Supporting Information

DMD: an efficient and versatile simulation method for fine protein characterization

David Shirvanyants[1], Feng Ding[1], Douglas Tsao[2], Srinivas Ramachandran[1], Nikolay V. Dokholyan[1]

[1]Department of Biochemistry and Biophysics, School of Medicine
[2]Department of Chemistry
University of North Carolina, Chapel Hill, NC 27599, USA
*Correspondence: dokh@unc.edu

1. Summary of simulated proteins.

Name	Reference	Composition	Exp. τ_f	Simulation time	CPU days
villin headpiece	1YRF	α	1 μs	10 μs	200[a]
WW domain	2F21	β	2 μs	12 μs	200[a]
ACBP	2ABD	α	6 ms	11.2 μs	1603[b]
Ubiquitin	1UBQ	α/β	2 ms	11.0 μs	1290[b]
SH3	1FYN	β	1 ms	9.6 μs	1478[b]
λ-repressor	1LMB	α	2 ms	13.2 μs	600[a]
villin 14T	2VIK	α/β	15 ms	12.2 μs	1165[b]

Table S1. Summary of simulated proteins. CPU days is total CPU time used for all trajectories, simulation time is total length of all trajectories.

[a] using 1 CPU per trajectory (serial DMD).

[b] using 4 CPUs per trajectory (parallel DMD), simulation wall-clock time is 1/4 of the CPU time.

2. DMD algorithm and force field modifications

DMD algorithm. DMD is a special type of molecular dynamics simulation where pairwise interaction potentials are modeled with discontinuous functions[1]. In DMD simulations, each atom moves with a constant velocity until it collides with another atom at the distance where their pairwise potential energy changes. During each collision event, the two colliding atoms change their velocities instantly based on the conservation laws of energy, momentum, and angular momentum. Therefore, the dynamics of the molecular system is computed by iterative prediction of the next collision and update of colliding atoms.[2,3]

In order to efficiently compute collisions between spatially close atoms, the simulation box is usually divided into sub-cells, where each sub-cell holds a list of atoms inside itself.[2,3] Dimensions of the sub-cells are chosen to contain an optimal number of atoms, which is algorithm specific and depends on system density and interaction range of atom pairs. List of atoms in a sub-cell updated every time an atom moves from one sub-cell to another. In DMD, this atom transition is treated as a special pseudo-collision event. Using sub-cells reduces amount of computing required to predict collisions for each atom *i*, as only the collisions between atom *i* and the atoms in the neighboring 3^3=27 cells need to be computed. Different strategies have been proposed for scheduling of predicted events. Rapaport [4] proposed a priority tree containing all possible collisions between an atom and all atoms in the neighboring 27 cells. The priority tree is sorted according to the collision time. Another scheduling algorithm, proposed by Lubachevsky[2,5], is a single-event scheduling approach, where only the soonest collision for each atom is stored in a fixed-length binary tree. The advantage of single-event scheme is reduced memory requirements and usage of fixed-size data structures for storing predicted collisions, which results in a more optimal code. On the other hand, multiple-event scheme eliminates the need to recompute the predicted events in case when the single stored event becomes invalid. Comparison of these two scheduling schemes shows that either single-event[6] or multiple-event[7] scheme may result in higher efficiency, likely to be subject to hardware organization, as well as the simulated system and algorithm details. Finally, in the approach suggested by Marin[8,9] all predicted events are stored but only the soonest events for every atom are scheduled to event queue, which offers a good compromise between single-event and multiple event approaches. Thus, if the soonest event of an atom becomes invalid, a next soonest event of that atom can be extracted from storage and scheduled for execution. We use scheduling approach of Marin for serial DMD simulations. In parallel DMD code, we have adopted a simplified approach of storing n>1 predicted events for every atom. We have determined that optimal n is 2-3 events per atom. This simple event scheduling method offers somewhat lower performance than Marin's algorithm, but reduces data complexity, crucial for development of parallel algorithms.

In order to further increase the computational efficiency of DMD simulations, we developed several additional optimization approaches. 1) We propose to split each cell into sub-cells with each dimension evenly divided by N_s. The neighboring sub-cells can be counted in a spherical manner with the distances less than the maximum interaction range. As the N_s value increases, the number of neighboring atoms for the calculation of next collisions decreases while the rate of cell-crossing increases. Therefore, there is an optimal value of N_s for maximal increase of DMD efficiency. In our calculations, we found N_s of 6 as the optimal value. 2) The most expensive calculation in the DMD algorithm is performed after each collision, when the DMD algorithm re-evaluates the collision times between the colliding atoms and their neighboring atoms. The costly square root calculation is executed to accurately compute the collision time. However, only one or even none of the predicted collisions will take place. On the other hand, the evaluation of whether the collision between two atoms will happen within a given time, Δt, does not require the square-root calculation. Therefore, we only need to perform the accurate but expensive calculation of the collision time if a collision is within the given time of Δt. The choice of either too small or large Δt will result into extra calculations without improving the efficiency. We propose to set a cutoff Δt for a given atom based on the atom's average collision time, ~<t_{col}>, which can be updated periodically. In our simulations, we set Δt = 4<t_{col}> for DMD optimization. In addition, we also adopt a new *O(1)* sorting algorithm proposed by Paul[10]. The detailed algorithm of these optimizations can be found in ref.[11].

