

1 *Submitted to CrystEngComm*

2

3 **Molecular-level insights of small organic molecule dipyrone**  
4 **crystallization by MD-based strategies**

5

6 *Yi Sui<sup>a</sup>, Wenchun Jiang<sup>a, b, \*</sup>, Yingzheng Meng<sup>b</sup>, Zhuwen Shao<sup>b, c</sup>, Huibo Meng<sup>b, \*</sup>*

7

8 <sup>a</sup> School of Petroleum Engineering, State Key Laboratory of Heavy Oil Processing, China

9 University of Petroleum (East China), Qingdao 266580, P.R. China

10 <sup>b</sup> College of New Energy, State Key Laboratory of Heavy Oil Processing, China

11 University of Petroleum (East China), Qingdao 266580, P.R. China

12 <sup>c</sup> College of Electromechanical Engineering, Qingdao University of Science and

13 Technology, Qingdao 266061, P.R. China

14

15 \* Corresponding author, E-mail: [jiangwenchun@126.com](mailto:jiangwenchun@126.com) (W. Jiang);

16 [huibomeng@126.com](mailto:huibomeng@126.com) (H. Meng)

17

Table S1. Atom type and the atom charges of the Restrained Electrostatic Potential (RESP).

Atom No.	Atom index	Atom type	Atom charge	Atom mass
1	N1	n	-0.2072	14.0067
2	N2	na	-0.20837	14.0067
3	C3	cc	0.103756	12.01074
4	C4	cc	-0.03843	12.01074
5	C5	c	0.575799	12.01074
6	C6	c3	-0.10902	12.01074
7	C7	c3	-0.30428	12.01074
8	O8	o	-0.61416	15.99941
9	N9	nh	-0.10324	14.0067
10	C10	c3	-0.40563	12.01074
11	C11	c3	-0.24019	12.01074
12	S12	s6	0.976247	32.06479
13	O13	o	-0.6373	15.99941
14	O14	o	-0.67735	15.99941
15	O15	o	-0.6078	15.99941
16	C16	ca	0.300484	12.01074
17	C17	ca	-0.20604	12.01074
18	C18	ca	-0.11628	12.01074
19	C19	ca	-0.18898	12.01074
20	C20	ca	-0.11125	12.01074
21	C21	ca	-0.25108	12.01074
22	H22	h1	0.101548	1.007941
23	H23	h1	0.101548	1.007941
24	H24	h1	0.101548	1.007941
25	H25	hc	0.123915	1.007941
26	H26	hc	0.123915	1.007941
27	H27	hc	0.123915	1.007941
28	H28	h1	0.139866	1.007941
29	H29	h1	0.139866	1.007941
30	H30	h1	0.139866	1.007941
31	H31	h2	0.139533	1.007941
32	H32	h2	0.139533	1.007941
33	H33	ha	0.13102	1.007941
34	H34	ha	0.133382	1.007941
35	H35	ha	0.138541	1.007941
36	H36	ha	0.132591	1.007941
37	H37	ha	0.159745	1.007941

Table S2. Bond information.

Bond type	Bond length (nm)	$k$ (kJ/mol/nm <sup>2</sup> )
N1-N2	0.14071	3.72E+05
N1-C5	0.13789	3.58E+05
N1-C16	0.14121	3.21E+05
N2-C3	0.13802	3.56E+05
N2-C6	0.14629	2.74E+05
C3-C4	0.14278	3.51E+05
C3-C7	0.15015	2.80E+05
C4-C5	0.14676	3.10E+05
C4-N9	0.13735	3.64E+05
C5-O8	0.12183	5.34E+05
C6-H22	0.10969	2.77E+05
C6-H23	0.10969	2.77E+05
C6-H24	0.10969	2.77E+05
C7-H25	0.10969	2.77E+05
C7-H26	0.10969	2.77E+05
C7-H27	0.10969	2.77E+05
N9-C10	0.1464	2.73E+05
N9-C11	0.1464	2.73E+05
C10-H28	0.10969	2.77E+05
C10-H29	0.10969	2.77E+05
C10-H30	0.10969	2.77E+05
C11-S12	0.18075	1.95E+05
C11-H31	0.10961	2.78E+05
C11-H32	0.10961	2.78E+05
S12-O13	0.14533	4.29E+05
S12-O14	0.14533	4.29E+05
S12-O15	0.14533	4.29E+05
C16-C17	0.13984	3.86E+05
C16-C21	0.13984	3.86E+05
C17-C18	0.13984	3.86E+05
C17-H33	0.1086	2.89E+05
C18-C19	0.13984	3.86E+05
C18-H34	0.1086	2.89E+05
C19-C20	0.13984	3.86E+05
C19-H35	0.1086	2.89E+05
C20-C21	0.13984	3.86E+05
C20-H36	0.1086	2.89E+05
C21-H37	0.1086	2.89E+05

Table S3. Angle information.

