Blatt 1

10/10
10/10
10/10

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise 1 10/10 *

1. FORMUL record, colum 19 (* for water) → HETATM way more common
2. 77-78 Element Symbol
3. x:31-38, y:39-46, z:47-54 coordinates
4. 18-20 residue name
5. 22 (chainID) chain name

2 Exercise 2 10/10

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>

int main(int argc, char* argv[]) {

  std::string line;
  std::ifstream pdbfile; //(argv[1], std::ios::in);
  std::ofstream coordfile; //(argv[2], std::ios::out);

  //Sanity check for correct number of arguments

  switch(argc){

    case 2:
      pdbfile.open(argv[1], std::ios::in);
      break;

    case 3:
      pdbfile.open(argv[1], std::ios::in);
      coordfile.open(argv[2], std::ios::out);
      break;

    default:
      std::cout << "Wrong number of arguments.\n\n Useage:
    ParsePDB inFile [outFile]" << "\n";
      return 1;
  }
```

✓ well done

```
29
30    if (pdbfile.is_open()){
31
32      while (getline(pdbfile,line)){
33
34        if (line.substr(0,4) == "ATOM"){
35
36          //x:31-38, y:39-46, z:47-54; one less because arrays in C
      ++ are 0-based
37          //a single space is added because it may be, that all
      coordinates are the maximum allowed size.
38          //no trimming is done. could be added later on.
39          switch(argc){
40            case 2:
41              std::cout << line.substr(30,8) << ' ' << line.substr
      (38,8) << ' ' << line.substr(46,8) << '\n';//substr(position,
      length); just a reminder for myself
42              break;
43            case 3:
44              coordfile << line.substr(30,8) << ' ' << line.substr
      (38,8) << ' ' << line.substr(46,8) << '\n';
45              break;
46
47          }
48
49        }
50      }
51      pdbfile.close();
52      coordfile.close();
53    }
54
55    else std::cout << "Unable to open file";
56
57    return 0;
58  }
```

3 Exercise 3

10

```
1  #include <iostream>
2  #include <fstream>
3  #include <sstream>
4  #include <vector>
5  #include <string>
6
```

```
7  int main(int argc, char* argv[]) {
8
9    std::string line;
10   std::ifstream pdbfile;
11   std::ofstream translatefile;
12   std::vector<double> d;
13
14   if(argc == 3){
15
16     pdbfile.open(argv[1], std::ios::in);
17     translatefile.open(argv[2], std::ios::out);
18     d.assign(3,1.000);
19
20   }else if(argc == 6){
21
22     pdbfile.open(argv[1], std::ios::in);
23     translatefile.open(argv[2], std::ios::out);
24
25     d.push_back(std::stod(argv[3]));
26     d.push_back(std::stod(argv[4]));
27     d.push_back(std::stod(argv[5]));
28
29   }else{
30
31     std::cout << "Wrong amount of Parameters\n\n Useage:
       HandlingPDB inFile outFile\n";
32     return 1;
33
34   }
35
36   //Basic idea:
37   // Checking ATOM
38   // Grabbing Coordinates based on columns
39   // Parsing to double, adding offset
40   // And the c++ way to parsing double to string with fixed size
       and precision is to set up an outsream correctly
41   // Important: If you want a precision of 3, i.e 3 digits after
       the dot, you need an outstream.precision of 4. Otherwise
42   // it get rounded to early.
43   // Lastly, replacing the new string in the selected line
44   if (pdbfile.is_open()){
45     while (getline(pdbfile,line)){
46
47       if (line.substr(0,4) == "ATOM"){
48
49         double xCoord = stod(line.substr(30,8));
50         double yCoord = stod(line.substr(38,8));
51         double zCoord = stod(line.substr(46,8));
52
```

```
53
54              xCoord += d[0];
55              yCoord += d[1];
56              zCoord += d[2];
57
58              std::ostringstream xCoordStrs;
59              std::ostringstream yCoordStrs;
60              std::ostringstream zCoordStrs;
61
62              xCoordStrs.width(8);
63              xCoordStrs.precision(4);
64              yCoordStrs.width(8);
65              yCoordStrs.precision(4);
66              zCoordStrs.width(8);
67              zCoordStrs.precision(4);
68
69              xCoordStrs << xCoord;
70              yCoordStrs << yCoord;
71              zCoordStrs << zCoord;
72
73              line.replace(30,8,xCoordStrs.str());
74              line.replace(38,8,yCoordStrs.str());
75              line.replace(46,8,zCoordStrs.str());
76          }
77
78          translatefile << line << '\n';
79
80      }
81
82  }else{
83
84      std::cout << "Unable to open file";
85
86  }
87
88  return 0;
89
90 }
```

Blatt 2

10/10
10/10
③ 10/10

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise 1

```
1
2  #include <BALL/FORMAT/PDBFile.h>
3  #include <BALL/KERNEL/system.h>
4  #include <BALL/KERNEL/chain.h>
5  #include <BALL/KERNEL/residue.h>
6  #include <BALL/KERNEL/atom.h>
7  #include <BALL/KERNEL/protein.h>
8
9
10 int main(int argc, char* argv[]){
11
12    BALL::Vector3 translationVector(1,1,1);
13    BALL::System translationSystem;
14    BALL::PDBFile sourceFile;
15    BALL::PDBFile translateFile;
16
17    //Sanity checks for command-line arguments
18    if(argc == 3){
19
20       sourceFile.open(argv[1], std::ios::in);
21       translateFile.open(argv[2], std::ios::out);
22
23    }else if(argc == 6){
24
25    sourceFile.open(argv[1], std::ios::in);
26    translateFile.open(argv[2], std::ios::out);
27
28    translationVector = BALL::Vector3(std::stod(argv[3]), std::stod
      (argv[4]), std::stod(argv[5]));
29
30    }else{
31
32       std::cout << "Wrong amount of Parameters\n\n Useage: prog
      inFile outFile\n";
33       return 1;
34
35    }
36
37    if(sourceFile.is_open()){
38
39       //the read action for a pdbfile reads the data into a system
      which can be manipulated afterwards - and saved
40       sourceFile.read(translationSystem);
```

```
41      sourceFile.close();
42
43      // iterate over all proteins
44      for(BALL::MoleculeIterator m_it = translationSystem.
    beginMolecule(); +m_it; ++m_it){
45
46          //We need to check if the molecule we just grabbed with
    m_it is indeed a protein. if it is not,
47          //We cannot iterate over chains and residues.
48          if (BALL::RTTI::isKindOf<BALL::Protein>(*(m_it))){
49
50              BALL::Protein* protein = BALL::RTTI::castTo<BALL::
    Protein>(*(m_it));
51
52              if(protein->countChains() > 0){
53
54                  for(BALL::ChainIterator ch_it = protein->beginChain
    (); +ch_it; ++ch_it){
55
56                      for(BALL::ResidueIterator r_it = ch_it->
    beginResidue(); +r_it; ++r_it){
57
58                          for(BALL::AtomIterator a_it = r_it->beginAtom
    (); +a_it; ++a_it){
59
60                              //Exercise 1: Translating Atom-Positions
61                              a_it->setPosition(a_it->getPosition() +
    translationVector);
62                          }
63                      }
64                  }
65              }
66          }
67      }
68  }else{
69
70      std::cout << "Could not open PDB file." << '\n';
71
72  }
73
74  //Writing translatedFile
75
76
77  if(translateFile.is_open()){
78
79      translateFile.write(translationSystem);
80      translateFile.close();
81
82  }else{
```