Screened charge-charge interactions. In addition to the previous version of the atomistic DMD force field[12], we also add electrostatic interactions between charged residues, including the basic and acidic residues. We assign integer charges to the central atoms of charged groups: CZ for Arginine, NZ for Lysine, CG for Aspartic acid, and CD for Glutamic acid. We use the Debye-Hückel approximation to model the screened charge-charge interactions. The Debye length is set at approximately 10 Å assuming water relative permittivity of 80 and a monovalent electrolyte concentration of 0.1 mM. We discretize the continuous electrostatic interaction potential with an interaction range of 30 Å, where the screened potential approaches zero (Fig. S1). We use the constant volume DMD simulations with periodic boundary conditions and control the simulation temperature using the Anderson thermostat [13].

Sequence-dependent local backbone interactions. For each three neighboring amino acids, $(\sigma_{i-1}, \sigma_i, \sigma_{i+1})$, we propose to assign a bias potential between the C_α of residues *i-1* and *i+1*, $E(d_{i-1,i+1}|\sigma_{i-1}, \sigma_i, \sigma_{i+1})$, which depends on the amino acid sequence. The value of C_α distance $d_{i-1,i+1}$ solely depends on the backbone dihedral angel of amino acid, (ϕ_i, ψ_i). As a result, there are three highly populated distance ranges of $d_{i-1,i+1}$, which corresponds to α-helix, β-strand, and poly-proline II regions in the Ramachandran plot[14]. Therefore, we use a potential function with three square-well basins corresponding to the three populated states.

In order to determine the relative energy of these three different regions, we perform statistical analysis of the structures in protein databank [15]. We use a subset of 4,888 non-redundant high-resolution PDB structures with the percentage identity cutoff of 25%, the resolution cutoff of 2.0 Å, and the R-factor cutoff of 0.25[16]. Due to the large number of triplet sequences (8,000), the C_α distance data for many sequences is sparsely populated. In order to accurately estimate the relative distribution of these three states for the sparsely populated sequences, we adopt a Bayesian statistical analysis proposed by Dunbrack and Cohen[17] to derive the sidechain rotamer library. Bayesian statistics requires the assumption of a prior distribution for parameters over their range of possible values. This prior distribution can be derived from previous data or from pooling some of the present data. In our case, we obtained the prior distributions from the product of the probabilities that are dependent of either the first amino acid or the last amino acid. For a given sequence, we assign the relative potential energy of each state according to the estimated probabilities as well as the averaged sequence-independent distributions:

$$E^i_{state} = E_0\left(\Delta E_\alpha - \log\frac{P^i_{state}}{P^0_{state}}\frac{P^0_\alpha}{P^i_\alpha}\right),$$

where E_0 and ΔE_α are adjustable parameters, independent of amino acid type, that can be used to emphasize the bias potential and relative weight of the α-helical state; $P^i{}_{state}$ is the relative weight of the given state (α-helix, β-strand, or poly-proline II) for the triplet at position *i* and $P^0{}_{state}$ is the relative weight of the state observed in simulation without the biased potential. We provide the $P^0{}_{state}$ determined from DMD simulations of individual triplets in **Table S2**.

Amino acid	**α**	**PPII**	**β**
Glycine	0.3		0.7
Proline	0.4	0.6	
All other	0.2	0.6	0.2

Table S2. Relative weight of secondary structure in the absence of the bias potential.

3. Stabilizing effect of the force field modifications

Electrostatic stabilization

It has been shown that long-range electrostatic interactions are crucial force that guides protein along its folding pathways and stabilizes the native state. We analyze the stabilizing effect by comparing conformational states population achieved in equilibrium simulations initialized from crystal structure conformations in the presence and in the absence of the long range interactions.

We implement a simple model of electrostatic interactions assuming constant and uniform dielectric permittivity, using formal charges and restricting charge-charge interactions between 4.5 and 30Å.

When compared to the simulations with only short-range interactions, trajectories with electrostatic interactions contain higher population of near-native conformations. The higher density of conformations at the native basin indicates higher barrier between folded and unfolded states compared to the simulations without long-range interactions (**Figure S1**).

Sequence-based statistical potential

It is well known that the secondary structure formation has a strong dependence on the local amino acid sequence. A force-field constructed from pairwise potentials, such as most of the MD force-fields, does not account for the environment effect, such as modulation of residue interactions by its neighbors along the chain. Here we attempt to take into account one aspect of this modulation, namely sequence based propensity toward a specific secondary structure. Analyzing protein crystal structures available in Protein Databank for all of 3-letter sequences we determine the probability to form α-helix, β-strand or polyproline II left-handed helix. This probability is then used to tweak the DMD force-field to modulate the propensity of protein sequences to form specific secondary structure in the course of simulation. This force-field adjustment is done once and applied to all proteins without additional corrections.

Sequence based force-field modulation also has a stabilizing effect on the native conformation. Compared to the bare DMD force-field the addition of sequence based contribution marginally shifts RMSD distribution towards smaller values for both of the villin headpiece and WW-domain (**Figure S1**). Interestingly though, combination of this statistical potential with electrostatic potential results in drastic increase of near-native populations.

