

Thermodynamics of structural transitions, folding & denaturation

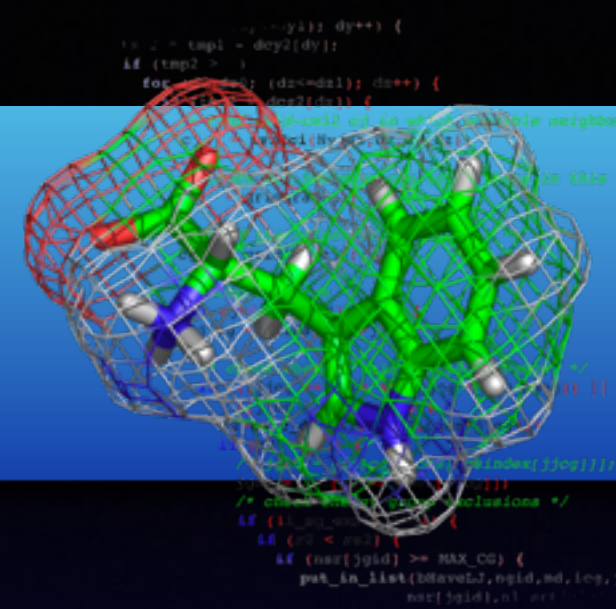
Magnus Andersson

magnus.andersson@scilifelab.se

Theoretical & Computational Biophysics



Recap



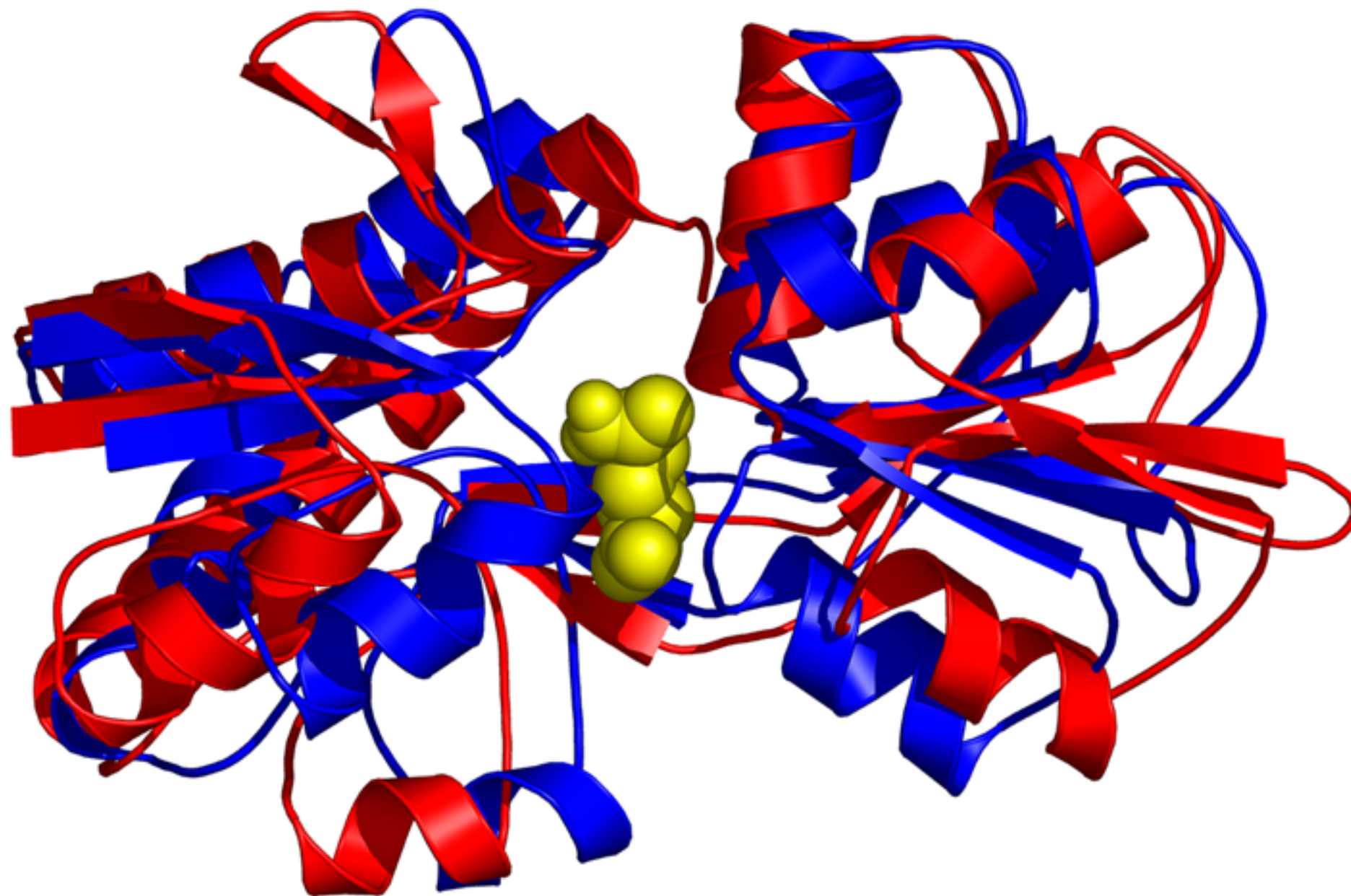
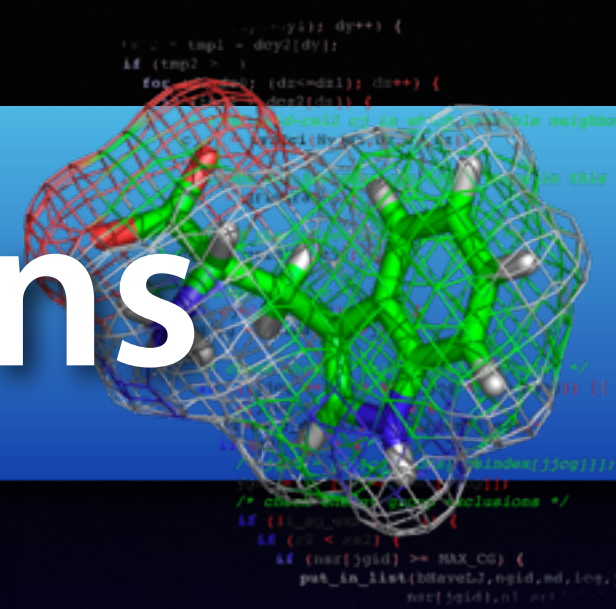
- Structural classification
- Evolution of structure and sequence
- Size of proteins, helices, sheets
- Stabilization of a few kcal/mol
- Common folds are common because they can accommodate lots of sequences
- Most sequences would not form proteins

Today



- Properties of structural transitions
- How do proteins fold and unfold?
- What does it mean for stabilization?
- Atomic models/theories of folding
 - Importance of hydrophobic effect
- Molten globule
- Side-chain packing
- Energy gap stabilization

Structural transitions

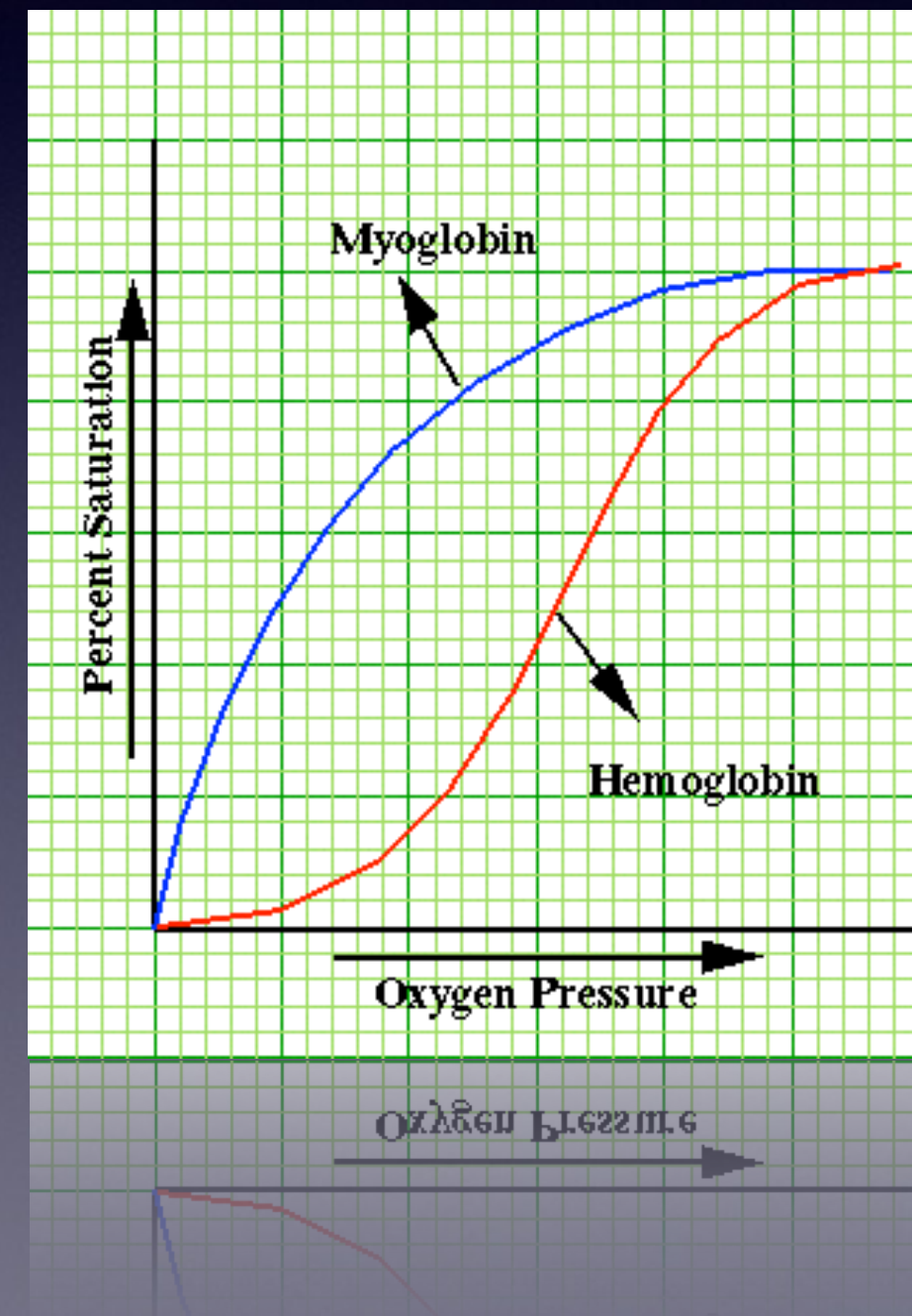
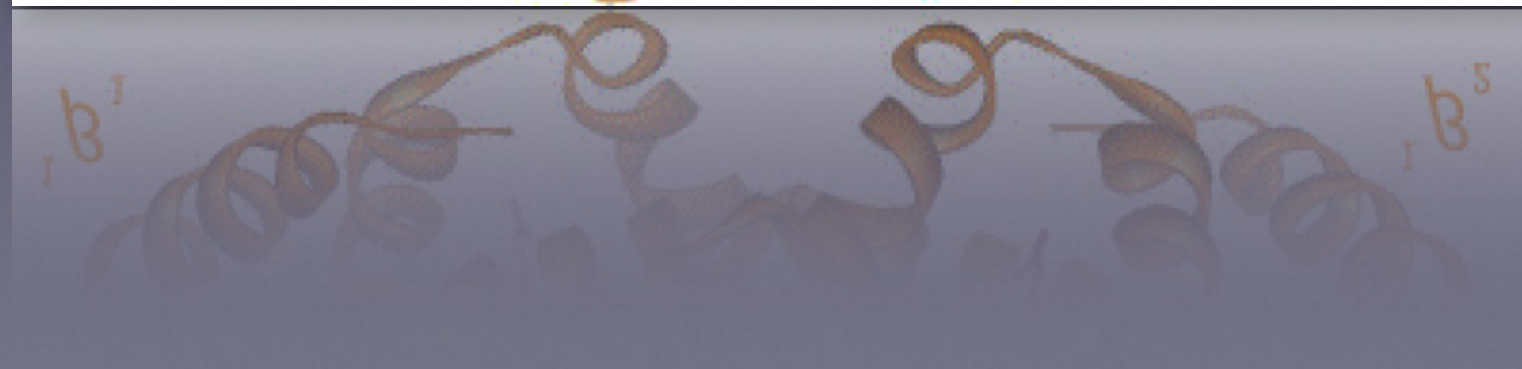


