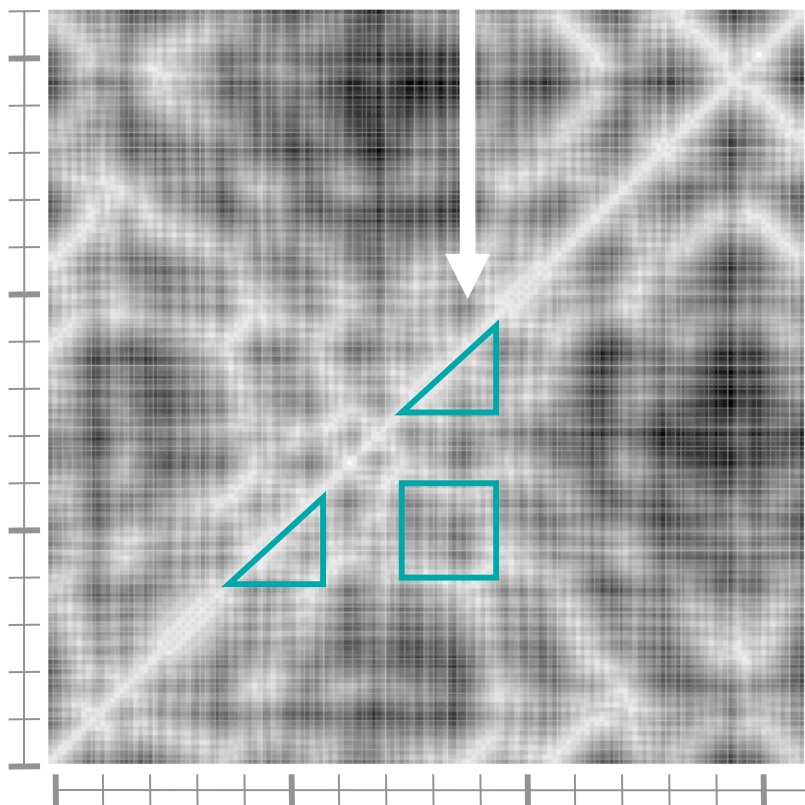
**high scores to pairs list**

Each pair of 6x6's corresponds to a gapped alignment



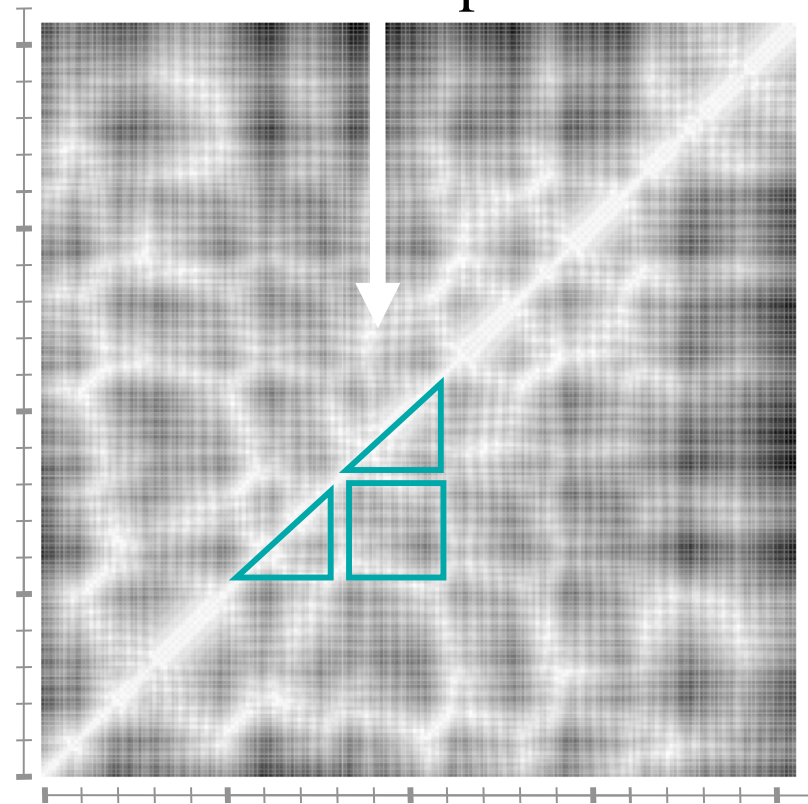
# DALI algorithm: start with 6x6 pairs from *pairs list*