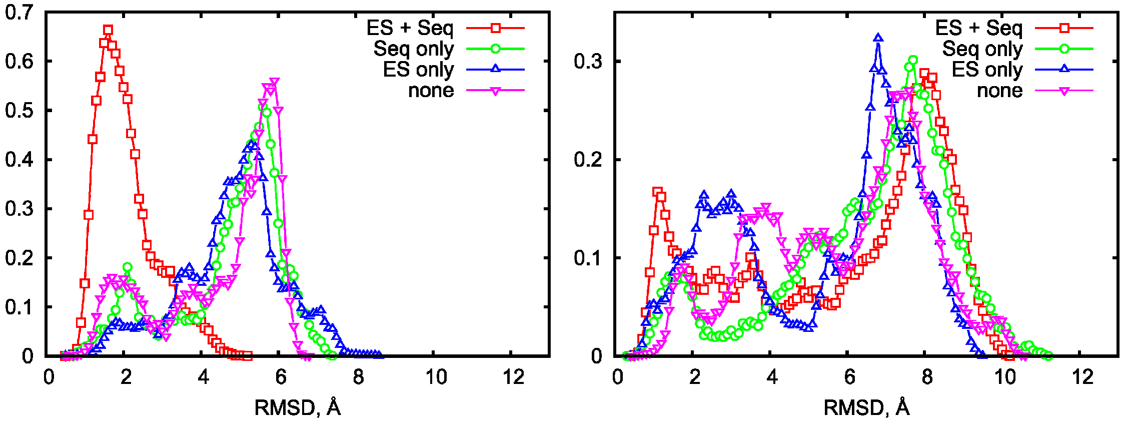

Figure S1. Distribution of root-mean-square deviations in the sampled trajectory for villin headpiece (left panel) and WW-domain (right panel). Starting configuration is crystal structure, trajectory length is 50ns.

4. Sequence-based statistical potential

	Alpha		**PPII**		**Beta**	
Amino acid	*Min*	*Max*	*Min*	*Max*	*Min*	*Max*
A	5.32	5.58	6.20	6.75	6.90	7.27
C	5.32	5.62	6.15	6.70	6.85	7.27
D	5.32	5.65	6.10	6.75	6.90	7.27
E	5.32	5.58	6.18	6.70	6.85	7.22
F	5.32	5.68	6.18	6.70	6.85	7.24
G	5.34	5.78			6.50	7.34
H	5.32	5.68	6.15	6.70	6.85	7.26
I	5.32	5.58	6.16	6.70	6.85	7.20
K	5.32	5.60	6.16	6.70	6.85	7.20
L	5.32	5.60	6.16	6.70	6.85	7.16
M	5.32	5.56	6.18	6.70	6.85	7.27
N	5.32	5.70	6.10	6.75	6.85	7.26
P	5.30	5.64	6.20	6.70		
Q	5.32	5.62	6.18	6.70	6.85	7.24
R	5.32	5.60	6.16	6.70	6.85	7.24
S	5.32	5.62	6.30	6.76	6.90	7.27
T	5.32	5.62	6.30	6.76	6.90	7.26
V	5.32	5.58	6.18	6.70	6.85	7.20
W	5.32	5.62	6.18	6.80	6.85	7.24
Y	5.32	5.64	6.18	6.80	6.85	7.26

Table S3. Distance-based constraints on β-carbon atoms for each of the secondary structure types. Minimum and maximum distance define the potential well between Cα atoms of residues *i-1* and *i+1* depending on the type of residue *i* (first column) and secondary structure type (first row). Depth of the well is determined by the probability of the given amino acid to form the corresponding secondary structure type. These probabilities are listed in the supplementary Excel sheet.

5. Parallel discrete molecular dynamics

The parallel DMD is derived from the serial DMD algorithm[1,10,11] by distributing event processing between multiple threads. Several possible approaches are described in. Our approach here is similar to methods proposed in [7], with few important modifications. First, we avoid using spinlocks and prevent threads from waiting to get their share of work. We store events in the shared Paul's type event queue[10], and let each thread to scan this queue continuously starting from the queue's head until an event is found that needs to be processed and is not yet taken by another thread (**Figure S2**). Such approach provides automatic load balancing between threads – every thread takes as much work as it is able to complete. One of the threads, called master thread, in addition to event processing performs inherently serial tasks, such as advancing time and writing trajectory files. Event processing is performed by all threads, although master thread, being additionally occupied with serial tasks, processes fewer events than any other thread. Events can be committed either by all threads or only by the master thread – we did not see significant difference in performance between these approaches.

Scaling of parallel DMD depends on the probability that two consecutive events in the queue are coupled, that is the outcome of an earlier event may change the predictions based on the execution time of the following event. If an event has been already evaluated, then as a result of coupling with preceding event it will be re-evaluated, thus wasting computing resource. Events become coupled when atoms participating in them are within the atomic interaction range. In