Selected Fit

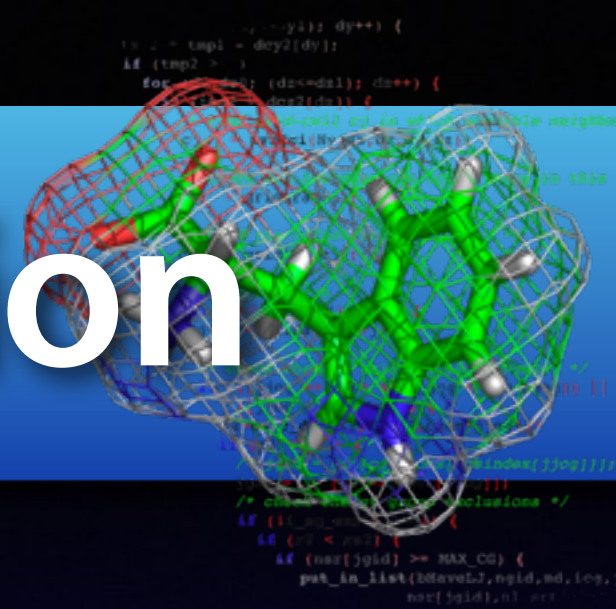
~6Å motion

**Fully
reversible**

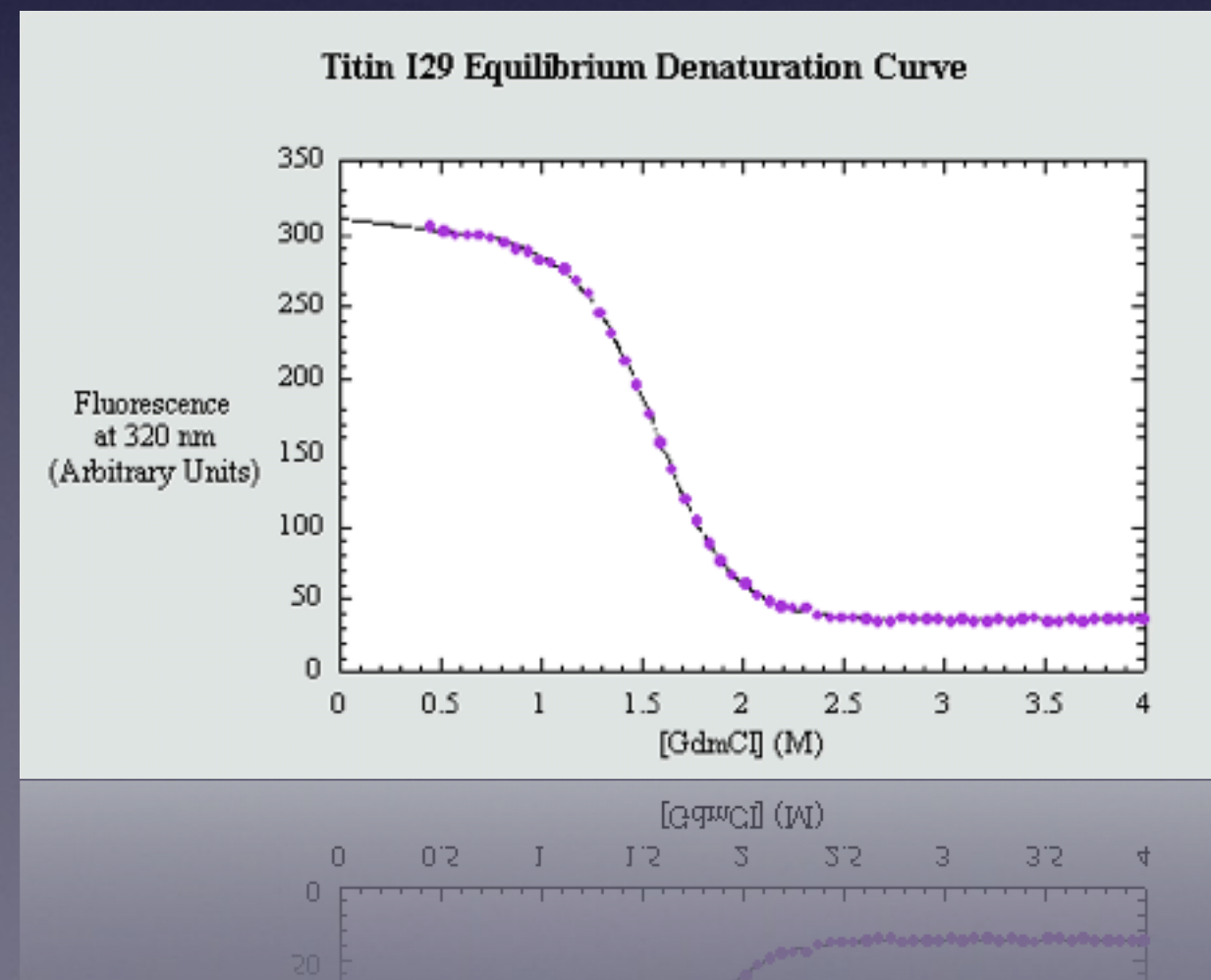
ion



Folding / denaturation

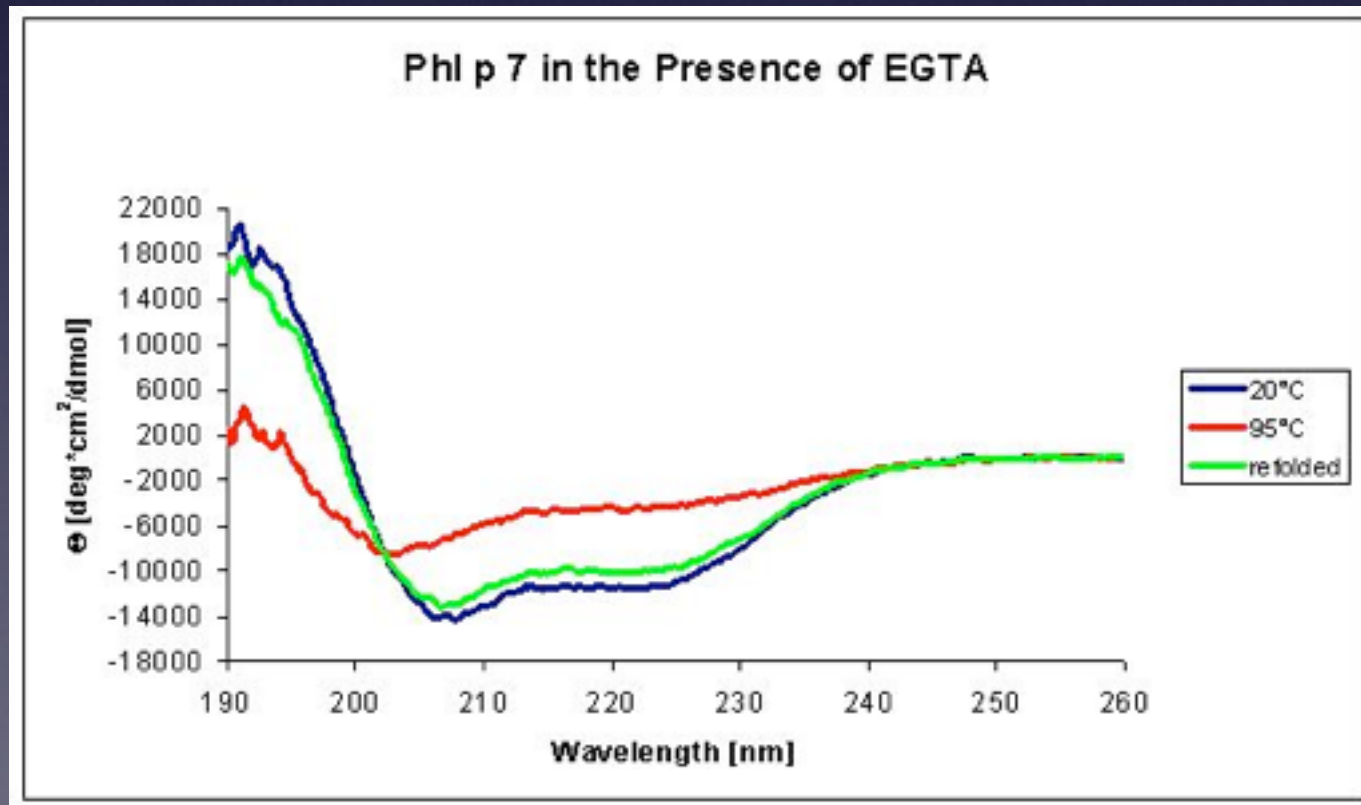
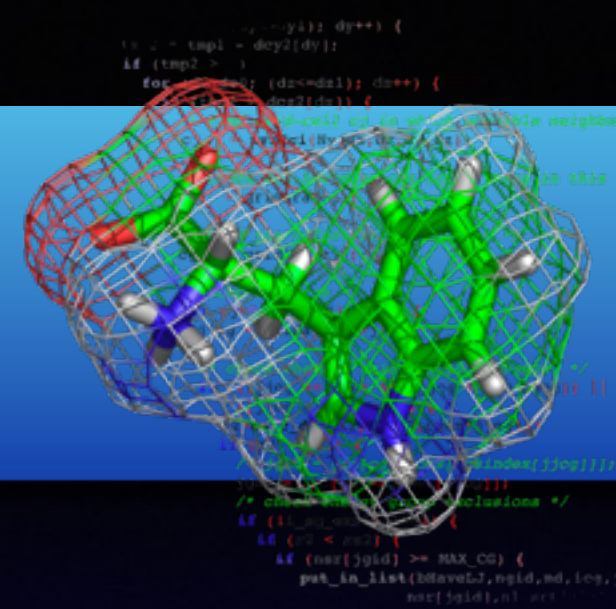


- Thermodynamic stability
- Separate issue: kinetic properties
- S-curves for observables: abrupt change
- Cooperative transition
- Salt concentration
 - Urea, GuHCl
- pH
- Temperature

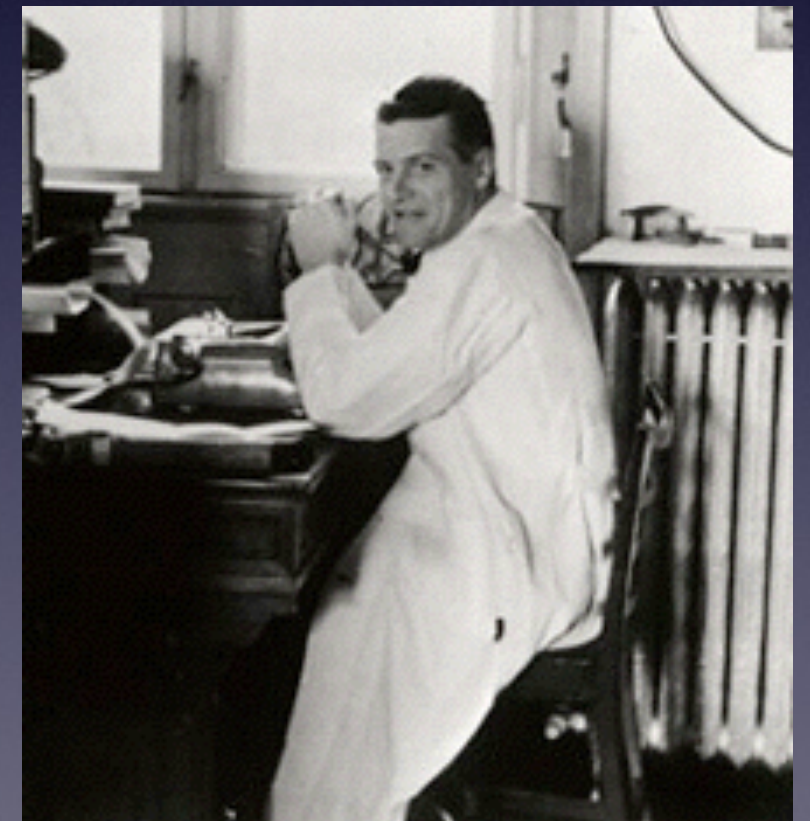


Refolding

- Christian Anfinsen, 1967
"Reductive Cleavage of Disulfide Bridges in Ribonuclease"
- Nobel Prize 1969



CD spectroscopy example

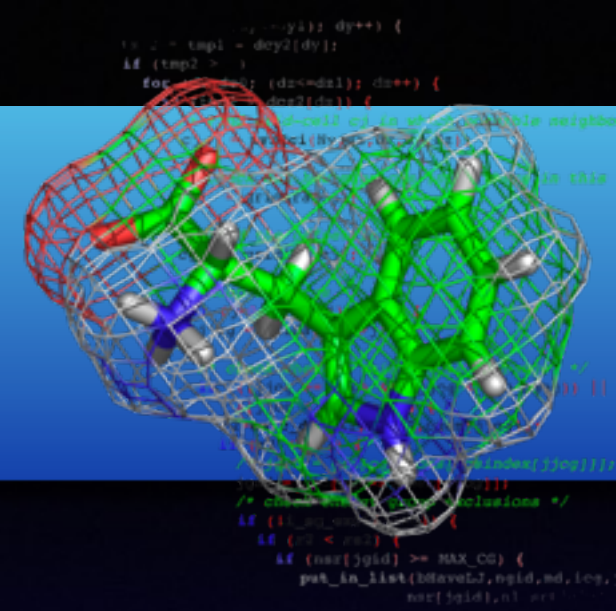


one

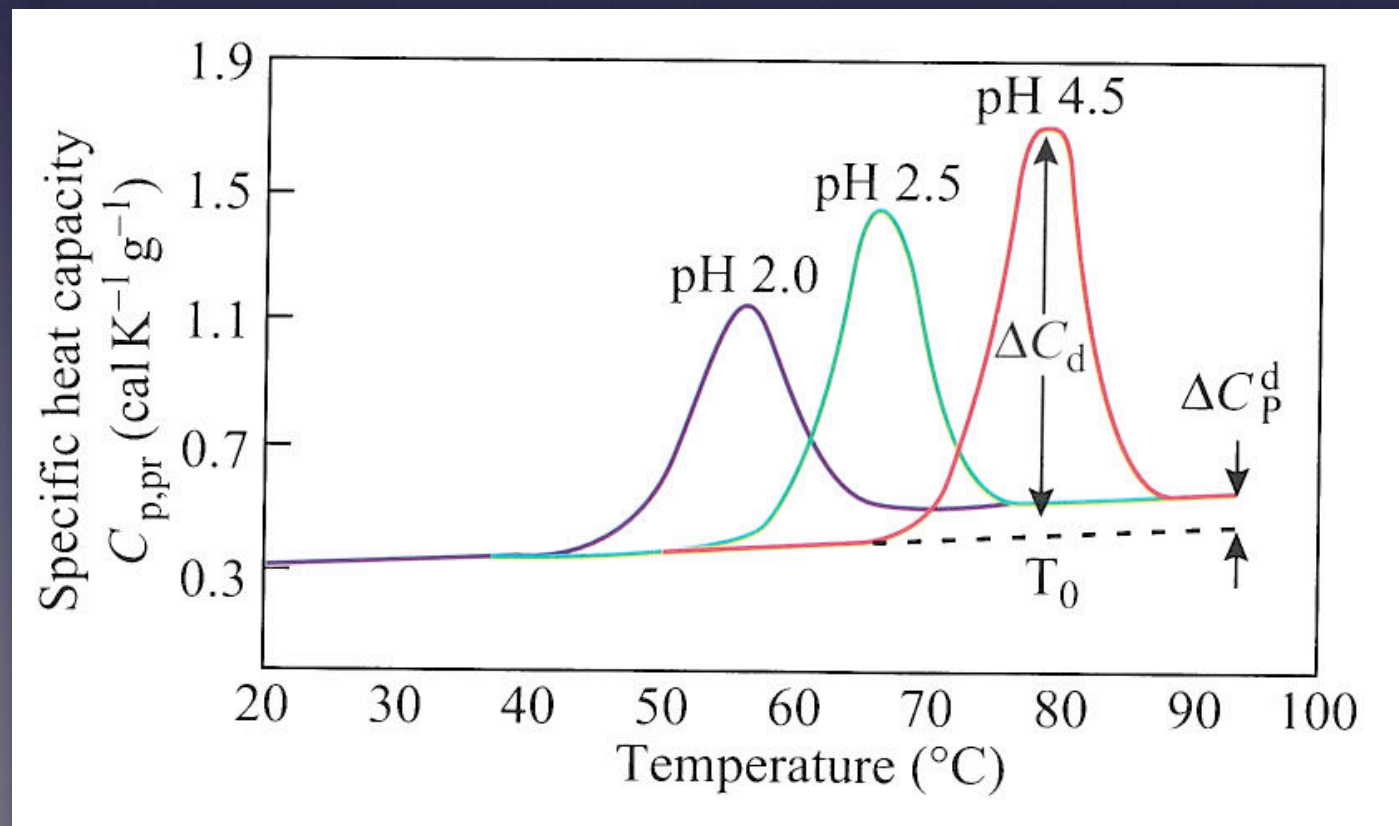


Not obvious from experiment!