one 6x6 pair



**4dfr**

one 6x6 pair

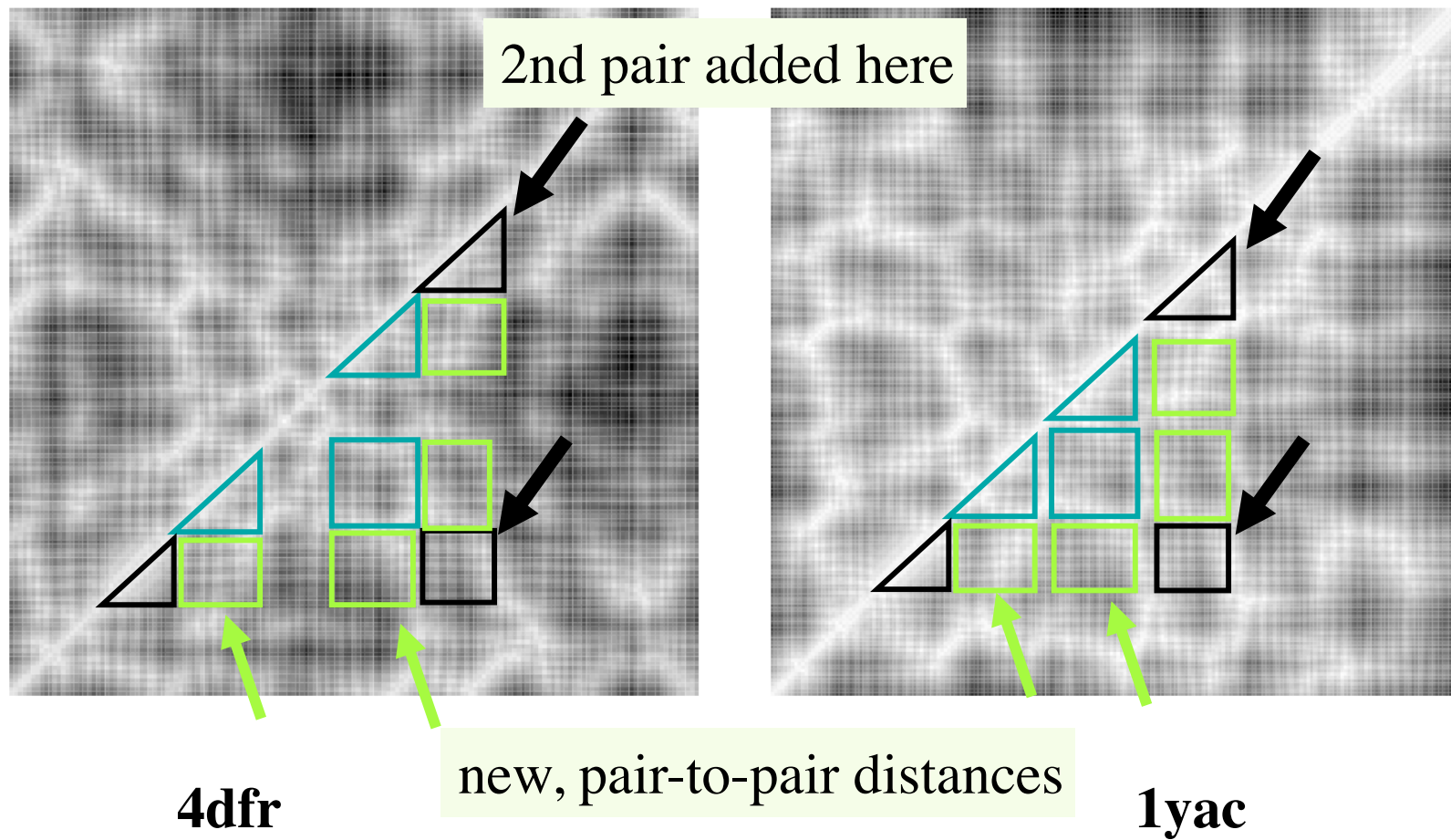


**1yac**

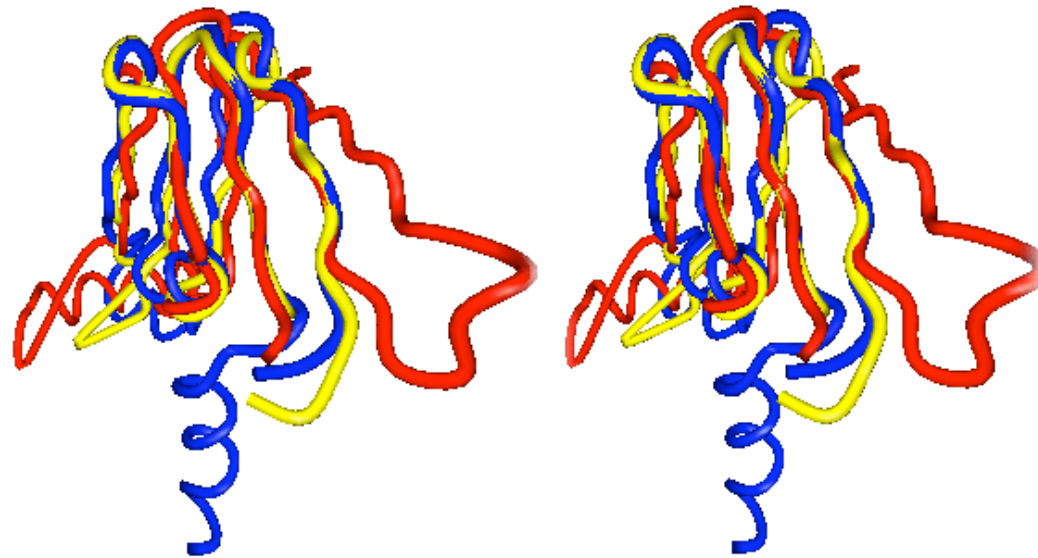
Axes = sequence position  
shade=distance,  
shorter distances are lighter shade.

# DALI algorithm

extend using another pair,  
consider new blocks



# DALI alignment output



DR2_3-91:	DR2_92-155:	DR2_156-230:	DR2_3-91:	DR2_92-155:	DR2_156-230:
3	92	156	..	TP	FG
			DMRMRFTIDQNM	GQIVALELGE	SIKKITLDGGT
			QFPLVEIDLEHGGSVYL	KQYRLNDGAF	NAHVVAUSRELDYDIHLE
			QOQGS	LALDGS	RELDYDIHLE
			MVYHTE	YKMERQ	DIHLE
			-NVT	---	---
			LN	---	---
			TKL	---	---
			GLGLKLV	---	---
			GAI	---	---
			GRSMV	---	---
			SGESM	---	---
			FFIT	---	---
			QOQMS	---	---
			NGDGKL	---	---
			LALAPN	---	---
			.....	---	---
				L-FV	---
				MTTEG	---
				LG	---
				TLLANS	---
				---	---
				EQF	---
				AG	---
				TL	---
				KRYL	---

# Uses of structural alignment in modeling

- A structure-based alignment is the **Gold Standard** for a sequence alignment.
- Aligned structures tell you where **structurally conserved regions** are, versus where insertions/deletions are allowed.
- Structural analogs provide a source of **plausible loop structures**.
- Multiple aligned structures show **evolutionary plasticity**.

