```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Threading;
 5
 6  using XDMessaging;
 7
 8  namespace ClusterSim.Standalone
 9  {
10      using System.CodeDom.Compiler;
11      using System.Collections.Concurrent;
12      using System.Diagnostics;
13      using System.Runtime.CompilerServices;
14      using System.Threading.Tasks;
15      using System.Windows.Forms.VisualStyles;
16
17      using ClusterSim.ClusterLib.Analysis;
18      using ClusterSim.ClusterLib.Calculation;
19      using ClusterSim.ClusterLib.Calculation.Cluster;
20      using ClusterSim.ClusterLib.Utility;
21
22      public class Program
23      {
24          private static bool abort = false;
25
26          public static int SaveInterval { get; set; } = 100;
27
28          public static double MinDAcc { get; set; } = 0.001;
29
30          public static void Main(string[] args)
31          {
32              XDMessagingClient client = new XDMessagingClient(); // https://
                  github.com/TheCodeKing/XDMessaging.Net
33              IXDBroadcaster broadcaster =
                  client.Broadcasters.GetBroadcasterForMode
                  (XDTransportMode.HighPerformanceUI);
34              string rTable = string.Empty;
35              Console.ForegroundColor = ConsoleColor.DarkRed;
36
37              if (args.Length > 0) // if the program gets called with arguments
38              {
39                  List<String> list = SQL.readTables();
40                  if (list != null)
41                  {
42                      List<string> res = new List<string>();
43                      foreach (string s in args)
44                      {
45                          if (list.Contains(s))//check if argument is a valid
                          table name
46                              rTable = s;
47
48                          Console.WriteLine(rTable);
49                      }
50                  }
51              }
52
```

```
53              if (rTable == "")//if argument handover failed or was not  given
54              {
55                  Console.WriteLine("Auswahltabelle: "); // input name manually
56                  rTable = Console.ReadLine();
57              }
58
59
60              int last = 0;
61              Console.WriteLine("\nLeer lassen, für gleiche Liste, oder
                  Speichern nach: ");
62              string wTable = Console.ReadLine();
63
64
65              if (wTable.Equals(string.Empty))
66              {
67                  wTable = rTable;
68                  last = SQL.lastStep(rTable);//get last step of given table
69              }
70
71
72              Console.WriteLine("\nDelta t in Tagen: ");
73
74              double dt = Convert.ToDouble(Console.ReadLine());
75
76              //Console.WriteLine("\nSchritte: ");
77
78              int n = 2;//Convert.ToInt32(Console.ReadLine());
79
80              int year = last * SaveInterval;
81              Console.WriteLine(@"Warte auf die Beendigung von {0} Speicher
                  Threads", Math.Round((dt * n) / 365, 2));
82              Thread.Sleep(2000);
83              broadcaster.SendToChannel("steps", "s" + n);// send max step to
                  steps channel
84
85
86              Thread Key = new Thread(listen);
87              Key.Start();
88              var cluster = new BoxCluster(SQL.readStars(rTable, last), dt); //
                  instatiate Starcluster
89              var Sub = new SubCluster(SQL.readStars(rTable, last), dt);
90
91              var time = 0d;
92
93
94              cluster.ParentDt = 100;
95              cluster.DoStep(Misc.Method.Rk5, true, 0, -1);
96
97              Sub.Stars = new List<Star>(cluster.Stars.Select(x=>x.Clone()));
98              Sub.Dt = cluster.Dt;
99
100             var X = new List<double>();
101
102             var Y = new List<double>();
103
104
```

```
105             for (int i = (last /** SaveInterval * 365*/) + 1;
106                  !abort; i++)
107             {
108                 var maxDAcc = cluster.Stars.Max(x => x.DAcc);
109                 if (maxDAcc > 0)
110                 {
111                     cluster.ParentDt = 20000;
112                     Sub.ParentDt = 20000;
113                     Stopwatch watch = Stopwatch.StartNew();
114 
115                     for (int j = 0; j < 1; j++)
116                     {
117                          cluster.DoStep(Misc.Method.Rk5, true, 0, -1);
118                         //DoStep(ref Sub);
119                     }
120 
121                     watch.Stop();
122 
123                     //SQL.addRows(cluster.Stars, i, wTable);
124 
125                     Console.WriteLine("n: " + watch.ElapsedMilliseconds /
                        1.0 / 1000.0);
126 
127                     X.Add(watch.ElapsedMilliseconds / 1.0 / 1000.0);
128 
129                     watch.Restart();
130 
131                     for (int j = 0; j < 1; j++)
132                     {
133                         //DoStep(ref Sub);
134                         Sub.DoStep(Misc.Method.Rk5, true, 0, -1);
135                     }
136 
137                     watch.Stop();
138 