```
83
84          std::cout << "Could not write translated PDB file." << '\n';
85
86      }
87
88      return 0;
89
90  }
```

Much more difficult than it was required!

2 Exercise 2

```
1
2  #include <BALL/FORMAT/PDBFile.h>
3  #include <BALL/KERNEL/system.h>
4  #include <BALL/KERNEL/chain.h>
5  #include <BALL/KERNEL/residue.h>
6  #include <BALL/KERNEL/atom.h>
7  #include <BALL/KERNEL/protein.h>
8
9
10 int main(int argc, char* argv[]){
11
12     BALL::Vector3 translationVector(1,1,1);
13     BALL::System translationSystem;
14     BALL::PDBFile sourceFile;
15     BALL::PDBFile translateFile;
16
17     //Sanity checks for command-line arguments
18     if(argc == 2){
19
20         sourceFile.open(argv[1], std::ios::in);
21
22     }else{
23
24         std::cout << "Wrong amount of Parameters\n\n Useage: prog
       inFile\n";
25         return 1;
26
27     }
28
29     if(sourceFile.is_open()){
30
31         //the read action for a pdbfile reads the data into a system
        which can be manipulated afterwards - and saved
```

```
32        sourceFile.read(translationSystem);
33        sourceFile.close();
34
35        // iterate over all proteins
36        for(BALL::MoleculeIterator m_it = translationSystem.
      beginMolecule(); +m_it; ++m_it){
37
38            //We need to check if the molecule we just grabbed with
      m_it is indeed a protein. if it is not,
39            //We cannot iterate over chains and residues.
40            if (BALL::RTTI::isKindOf<BALL::Protein>(*(m_it))){
41
42                BALL::Protein* protein = BALL::RTTI::castTo<BALL::
      Protein>(*(m_it));
43
44                if(protein->countChains() > 0){
45
46                  for(BALL::ChainIterator ch_it = protein->beginChain
      (); +ch_it; ++ch_it){
47
48                    for(BALL::ResidueIterator r_it = ch_it->
      beginResidue(); +r_it; ++r_it){
49
50                      for(BALL::AtomIterator a_it = r_it->beginAtom
      (); +a_it; ++a_it){
51
52                        //Exercise 2: Printing C-Alphas
53                        if(a_it->getName() == "CA"){
54
55                          std::cout << a_it->getPosition() << '\n
      ';
56
57                        }
58                      }
59                    }
60                  }
61                }
62              }
63            }
64        }else{
65
66          std::cout << "Could not open PDB file." << '\n';
67
68        }
69
70        return 0;
71
72    }
```

3 Exercise 3

3.1 $y''(x) + y(x) = \sin(2x)$

General solution will be from the form $y(x) = y_c(x) + y_p(x)$, which is the sum of the complementary and particular solution.

First determine the complementary solution:

$$y''(x) + y(x) = 0$$

Which can be solved by a sin(x) as well as a cos(x). Because

$$\partial_x^2 \sin(x) = -\sin(x)$$
$$\partial_x^2 \cos(x) = -\cos(x)$$

Therefore

$$y_c(x) = A\sin(x) + B\cos(x)$$

Second step is to determine the particular solution:

$$y_p(x) = C\sin(2x) + D\cos(2x)$$
$$\partial_x^2 y_p(x) = -4C\sin(2x) - 4D\cos(2x)$$

$$-4C\sin(2x) - 4D\cos(2x) + C\sin(2x) + D\cos(2x) = \sin(2x)$$
$$-3C\sin(2x) - 3D\cos(2x) = \sin(2x)$$
$$\Rightarrow D = 0 \quad C = -\frac{1}{3}\sin(2x)$$

$$\boxed{y(x) = A\sin(x) + B\cos(x) - \frac{1}{3}\sin(2x)}$$

Proof:

$$y''(x) = -A\sin(x) - B\cos(x) - \frac{4}{3}\sin(2x)$$

$$y''(x)+y(x) = \sin(x)+B\cos(x)-\frac{1}{3}\sin(2x)-A\sin(x)-B\cos(x)-\frac{4}{3}\sin(2x) = \sin(2x)$$

3.2 $y''(x) + y(x) = \sin(x)$

Complementary solution same as in section 3.1.

$$y_c(x) = A\sin(x) + B\cos(x)$$

Guess for particular solution:

$$y_p(x) = Cx\cos(x)$$

$$y_p'(x) = C\cos(x) - Cx\sin(x)$$

$$y_p''(x) = -C\sin(x) - Cx\cos(x) - C\sin(x) = -2C\sin(x) - Cx\cos(x)$$

$$-2C\sin(x) - Cx\cos(x) + Cx\cos(x) = \sin(x)$$

$$\Rightarrow C = -\frac{1}{2}$$

$$\boxed{y(x) = A\sin(x) + B\cos(x) - \frac{1}{2}x\cos(x)}$$

Proof:

$$y''(x) = -A\sin(x) - B\cos(x) + \sin(x) - \frac{1}{2}x\cos(x)$$

$$y''(x)+y(x) = A\sin(x)+B\cos(x)-\frac{1}{2}x\cos(x)-A\sin(x)-B\cos(x)+\sin(x)-\frac{1}{2}x\cos(x)$$

$$= \sin(x)$$

3.3 $y''(x) - y(x) = 0$

This is a homogeneous differential equation. Therefore it is only necessary to find the homogeneous solution $y_c(x)$.

The equation can be solved by e^x and e^-x.

$$\boxed{y(x) = Ae^x + Be^{-x}}$$

Proof:

$$y''(x) = Ae^x + (-1)^2Be^{-x} = Ae^x + Be^{-x}$$

$$y''(x) - y(x) = Ae^x + Be^{-x} - Ae^x - Be^{-x} = 0$$

Blatt 4

10
9
10

SUBMITTED BY: BREITENBERGER, NICODEMUS

10

1 Exercise: Solving a simple model – system differential equations

Consider a water molecule. Assuming that the position of the oxygen atom is fixed and the angle between the hydrogen atom is also fixed the only thing that can change is the distance between the oxygen and hydrogen atoms. These are defined as ρ_1 and ρ_2. In the next steps described with ρ_j, $j \in \{1, 2\}$.

The connection of the oxygen und hydrogen atom can be approximated with a spring with a spring constant k.

Therefore equating Newton's force with the spring force results in:

$$m\partial_t^2 \rho_j = -k\rho_j$$

$$\partial_t^2 \rho_j + \frac{k}{m}\rho_j = 0$$

with the initial conditions:

$$\rho_j(0) = \tilde{\rho_j}$$

$$\partial_t \rho_j(0) = v_j$$

$\rho_j(t)$ has to fullfill the equation as well as the initial conditions. Assume an exponential function:

$$\rho_j(t) = ce^{\lambda t}$$

$$\partial_t^2 \rho_j(t) = c\lambda^2 e^{\lambda t}$$

Insert into differential equation:

$$c\lambda^2 e^{\lambda t} + c\frac{k}{m}e^{\lambda t} = 0$$

$$\lambda^2 + \frac{k}{m} = 0$$

$$\lambda_{1,2} = \pm\sqrt{-\frac{k}{m}} = \pm i\sqrt{\frac{k}{m}}$$

Add the two complex solutions:

$$\rho_j(t) = c_1 \cdot e^{i\sqrt{\frac{k}{m}}t} + c_2 \cdot e^{-i\sqrt{\frac{k}{m}}t}$$

Initial conditions:

$$\rho_j(0) = c_1 + c_2 = \tilde{\rho}_j$$

This is fullfilled for:

$$\rho_j(t) = \tilde{\rho}_j \cos\left(\sqrt{\frac{k}{m}}t\right)$$

with $c_1 = c_2 = \frac{1}{2}\tilde{\rho}_j$

$$\partial_t \rho_j(t) = -\tilde{\rho}_j \sqrt{\frac{k}{m}} \sin\left(\sqrt{\frac{k}{m}}t\right)$$

$$\partial_t \rho_j(0) = -\tilde{\rho}_j \sqrt{\frac{k}{m}} \sin(0) = 0$$

This means for the velocity at t = 0:

$$v_j(t = 0) = 0$$

If we approximate the binding between the oxygen and the hydrogen atoms each with a spring force the initial conditions lay down that at the starting point t=0 the distances are at their maximum $\tilde{\rho}_j$. At time t=0 the velocity is zero because the motion is at a turning point. This motion of the water molecule is called the symmetric stretch mode.

2 Exercise: Lennard–Jones