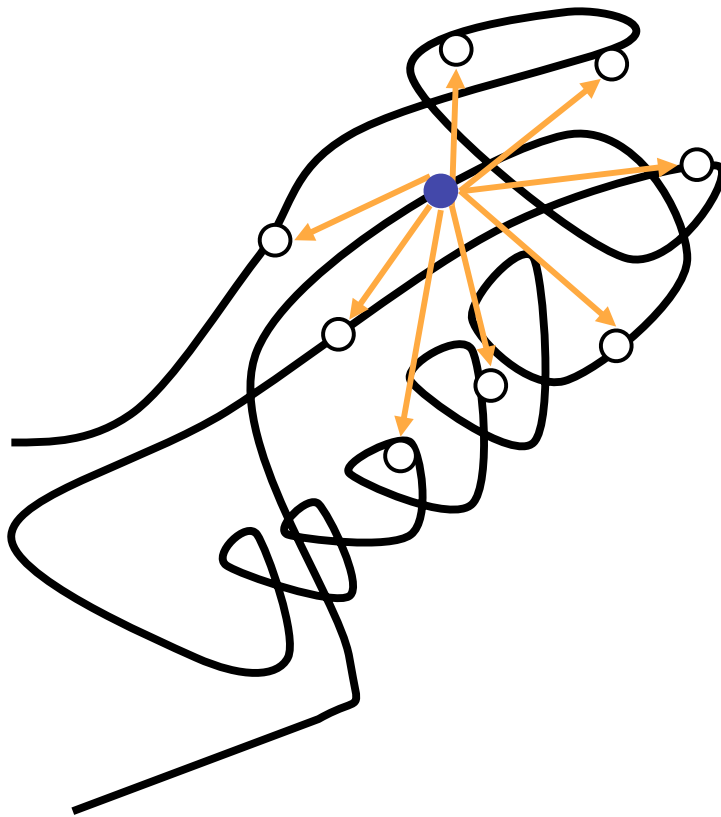
## Exercise 4.1: Superimpose

- Open two PDB files: **2ptl.pdb**, **2gb1.pdb**
- Use the first chain if there are multiple chains.
- SEQ:Alignment/Superpose, Set “Force Realignment”.
- Find the best algorithm for superposition.

Do these pairs:	2ptl.pdb	2gb1.pdb	
	3sdh.pdb	1h97.pdb	
	3sdh.pdb	1phn.pdb	(hard)

## 4.4 Supplementary slides

# SSAP alignment

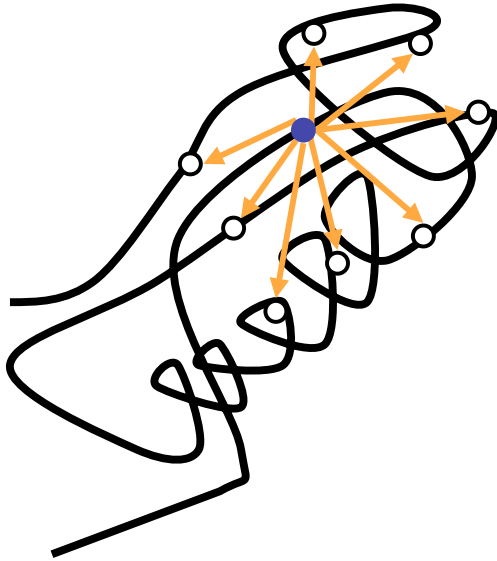


A View is the set of all vectors from one residue.

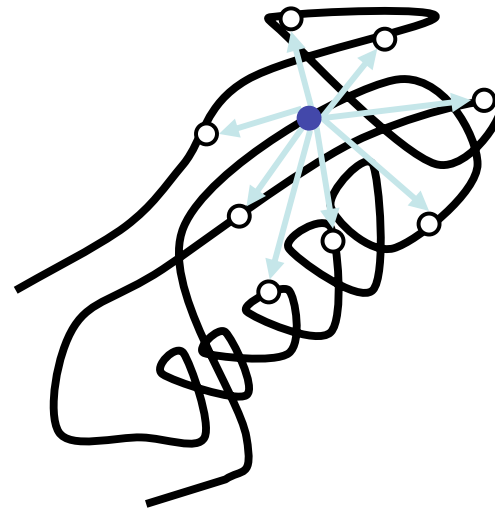
Each residue has its own "View", which is a set of vectors to nearest neighbor residues.