139                     //Sub.CalcDt();
140 
141                     SQL.addRows(Sub.Stars, i, wTable);
142                     Console.WriteLine("sub: " + watch.ElapsedMilliseconds /
                        1.0 / 1000.0);
143                     Y.Add(watch.ElapsedMilliseconds / 1.0 / 1000.0);
144 
145                     //cluster.CalcDt();
146                      //Sub.GetSubsetSeeds().ForEach(s => Console.Write($"{s},
                        "));
147                 }
148 
149                 time += dt;
150                 GnuPlot.HoldOn();
151                 GnuPlot.Unset("logscale y");
152                 GnuPlot.Set("key top left", "xlabel 'Dauer Normal'", "ylabel
                      'Gesamtdauer'");
153                 GnuPlot.Plot(X.ToArray(), Y.ToArray(), "title 'SubCluster' ");
154                 GnuPlot.Plot(X.ToArray(), X.ToArray(), "title 'Normal' w
                      linespoints");
155 
```

```
156                 //broadcaster.SendToChannel("steps", $"i{i}");
157
158                 // send "i"+step in channel steps
159                 // Console.WriteLine("\n");//+ i + "\n ");
160                 cluster.Stars.MoveCenter(cluster.Stars.GetCenter());
161
162
163                 if (Math.Ceiling((time - dt) / 365) < Math.Ceiling(time / 365)
                    && ++year % SaveInterval == 0)
164                 {
165                     Console.WriteLine($@"Exportiere Daten... Jahr: {(int)i *
                    dt / 365} = {year}");
166 //                    while (!SQL.addRows(cluster.Stars, year / SaveInterval,
      wTable))
167 //                    {
168 //                        Thread.Sleep(100);
169 //                    }
170                 }
171             }
172
173             Key.Abort();
174
175             SQL.order(wTable);
176             broadcaster.SendToChannel("steps", "abort");
177
178             Console.WriteLine("Direkt in Dataview öffnen? (y/n)");
179             string view = Console.ReadLine();                          //wait
                  for input
180             if (view == "y" || view == "Y")
181                 System.Diagnostics.Process.Start(@"DataView.exe", wTable);
182         }
183
184         private static void DoStep(ref SubCluster cluster, double dt = 30)
185         {
186             List<Star> newStars;
187
188             for (double time = 0; time < cluster.ParentDt;)
189             {
190                 var subClusters = cluster.DivideIntoSubClusters(true);
191                 var temp = new ConcurrentBag<Star>();
192
193                 time += subClusters.First().Dt;
194
195                 Parallel.ForEach(
196                     subClusters,
197                     c =>
198                         {
199                             var stars = c.DoStep(Misc.Method.Rk5, true);
200                             foreach (var star in stars)
201                             {
202                                 temp.Add(star);
203                             }
204                         });
205
206
207
```

```
208                 newStars = temp.OrderBy(s => s.id).ToList();
209
210
211                 if (newStars.Count != cluster.Stars.Count)
212                 {
213                     var duplicates = newStars.Where(x => newStars.Count(c =>
                    c.id == x.id) > 1).Select(d => d.id).Distinct()
214                         .ToList();
215                     foreach (var duplicate in duplicates)
216                     {
217                         while (newStars.Remove(newStars.Where(x => x.id ==
                    duplicate).OrderBy(c => c.DAcc).First()) && newStars.Count
                    (c => c.id == duplicate) > 1)
218                         {
219                         }
220                     }
221
222                     // throw new Exception("Duplikate!");
223                 }
224
225                 cluster = new SubCluster(newStars, dt: subClusters.First().Dt)
226                             {
227                                 ParentDt = cluster.ParentDt,
228                                 Stars = newStars.Select(
229                                     x => x.Clone()).ToList()
230                             };
231                 cluster.Stars.ForEach(x => x.ToCompute = false);
232             }
233
234
235             //cluster.Stars = newStars;
236         }
237
238
239         private static void listen()
240         {
241             XDMessagingClient client = new XDMessagingClient(); //https://
                github.com/TheCodeKing/XDMessaging.Net
242             IXDBroadcaster broadcaster =
                client.Broadcasters.GetBroadcasterForMode
                (XDTransportMode.HighPerformanceUI);
243
244             ConsoleKeyInfo keyinfo;
245             do
246             {
247                 keyinfo = Console.ReadKey();
248
249             }
250             while (keyinfo.Key != ConsoleKey.X);
251             Console.WriteLine("\n\n\n\n Beenden Eingeleitet\n\n\n\n");
252             abort = true;
253             broadcaster.SendToChannel("steps", "abort");
254         }
255     }
256 }
257
```