```
# Gnuplot script file for plotting lennard jones potential and
    force
set   autoscale                        # scale axes automatically
unset log                              # remove any log-scaling
unset label                            # remove any previous labels
set xtic auto                          # set xtics automatically
set ytic auto                          # set ytics automatically

set title "Lennard-Jones potential and approximation"
set xlabel "r"
set ylabel "V(r)"
```

```
11
12  set xrange [11:13]
13  set yrange [-3:0]
14
15  LJV(x) = 2.0*((12.0/x)**12.0-2.0*(12.0/x)**6.0)
16  V(x) = -2.0 + (6.0*2.0)/(12.0**2.0)*(13.0-7.0)*(x-12.0)**2.0
17
18  set multiplot layout 1,2
19  set size 1,0.5
20  set origin 0,0
21
22  plot V(x) title "Approximation close to equilibrium point", LJV(x
      ) title "Lennard-Jones Potential"
23
24  #set term png
25  #set output "LennardJones.png"
26  #replot
27  #set term x11
28
29  # o.b.d.A. vector r = |r|*(1,0,0)
30  # only in x direction because gnuplot does not provide
31  # a 3d vector plot of a function
32
33  LJF(x) = 12.0*2.0/x * ((12.0/x)**12.0 - (12.0/x)**6.0)
34  F(x) = -72.0*(x-12.0)/(12.0**2.0)
35
36  set title "Lennard-Jones force and approximation"
37  set xlabel "x"
38  set ylabel "F(x)"
39
40  set xrange [10:15]
41  set yrange [-20:20]
42
43  set size 1,0.5
44  set origin 0,0.5
45  plot F(x) title "Approximation close to equilibrium point", LJF(x
      ) title "Lennard-Jones Force"
46
47  unset multiplot
48
49
```

The plots verify that the approximation is reasonable close to the equilibrium point. Further away from this point the approximated curves are very different from the Lennard-Jones potential and force.

Morse potential: $V(r) = \epsilon \left(1 - e^{-\sigma(r-r_0)}\right)^2$

Corresponding force:

$$\vec{F}(\vec{r}) = -\nabla V = 2\epsilon\sigma e^{-\sigma(r-r_0)} \left(1 - e^{-\sigma(r-r_0)}\right) \frac{\vec{r}}{r}$$

Approximation: Taylor expansion of the potential as in the exercise description for minimum (equilibrium point $\tilde{r}$).

$$\left.\frac{\partial V}{\partial r}\right|_{r=\tilde{r}} = 0 = 2\epsilon\sigma e^{-\sigma(\tilde{r}-r_0)} \left(1 - e^{-\sigma(\tilde{r}-r_0)}\right) \Leftrightarrow \tilde{r} = r_0$$

$$\left.\frac{\partial^2 V}{\partial r^2}\right|_{r=\tilde{r}} = 2\epsilon\sigma^2 e^{-2\sigma(\tilde{r}-r_0)} - 2\epsilon\sigma^2 e^{-\sigma(\tilde{r}-r_0)} \left(1 - e^{-\sigma(\tilde{r}-r_0)}\right)$$

$$V(r) = V(r_0) + \frac{\partial V(r_0)}{\partial r}(r - r_0) + \frac{1}{2}\frac{\partial^2 V(r_0)}{\partial r^2}(r - r_0)^2 + O((r - r_0)^3)$$

$$V(r) \approx 0 + 0 + \frac{2\epsilon\sigma^2}{2}(r - r_0)^2 = 2\epsilon\sigma^2(r - r_0)^2 \checkmark$$ calculation error

$$\vec{F}(\vec{r}) = -\nabla V \approx 4\epsilon\sigma^2(r - r_0)\frac{\vec{r}}{r}$$

Then the linear approximation constant equals $k_M = 4\epsilon\sigma^2$.

3 Exercise: Euler Method – C++

```
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>

//could be replaced by literal constants; define is the C way
#define K 1
#define M 1
#define H 0.01
#define STEPS 1000
//some constants regarding x_0 and v_0
#define X_0 0
#define V_0 2

```

```
15
16 double computeNextForce(double x_n_1){
17
18   return -1 * K * x_n_1;
19
20 }
21
22 double computeNextVelocity(double v_n_1, double F_n){
23
24   return v_n_1 + H / M * F_n;
25
26 }
27
28 double computeNextPosition(double x_n_1, double v_n_1){
29
30   return x_n_1 + H * v_n_1;
31
32 }
33
34 int main(int argc, char* argv[]){
35
36   double x_n_1 = X_0;
37   double v_n_1 = V_0;
38   double f_n = 0;
39   std::ofstream eulerFile, gnuPlot;
40
41   if(argc == 2){
42
43     eulerFile.open(argv[1], std::ios::out);
44     gnuPlot.open("gp.gp", std::ios::out);
45
46   }else{
47
48     std::cout << "Wrong number of arguments.\n\n Useage: prog
       outfile" << '\n';
49     return 1;
50
51   }
52
53   int i = 1;
54   for(i = 1; i <= STEPS; i++){
55
56     f_n = computeNextForce(x_n_1);
57     double v_n_temp = computeNextVelocity(v_n_1, f_n);
58     double x_n_temp = computeNextPosition(x_n_1, v_n_1);
59
60     v_n_1 = v_n_temp;
61     x_n_1 = x_n_temp;
62
```