**Algorithm type: Geometric--intramolecular**

# SSAP alignment: views



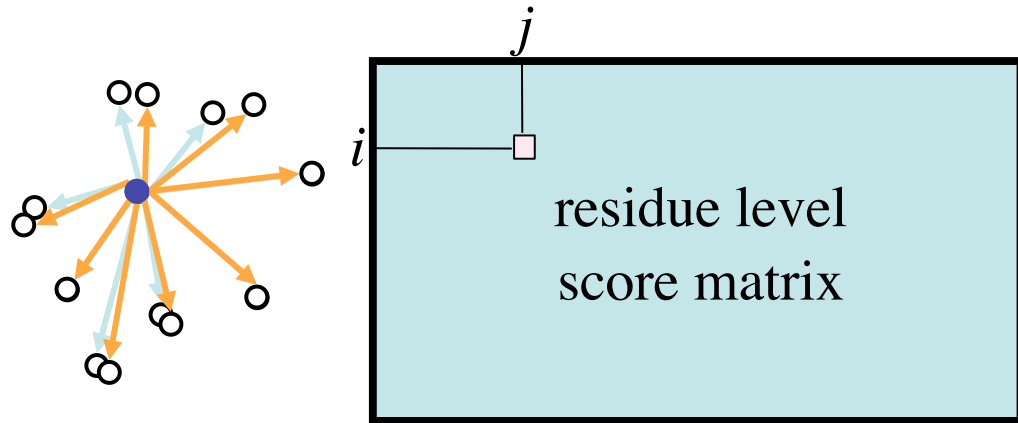
View for Template residue  $i$



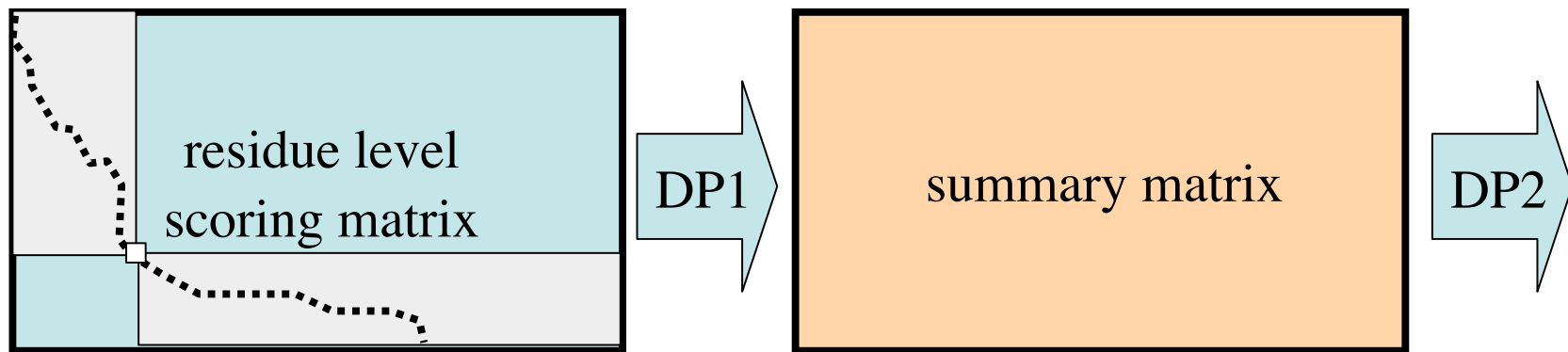
View for Target residue  $j$

$i$  and  $j$  must have similar backbone angles, otherwise the score is zero.

The difference between the two *views* is a measure of how similar the structures are, when viewed from  $i$  and  $j$ .



# SSAP algorithm: Double Dynamic Programming



For each  $ij$  pair, we find the **best DP alignment that includes  $ij$** . Keep the DP score at position  $(i,j)$  in the “summary matrix”.

The “summary matrix” is subjected to a **second round of DP**, to give the optimal alignment.

# Two other servers for structure-based alignment

CE: Combinatorial Extension

<http://cl.sdsc.edu/ce.html>

VAST:

<http://www.ncbi.nlm.nih.gov/Structure/VAST/vast.shtml>

**Algorithm type: Geometric--intermolecular**

Location: [http://cl.sdsc.edu/ce\\_scratch/ata2912.html](http://cl.sdsc.edu/ce_scratch/ata2912.html)