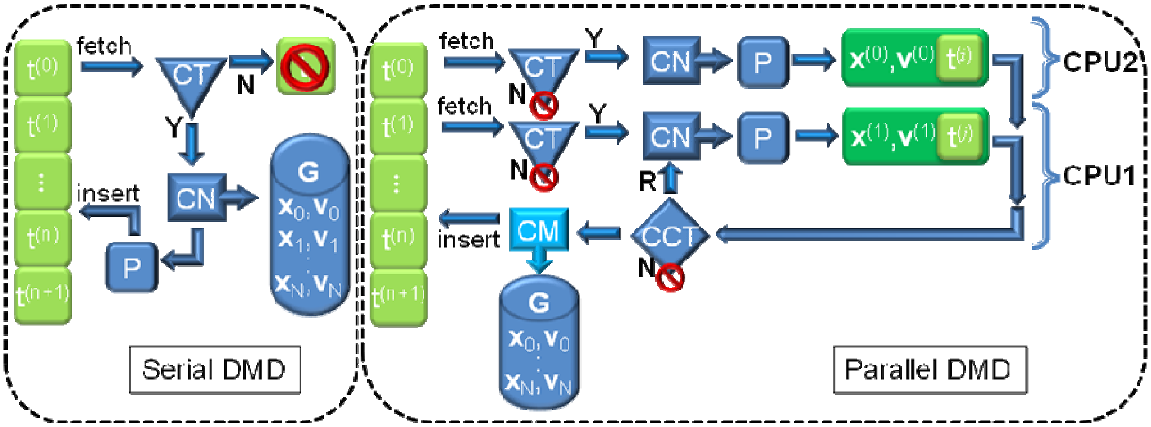

Figure S2. Serial and parallel DMD algorithm. In **serial DMD** (left panel) collision events predicted earlier to occur at times $t^{(0)}$, $t^{(1)}$… are extracted from the top of the sorted queue (green bricks) one by one. Next, an extracted collision is tested (CT triangle) to ensure that it has not been cancelled by previous collisions. If the collision has been cancelled (N), it is discarded. Otherwise (Y) DMD computes new coordinates and velocities (CN bar) of the participating atoms and updates global vectors (G) of atom states. Then DMD predicts new collisions for the participating atoms and inserts future collisions into the sorted queue. In **parallel** DMD several events are fetched from the queue by different threads (two are shown here) in parallel, though not necessarily simultaneously. Parallel processing continues with testing of extracted events (CT), evaluation of new positions and velocities and prediction of future collisions of their corresponding atoms (CN). Finally, each thread stores the new coordinates $\mathbf{x}^{(i)}$, velocities $\mathbf{v}^{(i)}$ and collision times $t^{(i)}$ in temporary data structures (dark green bars), and extracts next event from the queue. One of the threads has special duty of committing (CM) the stored atom data. This master thread tests if event has been cancelled or results of the earlier processing have become stale due to earlier collisions (CCT). Then it either discards the event or requests re-processing (R). If event has not been cancelled and evaluation results are valid, event is committed (CM) that is atoms' coordinates and velocities in global vectors (G) are updated, and predicted events are inserted into the sorted queue. Locks are used to ensure that no two threads fetch the same event for processing. All locks are non-blocking, that is if a thread finds that next event in the queue is already claimed, it tries the next event and so on, until it finds one that is available. Aside for event extraction, no additional synchronization or load balancing is required for the threads.

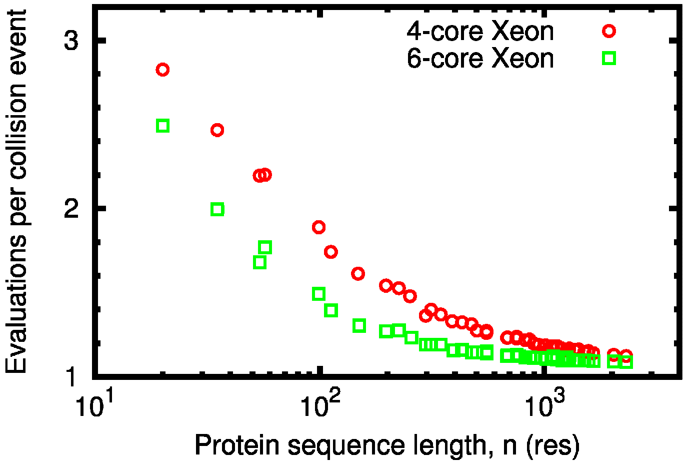

Figure S3. Average number of times a collision event is being evaluated by multithreaded code as function of protein size. In serial code every event is evaluated exactly once. In parallel code the number of event evaluations depends on the fraction of coupled events, which goes down as proteins size increases. Red circles correspond to 4-thread simulation on the 4-core Xeon, green squares are for 6-thread simulation on the 6-core Xeon.

small proteins their size is close to the interaction cut-off of 6.5Å for most pairs of atoms, which results in high fraction of coupled events in the queue (**Figure S3**). For larger proteins the number *k* of event evaluations falls approximately following the power law k=1+11$n^{-0.6}$.

To estimate the scaling of parallel DMD with number of threads we have compared the average speed increase over the serial code achieved on the 4-core Xeon and 6-core Xeon for proteins of length >1000 residues (**Figure S5**). We observe that maximum performance gain is achieved when number of threads equals the number of CPU cores on a dye. This can be explained by the architecture of modern CPUs, where data exchange is highly optimized between cores within one socket, as opposed to exchange between cores in different sockets via the dedicated data bus. This limitation on scaling contrasts the results of Khan and Herbordt [7], and is possibly caused by significantly higher rate of collisions (**Figure S4**), putting a higher load on the memory bus.