```
63      eulerFile << i * H << ' ' << f_n   << '\n';
64      eulerFile << i * H << ' ' << v_n_1  << '\n';
65      eulerFile << i * H << ' ' << x_n_1  << '\n';
66
67    }
68
69    eulerFile.close();
70
71    //preparing a gnu-script file to print the data to a png-file
72    //basically, i set a gnuplot file to read the generated file and
        print it to a png file
73    //most of the stuff is hardcoded. There has to be a more elegant
        way. For now, this will suffice
74    gnuPlot << "set terminal png" << '\n';
75    gnuPlot << "plot \"" << argv[1] << "\" using 1:2 title \"Euler
        Method Plot for Force, Velocity and Position\"" << '\n';
76    gnuPlot.close();
77
78    std::system("gnuplot gp.gp > eulerMethod.png");
79
80
81  }
```

0|20
5|10
7|10

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise: Simple coarse grained model

2 Exercise: Nosè-Hoover

```cpp
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

//v is fixed, calculation of v(t+1/2dt) not necessary

double v(double t, double dt, double fixed){
    if (t <= 0){
        return 1.0;
    }
    else {
        // fix velocity to 0.8 or 1.2
        return fixed;  //1/2*(v(t-1/2*dt,dt)+v(t+1/2*dt,dt));
    }
}

// x(t) for a harmonic oszillator with m=k=1
double x(double t){
    double x0 = 1.0;

    if(t == 0){
        return x0;
    }
    else {
        return (x0*cos(t));
    }
}

double x(double t, double dt, double fixedv){
    return (x(t) + dt*v(t,dt,fixedv));
}

int main(int argc, char* argv[]) {

    ofstream outfile1;
    ofstream outfile2;
    outfile1.open("NoseHooverOut08.txt");
    outfile2.open("NoseHooverOut12.txt");
```

Not putting equal, but constrain

```
41
42      outfile1 << "t x(t) v(t)" << endl;
43      outfile2 << "t x(t) v(t)" << endl;
44      double dt = 0.1;
45
46      for (double t = 0; t < stod(argv[1]); t+=dt){
47          outfile1 << t << "  " << x(t, dt, 0.8) << "  " << v(t, dt,
        0.8) << endl;
48          outfile2 << t << "  " << x(t, dt, 1.2) << "  " << v(t, dt,
        1.2) << endl;
49      }
50
51      outfile1.close();
52      outfile2.close();
53
54      // plot x(t) and v(t)
55      system("gnuplot plot.gnu ");
56
57      return 0;
58  }
```

```
# gnuplot script to plot Nose-Hoover output
set   autoscale                         # scale axes automatically
unset log                               # remove any log-scaling
unset label                             # remove any previous
    labels
set xtic auto                           # set xtics automatically
set ytic auto                           # set ytics automatically
set title "Simulation of a harmonic oszillator with fixed velocity
    "
set xlabel "t"
set ylabel "x(t), v(t)"
set xr [0.0:100]
set yr [-2:2]


set size 2,2
set origin 0,0
set multiplot layout 2,1 columnsfirst scale 1,1

plot "NoseHooverOut08.txt" using 1:3 title 'v(t) fixed to 0.8'
    with linespoints, "NoseHooverOut08.txt" using 1:2 title 'x(t)'
     with linespoints

plot "NoseHooverOut12.txt" using 1:3 title 'v(t) fixed to 1.2'
    with linespoints, "NoseHooverOut12.txt" using 1:2 title 'x(t)'
     with linespoints

unset multiplot
```

3 Exercise: Molecular Dynamics - BALL

```
#include <BALL/KERNEL/system.h>
#include <BALL/KERNEL/selector.h>
#include <BALL/FORMAT/PDBFile.h>
#include <BALL/MOLMEC/MDSIMULATION/microCanonicalMD.h>
#include <BALL/STRUCTURE/fragmentDB.h>
#include <BALL/STRUCTURE/residueChecker.h>
#include <BALL/MOLMEC/AMBER/amber.h>
#include <BALL/MOLMEC/MINIMIZATION/conjugateGradient.h>

using namespace std;
using namespace BALL;

int main(int argc, char* argv[]){

    PDBFile sourceFile;
    System mdSystem;

    //Sanity checks for command-line arguments
    if(argc == 3){
        sourceFile.open(argv[1], ios::in);
    }else{
        cout << "Wrong amount of Parameters\n\n Useage: prog
     sourceFile simulationTime\n";
        return 1;
    }

    if(sourceFile.is_open()){
        sourceFile.read(mdSystem);
        sourceFile.close();
    }

    FragmentDB db("");
    mdSystem.apply(db.normalize_names);
    mdSystem.apply(db.add_hydrogens);
    mdSystem.apply(db.build_bonds);

    ResidueChecker rc(db);
    mdSystem.apply(rc);

    // create hydrogen bonds and force field
    AmberFF amber(mdSystem);
    Selector hydrogen_selector("element(H)");
    mdSystem.apply(hydrogen_selector);

    amber.options[PeriodicBoundary::Option::PERIODIC_BOX_ENABLED]="
     true";
```

```
45      amber.setup(mdSystem);
46
47      MicroCanonicalMD md(amber);
48      md.setReferenceTemperature(300.0);
49      md.setEnergyOutputFrequency(500.0);
50
51      // redirect std::cout to file. Found no other possibility to
         write MD simulation output directly to file
52      ofstream finalMD("finalMD.txt");
53      streambuf *coutbuf = cout.rdbuf();
54      cout.rdbuf(finalMD.rdbuf());
55      md.simulateTime(stod(argv[2]));
56      cout.rdbuf(coutbuf);
57      finalMD.close();
58      return 0;
59  }
```

simulate one step, get energy and again...

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise: Numerical integration

9

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>

using namespace std;

int main(int argc, char* argv[]) {

    double k = 1.0;
    double m = 1.0;
    double h = 0.01;

    // double h = 0.01; does not affect the simulation in the range of t = 40
    // greater timessteps lead to a divergence of the Ekin and Pges especially for the euler method

    ofstream euler;
    ofstream verlet;
    euler.open("euler.txt");
    verlet.open("verlet.txt");

    euler << "t Ekin Pges" << endl;
    verlet << "t Ekin Pges" << endl;

    double t0 = 0.0;

    vector<double> eulerx;
    vector<double> eulerv;
    vector<double> eulerF;

    vector<double> verletx;
    vector<double> verletv;
    vector<double> verletF;

    eulerx.push_back(1.0);
    eulerv.push_back(1.0);
    eulerF.push_back(0);

    verletx.push_back(1.0);
    verletv.push_back(1.0);
    verletF.push_back(0);
```

Not a good choice
Typically, $h \sim 10^{-4}$

```
42
43      for(int n=1;n<10000;n++){
44          eulerx.push_back(eulerx[n-1] + h*eulerv[n-1]);
45          eulerF.push_back(-k*eulerx[n-1]);
46          eulerv.push_back(eulerv[n-1] + h/m*eulerF[n]);
47
48          verletx.push_back(verletx[n-1]+verletv[n-1]*h+verletF[n]/(2*
        m)*pow(h,2));
49          verletF.push_back(-k*verletx[n]);
50          verletv.push_back(verletv[n-1] + h/(2*m)*(verletF[n-1] +
        verletF[n]));
51      }
52
53      for(int n=0;n<10000;n++){
54          euler << t0+n*h << " " << 3/2*m*pow(eulerv[n],2) << " " << 3
        *m*eulerv[n] << endl;
55          verlet << t0+n*h << " " << 3/2*m*pow(verletv[n],2) << " " <<
         3*m*verletv[n] << endl;
56      }
57
58      euler.close();
59      verlet.close();
60
61      system("gnuplot plot.gnu");
62
63      return 0;
64  }
```