# CE interface

?	ID	Z-Score	RMSD(Å)	Seq.(%)	Aligned/ Size	Gap	Exp.	Name	
	<a href="#">2DRC:A</a>	Representative						X-Ray	DIHYDROFOLATE REDUCTASE (DHFR) (E.C.1.5.1.3) MUTANT WITH TRP 22 REPLACED BY PHE
<input checked="" type="checkbox"/>	<a href="#">IBZF:</a> <a href="#">Neighbors</a>	6.3	2.0	27.7	155 / 162	9	X-Ray	MOL_ID: 1; MOLECULE: DIHYDROFOLATE REDUCTASE; CHAIN: NULL; SYNONYM: DHFR; EC: 1.	
<input checked="" type="checkbox"/>	<a href="#">ICZ3:A</a> <a href="#">Neighbors</a>	6.1	2.1	24.3	148 / 168	26	X-Ray	MOL_ID: 1; MOLECULE: DIHYDROFOLATE REDUCTASE; CHAIN: A, B; EC: 1.5.1.3; ENGINEER	
<input checked="" type="checkbox"/>	<a href="#">IDRF:</a> <a href="#">Neighbors</a>	6.0	2.0	29.5	156 / 186	30	X-Ray	DIHYDROFOLATE REDUCTASE (E.C.1.5.1.3) COMPLEX WITH FOLATE	
<input checked="" type="checkbox"/>	<a href="#">IAI9:B</a> <a href="#">Neighbors</a>	5.9	2.1	30.1	153 / 192	41	X-Ray	MOL_ID: 1; MOLECULE: DIHYDROFOLATE REDUCTASE; CHAIN: A, B; SYNONYM: DHFR; EC: 1.	
<input checked="" type="checkbox"/>	<a href="#">IDAJ:</a> <a href="#">Neighbors</a>	5.7	1.9	30.6	157 / 206	45	X-Ray	MOL_ID: 1; MOLECULE: DIHYDROFOLATE REDUCTASE; CHAIN: NULL; EC: 1.5.1.3; ENGINEER	
<input checked="" type="checkbox"/>	<a href="#">IYAC:A</a> <a href="#">Neighbors</a>	4.4	4.6	5.9	118 / 208	51	X-Ray	MOL_ID: 1; MOLECULE: YCAC GENE PRODUCT; CHAIN: A, B; SYNONYM: YCACGP; BIOLOGICAL	
<input checked="" type="checkbox"/>	<a href="#">IJPU:A</a> <a href="#">Neighbors</a>	4.2	4.0	9.9	91 / 370	39	X-Ray	MOL_ID: 1; MOLECULE: GLYCEROL DEHYDROGENASE; CHAIN: A; EC: 1.1.1.6; ENGINEERED:	
<input checked="" type="checkbox"/>	<a href="#">IPOW:A</a> <a href="#">Neighbors</a>	4.1	3.3	6.4	93 / 587	36	X-Ray	PYRUVATE OXIDASE (E.C.1.2.3.3) (WILD TYPE)	
<input checked="" type="checkbox"/>	<a href="#">IGSO:A</a> <a href="#">Neighbors</a>	4.1	3.8	9.6	73 / 431	12	X-Ray	MOL_ID: 1; MOLECULE: GLYCINAMIDE RIBONUCLEOTIDE SYNTHETASE; CHAIN: A; SYNONYM: P	
<input checked="" type="checkbox"/>	<a href="#">IZPD:A</a> <a href="#">Neighbors</a>	4.1	3.9	7.1	98 / 568	35	X-Ray	MOL_ID: 1; MOLECULE: PYRUVATE DECARBOXYLASE; CHAIN: A, B, E, F; EC: 4.1.1.1; ENG	
<input checked="" type="checkbox"/>	<a href="#">IXVA:A</a> <a href="#">Neighbors</a>	4.1	4.0	8.5	94 / 292	32	X-Ray	MOL_ID: 1; MOLECULE: GLYCINE N-METHYLTRANSFERASE; CHAIN: A, B; SYNONYM: GNMT; S-	
<input checked="" type="checkbox"/>	<a href="#">IB8G:A</a> <a href="#">Neighbors</a>	4.1	4.1	8.1	99 / 429	56	X-Ray	MOL_ID: 1; MOLECULE: 1-AMINOCYCLOPROPANE-1-CARBOXYLATE SYNTHASE; CHAIN: A, B; SY	
<input checked="" type="checkbox"/>	<a href="#">IIG3:A</a> <a href="#">Neighbors</a>	4.1	4.2	6.7	104 / 263	55	X-Ray	MOL_ID: 1; MOLECULE: THIAMIN PYROPHOSPHOKINASE; CHAIN: A, B; EC: 2.7.6.2; ENGINE	
<input checked="" type="checkbox"/>	<a href="#">IFBN:A</a> <a href="#">Neighbors</a>	4.1	4.4	10.3	116 / 230	87	X-Ray	MOL_ID: 1; MOLECULE: MJ FIBRILLARIN HOMOLOGUE; CHAIN: A; ENGINEERED: YES	
<input checked="" type="checkbox"/>	<a href="#">IIC3:A</a> <a href="#">Neighbors</a>	4.1	5.0	9.2	130 / 285	107	X-Ray	MOL_ID: 1; MOLECULE: ABD1 PROTEIN; CHAIN: A; SYNONYM: MRNA (GUANINE-N7)-METHYL T	

# CE alignment

## Structure Alignment - 2DRC:A Neighbors

Sequence alignment based on structure alignment between 2DRC:A and its neighbors. Light color indicates not-aligned residues in structural neighbors. Position numbers according to sequ PDB are given as \$\$\$\$/PPPP, \$\$\$\$ - sequence, PPPP - PDB.

```

2DRC:A  16/17  MENAM---PFNL-PADLAWFKRNTL---DKFVIMGRH----TWESIG--RPLPGRKNIIL
LYAC:A  24/25  LLSLVRDIEFDKFKNNVLALGLAKYFNLPTILITSAITGPNGLVPEYKAQFPDAPYIA

2DRC:A  63/64  SSQPGTDDRVTWVK----SVDEAIAAC-GDVP-EIMVIGGG--RVYEQFLPKA-----QK
LYAC:A  84/85  RFG-----NINARDNEDFVKAVRATGKQLIAGVVFVVCVAFPAISATREGIDV

2DRC:A  110/111 LYLTHIDAEVEGDTHFPDYE--PDDWESVFSEFHDADAQNSHSYCFEIL
LYAC:A  134/135 FVVTDASGTFNEITRHSAWDRMSQAGAQLMT-----WFGVA
  