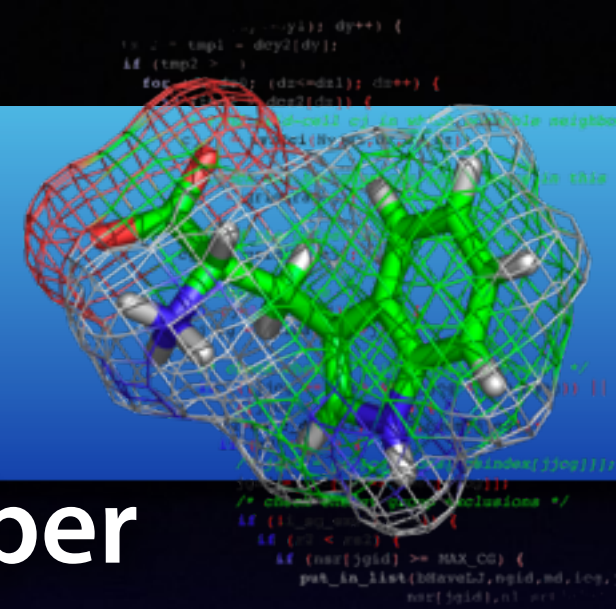
Calorimetry



- Measurements of heat capacity per protein
- Does an abrupt change imply cooperativity of the transition?
- No, just that it happens fast!



van't Hoff criterion



- What is the specific heat change “per melting unit” upon denaturation?
- Compare to specific heat per molecule (easy to calculate from concentration)
- if melting unit = full protein -> all-or-none
- if melting unit < full protein -> unfolds in smaller parts
- if melting unit > full protein -> aggregate

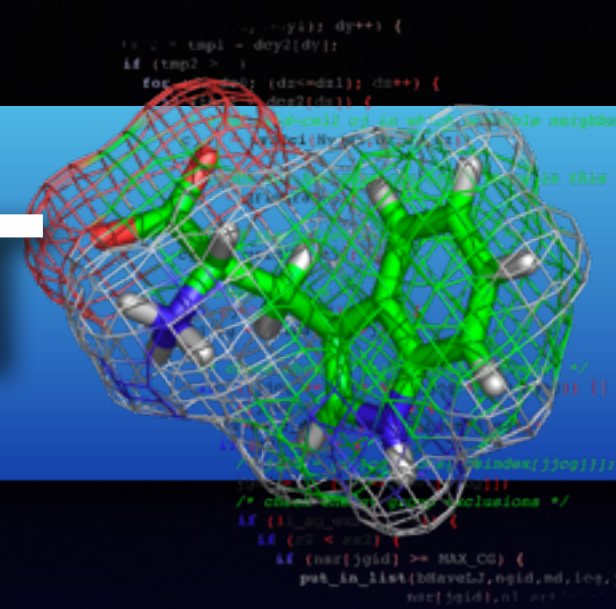
Fold fraction



- Native: Energy E , entropy S
- Molten: Energy E' , entropy S'
- Free energy $G=E-TS$ and $G'=E'-TS'$
- Assume Boltzmann state distribution

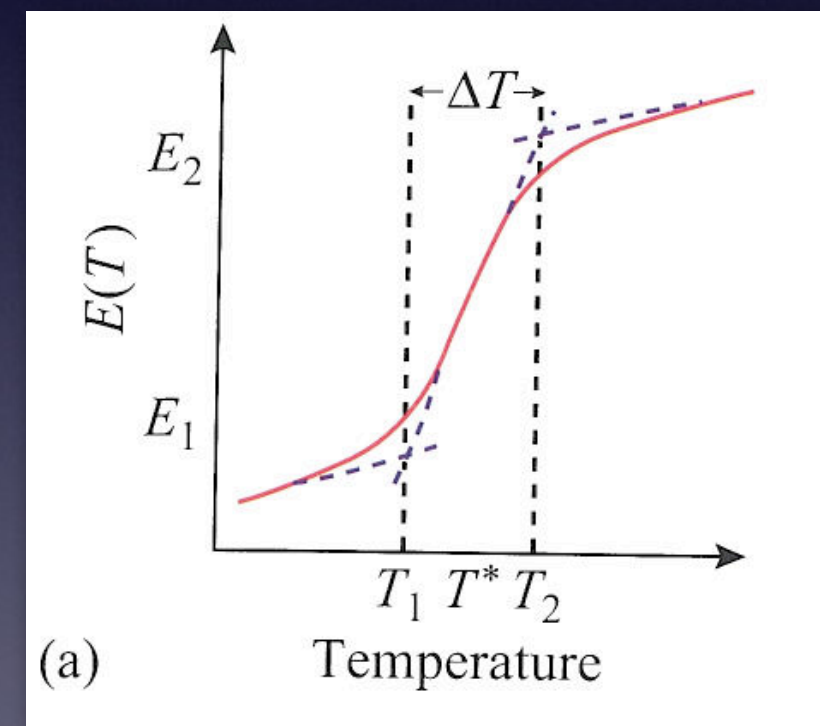
$$\begin{aligned} P_{\text{molten}} &= \frac{\exp[-(E' - TS')/kT]}{\exp[-(E - TS)/kT] + \exp[-(E' - TS')/kT]} \\ &= \frac{1}{1 + \exp[-(\Delta E - T\Delta S)/kT]} \end{aligned}$$

Transition width ΔT



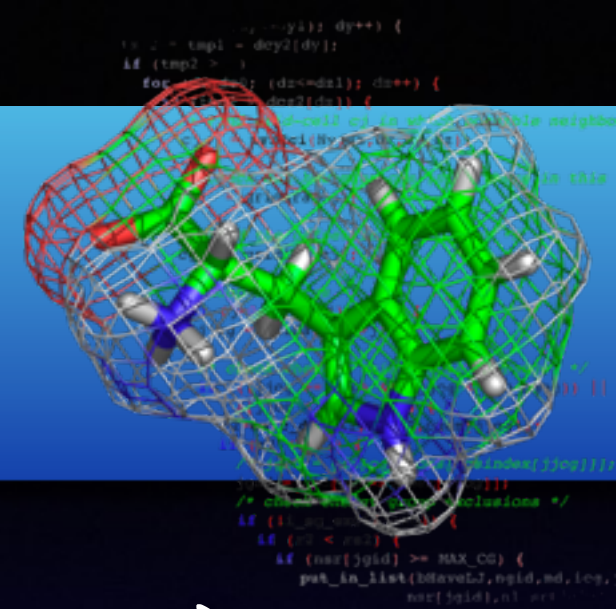
- Fold fraction P and the temperature width ΔT can be obtained from calorimetry

$$\frac{d}{dT}P_{\text{molten}} \approx \frac{\Delta P_{\text{molten}}}{\Delta T} \approx \frac{1}{\Delta T}$$



- Can we calculate the derivative in terms of energy E from the P_{molten} expression?

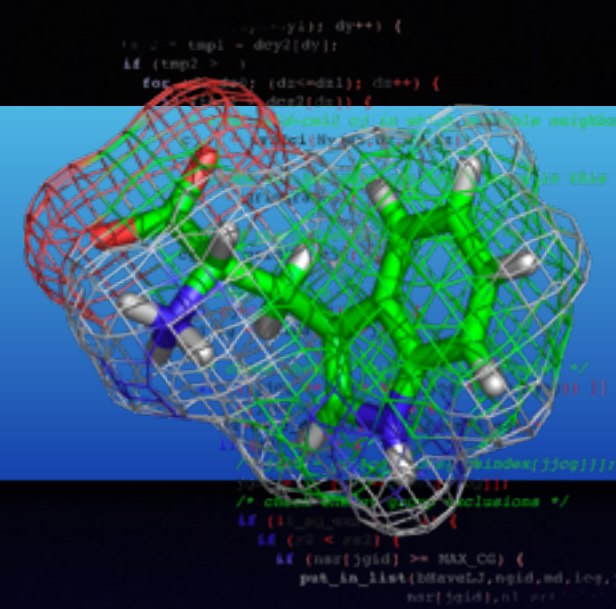
Specific heat



$$\begin{aligned} \frac{d}{dT} P_{\text{molten}} &= \frac{d}{dT} \left\{ \frac{1}{1 + \exp [(\Delta E - T \Delta S) / kT]} \right\} \\ &= - \left\{ \frac{1}{1 + \exp [(\Delta E - T \Delta S) / kT]} \right\}^2 \{ \exp [(\Delta E - T \Delta S) / kT] \} \\ &\quad \times \left(- \frac{\Delta E}{kT^2} \right) \end{aligned}$$

$$= P_{\text{molten}} (1 - P_{\text{molten}}) (\Delta E / kT^2)$$

Specific heat



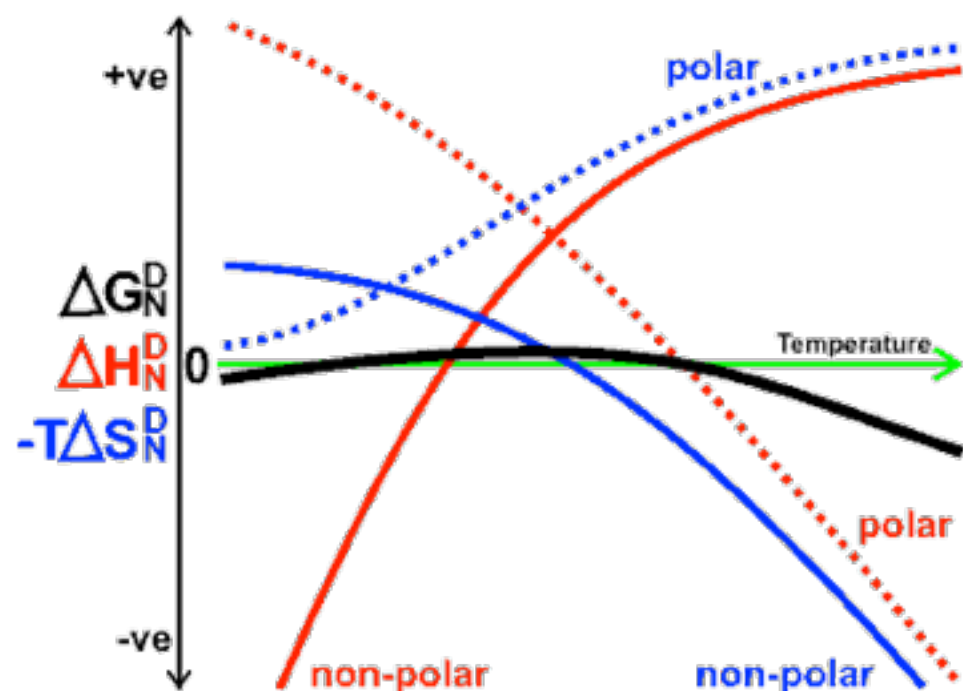
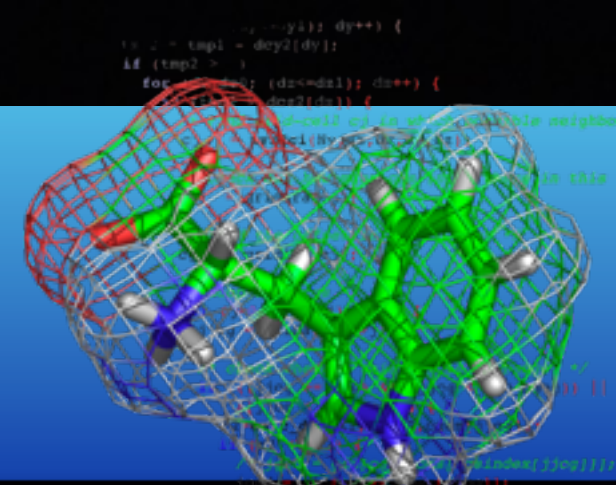
- At transition temperature, $P_{\text{molten}}=0.5$
- If $\Delta E/kT \gg 1$, this is close to middle point
- $dP/dT = 0.25 \Delta E/kT_0^2$
- Combine with $dP/dT=1/\Delta T$:
- $\Delta E=4kT_0^2/\Delta T$ for a “melting unit”
- $\Delta E=\Delta H/N$ for entire protein molecules

Denaturation



- Why is the protein unfolding?
- $\Delta G = \Delta E - T\Delta S$
- Explained by hydrophobic effect
 - ΔE increases with temperature
 - ΔS positive for unfolding
- But S will drop with temperature (less mobile solvent molecules)

Cold denaturation



Protein Cold Denaturation as Seen From the Solvent

Monika Davidovic, Carlos Mattea[†], Johan Qvist and Bertil Halle*

Department of Biophysical Chemistry, Center for Molecular Protein Science, Lund University, SE-22100 Lund, Sweden

J. Am. Chem. Soc., 2009, 131 (3), pp 1025-1036

DOI: 10.1021/ja8056419

Publication Date (Web): December 30, 2008

Copyright © 2008 American Chemical Society

[†] Present address: Institute of Physics, Technical University of Ilmenau, D-98684 Ilmenau, Germany.

Abstract

[Full Text HTML](#)

[Hi-Res PDF \[1189 KB\]](#)

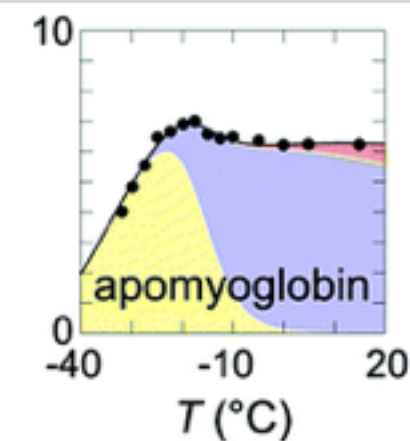
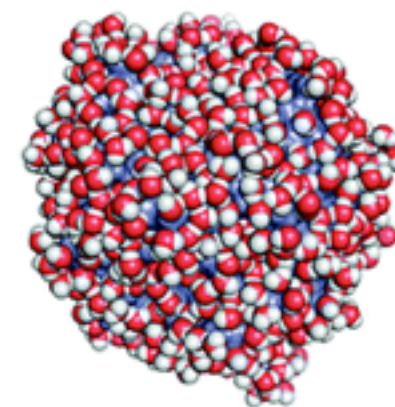
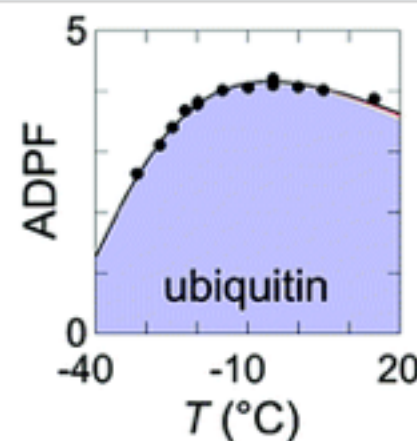
[PDF w/ Links \[349 KB\]](#)

[Supporting Info](#)

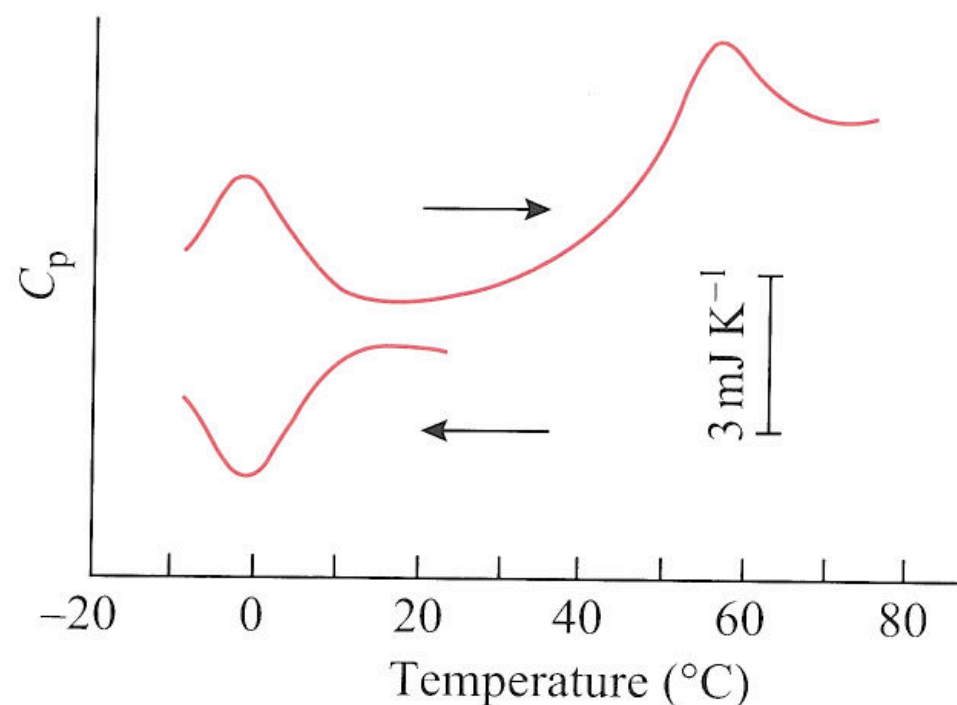
[Figures](#)

[Citing Articles](#)