To test the assumption that parallel DMD scaling is limited by the speed of shared memory access we have inserted dummy loops into the event processing routine. The dummy loops waste

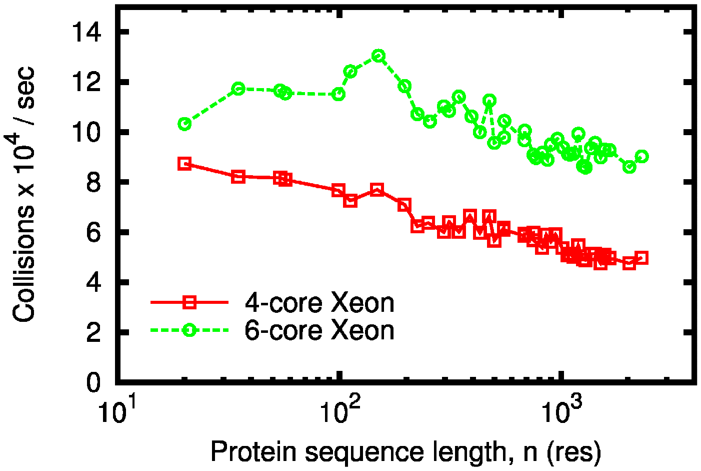

Figure S4. Number of pairwise atom collisions per second for proteins of different length. Red circles correspond to 4-thread simulation on the 4-core Xeon, green squares are for 6-thread simulation on the 6-core Xeon.

CPU cycles, but do not access any shared data. We have adjusted the added delay to match the event rate of serial code described in [7], to make it possible to compare results. Indeed, the delay loops have reduced the load on shared memory bus, and parallel DMD performance has scaled up to 8 cores on the dual-socket system (**Figure S5**, blue bars), reproducing the results of [7]. Of course, despite the better scaling the absolute performance of the code with delay loops is still below that of the code without delay loops.

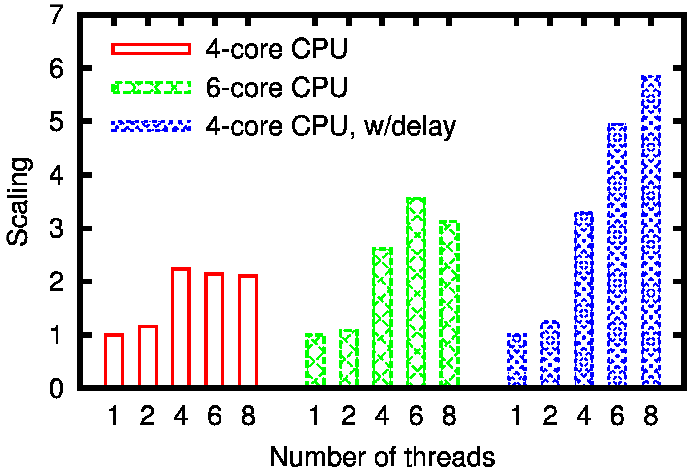

Figure S5. Parallel DMD performance scaling when using different number of threads. Red and green bars correspond to unmodified code running on 4- and 6-core Xeons. Blue bars show scaling of code with delay loops, which consume CPU cycles without accessing memory.

6. Equilibrium folding simulations

This section provides state density diagrams and native contacts frequencies for all proteins not displayed in the main text. In order to characterize the DMD force-field ability to predict native protein structure we plot state density diagrams using α-carbon RMSD, native contacts fraction (Q-value) and potential energy as order parameters (Figures S1-S6). In the state diagrams (panels *a*, *b* and *c*) we plot negative logarithm of the probability to observe a protein conformation with given values of order parameters: $U = -\log P_{state}$, where U has the meaning of the free energy. U is not a free energy since in our simulations we did not sample an entire conformational space of the proteins, and therefore can not accurately estimate the partition sum.

We compare the crystal structure with the closest sampled conformation by aligning the crystal structure with the simulated conformation having the smallest α-carbon RMSD (panels *d*, Figures S1-S6). The smallest RMSD itself is reported in Table 1 of the main text.

To observe sequence of structure formation in different segments, we compute the evolution of the fraction of native contacts for each individual residue for the trajectory (panels *e*, Figures S1-S6). The trajectory shown is the one in which conformation shown in panel *d* was sampled. All-residue average is shown as a 3-point strip on the top. Native contacts are defined as contacts between β-carbons (α-carbons for glycin) with spatial separation of less than 8Å and separation along the chain greater than two residues.

Time- and trajectory-average probabilities to form native contacts that were observed throughout the simulation with probability $P_{threshold}$ ≥1/32 are shown in panels *f*, Figures S1-S6. The observed native contacts are shown in upper triangle with gray level proportional to contact probability, while native contacts with probability below $P_{threshold}$ are marked with open circles. The lower triangle displays native contacts indicated by solid black squares.