```

LYAC:A

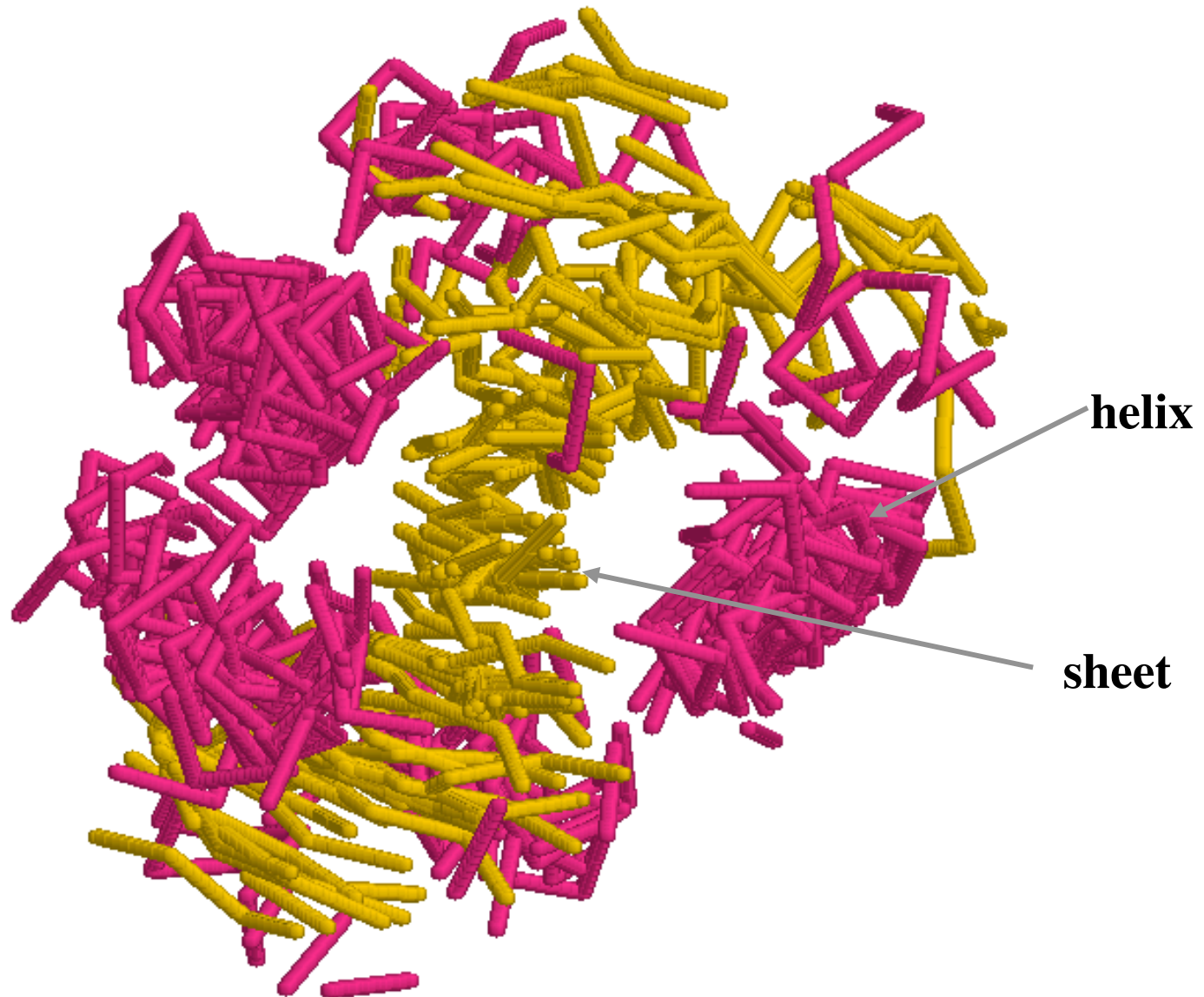
4.6	5.9
118	4.4

2DRC:A

*Each cell in distance matrix provides:*

RMSD(Å)	Sequence identity(%)
Length of alignment	Z-score

# CE alignment of 15 *analogs*



# HOW to use CE to find and align structural homologs

Set your browser to <http://cl.sdsc.edu/ce.html>

Find structural alignments by selecting from ALL or **REPRESENTATIVES** from the PDB.

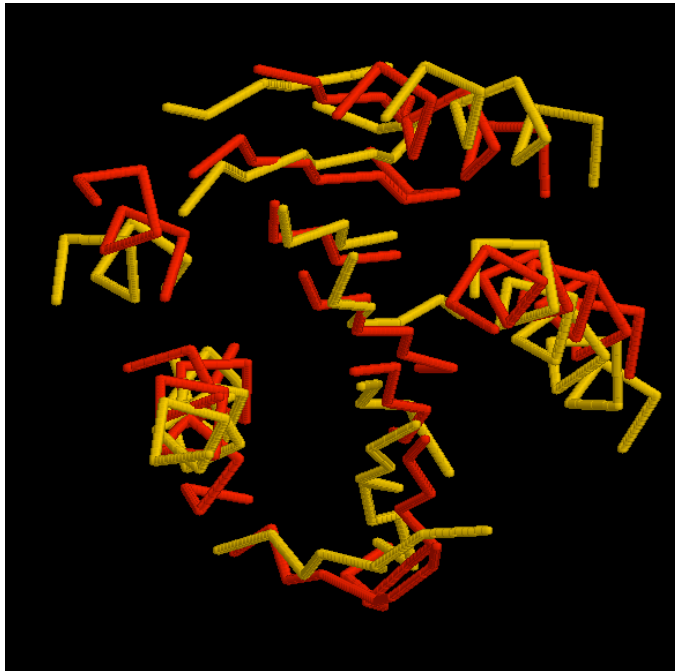
Submit your protein and chain. Or use 4dfr:A

Select 2 structures. Then hit: “Get alignment” (or use 4DFR and 1YAC)

Download as PDB file. Save it.

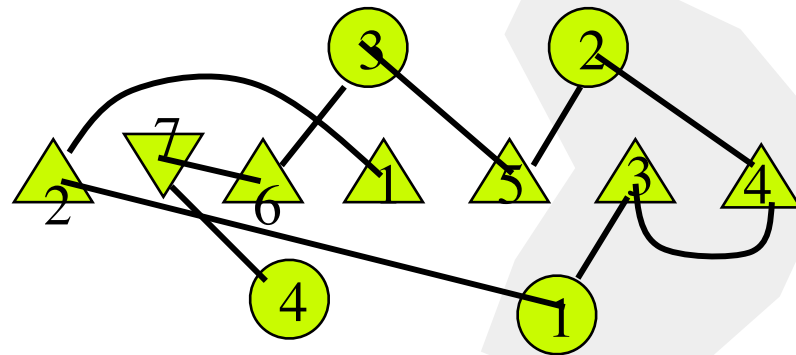
For use in MOE, divide the file into 2, one for each protein.