Abstract



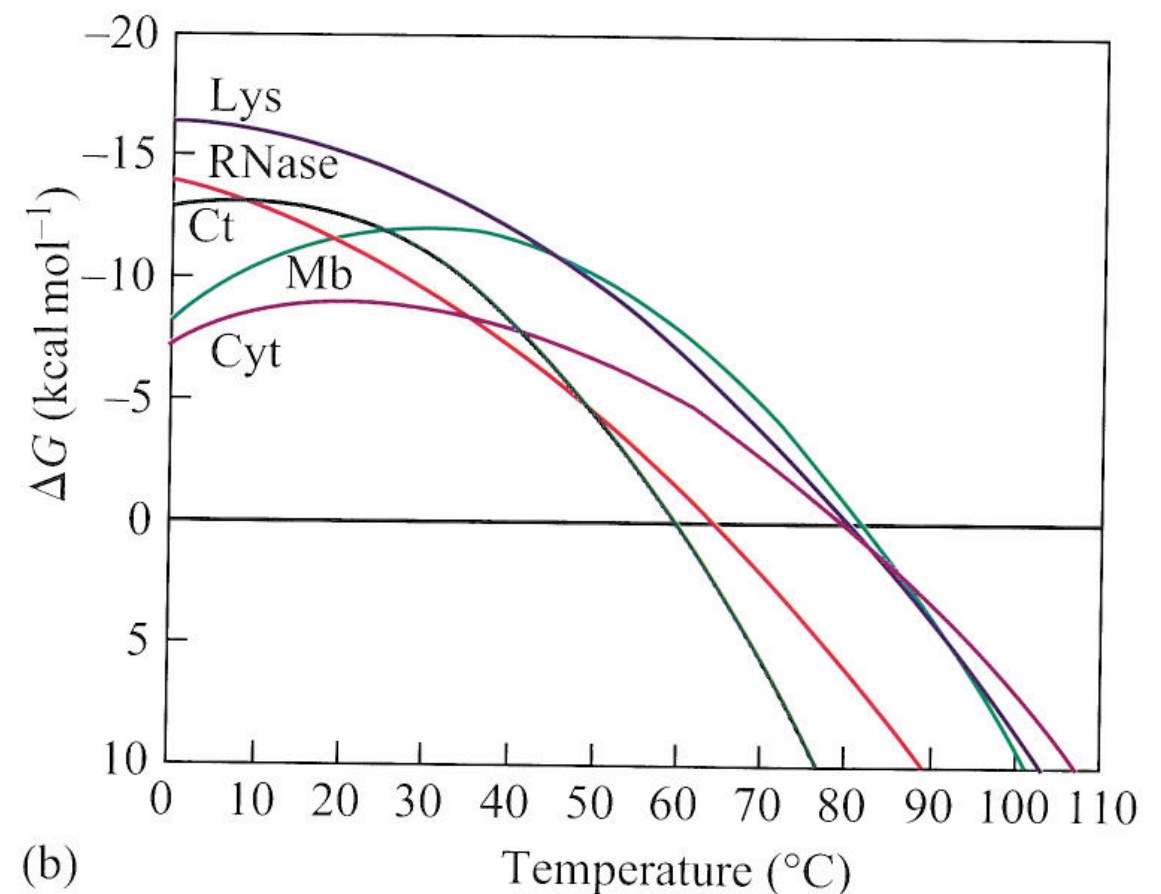
Unlike most ordered molecular systems, globular proteins exhibit a temperature of maximum stability, implying that the structure can be disrupted by cooling. This cold denaturation phenomenon is usually linked to the temperature-dependent hydrophobic driving force for protein folding. Yet, despite the key role played by protein-water interactions, hydration changes during cold denaturation have not been investigated

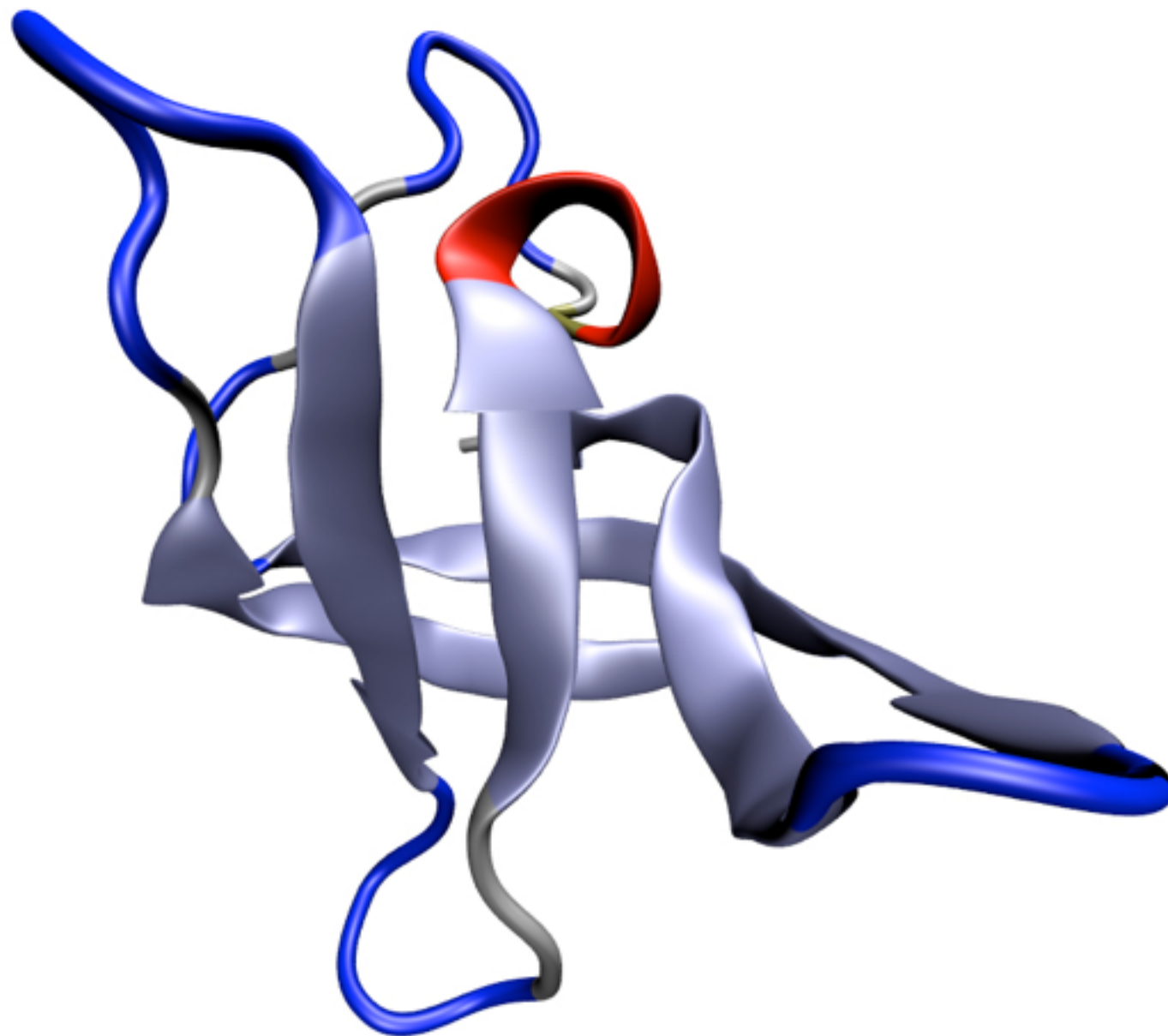



```

    dy = dy1; dy++} {
    for (i = tmp1; i <= tmp2; i++) {
        for (j = tmp3; j <= tmp4; j++) {
            for (k = tmp5; k <= tmp6; k++) {
                for (l = tmp7; l <= tmp8; l++) {
                    for (m = tmp9; m <= tmp10; m++) {
                        for (n = tmp11; n <= tmp12; n++) {
                            for (o = tmp13; o <= tmp14; o++) {
                                for (p = tmp15; p <= tmp16; p++) {
                                    for (q = tmp17; q <= tmp18; q++) {
                                        for (r = tmp19; r <= tmp20; r++) {
                                            for (s = tmp21; s <= tmp22; s++) {
                                                for (t = tmp23; t <= tmp24; t++) {
                                                    for (u = tmp25; u <= tmp26; u++) {
                                                        for (v = tmp27; v <= tmp28; v++) {
                                                            for (w = tmp29; w <= tmp30; w++) {
                                                                for (x = tmp31; x <= tmp32; x++) {
                                                                    for (y = tmp33; y <= tmp34; y++) {
                                                                        for (z = tmp35; z <= tmp36; z++) {
                                                                            for (aa = tmp37; aa <= tmp38; aa++) {
                                                                                for (ab = tmp39; ab <= tmp40; ab++) {
                                                                                    for (ac = tmp41; ac <= tmp42; ac++) {
                                                                                        for (ad = tmp43; ad <= tmp44; ad++) {
                                                                                            for (ae = tmp45; ae <= tmp46; ae++) {
                                                                                                for (af = tmp47; af <= tmp48; af++) {
                                                                                                    for (ag = tmp49; ag <= tmp50; ag++) {
                                                                                                        for (ah = tmp51; ah <= tmp52; ah++) {
                                                                                                            for (ai = tmp53; ai <= tmp54; ai++) {
                                                                                                                for (aj = tmp55; aj <= tmp56; aj++) {
                                                                                                                    for (ak = tmp57; ak <= tmp58; ak++) {
                                                                                                                        for (al = tmp59; al <= tmp60; al++) {
                                                                                                                            for (am = tmp61; am <= tmp62; am++) {
                                                                                                                                for (an = tmp63; an <= tmp64; an++) {
                                                                                                                                    for (ao = tmp65; ao <= tmp66; ao++) {
                                                                                                                                        for (ap = tmp67; ap <= tmp68; ap++) {
                                                                                                                                            for (aq = tmp69; aq <= tmp70; aq++) {
                                                                                                                                                for (ar = tmp71; ar <= tmp72; ar++) {
                                                                                                                                                    for (as = tmp73; as <= tmp74; as++) {
                                                                                                                                                        for (at = tmp75; at <= tmp76; at++) {
                                                                                                                                                            for (au = tmp77; au <= tmp78; au++) {
                                                                                                                                                                for (av = tmp79; av <= tmp80; av++) {
                                                                                                                                                                    for (aw = tmp81; aw <= tmp82; aw++) {
                                                                                                                                                                        for (ax = tmp83; ax <= tmp84; ax++) {
                                                                                                                                                                            for (ay = tmp85; ay <= tmp86; ay++) {
                                                                                                                                                                                for (az = tmp87; az <= tmp88; az++) {
                                                                                                                                                                                    for (ba = tmp89; ba <= tmp90; ba++) {
                                                                                                                                                                                        for (bb = tmp91; bb <= tmp92; bb++) {
                                                                                                                                                                                            for (bc = tmp93; bc <= tmp94; bc++) {
                                                                                                                                                                                                for (bd = tmp95; bd <= tmp96; bd++) {
                                                                                                                                                                                                    for (be = tmp97; be <= tmp98; be++) {
                                                                                                                                                                                                        for (bf = tmp99; bf <= tmp100; bf++) {
                                                                                                                                                                                                            for (bg = tmp101; bg <= tmp102; bg++) {
                                                                                                                                                                                                                for (bh = tmp103; bh <= tmp104; bh++) {
                                                                                                                                                                                                                    for (bi = tmp105; bi <= tmp106; bi++) {
                                                                                                                                                                                                                        for (bj = tmp107; bj <= tmp108; bj++) {
                                                                                                                                                                                                                            for (bk = tmp109; bk <= tmp110; bk++) {
                                                                                                                                                                                                                                for (bl = tmp111; bl <= tmp112; bl++) {
                                                                                                                                                                    ...
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```



[illegible]

Structures evolved to survive low temperatures

[illegible]

- What does the denatured state look like?
- Extended coil?
- Surprisingly, experiments indicate they can be quite compact - almost like a native protein
- Secondary structure still present
- Complete unfolding requires strong denaturants like GuHCl in high conc.

[illegible]

The graph plots the concentration of GuHCl (M) on the y-axis (0 to 4) against temperature (°C) on the x-axis (0 to 60). The regions are labeled COIL, NATIVE, and MOLTEN. A red curve represents the denaturation transition, with a red dot at the midpoint of denaturation at approximately 40°C and 1.4 M GuHCl.

Temp (°C)	Concentration of GuHCl (M)	Region
0	2.1	NATIVE
10	2.2	NATIVE
20	2.2	NATIVE
30	2.0	NATIVE
40	1.4	Midpoint of Denaturation
45	0.5	COIL
50	1.8	MOLTEN
60	2.2	MOLTEN

Molten globule

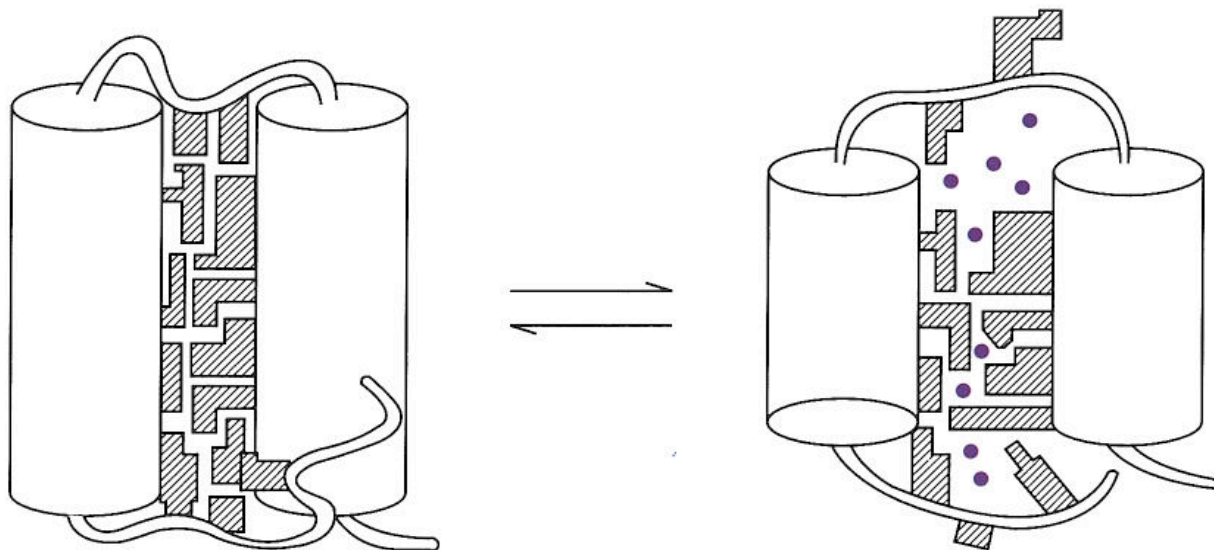
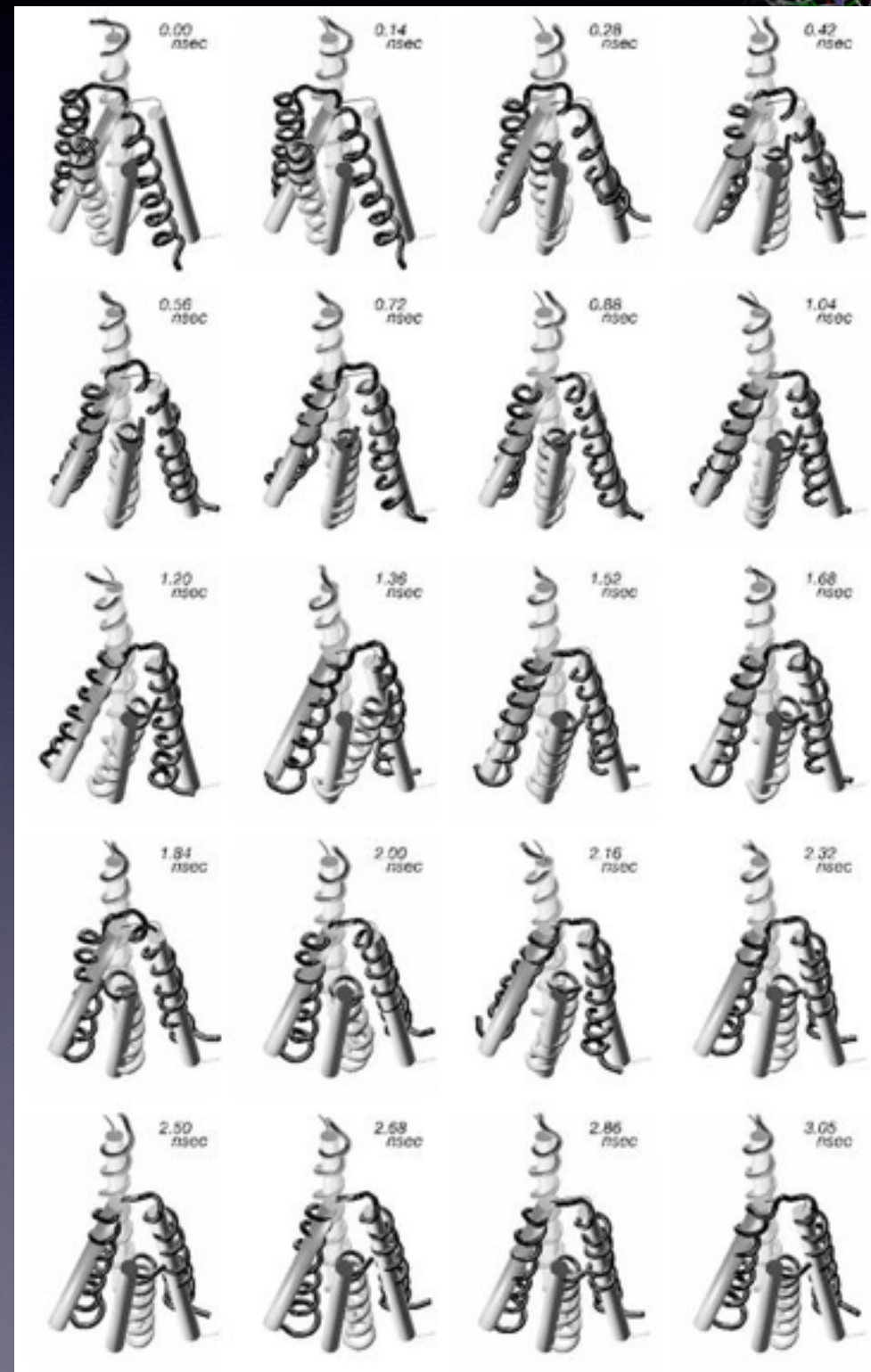
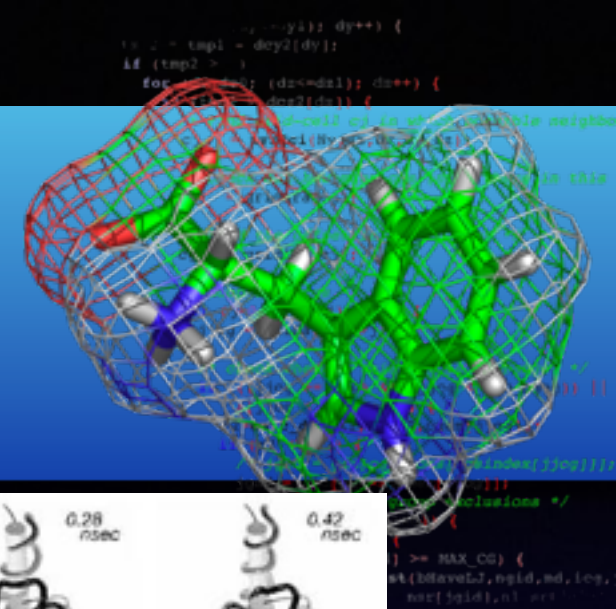


- What is the molten globule?
- Main chain ordering (trace & structure)
- Hydrophobic core size & density
- Volume of the protein molecule (radius)
- Transition native-globule is well defined, yet the structures are quite similar
- Transition globule-coil less sharp

[illegible]

Like native	Like unfolded
Compact, hydrophobic core	Not rigid structure
Secondary structure	No second melting transition to coil
Partial sidechain order (TRP buried)	No unique sidechain packing
Partial S-S bond formation	Not functional

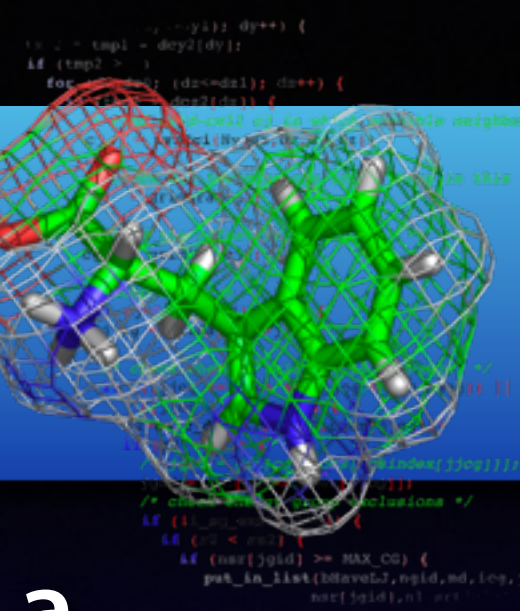
Molten globule