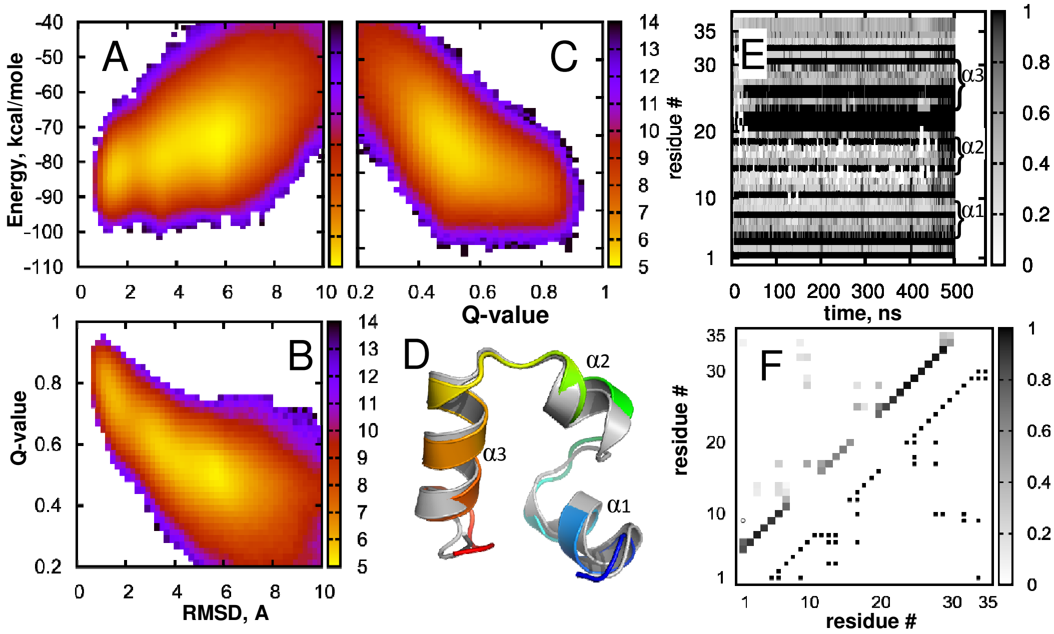

Figure S6. Villin headpiece. Panel contents and notation is same as in Figure 2.

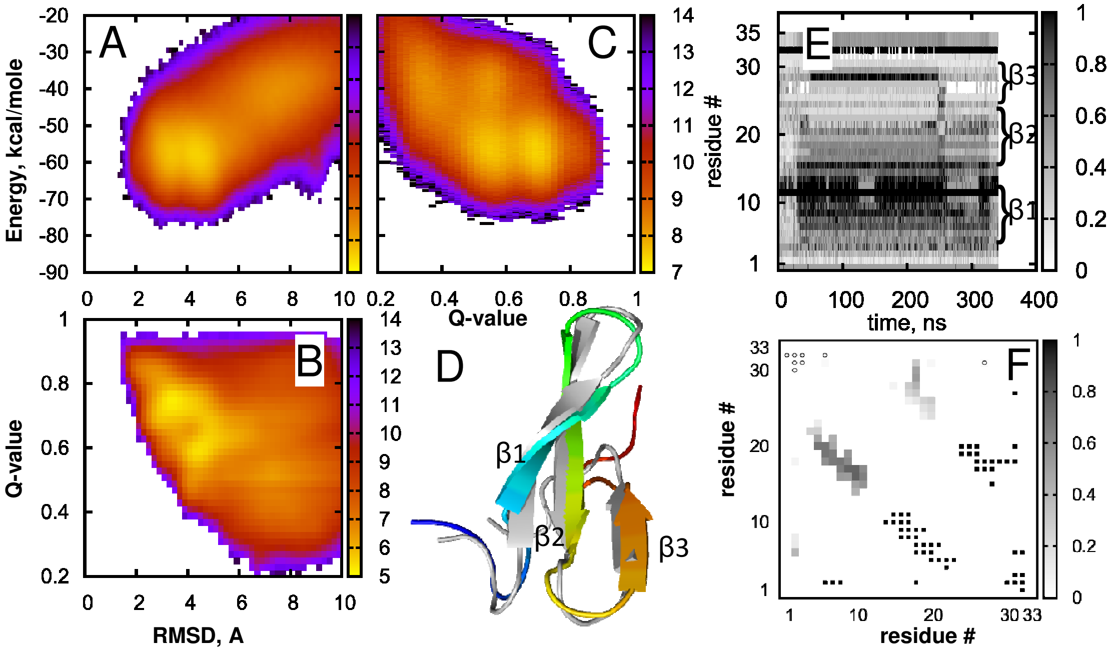

Figure S7. WW-domain. Panel contents and notation is same as in Figure 2.