File plot.gnu:

```
1  # gnuplot script to plot NumInt output
2  set   autoscale                          # scale axes automatically
3  unset log                                # remove any log-scaling
4  unset label                              # remove any previous
       labels
5  set xtic auto                            # set xtics automatically
6  set ytic auto                            # set ytics automatically
7  set xlabel "t"
8  set ylabel "Ekin(t), Pges(t)"
9
10
11 set size 2,2
12 set origin 0,0
13 set multiplot layout 2,1 columnsfirst scale 1,1
14
15 set xr [0.0:20.0]
16 set yr [-20:20]
```

```
17 set title "Numerical integration - harmonic oscillator - Euler
       algorithm"
18 plot "euler.txt" using 1:2 title 'Ekin(t)' with points, "euler.txt
       " using 1:3 title 'Pges(t)' with points
19
20 set xr [0.0:20.0]
21 set yr [-20:20]
22 set title "Numerical integration - harmonic oscillator - Velocity-
       Verlet algorithm"
23 plot "verlet.txt" using 1:2 title 'Ekin(t)' with points, "verlet.
       txt" using 1:3 title 'Pges(t)' with points
24
25 unset multiplot
```

2 Exercise: Local Minimizer - C++

Given function:

$$f(x,y) = e^{-x^2-y^2}$$

For local minima/maxima the condition $\partial_x f(x,y) = 0$ and $\partial_y f(x,y) = 0$ has to be fullfilled.

$$\partial_x f(x,y) = -2xe^{-x^2-y^2}$$

and

$$\partial_y f(x,y) = -2ye^{-x^2-y^2}$$

Because e^z is never zero for any z, so $-2x = 0$ and $-2y = 0$, equal to $x = 0$ and $y = 0$. That means a local extremal point is at (0,0).

To determine if it is a local minimum or maximum the second derivatives have to be calculated. H is the Hessian matrix :

$$H(x,y) = \begin{pmatrix} f_{xx}(x,y) & f_{xy}(x,y) \\ f_{yx}(x,y) & f_{yy}(x,y) \end{pmatrix}$$

$$D(x_c,y_c) = \det(H(x,y)) = f_{xx}(x_c,y_c)f_{yy}(x_c,y_c) - [f_{xy}(x_c,y_c)]^2$$

With the cases:

- If $D(a,b) > 0$ and $f_{xx}(a,b) > 0$ (a,b) is a local minimum of f.

- If $D(a,b) > 0$ and $f_{xx}(a,b) < 0$ (a,b) is a local maximum of f.
- If $D(a,b) < 0$ then (a,b) is a saddle point of f.
- If $D(a,b) = 0$ then the derivative test is inconclusive.

$$f_{xx}(x,y) = -2e^{-x^2-y^2} + 4xe^{-x^2-y^2}$$
$$f_{yy}(x,y) = -2e^{-x^2-y^2} + 4ye^{-x^2-y^2}$$

$$f_{xy}(x,y) = f_{yx}(x,y) = 4xye^{-x^2-y^2}$$

$$D(0,0) = (-2+0)\cdot(-2+0) - 0 = 4 > 0$$
$$f_{xx}(0,0) = -2 + 0 = -2 < 0$$

This means that there is a local maximum at (0,0). 6/10 minima?

Blatt 7

SUBMITTED BY: BREITENBERGER, NICODEMUS 8/10

1 Exercise: Calculation of the work

Point moving in force field:

$$\vec{F} = (3x + y)\vec{i} + (x + 2y)\vec{j}$$

Two paths:

$$\gamma_1(t) = \begin{cases} t\vec{i} & t \in [0, 3) \\ 3\vec{i} + (t-3)\vec{j} & t \in [3, 6] \end{cases}$$

$$\gamma_2(t) = \left\{ t\vec{i} + t\vec{j} \quad t \in [0, 3) \right.$$

Calculate work :

$$W_{ab} = \int_{\vec{r}_a}^{\vec{r}_b} \vec{F}(\vec{r})d\vec{r} = \int_{t_a}^{t_b} \vec{F}(\vec{r})\frac{d\vec{r}}{dt}dt$$

With $\vec{r}(t) = \gamma_1(t)$:

$$W_{ab} = \int_0^3 ((3t+0)\vec{i}+(t+0)\vec{j})\frac{d(t\vec{i})}{dt}dt + \int_3^6 ((9+(t-3))\vec{i}+(3+2(t-3))\vec{j})\frac{d(3\vec{i}+(t-3)\vec{j})}{dt}dt$$

NO!

$$W_{ab} = \int_0^3 (9 + (t-3))dt + \int_3^6 (3 + 2(t-3))dt$$

$$= \int_0^3 (6+t)dt + \int_3^6 (2t-3)dt = \left[6t + \frac{1}{2}t^2\right]_0^3 + \left[t^2 - 3t\right]_3^6$$

$$= 18 + 4.5 + 36 - 18 - 9 + 9 = 40.5$$

With $\vec{r}(t) = \gamma_2(t)$:

$$W_{ab} = \int_0^3 ((3t+t)\vec{i} + (t+2t)\vec{j})\frac{d(t\vec{i}+t\vec{j})}{dt}dt$$

$$= \int_0^3 ((3t+t)\vec{i} + (t+2t)\vec{j})(\vec{i}+\vec{j})dt$$

$$= \int_0^3 (3t+t+t+2t)dt = \int_0^3 (3t+t+t+2t)dt = \int_0^3 7tdt$$

$$= \left[\frac{7}{2}t^2\right]_0^3 = \frac{7}{2}9 = 31.5$$

The work depends on the path the point goes through the force field.