# Non-sequential alignment!

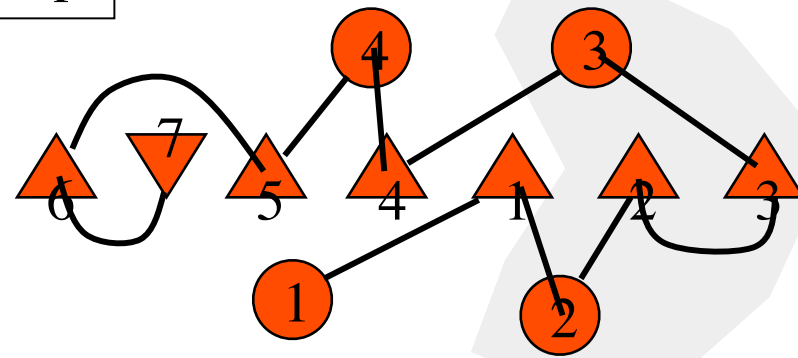


SCALI non-sequential alignment.

1alk



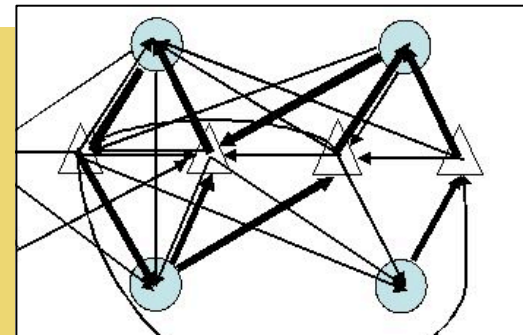
1vpt



# SCALI

Non-sequential structure-based alignments can be used to identify similar motifs in the packing geometry of SSEs. For example, it can find two proteins that have 2/4/2  $\alpha/\beta/\alpha$  3-layer sandwich architecture, regardless of how the SSEs are connected.

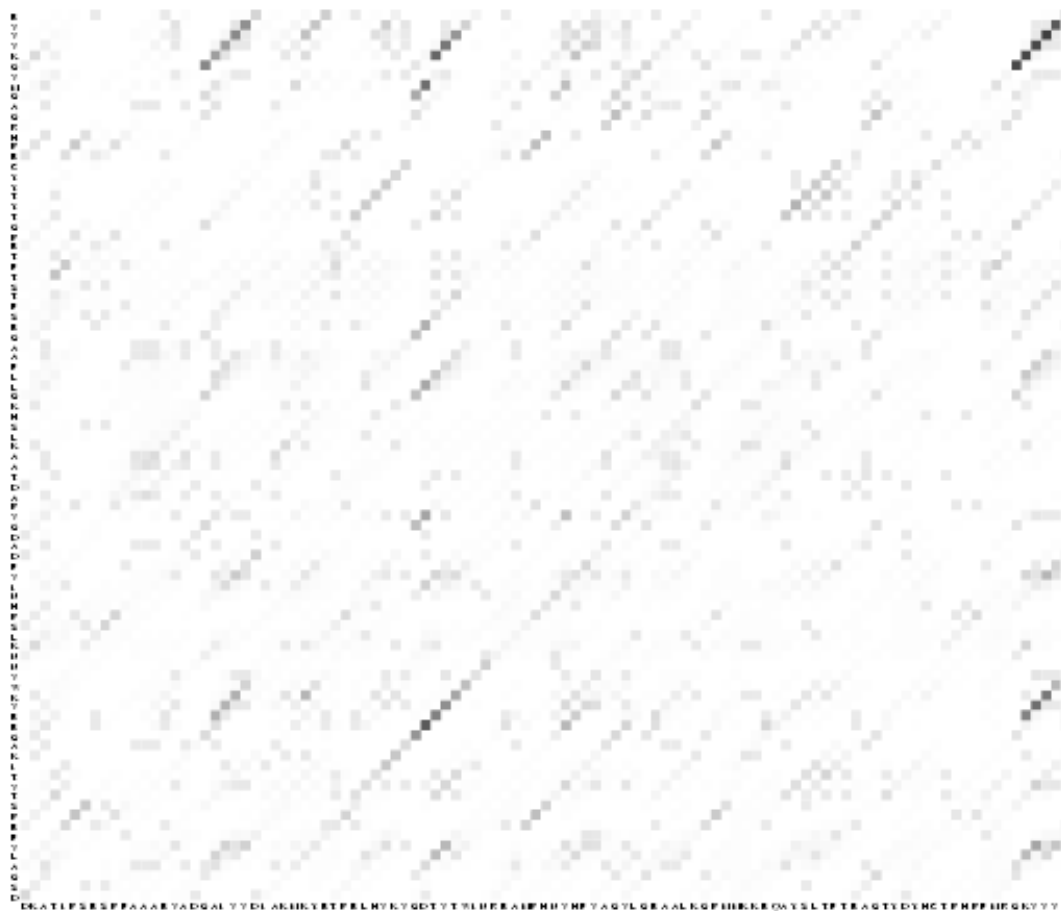
- (1) Exhaustive gapless alignments
- (2) FASTA-style assembly of alignment from fragment pairs.
- (3) Alignment score based on sequence similarity, local structure similarity and contact map similarity.