Angle type	Angle (degree)	$k$ (kJ/mol/rad <sup>2</sup> )
N2-N1-C5	111.5	5.77E+02
N2-N1-C16	119.3	5.51E+02
C5-N1-C16	123.71	5.34E+02
N1-N2-C3	110.05	5.81E+02
N1-N2-C6	112.88	5.56E+02
C3-N2-C6	126.46	5.18E+02
N2-C3-C4	117.77	5.74E+02
N2-C3-C7	122.73	5.46E+02
C4-C3-C7	115.97	5.41E+02
C3-C4-C5	122.69	5.32E+02
C3-C4-N9	119.72	5.71E+02
C5-C4-N9	125.703	5.00E+02
N1-C5-C4	112.7	5.78E+02
N1-C5-O8	123.05	6.21E+02
C4-C5-O8	123.93	5.78E+02
N2-C6-H22	108.78	4.17E+02
N2-C6-H23	108.78	4.17E+02
N2-C6-H24	108.78	4.17E+02
H22-C6-H23	108.46	3.28E+02
H22-C6-H24	108.46	3.28E+02
H23-C6-H24	108.46	3.28E+02
C3-C7-H25	110.49	3.95E+02
C3-C7-H26	110.49	3.95E+02
C3-C7-H27	110.49	3.95E+02
H25-C7-H26	107.58	3.30E+02
H25-C7-H27	107.58	3.30E+02
H26-C7-H27	107.58	3.30E+02
C4-N9-C10	119.72	5.33E+02
C4-N9-C11	119.72	5.33E+02
C10-N9-C11	114.51	5.29E+02
N9-C10-H28	109.79	4.15E+02
N9-C10-H29	109.79	4.15E+02
N9-C10-H30	109.79	4.15E+02
H28-C10-H29	108.46	3.28E+02
H28-C10-H30	108.46	3.28E+02
H29-C10-H30	108.46	3.28E+02
N9-C11-S12	115.571	5.00E+02
N9-C11-H31	110.01	4.14E+02
N9-C11-H32	110.01	4.14E+02

S12-C11-H31	106.51	3.62E+02
S12-C11-H32	106.51	3.62E+02
H31-C11-H32	110.2	3.26E+02
C11-S12-O13	108.61	5.47E+02
C11-S12-O14	108.61	5.47E+02
C11-S12-O15	108.61	5.47E+02
O13-S12-O14	120.05	6.16E+02
O13-S12-O15	120.05	6.16E+02
O14-S12-O15	120.05	6.16E+02
N1-C16-C17	120.19	5.68E+02
N1-C16-C21	120.19	5.68E+02
C17-C16-C21	120.02	5.57E+02
C16-C17-C18	120.02	5.57E+02
C16-C17-H33	119.88	4.03E+02
C18-C17-H33	119.88	4.03E+02
C17-C18-C19	120.02	5.57E+02
C17-C18-H34	119.88	4.03E+02
C19-C18-H34	119.88	4.03E+02
C18-C19-C20	120.02	5.57E+02
C18-C19-H35	119.88	4.03E+02
C20-C19-H35	119.88	4.03E+02
C19-C20-C21	120.02	5.57E+02
C19-C20-H36	119.88	4.03E+02
C21-C20-H36	119.88	4.03E+02
C16-C21-C20	120.02	5.57E+02
C16-C21-H37	119.88	4.03E+02
C20-C21-H37	119.88	4.03E+02
N2-N1-C5	111.5	5.77E+02
N2-N1-C16	119.3	5.51E+02
C5-N1-C16	123.71	5.34E+02

23

24

Table S4. Dihedrals (propers).

Dihedral type	Phase (degree)	$k_d$ (kJ/mol)	$p_n$
N1-N2-C3-C4	180	7.1128	2
N1-N2-C3-C7	180	7.1128	2
N1-N2-C6-H22	0	0	2
N1-N2-C6-H23	0	0	2
N1-N2-C6-H24	0	0	2
N1-C5-C4-C3	180	12.029	2
N1-C5-C4-N9	180	12.029	2

---

N1-C16-C17-C18	180	15.167	2
N1-C16-C17-H33	180	15.167	2
N1-C16-C21-C20	180	15.167	2
N1-C16-C21-H37	180	15.167	2
N2-N1-C5-C4	180	10.46	2
N2-N1-C5-O8	180	10.46	2
N2-N1-C16-C17	180	1.8828	2
N2-N1-C16-C21	180	1.8828	2
N2-C3-C4-C5	180	16.736	2
N2-C3-C4-N9	180	16.736	2
N2-C3-C7-H25	0	0	3
N2-C3-C7-H26	0	0	3
N2-C3-C7-H27	0	0	3
C3-N2-N1-C5	0	2.9288	2
C3-N2-N1-C16	0	2.9288	2
C3-N2-C6-H22	0	0	2
C3-N2-C6-H23	0	0	2
C3-N2-C6-H24	0	0	2
C3-C4-C5-O8	180	12.029	2
C3-C4-N9-C10	180	4.3932	2
C3-C4-N9-C11	180	4.3932	2
C4-C3-N2-C6	180	7.1128	2
C4-C3-C7-H25	0	0	3
C4-C3-C7-H26	0	0	3
C4-C3-C7-H27	0	0	3
C4-C5-N1-C16	180	10.46	2
C4-N9-C10-H28	0	0	2
C4-N9-C10-H29	0	0	2
C4-N9-C10-H30	0	0	2
C4-N9-C11-S12	0	0	2
C4-N9-C11-H31	0	0	2
C4-N9-C11-H32	0	0	2
C5-N1-N2-C6	0	2.9288	2
C5-N1-C16-C17	180	1.8828	2
C5-N1-C16-C21	180	1.8828	2
C5-C4-C3-C7	180	16.736	2
C5-C4-N9-C10	180	4.3932	2
C5-C4-N9-C11	180	4.3932	2
C6-N2-N1-C16	0	2.9288	2
C6-N2-C3-C7	180	7.1128	2
C7-C3-C4-N9	180	16.736	2