Wrong but consistent

2 Exercise: Generating a grid for Katchalski-Katzir

20/30

No hashgrid

```
#include <BALL/FORMAT/PDBFile.h>
#include <BALL/KERNEL/system.h>
#include <BALL/KERNEL/chain.h>
#include <BALL/KERNEL/residue.h>
#include <BALL/KERNEL/atom.h>
#include <BALL/KERNEL/protein.h>
#include <BALL/KERNEL/PTE.h>
#include <BALL/MATHS/vector3.h>
#include <array>
#include <iostream>
#include <fstream>

using namespace std;
using namespace BALL;

int alpha1 = 1;
int beta1 = 0;
int gamma1 = 2;

int main(int argc, char* argv[]){

  //   BALL::Vector3 translationVector(1,1,1);
  System kkSystem;
  PDBFile sourceFile;
  // BALL::PDBFile translateFile;

  vector<Vector3> atompos;
  Vector3 pos;
  vector<float> vwradii;
  float vwr;

  //Sanity checks for command-line arguments
  if(argc == 2){

    sourceFile.open(argv[1], ios::in);
  }else{

    cout << "Wrong amount of Parameters\n\n Useage: prog inFile \n
    ";
    return 1;
```

```
40
41  }
42
43  if(sourceFile.is_open()){
44
45    //the read action for a pdbfile reads the data into a system
      which can be manipulated afterwards - and saved
46    sourceFile.read(kkSystem);
47    sourceFile.close();
48
49    // get only first protein
50    Protein* protein = kkSystem.getProtein(0);
51
52    if(protein->countChains() > 0){
53
54      for(ChainIterator ch_it = protein->beginChain(); +ch_it; ++
      ch_it){
55
56        for(ResidueIterator r_it = ch_it->beginResidue(); +r_it;
      ++r_it){
57
58          for(AtomIterator a_it = r_it->beginAtom(); +a_it; ++a_it
      ){
59            Element element = a_it->getElement();
60            // name
61            cout << element.getName() << " ";
62            // position
63            pos = a_it->getPosition();
64            // save position in vector
65            atompos.push_back(pos);
66            cout << pos  << " ";
67            // van der waals radius
68            vwr = element.getVanDerWaalsRadius();
69            cout << vwr << endl;
70            vwradii.push_back(vwr);
71          }
72        }
73      }
74    }
75    for(int i(0); i < atompos.size(); i++){
76      cout << atompos[i] << vwradii[i] << endl;
77    }
78  }else{
79    cout << "Could not open PDB file." << '\n';
80  }
81
82  // dimension of katchalski katzir grid n*m*l
83  int n = 50;
84  int m = 50;
```

```
85  int l = 50;
86  float px;
87  float py;
88  float pz;
89
90  float kkgrid[n][m][l];
91  // initialize kkgrid
92  for (int i = 0; i < n; i++){
93
94    for (int j = 0; j < m; j++){
95
96      for (int k = 0; k < l; k++){
97        int c = 0;
98        while (c < atompos.size()){
99
100         px = 0.1*(i-n/2);
101         py = 0.1*(j-m/2);
102         pz = 0.1*(k-l/2);
103
104         // inside   you are using cubes, not spheres
105         if (px > (atompos[c][0] - 0.1*vwradii[c]) && px < (
    atompos[c][0] + 0.1*vwradii[c]) && py > (atompos[c][1] - 0.1*
    vwradii[c]) && py < (atompos[c][1] + 0.1*vwradii[c]) && pz > (
    atompos[c][2] - 0.1*vwradii[c]) && pz < (atompos[c][2] + 0.1*
    vwradii[c]) ){
106           kkgrid[i][j][k] = beta1;
107         }
108         // outside
109         else if  (px < (atompos[c][0] - 0.1*vwradii[c]) ||  px >
     (atompos[c][0] + 0.1*vwradii[c]) || py > (atompos[c][1] - 0.1
    *vwradii[c]) || py < (atompos[c][1] + 0.1*vwradii[c]) || pz >
    (atompos[c][2] - 0.1*vwradii[c]) || pz < (atompos[c][2] + 0.1*
    vwradii[c]) ){
110           kkgrid[i][j][k] = gamma1;
111         }
112         // boundary
113         else{
114           kkgrid[i][j][k] = alpha1;
115         }
116         c++;
117       }
118     }
119   }
120 }
121
122  // safe kkgrid to file
123  ofstream kkfile;
124  kkfile.open("kkgrid.txt");
125  kkfile << "x y z value" << endl;
```

```
126
127   for (int x(0); x < n; x++){
128
129     for (int y(0); y < m; y++){
130
131       for (int z(0); z < l; z++){
132
133         kkfile << 0.1*(x-n/2) << " " << 0.1*(y-m/2) << " " << 0.1*
      (z-l/2) << " " << kkgrid[x][y][z] << endl;
134       }
135     }
136   }
137
138   kkfile.close();
139
140   // countour plots
141   system("gnuplot plotkkgrid.gnu");
142
143   return 0;
144
145 }
```

Gnuplot script "plotkkgrid.gnu":

```
1  # gnuplot script to plot katchalski katzir output
2
3  set parametric
4  set contour base
5  set view 0,0,1
6  unset surface
7  set cntrparam levels 5
8  set dgrid3d
9
10 set title "Katchalski-Katzir Grid entries for different planes"
11
12 set size 2,2
13 set origin 0,0
14 set multiplot layout 3,1 columnsfirst scale 1,1
15
16 set xr [-10.0:10.0]
17 set yr [-10.0:10.0]
18 set zr [0:2]
19 set xlabel "x"
20 set ylabel "y"
21 splot "kkgrid.txt" using 1:2:4 title "x-y plane" with line
22
23 set xr [-10.0:10.0]
24 set yr [-10.0:10.0]
```

```
25 set zr [0:2]
26 set xlabel "y"
27 set ylabel "z"
28 plot "kkgrid.txt" using 2:3:4 title "y-z plane" with line
29
30 set xr [-10.0:10.0]
31 set yr [-10.0:10.0]
32 set zr [0:2]
33 set xlabel "x"
34 set ylabel "z"
35 plot "kkgrid.txt" using 1:3:4 title "x-z plane" with line
36
37 unset multiplot
```

Blatt 8

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise: Fourier transform

8/10

$$f_1(x) = \sin(x) \quad x \in [-\pi, \pi]$$
$$f_2(x) = \cos(x) \quad x \in [-\pi, \pi]$$
$$f_3(x) = e^{-x} \quad x \in [0, 2]$$

Fourier transformation:

$$F[f](k) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} f(x) e^{-ikx} dx$$

$$F[f_1](k) = \int_{-\pi}^{\pi} \frac{1}{\sqrt{2\pi}} \sin(x) e^{-ikx} dx = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \frac{e^{ix} - e^{-ix}}{2i} e^{-ikx} dx$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \frac{1}{2i} \left(e^{ix(1-k)} - e^{-ix(1+k)} \right) dx$$

$$= \frac{1}{2i\sqrt{2\pi}} \left[\frac{1}{i(1-k)} e^{ix(1-k)} + \frac{1}{i(1+k)} e^{-ix(1+k)} \right]_{-\pi}^{\pi}$$

What if $k = \pm 1$?

$$= \frac{-1}{2\sqrt{2\pi}} \left[\frac{1}{(1-k)} e^{i\pi(1-k)} + \frac{1}{(1+k)} e^{-i\pi(1+k)} - \frac{1}{(1-k)} e^{i(-\pi)(1-k)} - \frac{1}{(1+k)} e^{-i(-\pi)(1+k)} \right]$$

$$= \frac{-1}{2\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[(1+k) e^{i\pi(1-k)} + (1-k) e^{-i\pi(1+k)} - (1+k) e^{-i\pi(1-k)} - (1-k) e^{i\pi(1+k)} \right]$$

$$= \frac{-1}{2\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[-(1+k)(e^{-i\pi k} - e^{i\pi k}) - (1-k)(e^{-i\pi k} - e^{i\pi k}) \right]$$

$$= \frac{-1}{2\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[(-1-k-1+k)(e^{-i\pi k} - e^{i\pi k}) \right]$$

$$= \frac{-1}{2\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[-2(e^{-i\pi k} - e^{i\pi k}) \right] = \frac{1}{\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[(e^{-i\pi k} - e^{i\pi k}) \right]$$