```

    dy--y1); dy++) {
    for (j = tmp1 - day2(dy);
    if (tmp1 > 0)
        for (k = (dnc-dx1); dx++ ) {
            dx2(dx);
            /* cell c1 is within visible neighbor
            c1 = (c1+dx1+dy1+dz1);
            /* cell c2 is within visible neighbor
            c2 = (c1+dx2+dy2+dz2);
            /* cell c3 is within visible neighbor
            c3 = (c2+dx3+dy3+dz3);
            /* cell c4 is within visible neighbor
            c4 = (c3+dx4+dy4+dz4);
            /* cell c5 is within visible neighbor
            c5 = (c4+dx5+dy5+dz5);
            /* cell c6 is within visible neighbor
            c6 = (c5+dx6+dy6+dz6);
            /* cell c7 is within visible neighbor
            c7 = (c6+dx7+dy7+dz7);
            /* cell c8 is within visible neighbor
            c8 = (c7+dx8+dy8+dz8);
            /* cell c9 is within visible neighbor
            c9 = (c8+dx9+dy9+dz9);
            /* cell c10 is within visible neighbor
            c10 = (c9+dx10+dy10+dz10);
            /* cell c11 is within visible neighbor
            c11 = (c10+dx11+dy11+dz11);
            /* cell c12 is within visible neighbor
            c12 = (c11+dx12+dy12+dz12);
            /* cell c13 is within visible neighbor
            c13 = (c12+dx13+dy13+dz13);
            /* cell c14 is within visible neighbor
            c14 = (c13+dx14+dy14+dz14);
            /* cell c15 is within visible neighbor
            c15 = (c14+dx15+dy15+dz15);
            /* cell c16 is within visible neighbor
            c16 = (c15+dx16+dy16+dz16);
            /* cell c17 is within visible neighbor
            c17 = (c16+dx17+dy17+dz17);
            /* cell c18 is within visible neighbor
            c18 = (c17+dx18+dy18+dz18);
            /* cell c19 is within visible neighbor
            c19 = (c18+dx19+dy19+dz19);
            /* cell c20 is within visible neighbor
            c20 = (c19+dx20+dy20+dz20);
            /* cell c21 is within visible neighbor
            c21 = (c20+dx21+dy21+dz21);
            /* cell c22 is within visible neighbor
            c22 = (c21+dx22+dy22+dz22);
            /* cell c23 is within visible neighbor
            c23 = (c22+dx23+dy23+dz23);
            /* cell c24 is within visible neighbor
            c24 = (c23+dx24+dy24+dz24);
            /* cell c25 is within visible neighbor
            c25 = (c24+dx25+dy25+dz25);
            /* cell c26 is within visible neighbor
            c26 = (c25+dx26+dy26+dz26);
            /* cell c27 is within visible neighbor
            c27 = (c26+dx27+dy27+dz27);
            /* cell c28 is within visible neighbor
            c28 = (c27+dx28+dy28+dz28);
            /* cell c29 is within visible neighbor
            c29 = (c28+dx29+dy29+dz29);
            /* cell c30 is within visible neighbor
            c30 = (c29+dx30+dy30+dz30);
            /* cell c31 is within visible neighbor
            c31 = (c30+dx31+dy31+dz31);
            /* cell c32 is within visible neighbor
            c32 = (c31+dx32+dy32+dz32);
            /* cell c33 is within visible neighbor
            c33 = (c32+dx33+dy33+dz33);
            /* cell c34 is within visible neighbor
            c34 = (c33+dx34+dy34+dz34);
            /* cell c35 is within visible neighbor
            c35 = (c34+dx35+dy35+dz35);
            /* cell c36 is within visible neighbor
            c36 = (c35+dx36+dy36+dz36);
            /* cell c37 is within visible neighbor
            c37 = (c36+dx37+dy37+dz37);
            /* cell c38 is within visible neighbor
            c38 = (c37+dx38+dy38+dz38);
            /* cell c39 is within visible neighbor
            c39 = (c38+dx39+dy39+dz39);
            /* cell c40 is within visible neighbor
            c40 = (c39+dx40+dy40+dz40);
            /* cell c41 is within visible neighbor
            c41 = (c40+dx41+dy41+dz41);
            /* cell c42 is within visible neighbor
            c42 = (c41+dx42+dy42+dz42);
            /* cell c43 is within visible neighbor
            c43 = (c42+dx43+dy43+dz43);
            /* cell c44 is within visible neighbor
            c44 = (c43+dx44+dy44+dz44);
            /* cell c45 is within visible neighbor
            c45 = (c44+dx45+dy45+dz45);
            /* cell c46 is within visible neighbor
            c46 = (c45+dx46+dy46+dz46);
            /* cell c47 is within visible neighbor
            c47 = (c46+dx47+dy47+dz47);
            /* cell c48 is within visible neighbor
            c48 = (c47+dx48+dy48+dz48);
            /* cell c49 is within visible neighbor
            c49 = (c48+dx49+dy49+dz49);
            /* cell c50 is within visible neighbor
            c50 = (c49+dx50+dy50+dz50);
            /* cell c51 is within visible neighbor
            c51 = (c50+dx51+dy51+dz51);
            /* cell c52 is within visible neighbor
            c52 = (c51+dx52+dy52+dz52);
            /* cell c53 is within visible neighbor
            c53 = (c52+dx53+dy53+dz53);
            /* cell c54 is within visible neighbor
            c54 = (c53+dx54+dy54+dz54);
            /* cell c55 is within visible neighbor
            c55 = (c54+dx55+dy55+dz55);
            /* cell c56 is within visible neighbor
            c56 = (c55+dx56+dy56+dz56);
            /* cell c57 is within visible neighbor
            c57 = (c56+dx57+dy57+dz57);
            /* cell c58 is within visible neighbor
            c58 = (c57+dx58+dy58+dz58);
            /* cell c59 is within visible neighbor
            c59 = (c58+dx59+dy59+dz59);
            /* cell c60 is within visible neighbor
            c60 = (c59+dx60+dy60+dz60);
            /* cell c61 is within visible neighbor
            c61 = (c60+dx61+dy61+dz61);
            /* cell c62 is within visible neighbor
            c62 = (c61+dx62+dy62+dz62);
            /* cell c63 is within visible neighbor
            c63 = (c62+dx63+dy63+dz63);
            /* cell c64 is within visible neighbor
            c64 = (c63+dx64+dy64+dz64);
            /* cell c65 is within visible neighbor
            c65 = (c64+dx65+dy65+dz65);
            /* cell c66 is within visible neighbor
            c66 = (c65+dx66+dy66+dz66);
            /* cell c67 is within visible neighbor
            c67 = (c66+dx67+dy67+dz67);
            /* cell c68 is within visible neighbor
            c68 = (c67+dx68+dy68+dz68);
            /* cell c69 is within visible neighbor
            c69 = (c68+dx69+dy69+dz69);
            /* cell c70 is within visible neighbor
            c70 = (c69+dx70+dy70+dz70);
            /* cell c71 is within visible neighbor
            c71 = (c70+dx71+dy71+dz71);
            /* cell c72 is within visible neighbor
            c72 = (c71+dx72+dy72+dz72);
            /* cell c73 is within visible neighbor
            c73 = (c72+dx73+dy73+dz73);
            /* cell c74 is within visible neighbor
            c74 = (c73+dx74+dy74+dz74);
            /* cell c75 is within visible neighbor
            c75 = (c74+dx75+dy75+dz75);
            /* cell c76 is within visible neighbor
            c76 = (c75+dx76+dy76+dz76);
            /* cell c77 is within visible neighbor
            c77 = (c76+dx77+dy77+dz77);
            /* cell c78 is within visible neighbor
            c78 = (c77+dx78+dy78+dz78);
            /* cell c79 is within visible neighbor
            c79 = (c78+dx79+dy79+dz79);
            /* cell c80 is within visible neighbor
            c80 = (c79+dx80+dy80+dz80);
            /* cell c81 is within visible neighbor
            c81 = (c80+dx81+dy81+dz81);
            /* cell c82 is within visible neighbor
            c82 = (c81+dx82+dy82+dz82);
            /* cell c83 is within visible neighbor
            c83 = (c82+dx83+dy83+dz83);
            /* cell c84 is within visible neighbor
            c84 = (c83+dx84+dy84+dz84);
            /* cell c85 is within visible neighbor
            c85 = (c84+dx85+dy85+dz85);
            /* cell c86 is within visible neighbor
            c86 = (c85+dx86+dy86+dz86);
            /* cell c87 is within visible neighbor
            c87 = (c86+dx87+dy87+dz87);
            /* cell c88 is within visible neighbor
            c88 = (c87+dx88+dy88+dz88);
            /* cell c89 is within visible neighbor
            c89 = (c88+dx89+dy89+dz89);
            /* cell c90 is within visible neighbor
            c90 = (c89+dx90+dy90+dz90);
            /* cell c91 is within visible neighbor
            c91 = (c90+dx91+dy91+dz91);
            /* cell c92 is within visible neighbor
            c92 = (c91+dx92+dy92+dz92);
            /* cell c93 is within visible neighbor
            c93 = (c92+dx93+dy93+dz93);
            /* cell c94 is within visible neighbor
            c94 = (c93+dx94+dy94+dz94);
            /* cell c95 is within visible neighbor
            c95 = (c94+dx95+dy95+dz95);
            /* cell c96 is within visible neighbor
            c96 = (c95+dx96+dy96+dz96);
            /* cell c97 is within visible neighbor
            c97 = (c96+dx97+dy97+dz97);
            /* cell c98 is within visible neighbor
            c98 = (c97+dx98+dy98+dz98);
            /* cell c99 is within visible neighbor
            c99 = (c98+dx99+dy99+dz99);
            /* cell c100 is within visible neighbor
            c100 = (c99+dx100+dy100+dz100);
            /* cell c101 is within visible neighbor
            c101 = (c100+dx101+dy101+dz101);
            /* cell c102 is within visible neighbor
            c102 = (c101+dx102+dy102+dz102);
            /* cell c103 is within visible neighbor
            c103 = (c102+dx103+dy103+dz103);
            /* cell c104 is within visible neighbor
            c104 = (c103+dx104+dy104+dz104);
            /* cell c105 is within visible neighbor
            c105 = (c104+dx105+dy105+dz105);
            /* cell c106 is within visible neighbor
            c106 = (c105+dx106+dy106+dz106);
            /* cell c107 is within visible neighbor
            c107 = (c106+dx107+dy107+dz107);
            /* cell c108 is within visible neighbor
            c108 = (c107+dx108+dy108+dz108);
            /* cell c109 is within visible neighbor
            c109 = (c108+dx109+dy109+dz109);
            /* cell c110 is within visible neighbor
            c110 = (c109+dx110+dy110+dz110);
            /* cell c111 is within visible neighbor
            c111 = (c110+dx111+dy111+dz111);
            /* cell c112 is within visible neighbor
            c112 = (c111+dx112+dy112+dz112);
            /* cell c113 is within visible neighbor
            c113 = (c112+dx113+dy113+dz113);
            /* cell c114 is within visible neighbor
            c114 = (c113+dx114+dy114+dz114);
            /* cell c115 is within visible neighbor
            c115 = (c114+dx115+dy115+dz115);
            /* cell c116 is within visible neighbor
           