---

O8-C5-N1-C16	180	10.46	2
O8-C5-C4-N9	180	12.029	2
N9-C11-S12-O13	0	0.60436	3
N9-C11-S12-O14	0	0.60436	3
N9-C11-S12-O15	0	0.60436	3
C10-N9-C11-S12	0	0	2
C10-N9-C11-H31	0	0	2
C10-N9-C11-H32	0	0	2
C11-N9-C10-H28	0	0	2
C11-N9-C10-H29	0	0	2
C11-N9-C10-H30	0	0	2
O13-S12-C11-H31	0	0.60436	3
O13-S12-C11-H32	0	0.60436	3
O14-S12-C11-H31	0	0.60436	3
O14-S12-C11-H32	0	0.60436	3
O15-S12-C11-H31	0	0.60436	3
O15-S12-C11-H32	0	0.60436	3
C16-C17-C18-C19	180	15.167	2
C16-C17-C18-H34	180	15.167	2
C16-C21-C20-C19	180	15.167	2
C16-C21-C20-H36	180	15.167	2
C17-C16-C21-C20	180	15.167	2
C17-C16-C21-H37	180	15.167	2
C17-C18-C19-C20	180	15.167	2
C17-C18-C19-H35	180	15.167	2
C18-C17-C16-C21	180	15.167	2
C18-C19-C20-C21	180	15.167	2
C18-C19-C20-H36	180	15.167	2
C19-C18-C17-H33	180	15.167	2
C19-C20-C21-H37	180	15.167	2
C20-C19-C18-H34	180	15.167	2
C21-C16-C17-H33	180	15.167	2
C21-C20-C19-H35	180	15.167	2
H33-C17-C18-H34	180	15.167	2
H34-C18-C19-H35	180	15.167	2
H35-C19-C20-H36	180	15.167	2
H36-C20-C21-H37	180	15.167	2

25

26

Table S5. Dihedrals (impropers).

Dihedral type	Phase (degree)	$k_d$ (kJ/mol)	$p_n$
---------------	----------------	----------------	-------

N2-C5-N1-C16	180	4.6024	2
N1-C3-N2-C6	180	4.6024	2
N1-C4-C5-O8	180	43.932	2
N1-C17-C16-C21	180	4.6024	2
C16-C18-C17-H33	180	4.6024	2
C17-C19-C18-H34	180	4.6024	2
C18-C20-C19-H35	180	4.6024	2
C19-C21-C20-H36	180	4.6024	2
C16-C20-C21-H37	180	4.6024	2

27

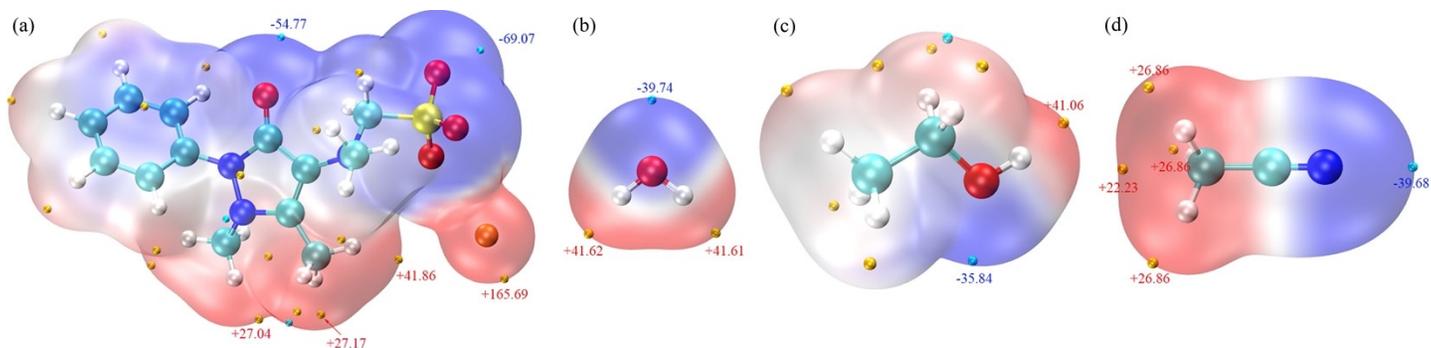


Figure S1. Molecular surface analysis over van der Waals surface colored by electrostatic potential (ESP) where cyan and yellow spheres in the maps highlight the surface ESP minima and maxima. The four molecules mentioned in the paper: (a) dipyrone; (b) water; (c) ethanol; (d) acetonitrile.

28

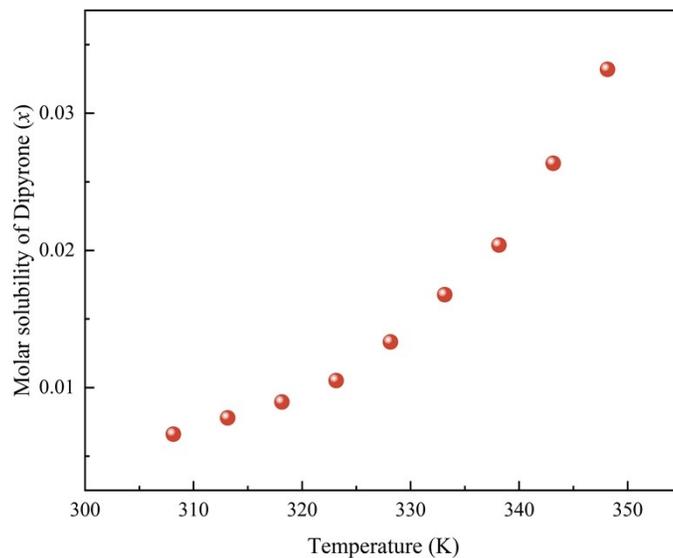


Figure S2. Mole fraction solubility of dipyrone in 95% ethanol at different temperature