$$F[f_1](k) = \frac{-i}{(1-k^2)} \sqrt{\frac{2}{\pi}} \sin(\pi k)$$

$$F[f_2](k) = \int_{-\pi}^{\pi} \frac{1}{\sqrt{2\pi}} \cos(x) e^{-ikx} dx = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \frac{e^{ix} + e^{-ix}}{2} e^{-ikx} dx$$

$$= \frac{1}{2\sqrt{2\pi}} \int_{-\pi}^{\pi} \left(e^{ix(1-k)} + e^{-ix(1+k)} \right) dx$$

$$= \frac{1}{2\sqrt{2\pi}} \left[\frac{1}{i(1-k)} e^{ix(1-k)} - \frac{1}{i(1+k)} e^{-ix(1+k)} \right]_{-\pi}^{\pi}$$

$$= \frac{1}{2i\sqrt{2\pi}} \left[\frac{1}{(1-k)} e^{i\pi(1-k)} - \frac{1}{(1+k)} e^{-i\pi(1+k)} - \frac{1}{(1-k)} e^{-i\pi(1-k)} + \frac{1}{(1+k)} e^{i\pi(1+k)} \right]$$

$$= \frac{1}{2i\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[(1+k) e^{i\pi(1-k)} - (1-k) e^{-i\pi(1+k)} - (1+k) e^{-i\pi(1-k)} + (1-k) e^{i\pi(1+k)} \right]$$

$$= \frac{1}{2i\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[(1+k)(e^{i\pi(1-k)} - e^{-i\pi(1-k)}) + (1-k)(e^{i\pi(1+k)} - e^{-i\pi(1+k)}) \right]$$

$$= \frac{1}{2i\sqrt{2\pi}} \frac{1}{(1-k^2)} \left[-(1+k)(e^{-i\pi k} - e^{i\pi k}) + (1-k)(e^{-i\pi k} - e^{i\pi k}) \right]$$

k = ±1

$$= \frac{-2k}{2i\sqrt{2\pi}} \frac{1}{(1-k^2)} (e^{-i\pi k} - e^{i\pi k}) = \frac{k}{(1-k^2)} \sqrt{\frac{2}{\pi}} \sin(\pi k)$$

$$F[f_3](k) = \int_0^2 \frac{1}{\sqrt{2\pi}} e^{-x} e^{-ikx} dx = \frac{1}{\sqrt{2\pi}} \int_0^2 e^{-x(1+ik)} dx$$

Calculation error!

$$= \frac{1}{\sqrt{2\pi}} \left[\frac{-1}{1-ik} e^{-x(1+ik)} \right]_0^2 = \frac{1}{\sqrt{2\pi}} \left[\frac{-1}{1+ik} (e^{-2(1+ik)} - 1) \right] = \frac{(1 - e^{-2(1+ik)})}{\sqrt{2\pi}(1+ik)}$$

2 Exercise: Convolution

consistent 10/10

Discrete convolution is defined as

$$\sum_{k \in D} f(k) \cdot g(n-k)$$

- Impressive but I wanted $\int \sin(x) \cos(x-y) \ldots$

with $f(x) = \sin(x)$ and $g(x) = \cos(x)$ in $[-\pi, \pi]$. Discretizing the signal gives 15 discrete points, which are

$$-\pi, -\pi\frac{6}{7}, -\pi\frac{5}{7}, -\pi\frac{4}{7}, -\pi\frac{3}{7}, -\pi\frac{2}{7}, -\pi\frac{1}{7}, 0, \pi\frac{1}{7}, \pi\frac{2}{7}, \pi\frac{3}{7}, \pi\frac{4}{7}, \pi\frac{5}{7}, \pi\frac{6}{7}, \pi$$

Caculating the discrete convolution gives:

$$F(0) = \sin(-\pi) \cdot \cos(-\pi + \pi) = 0$$

$$F(1) = \sin(-\pi) \cdot \cos(-\frac{6}{7}\pi + \pi) + \sin(-\frac{6}{7}\pi) \cdot \cos(-\frac{6}{7}\pi + \frac{6}{7}\pi) = -0.05$$

$$F(2) = \sin(-\pi)\cdot\cos(-\frac{5}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(-\frac{5}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(-\frac{5}{7}\pi+\frac{5}{7}\pi) = -8.7\cdot10^{-3}$$

$$F(3) = \sin(-\pi)\cdot\cos(-\frac{4}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(-\frac{4}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(-\frac{4}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi) \cdot \cos(-\frac{4}{7}\pi + \frac{4}{7}\pi) = -8.7 \cdot 10^{-3} = -0.01$$

$$F(4) = \sin(-\pi)\cdot\cos(-\frac{3}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(-\frac{3}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(-\frac{3}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi) \cdot \cos(-\frac{3}{7}\pi + \frac{4}{7}\pi) + \sin(-\frac{3}{7}\pi) \cdot \cos(-\frac{3}{7}\pi + \frac{3}{7}\pi) = -0.014$$

$$F(5) = \sin(-\pi)\cdot\cos(-\frac{2}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(-\frac{2}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(-\frac{2}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(-\frac{2}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(-\frac{2}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(-\frac{2}{7}\pi+\frac{2}{7}\pi)$$

$$F(6) = \sin(-\pi)\cdot\cos(-\frac{1}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(-\frac{1}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(-\frac{1}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(-\frac{1}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(-\frac{1}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(-\frac{1}{7}\pi+\frac{2}{7}\pi)$$

$$+\sin(-\frac{1}{7}\pi)\cdot\cos(-\frac{1}{7}\pi+\frac{1}{7}\pi)$$

$$F(7)=\sin(-\pi)\cdot\cos(\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(\frac{1}{7}\pi)+\sin(0)\cdot\cos(0+0)$$

$$F(8)=\sin(-\pi)\cdot\cos(\frac{1}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{1}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{1}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{1}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{1}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(\frac{1}{7}\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(\frac{1}{7}\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\frac{1}{7}\pi-\frac{1}{7}\pi)$$

$$F(9)=\sin(-\pi)\cdot\cos(\frac{2}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{2}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{2}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{2}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{2}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(\frac{2}{7}\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(\frac{2}{7}\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\frac{1}{7}\pi-\frac{1}{7}\pi)+\sin(\frac{2}{7}\pi)\cdot\cos(\frac{2}{7}\pi-\frac{2}{7}\pi)$$

$$F(10)=\sin(-\pi)\cdot\cos(\frac{3}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\frac{3}{7}\pi-\frac{1}{7}\pi)+\sin(\frac{2}{7}\pi)\cdot\cos(\frac{3}{7}\pi-\frac{2}{7}\pi)$$
$$+\sin(\frac{3}{7}\pi)\cdot\cos(\frac{3}{7}\pi-\frac{3}{7}\pi)$$

$$F(11)=\sin(-\pi)\cdot\cos(\frac{4}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{4}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{4}{7}\pi+\frac{5}{7}\pi)$$