```

- [illegible]

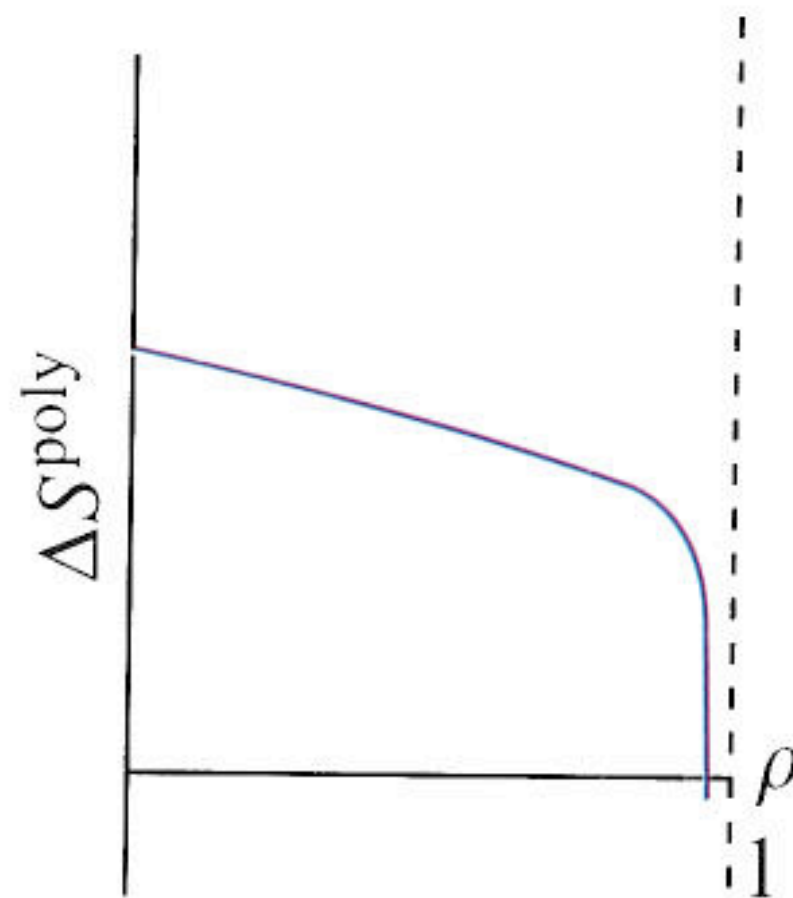
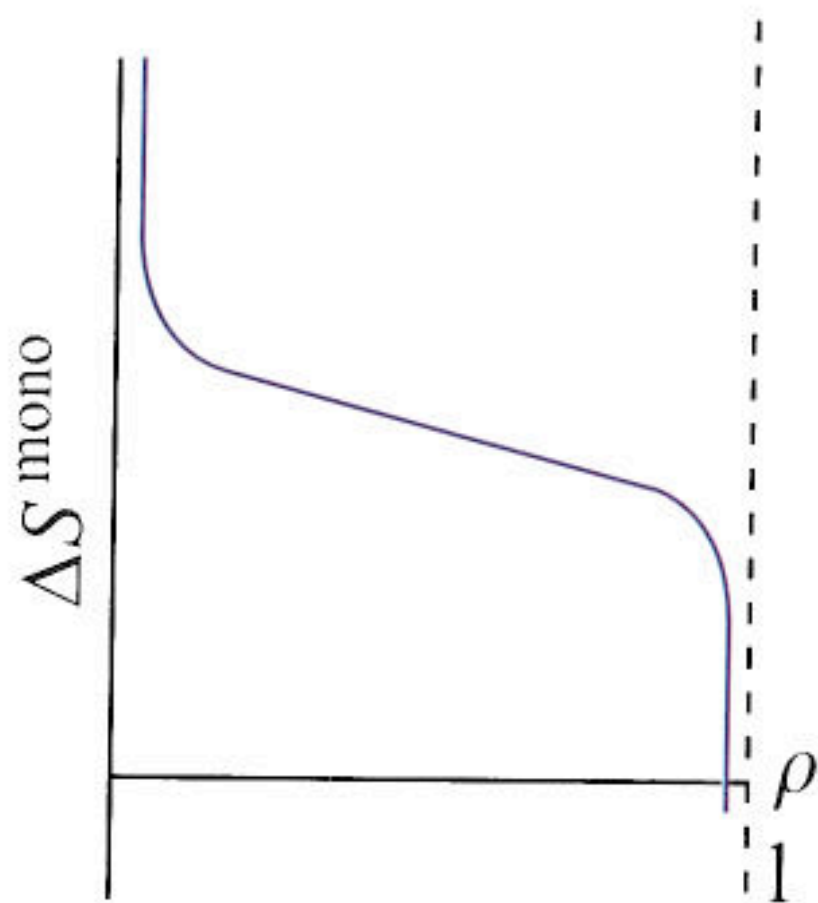
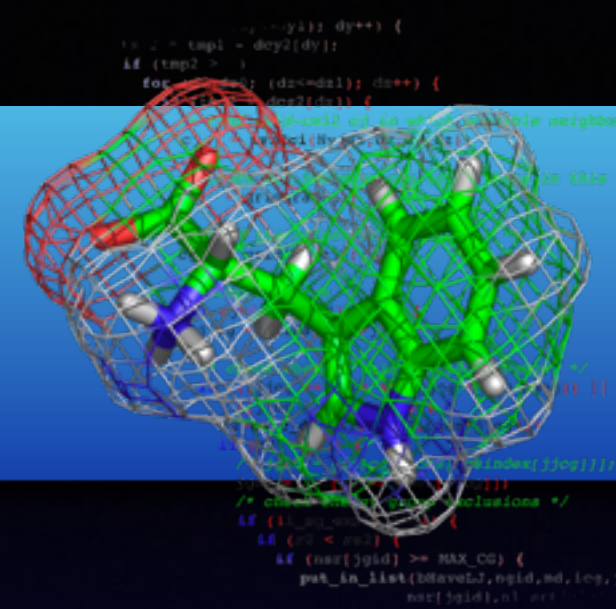


Entropy effects



- The energy per monomer is roughly the same in a cloud and chain
- How does the entropy change?
 - Accessible volume per monomer
- Cloud: $V' = (V - N\omega) / N = V / N(1 - \rho) = \omega / \rho(1 - \rho)$
- Chain: $V' = \Omega(1 - \rho)$
- Entropy $S = k \ln V'$

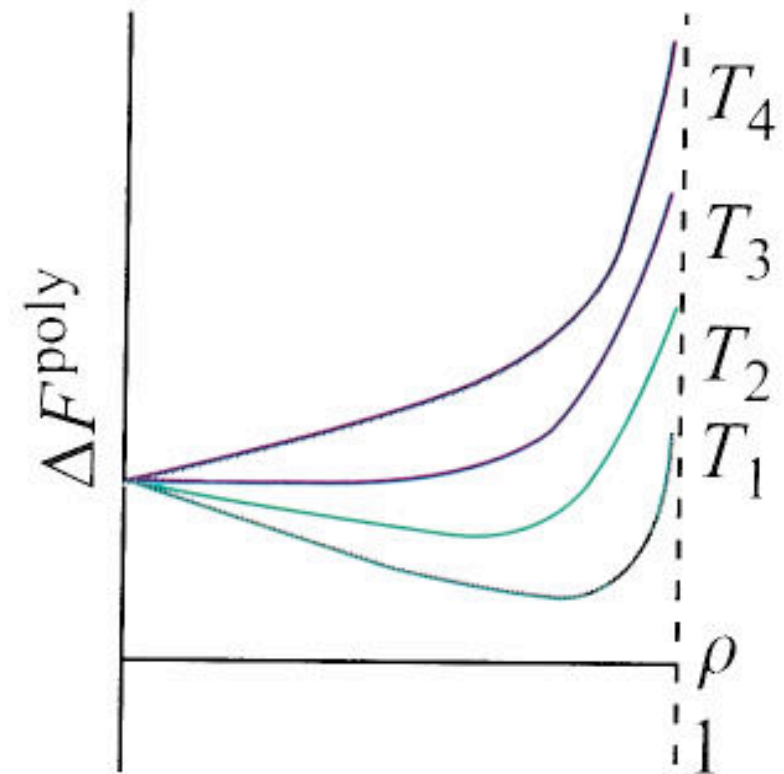
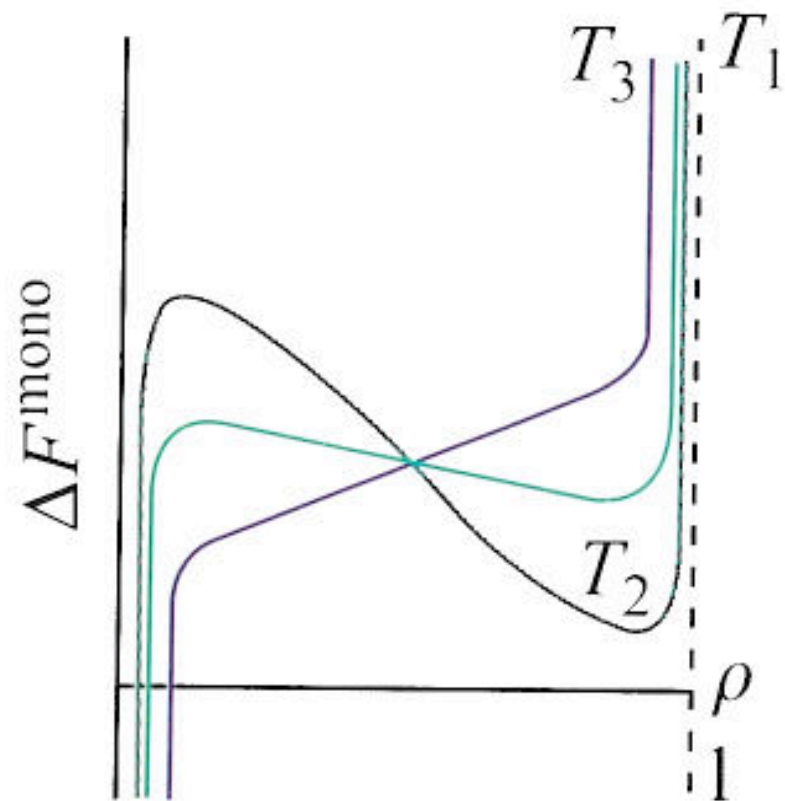
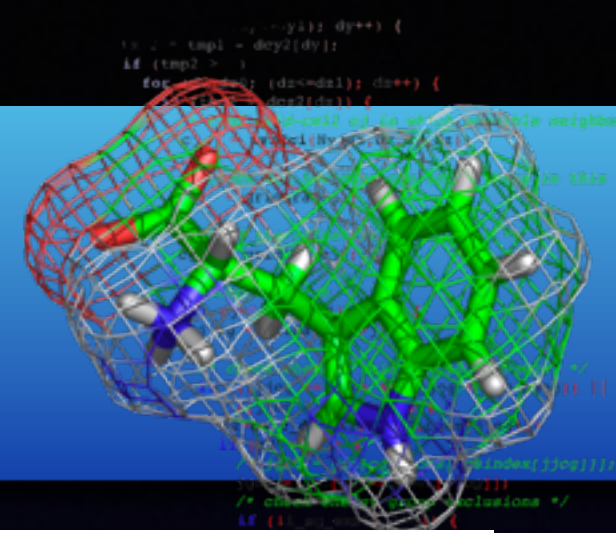
Homopolymers



$$S = k \ln \omega / \rho(1 - \rho)$$

$$S = k \ln \Omega(1 - \rho)$$

Homopolymers



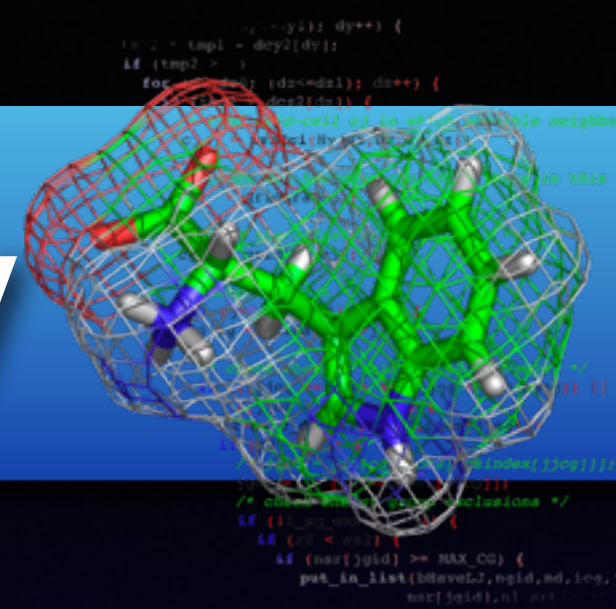
$$T_1 < T_2 < T_3 < T_4$$

Phase
transition

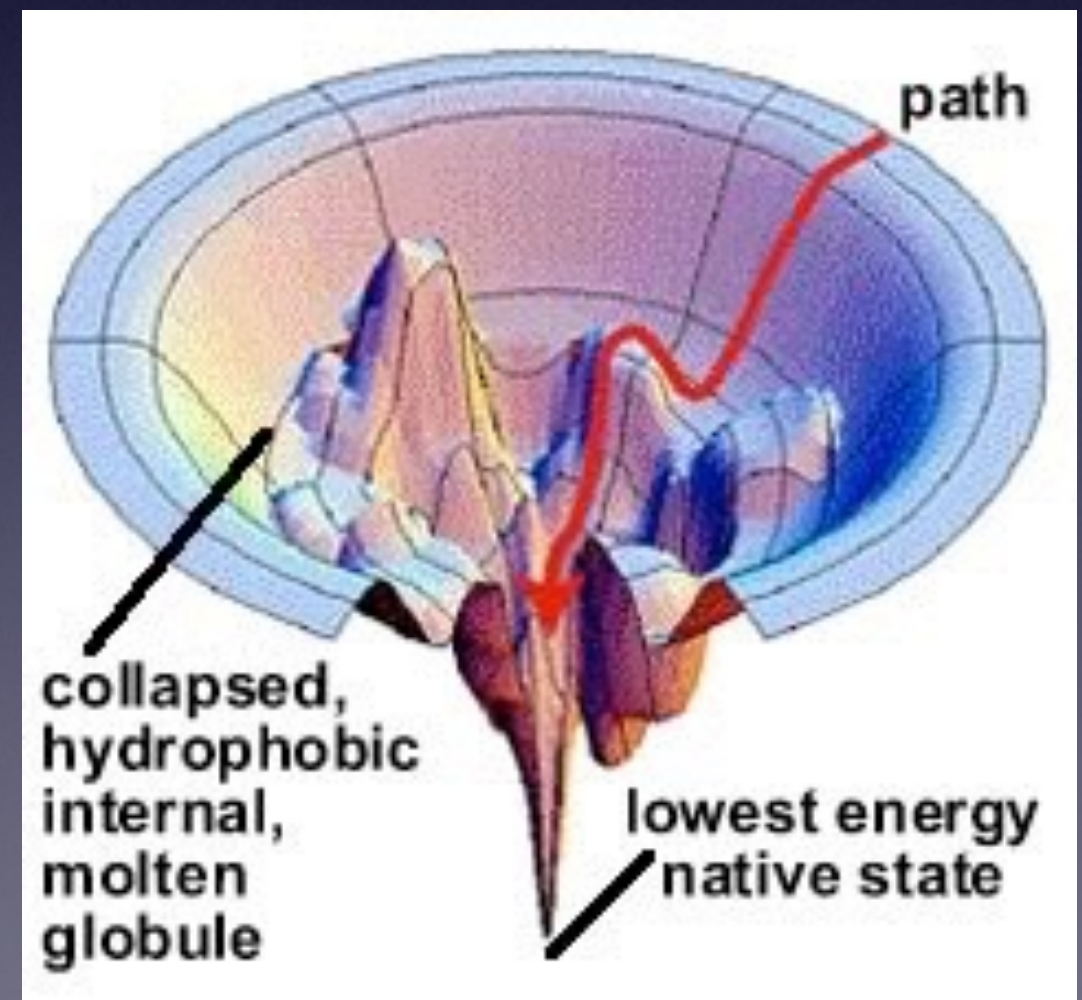
$$\rho_1 < \rho_2 < \rho_3 < \rho_4$$

No first-order
phase transition

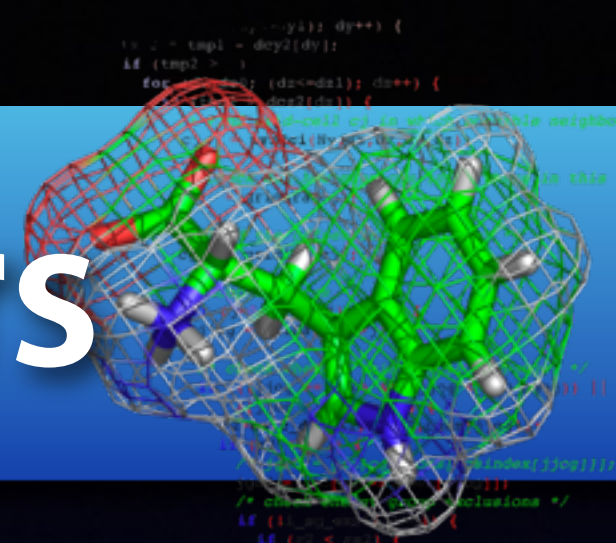
Protein free energy



- Free energy barriers is a basic requirement of an all-or-none / first-order phase transition
- Where does it come from in proteins?

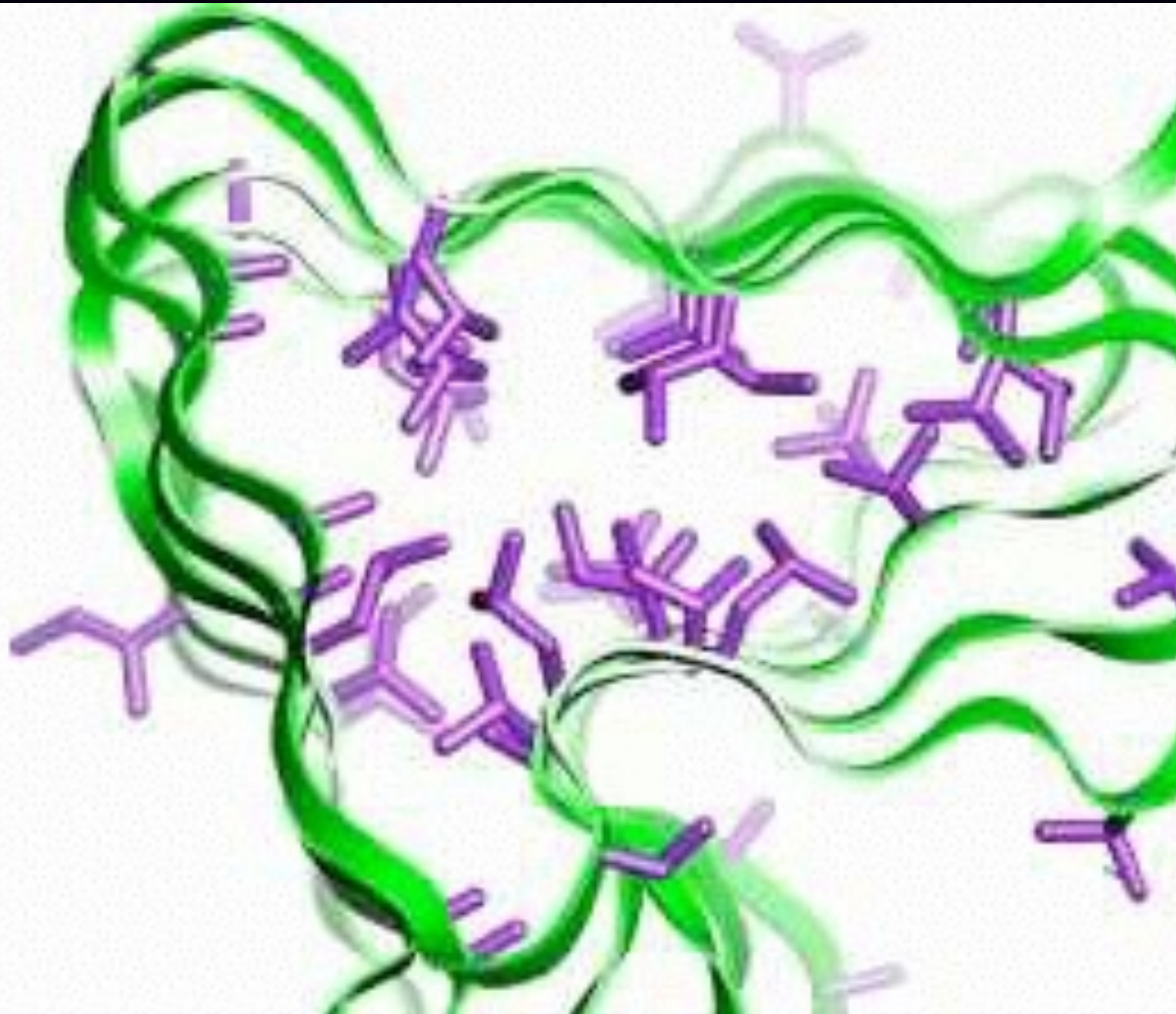


Free energy barriers

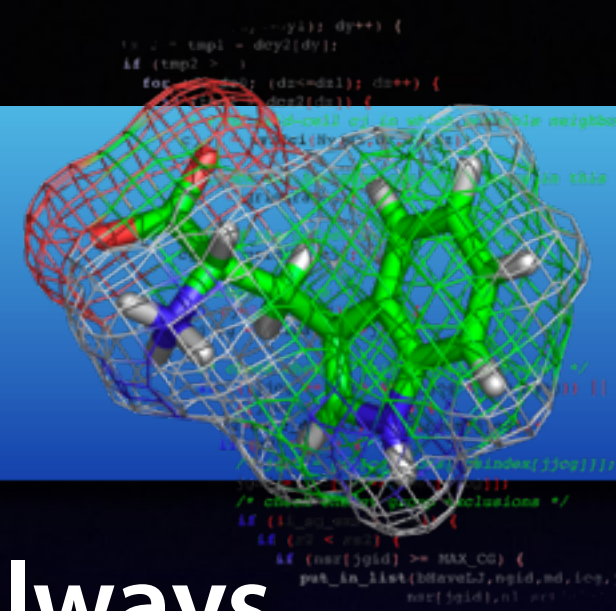


Sidechains are very well packed in native state

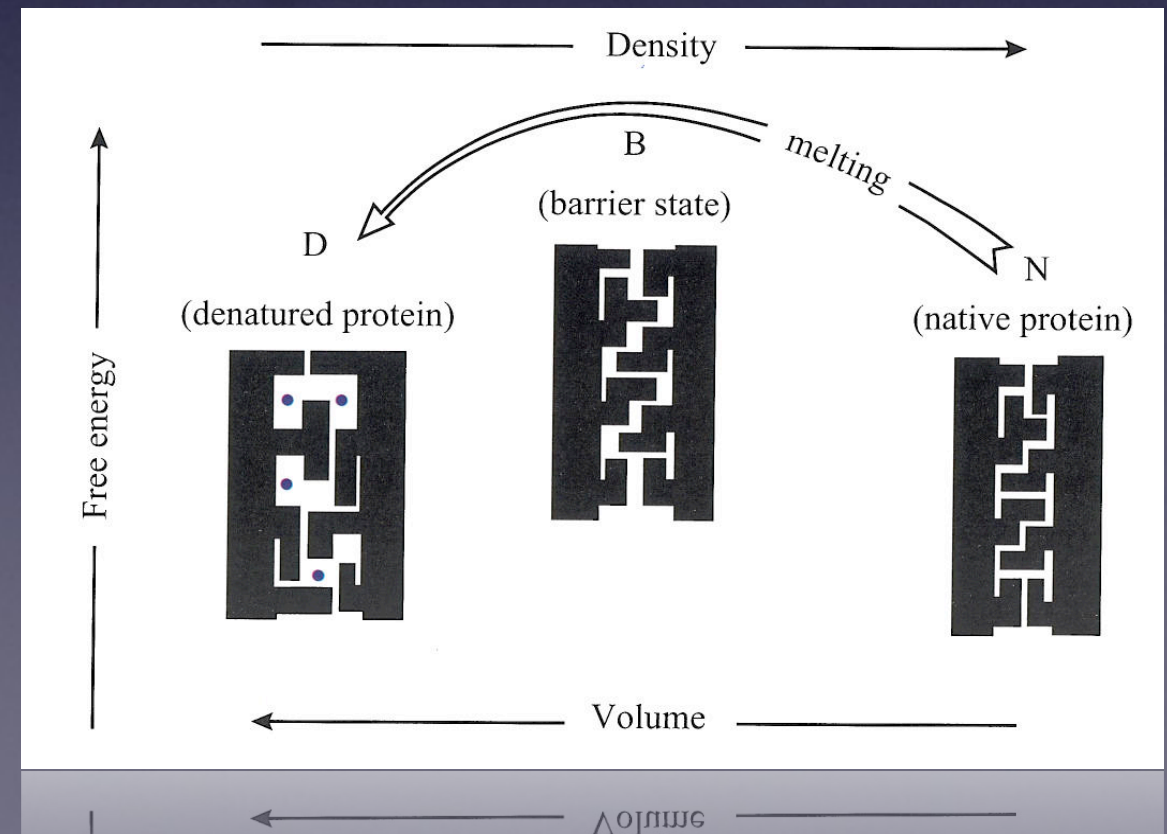