29

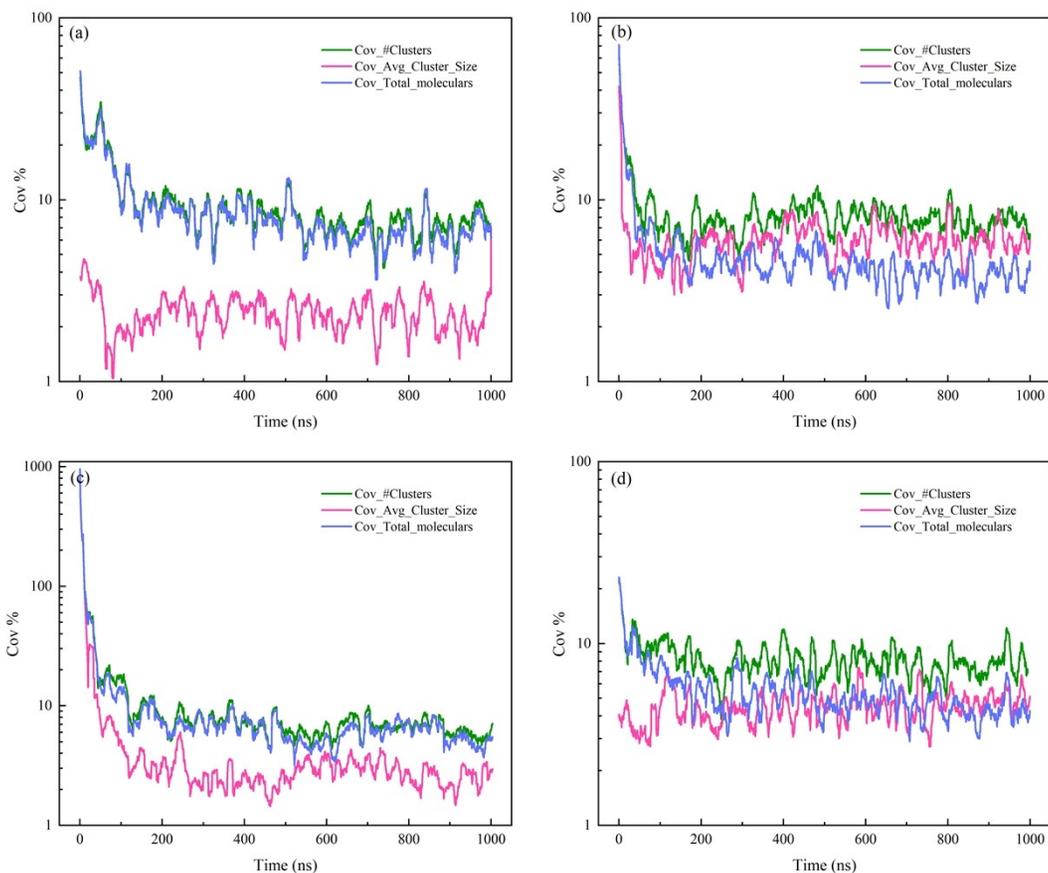


Fig. S3. Time-dependent curves of the coefficient of variation for the number, average size, and total number of dipyrone clusters formed in different solutions: (a) EtOH; (b) 95% EtOH; (c) ACN; (d) 95% ACN.

30

31

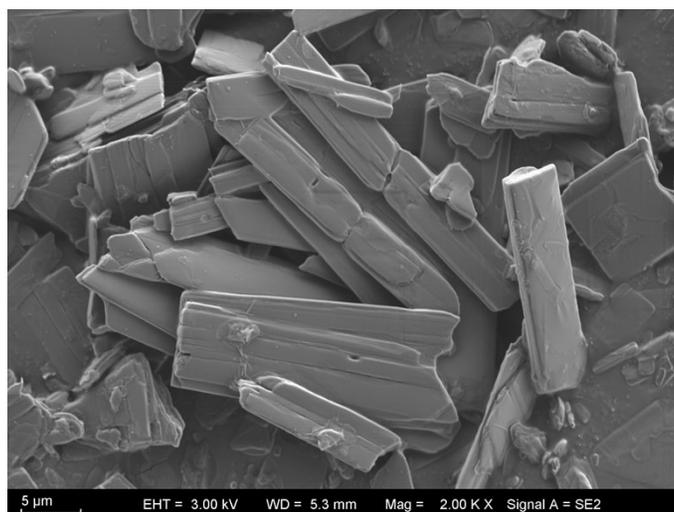


Figure S4. SEM image of the dipyrone crystal in 95% EtOH.

32

33 **Tcl code for molecular classification in Fig.3a:**

```

34
35 set range1_start 1
36 set range1_end 480
37 set range2_start 1441
38 set range2_end 1920
39 set start_frame 0
40 set end_frame 405
41 set displacement_threshold 4.5
42 set angle_change_threshold 50.0
43 set neighbor_distance 13.5
44 set z_min 42.0
45 set z_max 145.0
46
47 # set num_reps [molinfo top get numreps]
48 # for {set i 0} {$i < $num_reps} {incr i} {
49 #     mol delrep 0 top
50 # }
51
52 set outfile [open "frame_stats_prod3-50degree-13dot5.csv" w]
53 puts $outfile "frame,crystal_count,A_molecule_count,solu_count"
54
55 for {set frame $start_frame} {$frame < $end_frame} {incr frame} {
56     set current_frame $frame
57     set next_frame [expr {$frame + 1}]
58
59     puts "Analyzing frames $current_frame to $next_frame..."
60
61     array unset res_coords_frame0
62     array unset res_coords_frame1
63     array unset res_angles
64     array unset res_displacement
65     array unset res_class
66
67     foreach {type start end} [list range1 $range1_start $range1_end range2 $range2_start $range2_end]
68 {
69         for {set ires $start} {$ires <= $end} {incr ires} {
70             set sel_N1 [atomselect top "resid $ires and name N1" frame $current_frame]
71             set sel_N9 [atomselect top "resid $ires and name N9" frame $current_frame]
72
73             if {[$sel_N1 num] != 1 || [$sel_N9 num] != 1} {
74                 $sel_N1 delete