**Algorithm type: Geometric--intramolecular**

# Exhaustive alignment, no gaps.

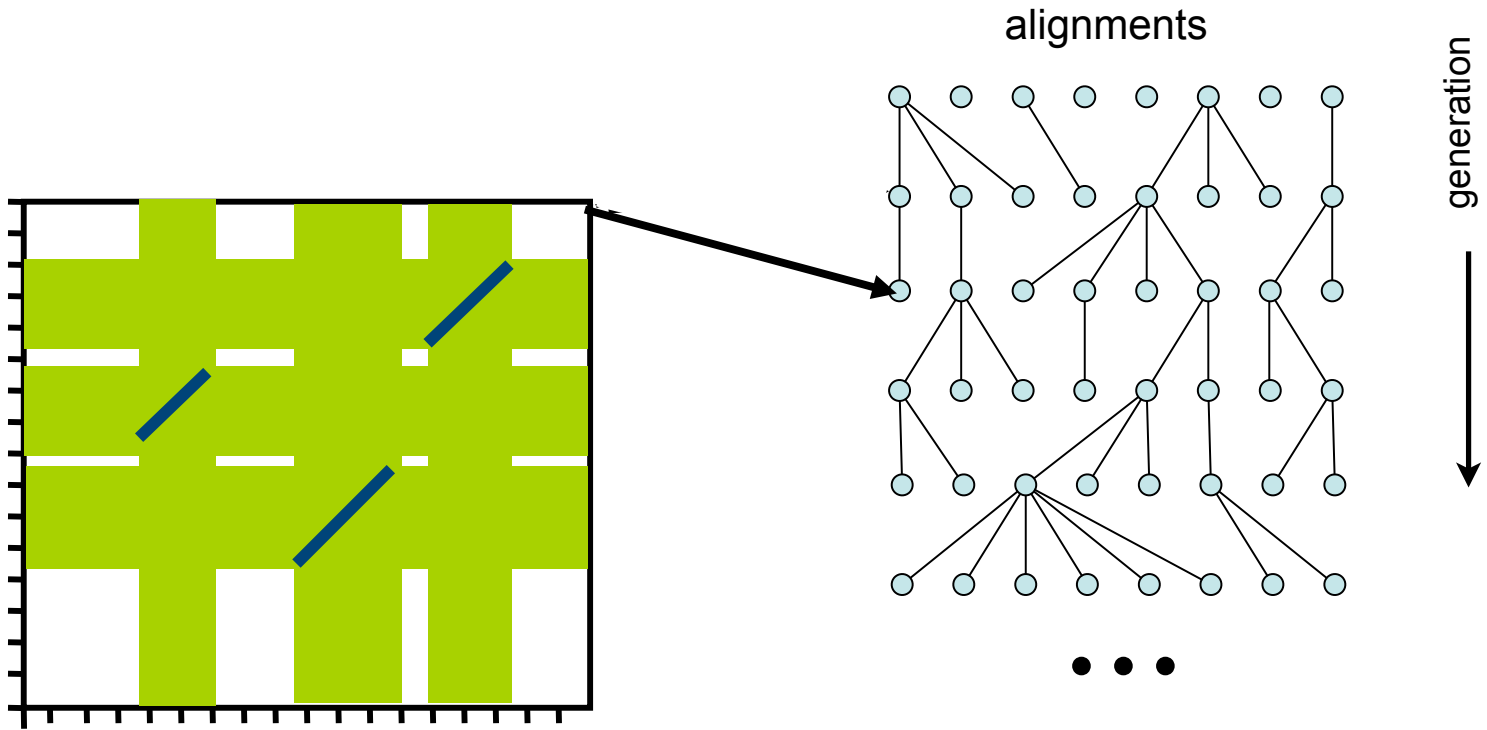
protein structure B



protein structure A

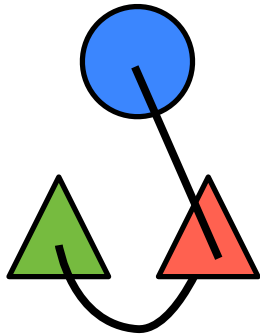
Each dot represents two positions that have the same local structure. Darker means similar in sequence. The SCALI alignment is constructed from fragment pairs (diagonal rows).

# FASTA-style search in alignment space

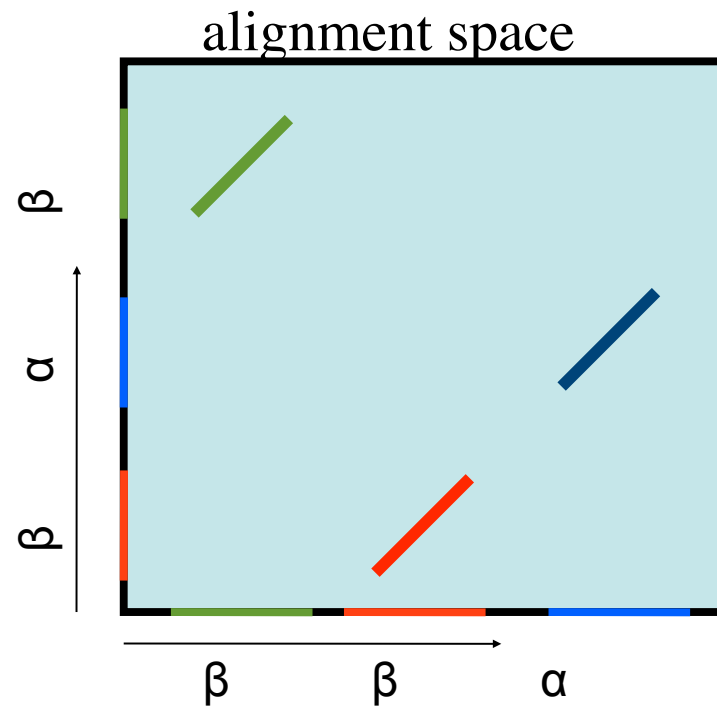
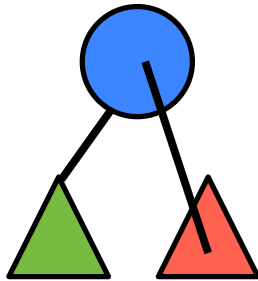


SCALI uses a near-greedy algorithm. Each candidate “parent” alignment generates up to 50 “children” alignments. Children are selected based on similarity (sequence, local structure, contacts). Selected children become parents in the next generation. The program is done when there is nothing left to align.

# Non-sequential alignment space



vs



Expressing a non-sequential alignment like a standard sequence alignment loses information. Using an alignment matrix is better, as in this example of two different  $\beta\beta\alpha$  units.