More like a solid than a liquid - 80% of the interior volume filled



Melting dynamics



- The initial denaturation process is always unfavorable, since the protein expands and ruins sidechain packing
- Water cannot yet enter (too packed)
- Eventually water gets in, barrier is crossed



3D molecular model of a protein-ligand complex. The protein is shown as a green mesh surface. The ligand is shown as a stick model with green carbon atoms, red oxygen atoms, and blue nitrogen atoms. The ligand is bound within a pocket of the protein. The background is a solid blue color.

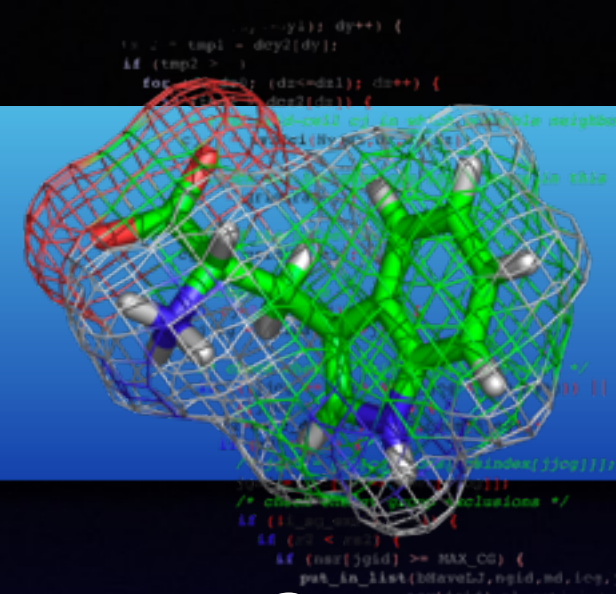


Solvent density in core

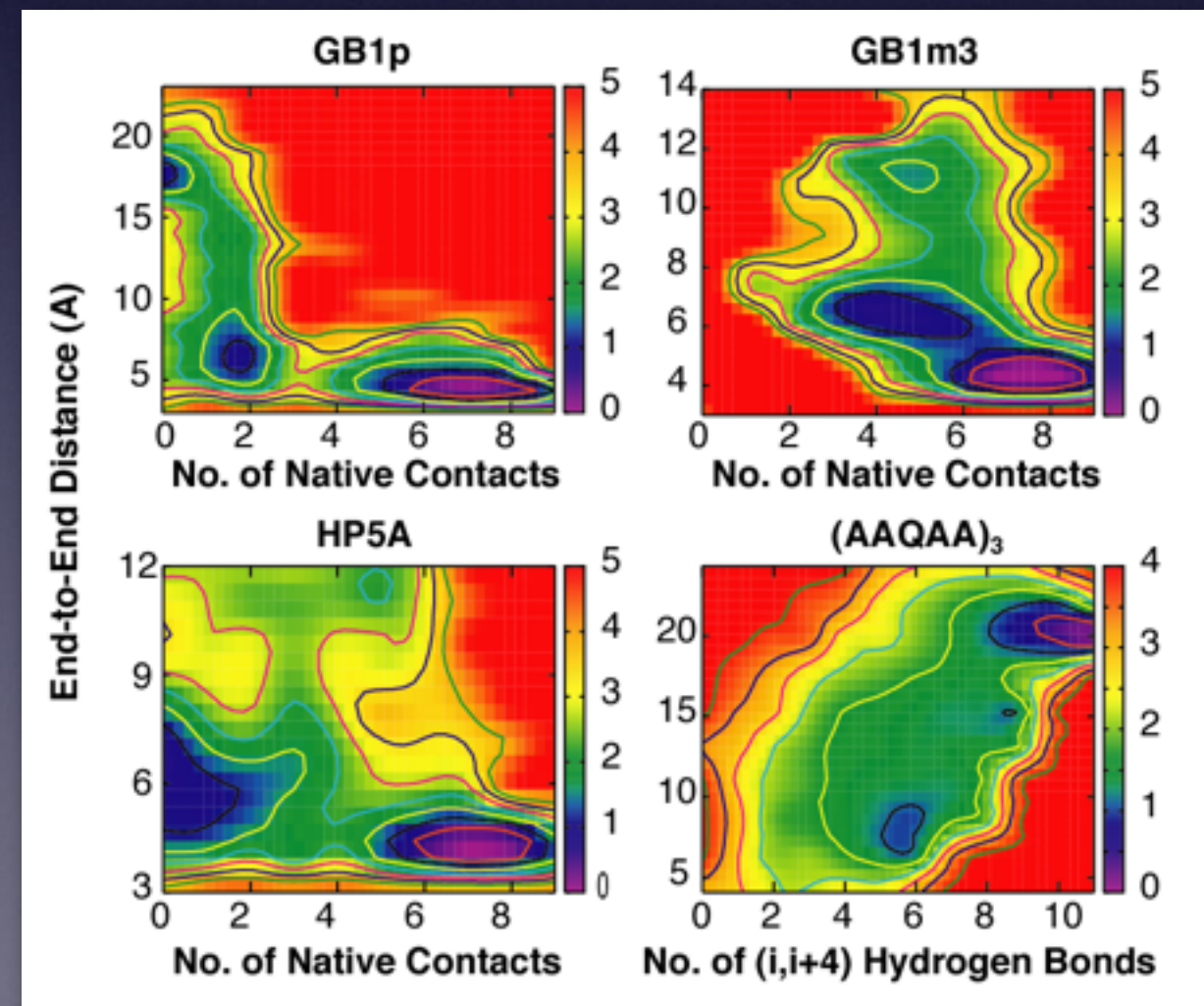
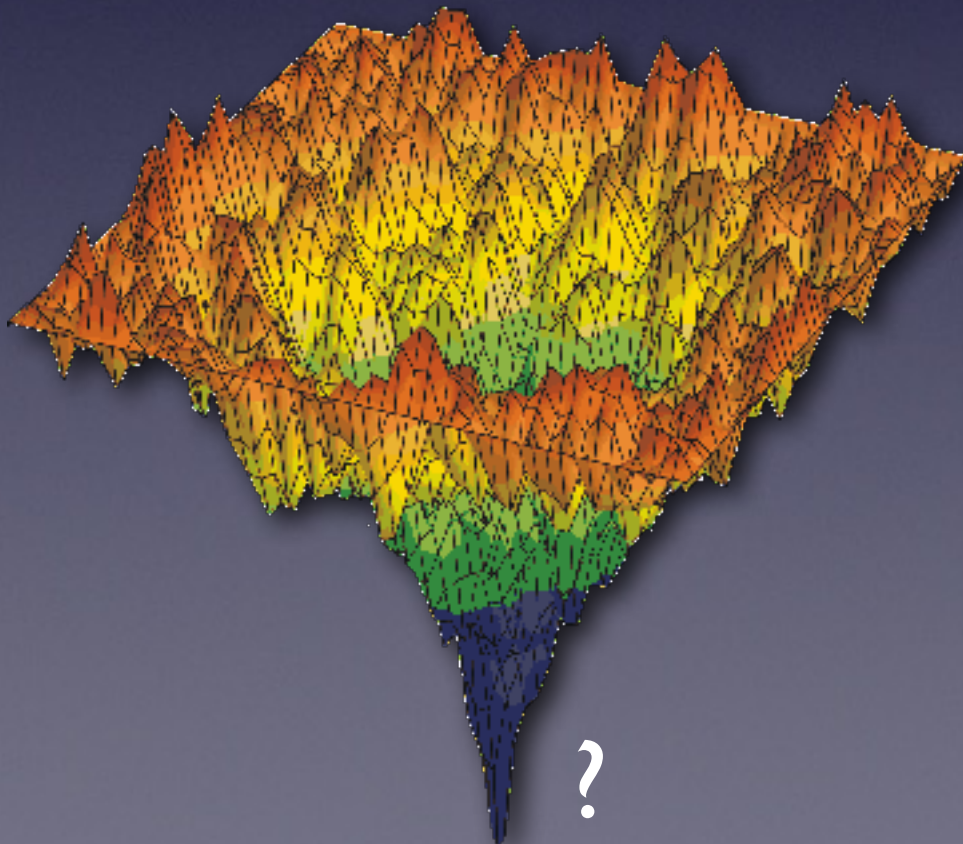
[illegible]

**Entropy shows a jump
when volume has
increased enough for
sidechains to move**

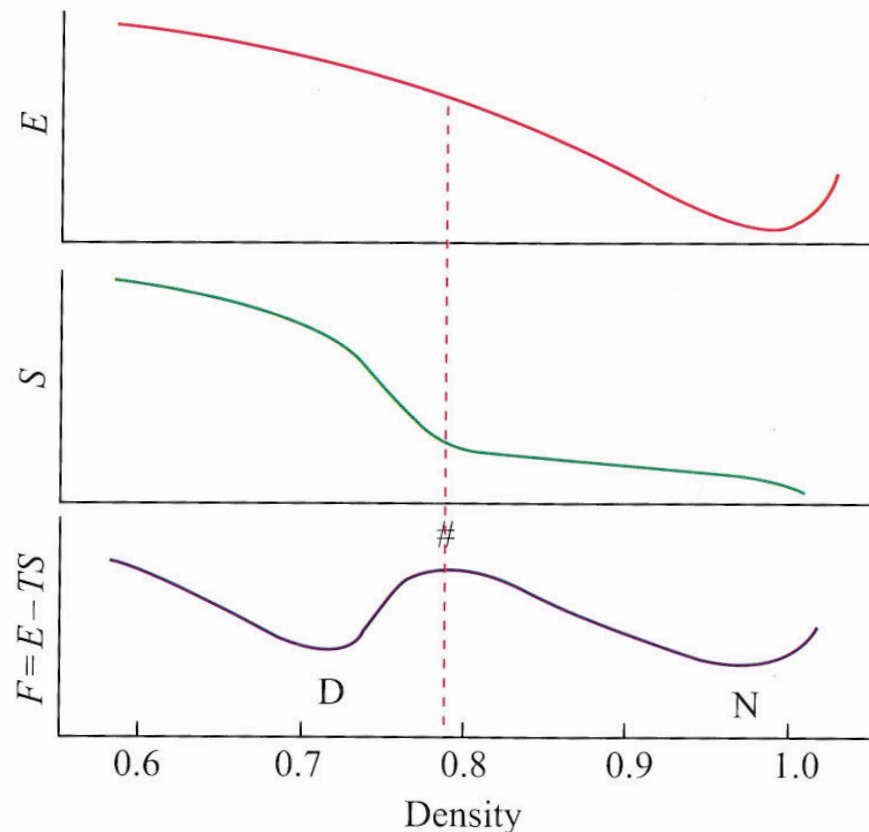
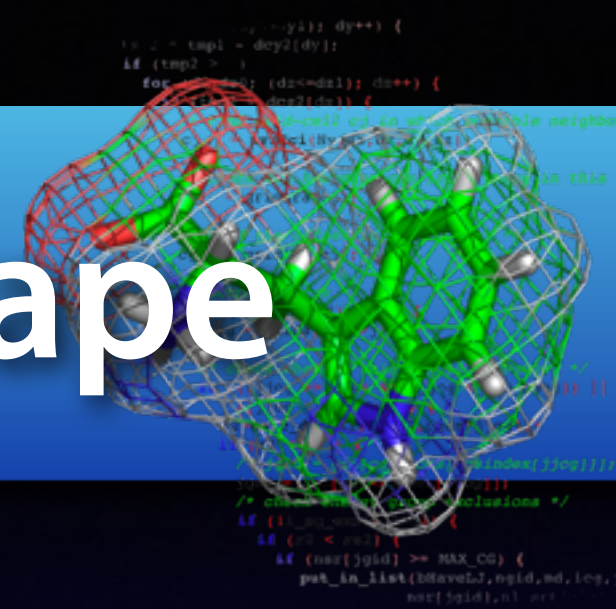
The native state



- What defines the native state? Uniqueness?
- Close packing
- Low energy



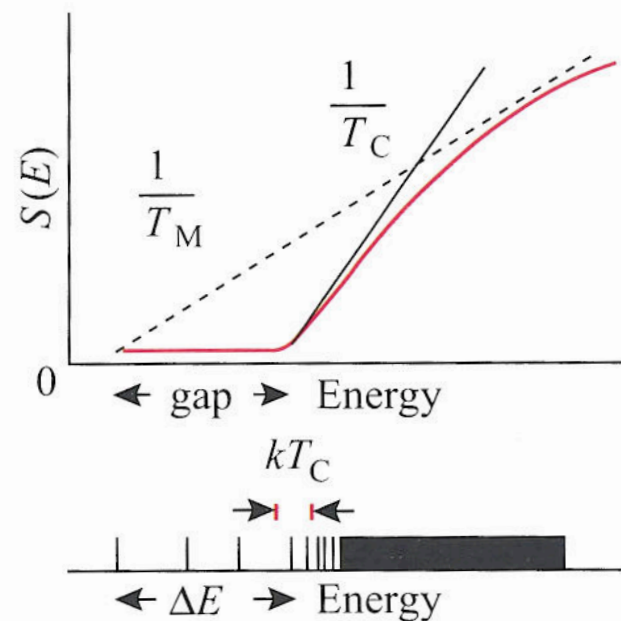
Entropy in the landscape



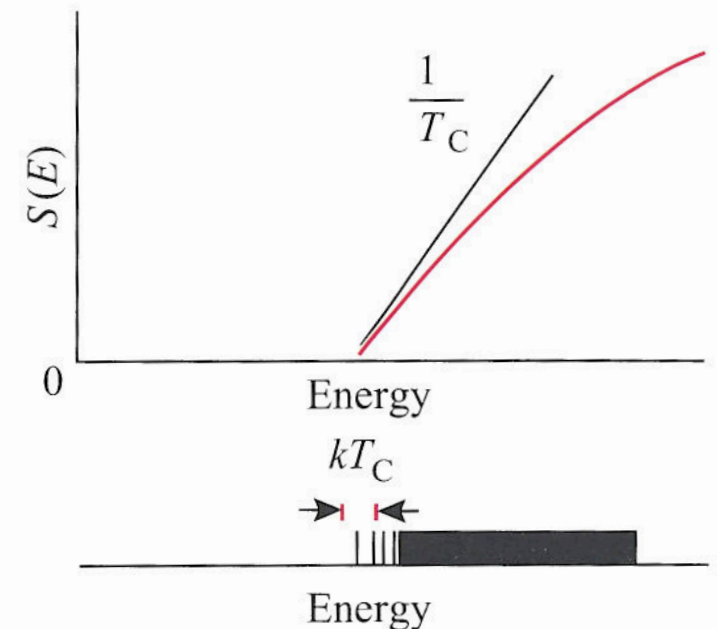
E, S, and F
vs. density

$$\rho(r) \propto \exp -\Delta E / kT$$

Plot S vs E:



(a) protein chain



(b) random heteropolymer

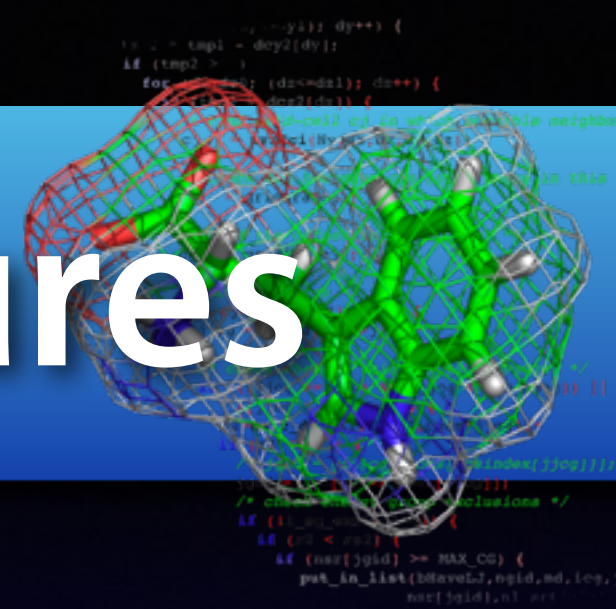