```

```

75         $sel_N9 delete
76         continue
77     }
78
79     set coords_N1 [$sel_N1 get {x y z}]
80     set coords_N9 [$sel_N9 get {x y z}]
81
82     if {[llength $coords_N1] != 1 || [llength $coords_N9] != 1} {
83         $sel_N1 delete
84         $sel_N9 delete
85         continue
86     }
87
88     set x1 [lindex $coords_N1 0 0]
89     set y1 [lindex $coords_N1 0 1]
90     set z1 [lindex $coords_N1 0 2]
91     set x9 [lindex $coords_N9 0 0]
92     set y9 [lindex $coords_N9 0 1]
93     set z9 [lindex $coords_N9 0 2]
94
95     if {[string is double $x1] && [string is double $y1] && [string is double $z1] &&
96         [string is double $x9] && [string is double $y9] && [string is double $z9]} {
97
98         set res_coords_frame0($ires) [list $x1 $y1 $z1 $x9 $y9 $z9]
99     }
100
101     $sel_N1 delete
102     $sel_N9 delete
103 }
104 }
105
106 foreach {type start end} [list range1 $range1_start $range1_end range2 $range2_start $range2_end]
107 {
108     for {set ires $start} {$ires <= $end} {incr ires} {
109         set sel_N1 [atomselect top "resid $ires and name N1" frame $next_frame]
110         set sel_N9 [atomselect top "resid $ires and name N9" frame $next_frame]
111
112         if {[$sel_N1 num] != 1 || [$sel_N9 num] != 1} {
113             $sel_N1 delete
114             $sel_N9 delete
115             continue

```

```

116         }
117
118         set coords_N1 [$sel_N1 get {x y z}]
119         set coords_N9 [$sel_N9 get {x y z}]
120
121         if {[llength $coords_N1] != 1 || [llength $coords_N9] != 1} {
122             $sel_N1 delete
123             $sel_N9 delete
124             continue
125         }
126
127         set x1 [lindex $coords_N1 0 0]
128         set y1 [lindex $coords_N1 0 1]
129         set z1 [lindex $coords_N1 0 2]
130         set x9 [lindex $coords_N9 0 0]
131         set y9 [lindex $coords_N9 0 1]
132         set z9 [lindex $coords_N9 0 2]
133
134         if {[string is double $x1] && [string is double $y1] && [string is double $z1] &&
135             [string is double $x9] && [string is double $y9] && [string is double $z9]} {
136
137             set res_coords_frame1($ires) [list $x1 $y1 $z1 $x9 $y9 $z9]
138         }
139
140         $sel_N1 delete
141         $sel_N9 delete
142     }
143 }
144
145 set molid [molinfo top]
146
147 set cell [pbc get -molid $molid -first $current_frame -last $current_frame]
148
149 if {[llength $cell] >= 6} {
150     set a [lindex $cell 0]
151     set b [lindex $cell 1]
152     set c [lindex $cell 2]
153     set alpha [lindex $cell 3]
154     set beta [lindex $cell 4]
155     set gamma [lindex $cell 5]
156 } else {

```

```

157         set a 50.0
158         set b 100.0
159         set c 180.0
160         set alpha 90.0
161         set beta 90.0
162         set gamma 90.0
163     }
164
165     set tolerance 0.1
166     if {abs($alpha-90.0)>$tolerance || abs($beta-90.0)>$tolerance || abs($gamma-90.0)>$tolerance} {
167         puts "Warning: Non-orthogonal box detected (angles: $alpha, $beta, $gamma). The PBC
168 correction may not be accurate for non-orthogonal boxes. This script currently only supports orthogonal
169 boxes."
170     }
171
172     proc pbc_correct_coords {x0 y0 z0 x1 y1 z1 a b c} {
173         set dx [expr {$x1 - $x0}]
174         set dy [expr {$y1 - $y0}]
175         set dz [expr {$z1 - $z0}]
176
177         if {abs($dx) > $a/2.0} {
178             if {$dx > 0} {
179                 set x1 [expr {$x1 - $a}]
180             } else {
181                 set x1 [expr {$x1 + $a}]
182             }
183         }
184
185         if {abs($dy) > $b/2.0} {
186             if {$dy > 0} {
187                 set y1 [expr {$y1 - $b}]
188             } else {
189                 set y1 [expr {$y1 + $b}]
190             }
191         }
192
193         if {abs($dz) > $c/2.0} {
194             if {$dz > 0} {
195                 set z1 [expr {$z1 - $c}]
196             } else {
197                 set z1 [expr {$z1 + $c}]

```

```

198         }
199     }
200
201     return [list $x1 $y1 $z1]
202 }
203
204 proc pbc_distance {coord1 coord2 a b c} {
205     lassign $coord1 x1 y1 z1
206     lassign $coord2 x2 y2 z2
207
208     set dx [expr {$x2 - $x1}]
209     set dy [expr {$y2 - $y1}]
210     set dz [expr {$z2 - $z1}]
211
212     if {abs($dx) > $a/2.0} {
213         if {$dx > 0} {
214             set dx [expr {$dx - $a}]
215         } else {
216             set dx [expr {$dx + $a}]
217         }
218     }
219
220     if {abs($dy) > $b/2.0} {
221         if {$dy > 0} {
222             set dy [expr {$dy - $b}]
223         } else {
224             set dy [expr {$dy + $b}]
225         }
226     }
227
228     if {abs($dz) > $c/2.0} {
229         if {$dz > 0} {
230             set dz [expr {$dz - $c}]
231         } else {
232             set dz [expr {$dz + $c}]
233         }
234     }
235
236     return [expr {sqrt($dx*$dx + $dy*$dy + $dz*$dz)}]
237 }
238