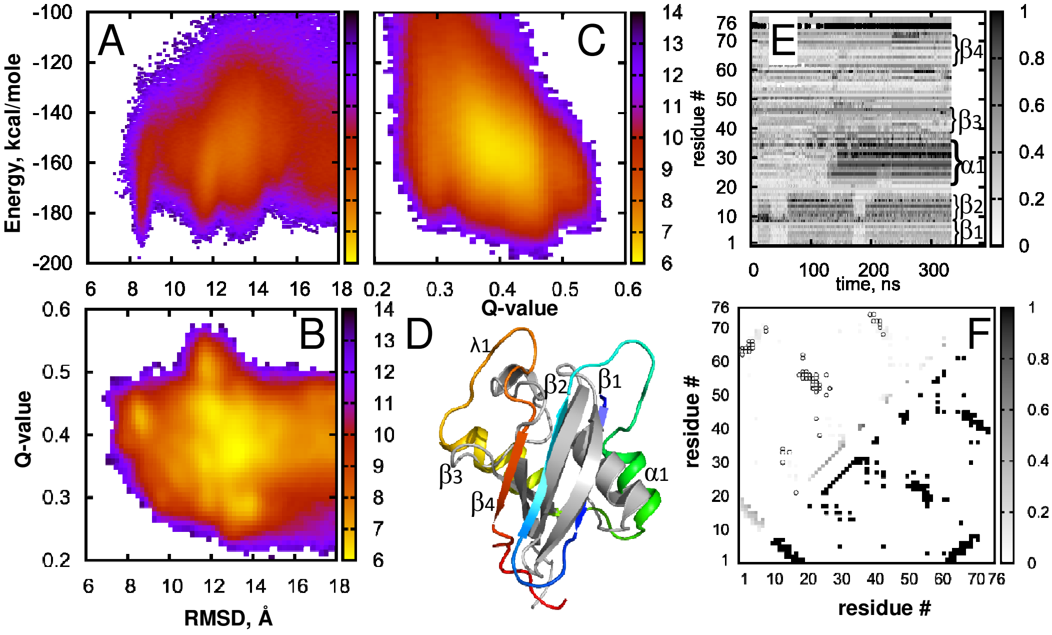

Figure S8. Human ubiquitin. Panel contents and notation is same as in Figure 2.

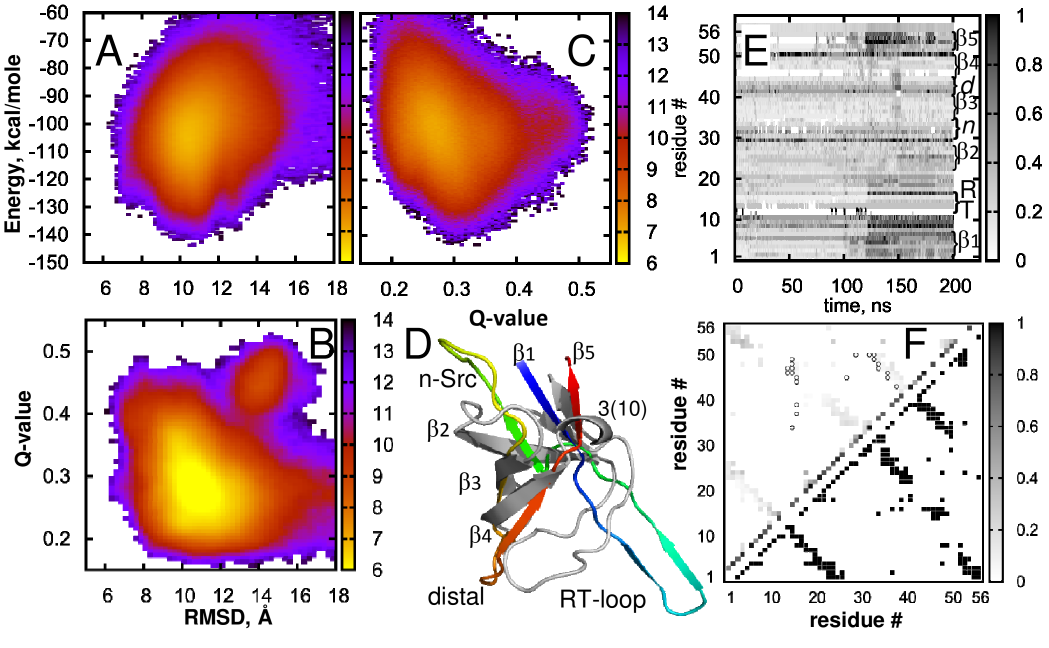

Figure S9. Src homology domain (SH3). Panel contents and notation is same as in Figure 2.

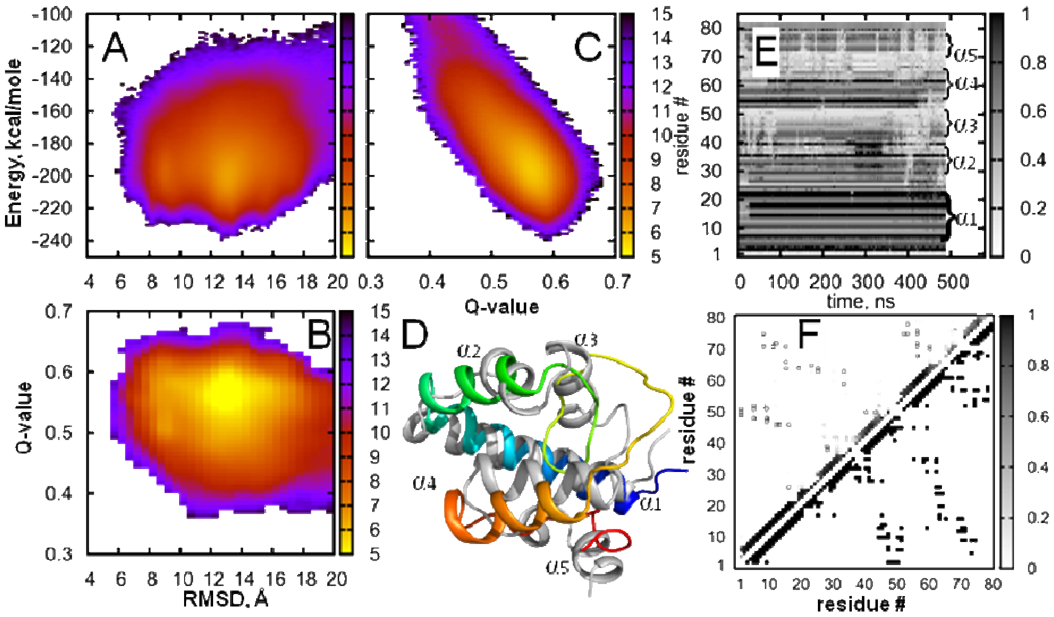

Figure S10. λ-repressor. Panel contents and notation is same as in Figure 2.