```

        dy--y1); dy++) {
    int i = tmp1 - dy2(dy);
    if (tmp1 > 0)
        for (i = 0; i < (dx+dx1); dx++) {
            if (i < 0 || i > dx1)
                continue;
            int cell = 0;
            for (j = 0; j < (dy+dy1); dy++) {
                if (j < 0 || j > dy1)
                    continue;
                int id = dx1 + i + (dy1 + j) * n;
                if (id < 0 || id > n * n)
                    continue;
                /* check if this cell is in the list of exclusions */
                if (!excl[id]) {
                    if (id < n * n) {
                        if (nax[id] >= MAX_CC) {
                            put_in_list(&naveLj, &id, nd, log,
                                nax[id], nl, set);
                        }
                    }
                }
            }
        }
    }
}

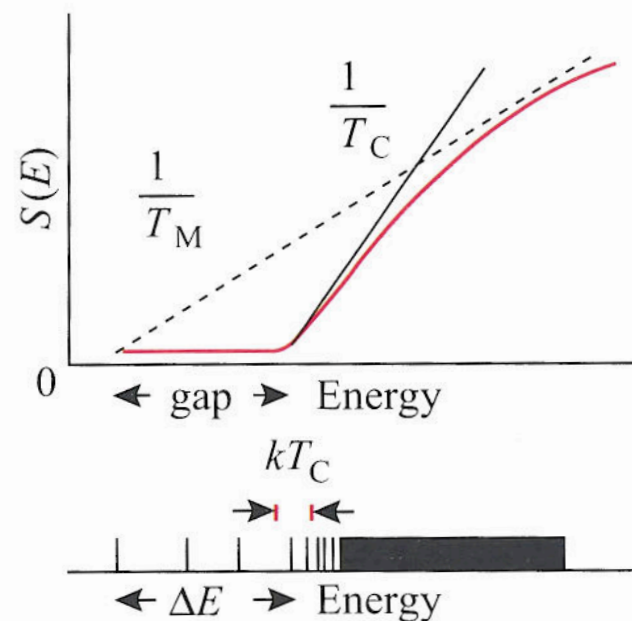
```

- Very few structures have low energy
- The lowest-energy structures seem to be separated from the rest by an energy gap
- If gap is large compared to kT , there will be a free energy barrier
- Specific for proteins because of packing
- *Not* valid for general polymers
- *Not* valid for random polypeptides!

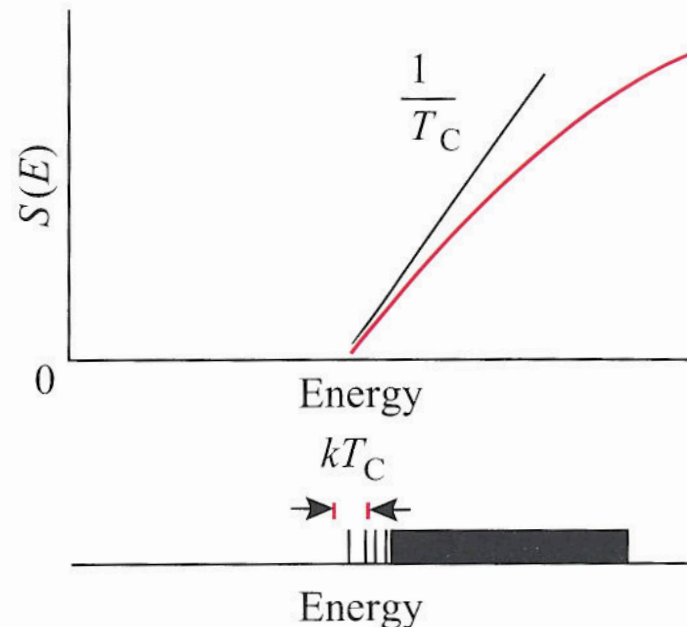
Transition temperatures



What determines if the barrier to the native state can be surmounted and if it is stable?



(a) protein chain



(b) random heteropolymer

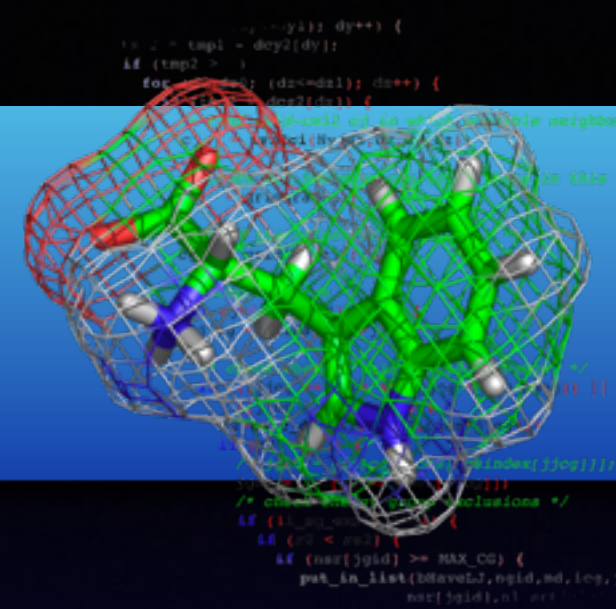
Folding/melting: transition between native and other states
Vitrification: chain gets stuck in glass-like low-energy state

What does $T_{\text{melting}} > T_{\text{vitrification}}$ mean for proteins?

[illegible]

- Sequences that fold into stable proteins do so because their native structure is separated by an energy gap from the rest
- Natural selection
- How common is this?
- Use the distribution from last lecture:
- $P_{\text{fold}} \approx \exp(-\Delta E/kT_{\text{vitrification}})$
- With $\Delta E \gg kT_{\text{vitrification}}$ (say, 20x) we get $P_{\text{fold}} \approx 10^{-8}$

Fold uniqueness



- But why is the native state unique?
- What about two stable native states?
- $P_{\text{fold}}^2 \simeq 10^{-16}$ $P_{\text{fold}}^3 \simeq 10^{-24}$
- Very rare, but it does happen!
 - Amyloid peptides
 - Remember: Prions!
 - Mad cow disease (BSE)

Summary



- Folding, denaturation, refolding
- Cooperative *and* all-or-none transition
- Cold & hot denaturation
- Molten globule and coil conformations
- Free energy barriers & energy gaps
- Proteins are different from random chains!

Book:
chapters 17 & 18