```

```

239 proc angle_with_y_axis {dx dy dz} {
240     set magnitude [expr {sqrt($dx*$dx + $dy*$dy + $dz*$dz)}]
241     if {$magnitude < 0.0001} {
242         return 0.0
243     }
244
245     set dot_product [expr {$dy / $magnitude}]
246
247     if {$dot_product > 1.0} {set dot_product 1.0}
248     if {$dot_product < -1.0} {set dot_product -1.0}
249
250     set angle [expr {acos($dot_product) * 180.0 / 3.14159265359}]
251     return $angle
252 }
253
254 proc vector_angle_between {v1 v2} {
255     lassign $v1 dx1 dy1 dz1
256     lassign $v2 dx2 dy2 dz2
257
258     set dot_product [expr {$dx1*$dx2 + $dy1*$dy2 + $dz1*$dz2}]
259     set mag1 [expr {sqrt($dx1*$dx1 + $dy1*$dy1 + $dz1*$dz1)}]
260     set mag2 [expr {sqrt($dx2*$dx2 + $dy2*$dy2 + $dz2*$dz2)}]
261
262     if {$mag1 < 0.0001 || $mag2 < 0.0001} {
263         return 0.0
264     }
265
266     set cos_theta [expr {$dot_product / ($mag1 * $mag2)}]
267     if {$cos_theta > 1.0} {set cos_theta 1.0}
268     if {$cos_theta < -1.0} {set cos_theta -1.0}
269
270     set angle [expr {acos($cos_theta) * 180.0 / 3.14159265359}]
271     return $angle
272 }
273
274 set pre_crystal_resids [list]
275 set disorder_resids [list]
276
277 foreach res [array names res_coords_frame0] {
278     if {[!info exists res_coords_frame1($res)]} {
279         continue

```

```

280     }
281
282     lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
283
284     #
285     if {($res >= $range1_start && $res <= $range1_end) || ($res >= $range2_start && $res <=
286 $range2_end)} {
287         #
288         lassign $res_coords_frame1($res) x1_1 y1_1 z1_1 x9_1 y9_1 z9_1
289         lassign [pbc_correct_coords $x9_0 $y9_0 $z9_0 $x9_1 $y9_1 $z9_1 $a $b $c] x9_1_corr
290 y9_1_corr z9_1_corr
291         set dx [expr {$x9_1_corr - $x9_0}]
292         set dy [expr {$y9_1_corr - $y9_0}]
293         set dz [expr {$z9_1_corr - $z9_0}]
294         set displacement [expr {sqrt($dx*$dx + $dy*$dy + $dz*$dz)}]
295         set res_displacement($res) $displacement
296
297         #
298         set dx0 [expr {$x9_0 - $x1_0}]
299         set dy0 [expr {$y9_0 - $y1_0}]
300         set dz0 [expr {$z9_0 - $z1_0}]
301         set vector0 [list $dx0 $dy0 $dz0]
302
303         #
304         set dx1 [expr {$x9_1 - $x1_1}]
305         set dy1 [expr {$y9_1 - $y1_1}]
306         set dz1 [expr {$z9_1 - $z1_1}]
307         set vector1 [list $dx1 $dy1 $dz1]
308
309         #
310         set angle0 [angle_with_y_axis $dx0 $dy0 $dz0]
311
312         #
313         set rotation_angle [vector_angle_between $vector0 $vector1]
314         set res_angles($res) [list $angle0 $rotation_angle]
315
316         if {$displacement <= $displacement_threshold &&
317             $rotation_angle <= $angle_change_threshold &&
318             (($angle0 >= 0.0 && $angle0 <= 50.0) || ($angle0 >= 130.0 && $angle0 <= 180.0))}
319     {
320         lappend pre_crystal_resids $res

```

```

321             set res_class($res) "pre_crystal"
322         } else {
323             lappend disorder_resids $res
324             set res_class($res) "disorder"
325         }
326     }
327 }
328
329 set crystal_resids [list]
330 set solu_resids [list]
331
332 foreach res $pre_crystal_resids {
333     lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
334     set coord1 [list $x1_0 $y1_0 $z1_0]
335
336     set neighbor_count 0
337
338     foreach other_res $pre_crystal_resids {
339         if {$res == $other_res} continue
340
341         lassign $res_coords_frame0($other_res) x1_other y1_other z1_other x9_other y9_other
342 z9_other
343         set coord2 [list $x1_other $y1_other $z1_other]
344
345         set dist [pbc_distance $coord1 $coord2 $a $b $c]
346
347         if {$dist <= $neighbor_distance} {
348             incr neighbor_count
349         }
350     }
351
352     if {$neighbor_count >= 2} {
353         lappend crystal_resids $res
354         set res_class($res) "crystal"
355     } else {
356         lappend solu_resids $res
357         set res_class($res) "solu"
358     }
359 }
360
361 set changed 1