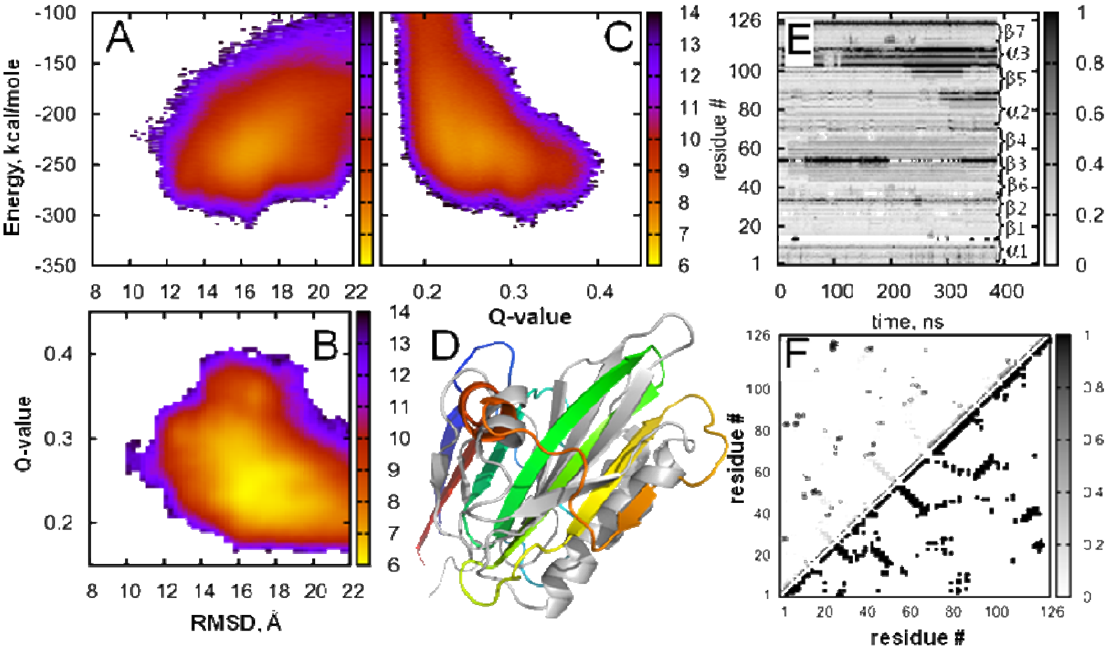

Figure S11. Villin 14T. Panel contents and notation is same as in Figure 2.

Supporting References

(1) Rapaport, D. C. *Journal of Physics A: Mathematical and General* **1978**, *11*, L213.

(2) Allen, M. P.; Tildesley, D. J. *Computer simulation of liquids*; Clarendon Press ; Oxford University Press: Oxford [England], 1989.

(3) Rapaport, D. C. *The art of molecular dynamics simulation*, 2nd ed.; Cambridge University Press: Cambridge, 2003.

(4) Rapaport, D. C. *Journal of Computational Physics* **1980**, *34*, 184.

(5) Lubachevsky, B. D. *Journal of Computational Physics* **1991**, *94*, 255.

(6) Smith, S. W.; Hall, C. K.; Freeman, B. D. *Journal of Computational Physics* **1997**, *134*, 16.

(7) Khan, M. A.; Herbordt, M. C. *Journal of Computational Physics* **2011**, *230*, 6563.

(8) Marin, M.; Risso, D.; Cordero, P. *Journal of Computational Physics* **1993**, *109*, 306.

(9) Berrouk, A. S.; Wu, C. L. *Powder Technology* **2010**, *198*, 435.

(10) Paul, G. *Journal of Computational Physics* **2007**, *221*, 615.

(11) Ding, F.; Dokholyan, N. V. Discrete Molecular Dynamics Simulation of Biomolecules. In *Computational Modeling of Biological Systems: From Molecules to Pathways*; Dokholyan, N. V., Ed.; Springer NY, 2011.

(12) Ding, F.; Tsao, D.; Nie, H.; Dokholyan, N. V. *Structure* **2008**, *16*, 1010.

(13) Andersen, H. C. *Journal of Chemical Physics* **1980**, *72*, 2384.

(14) Ramachandran, G. N.; Ramakrishnan, C.; Sasisekharan, V. *J Mol Biol* **1963**, *7*, 95.

(15) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. *Nucleic Acids Res* **2000**, *28*, 235.

(16) Wang, G.; Dunbrack, R. L., Jr. *Bioinformatics* **2003**, *19*, 1589.

(17) Dunbrack, R. L., Jr.; Cohen, F. E. *Protein Sci* **1997**, *6*, 1661.