```

```

362     set iteration 0
363     while {$changed && $iteration < 5} {
364         set changed 0
365         set new_crystal_resids [list]
366         set new_solu_resids [list]
367
368         array set current_crystal_set {}
369         foreach res $crystal_resids {
370             set current_crystal_set($res) 1
371         }
372
373         foreach res $crystal_resids {
374             lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
375             set coord1 [list $x1_0 $y1_0 $z1_0]
376             set neighbor_count 0
377             foreach other_res $crystal_resids {
378                 if {$res == $other_res} continue
379                 if {[info exists current_crystal_set($other_res)]} continue
380                 lassign $res_coords_frame0($other_res) x1_other y1_other z1_other x9_other
381                 y9_other z9_other
382                 set coord2 [list $x1_other $y1_other $z1_other]
383                 set dist [pbc_distance $coord1 $coord2 $a $b $c]
384
385                 if {$dist <= $neighbor_distance} {
386                     incr neighbor_count
387                 }
388             }
389             if {$neighbor_count >= 1} {
390                 lappend new_crystal_resids $res
391             } else {
392                 lappend new_solu_resids $res
393                 set res_class($res) "solu"
394                 set changed 1
395             }
396         }
397         set crystal_resids $new_crystal_resids
398         set solu_resids [concat $solu_resids $new_solu_resids]
399
400         incr iteration
401     }
402

```

```

403     set A_molecule_resids [list]
404     foreach res $disorder_resids {
405         if {[info exists res_coords_frame0($res)]} {
406             lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
407             if {$z1_0 < $z_min || $z1_0 > $z_max} {
408                 lappend crystal_resids $res
409                 set res_class($res) "crystal"
410             }
411         }
412     }
413     set new_disorder_resids [list]
414     foreach res $disorder_resids {
415         if {[info exists res_coords_frame0($res)]} {
416             lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
417             if {$z1_0 >= $z_min && $z1_0 <= $z_max} {
418                 lappend new_disorder_resids $res
419             }
420         }
421     }
422     set disorder_resids $new_disorder_resids
423     foreach res $disorder_resids {
424         lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
425         set coord1 [list $x1_0 $y1_0 $z1_0]
426         set has_crystal_neighbor 0
427         foreach crystal_res $crystal_resids {
428             lassign $res_coords_frame0($crystal_res) x1_crystal y1_crystal z1_crystal x9_crystal
429             y9_crystal z9_crystal
430             set coord2 [list $x1_crystal $y1_crystal $z1_crystal]
431             set dist [pbc_distance $coord1 $coord2 $a $b $c]
432             if {$dist <= $neighbor_distance} {
433                 set has_crystal_neighbor 1
434                 break
435             }
436         }
437         if {$has_crystal_neighbor} {
438             lappend A_molecule_resids $res
439             set res_class($res) "A_molecule"
440         } else {
441             lappend solu_resids $res
442             set res_class($res) "solu"
443         }

```

```

444     }
445
446     foreach res [array names res_coords_frame0] {
447         if {[info exists res_class($res)]} {
448             lassign $res_coords_frame0($res) x1_0 y1_0 z1_0 x9_0 y9_0 z9_0
449
450             if {$z1_0 < $z_min || $z1_0 > $z_max} {
451                 if {$res_class($res) ne "crystal"} {
452                     set idx [lsearch -exact $solu_resids $res]
453                     if {$idx != -1} {
454                         set solu_resids [lreplace $solu_resids $idx $idx]
455                     }
456
457                     set idx [lsearch -exact $A_molecule_resids $res]
458                     if {$idx != -1} {
459                         set A_molecule_resids [lreplace $A_molecule_resids $idx $idx]
460                     }
461
462                     set idx [lsearch -exact $disorder_resids $res]
463                     if {$idx != -1} {
464                         set disorder_resids [lreplace $disorder_resids $idx $idx]
465                     }
466
467                     lappend crystal_resids $res
468                     set res_class($res) "crystal"
469                 }
470             }
471         }
472     }
473
474     #
475     set crystal_count [llength $crystal_resids]
476     set A_molecule_count [llength $A_molecule_resids]
477     set solu_count [llength $solu_resids]
478
479     #
480     puts $outfile "$next_frame,$crystal_count,$A_molecule_count,$solu_count"
481
482     puts "Frame $next_frame: Crystal=$crystal_count, A_molecule=$A_molecule_count,
483 Solu=$solu_count"
484 }

```

```

485
486 close $outfile
487 puts "Multi-frame analysis completed. Results saved to frame_stats_0-3000-50degree-13dot5.csv"
488
489 Tcl code for cluster statistics in Fig.3b:
490 draw delete all
491
492 set range1_start 0
493 set range1_end 0
494 set range2_start 5723
495 set range2_end 5828
496 set crit 7
497 set output_file "connects.txt"
498 set min_cluster_size 3
499
500 set num_frames [molinfo top get numframes]
501
502 set outfile [open $output_file w]
503 puts $outfile "Frame\tYellow_Connects\tRed_Connects\tCluster_Count\tAvg_Cluster_Size"
504
505 for {set frame 0} {$frame < $num_frames} {incr frame} {
506     animate goto $frame
507     display update
508
509     array unset yellow_connected
510     array unset red_connected
511     array unset cluster_assigned
512     array unset cluster_size
513
514     foreach {start end} [list $range1_start $range1_end $range3_start $range3_end] {
515         for {set ires $start} {$ires <= $end} {incr ires} {
516             set sel [atomselect top "resid $ires and name S12" frame $frame]
517             if {[[$sel num] > 0]} {
518                 set coord [lindex [$sel get {x y z}] 0]
519                 set x($ires) [lindex $coord 0]
520                 set y($ires) [lindex $coord 1]
521                 set z($ires) [lindex $coord 2]
522             }
523             $sel delete
524         }
525     }

```

```

526
527 for {set ires $range2_start} {$ires <= $range2_end} {incr ires} {
528     set sel [atomselect top "resid $ires and name S12" frame $frame]
529     if {[ $sel num] > 0} {
530         set coord [lindex [ $sel get {x y z}] 0]
531         set x($ires) [lindex $coord 0]
532         set y($ires) [lindex $coord 1]
533         set z($ires) [lindex $coord 2]
534     }
535     $sel delete
536 }
537
538 foreach group [list [list $range1_start $range1_end] [list $range3_start $range3_end]] {
539     lassign $group group_start group_end
540     for {set ires $group_start} {$ires <= $group_end} {incr ires} {
541         if {[info exists x($ires)]} { continue }
542
543         for {set jres $range2_start} {$jres <= $range2_end} {incr jres} {
544             if {[info exists x($jres)]} { continue }
545
546             set dist [expr {
547                 sqrt(($x($ires)-$x($jres))**2 +
548                     ($y($ires)-$y($jres))**2 +
549                     ($z($ires)-$z($jres))**2)
550             }]
551             if {$dist < $crit} {
552                 set yellow_connected($jres) 1
553             }
554         }
555     }
556 }
557
558 array unset adjacency_list
559 for {set ires $range2_start} {$ires <= $range2_end} {incr ires} {
560     if {[info exists x($ires)]} { continue }
561
562     for {set jres [expr {$ires + 1}]} {$jres <= $range2_end} {incr jres} {
563         if {[info exists x($jres)]} { continue }
564
565         set dist [expr {
566             sqrt(($x($ires)-$x($jres))**2 +

```

```

567             ($y($ires)-$y($jres))**2 +
568             ($z($ires)-$z($jres))**2)
569     }]
570     if {$dist < $crit} {
571         set red_connected($ires) 1
572         set red_connected($jres) 1
573
574         lappend adjacency_list($ires) $jres
575         lappend adjacency_list($jres) $ires
576     }
577 }
578 }
579
580 set cluster_id 0
581 set total_cluster_size 0
582 set valid_cluster_count 0
583
584 for {set ires $range2_start} {$ires <= $range2_end} {incr ires} {
585     if {[info exists x($ires)]} { continue }
586     if {[info exists cluster_assigned($ires)]} { continue }
587
588     set queue [list $ires]
589     set current_cluster [list]
590     set cluster_assigned($ires) 1
591
592     while {[length $queue] > 0} {
593         set current [lindex $queue 0]
594         set queue [lrange $queue 1 end]
595         lappend current_cluster $current
596
597         if {[info exists adjacency_list($current)]} {
598             foreach neighbor $adjacency_list($current) {
599                 if {[info exists cluster_assigned($neighbor)]} {
600                     set cluster_assigned($neighbor) 1
601                     lappend queue $neighbor
602                 }
603             }
604         }
605     }
606
607     set size [length $current_cluster]

```

```

608         set cluster_size($cluster_id) $size
609
610         if {$size >= $min_cluster_size} {
611             incr valid_cluster_count
612             incr total_cluster_size $size
613         }
614
615         incr cluster_id
616     }
617
618     if {$valid_cluster_count > 0} {
619         set avg_cluster_size [expr {double($total_cluster_size) / $valid_cluster_count}]
620     } else {
621         set avg_cluster_size 0.0
622     }
623
624     set yellow_count [array size yellow_connected]
625     set red_count [array size red_connected]
626
627     puts $outfile [format "%d\t%d\t%d\t%d\t%.2f" $frame $yellow_count $red_count
628 $valid_cluster_count $avg_cluster_size]
629     puts "Frame $frame: Yellow=$yellow_count, Red=$red_count, Clusters=$valid_cluster_count,
630 AvgSize=[format "%.2f" $avg_cluster_size]"
631 }
632
633 close $outfile
634 puts "Multi-frame statistics saved to $output_file"
635
636 animate goto [expr $num_frames - 1]
637 draw delete all
638
639 draw color blue
640 foreach {start end} [list $range1_start $range1_end $range3_start $range3_end] {
641     for {set ires $start} {$ires <= $end} {incr ires} {
642         if {[info exists x($ires)]} {
643             draw sphere "$x($ires) $y($ires) $z($ires)" radius 1.5
644         }
645     }
646 }
647
648 draw color orange

```

```

649 for {set ires $range2_start} {$ires <= $range2_end} {incr ires} {
650     if {[info exists x($ires)]} {
651         draw sphere "$x($ires) $y($ires) $z($ires)" radius 1.5
652     }
653 }
654
655 draw color yellow
656 foreach group [list [list $range1_start $range1_end] [list $range3_start $range3_end]] {
657     lassign $group group_start group_end
658     for {set ires $group_start} {$ires <= $group_end} {incr ires} {
659         if {[info exists x($ires)]} { continue }
660
661         for {set jres $range2_start} {$jres <= $range2_end} {incr jres} {
662             if {[info exists x($jres)]} { continue }
663
664             set dist [expr {
665                 sqrt(($x($ires)-$x($jres))**2 +
666                     ($y($ires)-$y($jres))**2 +
667                     ($z($ires)-$z($jres))**2)
668             }]
669             if {$dist < $crit} {
670                 draw cylinder "$x($ires) $y($ires) $z($ires)" \
671                     "$x($jres) $y($jres) $z($jres)" \
672                     radius 0.5
673             }
674         }
675     }
676 }
677
678 draw color red
679 for {set ires $range2_start} {$ires <= $range2_end} {incr ires} {
680     if {[info exists x($ires)]} { continue }
681
682     for {set jres [expr {$ires + 1}]} {$jres <= $range2_end} {incr jres} {
683         if {[info exists x($jres)]} { continue }
684
685         set dist [expr {
686             sqrt(($x($ires)-$x($jres))**2 +
687                 ($y($ires)-$y($jres))**2 +
688                 ($z($ires)-$z($jres))**2)
689         }]

```

```
690     if {$dist < $crit} {
691         draw cylinder "$x($ires) $y($ires) $z($ires)" \
692             "$x($jres) $y($jres) $z($jres)" \
693             radius 0.5
694     }
695 }
696 }
```