$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{4}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{4}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(\frac{4}{7}\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(\frac{3}{7}\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\frac{4}{7}\pi-\frac{1}{7}\pi)+\sin(\frac{2}{7}\pi)\cdot\cos(\frac{4}{7}\pi-\frac{2}{7}\pi)$$
$$+\sin(\frac{3}{7}\pi)\cdot\cos(\frac{4}{7}\pi-\frac{3}{7}\pi)+\sin(\frac{4}{7}\pi)\cdot\cos(\frac{4}{7}\pi-\frac{4}{7}\pi)$$

$$F(12)=\sin(-\pi)\cdot\cos(\frac{5}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{5}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{5}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{5}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{5}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(-\frac{5}{7}\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(-\frac{5}{7}\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\frac{5}{7}\pi-\frac{1}{7}\pi)+\sin(\frac{2}{7}\pi)\cdot\cos(\frac{5}{7}\pi-\frac{2}{7}\pi)$$
$$+\sin(\frac{3}{7}\pi)\cdot\cos(\frac{5}{7}\pi-\frac{3}{7}\pi)+\sin(\frac{4}{7}\pi)\cdot\cos(\frac{5}{7}\pi-\frac{4}{7}\pi)+\sin(\frac{5}{7}\pi)\cdot\cos(\frac{5}{7}\pi-\frac{5}{7}\pi)$$

$$F(13)=\sin(-\pi)\cdot\cos(\frac{6}{7}\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\frac{6}{7}\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\frac{6}{7}\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\frac{6}{7}\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\frac{6}{7}\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(-\frac{6}{7}\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(-\frac{6}{7}\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\frac{6}{7}\pi-\frac{1}{7}\pi)+\sin(\frac{2}{7}\pi)\cdot\cos(\frac{6}{7}\pi-\frac{2}{7}\pi)$$
$$+\sin(\frac{3}{7}\pi)\cdot\cos(\frac{6}{7}\pi-\frac{3}{7}\pi)+\sin(\frac{4}{7}\pi)\cdot\cos(\frac{6}{7}\pi-\frac{4}{7}\pi)+\sin(\frac{5}{7}\pi)\cdot\cos(\frac{6}{7}\pi-\frac{5}{7}\pi)+\sin(\frac{6}{7}\pi)\cdot\cos(\frac{6}{7}\pi-\frac{6}{7}\pi)$$

$$F(14)=\sin(-\pi)\cdot\cos(\pi+\pi)+\sin(-\frac{6}{7}\pi)\cdot\cos(\pi+\frac{6}{7}\pi)+\sin(-\frac{5}{7}\pi)\cdot\cos(\pi+\frac{5}{7}\pi)$$
$$+\sin(-\frac{4}{7}\pi)\cdot\cos(\pi+\frac{4}{7}\pi)+\sin(-\frac{3}{7}\pi)\cdot\cos(\pi+\frac{3}{7}\pi)+\sin(-\frac{2}{7}\pi)\cdot\cos(\pi+\frac{2}{7}\pi)$$
$$+\sin(-\frac{1}{7}\pi)\cdot\cos(\pi+\frac{1}{7}\pi)+0+\sin(\frac{1}{7}\pi)\cdot\cos(\pi-\frac{1}{7}\pi)+\sin(\frac{2}{7}\pi)\cdot\cos(\pi-\frac{2}{7}\pi)$$
$$+\sin(\frac{3}{7}\pi)\cdot\cos(\pi-\frac{3}{7}\pi)+\sin(\frac{4}{7}\pi)\cdot\cos(\pi-\frac{4}{7}\pi)+\sin(\frac{5}{7}\pi)\cdot\cos(\pi-\frac{5}{7}\pi)+\sin(\frac{6}{7}\pi)\cdot\cos(\pi-\frac{6}{7}\pi)+0$$

10/10

3 Exercise: Fast Fourier Transformation

```
#include <iostream>
#include <complex>
#include <cmath>
#include <iterator>

using namespace std;

const double PI = 3.1415926536;

1. based

//based on
// https://www.safaribooksonline.com/library/view/c-cookbook
    /0596007612/ch11s18.html
unsigned int bitReverse(unsigned int x, int log2n) {
  int n = 0;
  int mask = 0x1;
  for (int i=0; i < log2n; i++) {
    n <<= 1;
    n |= (x & 1);
    x >>= 1;
  }
  return n;
}

template<class Iter_T>
void fft(Iter_T a, Iter_T b, int log2n)
{
  typedef typename iterator_traits<Iter_T>::value_type complex;
  const complex J(0, 1);
  int n = 1 << log2n;
  for (unsigned int i=0; i < n; ++i) {
    b[bitReverse(i, log2n)] = a[i];
  }
  for (int s = 1; s <= log2n; ++s) {
    int m = 1 << s;
    int m2 = m >> 1;
    complex w(1, 0);
    complex wm = exp(-J * (PI / m2));
    for (int j=0; j < m2; ++j) {
      for (int k=j; k < n; k += m) {
        complex t = w * b[k + m2];
        complex u = b[k];
        b[k] = u + t;
        b[k + m2] = u - t;
      }
      w *= wm;
```

```
46      }
47    }
48  }
49
50  int main(int argc, char* argv[]){
51
52    //building the arrays
53    complex<double> sinus[16];
54    complex<double> cosinus[16];
55    complex<double> e_fkt[16];
56
57    //result arrays
58    complex<double> sinus_fft[16];
59    complex<double> cosinus_fft[16];
60    complex<double> e_fkt_fft[16];
61    complex<double> sin_cos_con[16];
62
63    //step-array
64    double steps[16] = {-PI, -PI*6.0/7, -PI*5.0/7,  -PI*4.0/7, -PI*
        3.0/7,  -PI*2.0/7, -PI*1.0/7, 0, PI*1.0/7, PI*2.0/7,  PI*
        3.0/7, PI*4.0/7,  PI*5.0/7, PI*6.0/7, PI*7.0/8, PI};
65    double steps_e[16] = {0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75,
        0.875, 1, 1.125, 1.25, 1.375,  1.5, 1.75, 1.875, 2};
66
67    //filling arrays
68    for(int i = 0; i < 16; i++){
69
70      sinus[i] = complex<double>(sin(steps[i]), 0);
71      cosinus[i] = complex<double>(cos(steps[i]), 0);
72      e_fkt[i] = complex<double>(exp(steps_e[i]), 0);
73
74    }
75
76    //calculating ffts
77    fft(sinus, sinus_fft, 4);
78    fft(cosinus, cosinus_fft, 4);
79    fft(e_fkt, e_fkt_fft, 4);
80
81    //calculating convolution16
82    for(int i = 0; i < 16; i++){
83
84      sin_cos_con[i] = sinus_fft[i] * cosinus_fft[i];
85
86    }
87
88    //printing results
89    for(int i = 0; i < 16; i++){
90
91      cout << sinus_fft[i] << ' ';
```

please, write e for loop

```
92
93      }
94
95      cout << "\n\n";
96
97      for(int i = 0; i < 16; i++){
98
99        cout << cosinus_fft[i] << ' ';
100
101     }
102
103     cout << "\n\n";
104
105     for(int i = 0; i < 16; i++){
106
107       cout << e_fkt_fft[i] << ' ';
108
109     }
110
111     cout << "\n\n";
112
113     for(int i = 0; i < 16; i++){
114
115       cout << sin_cos_con[i] << ' ';
116
117     }
118
119     cout << '\n';
120
121     return 0;
122 }
```

Blatt 3

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise: Minimize RMSD

$$\text{RMSD}(A,B) = \min_{R,T} \sqrt{\frac{1}{N}\sum_i ||A_i - R(B_i - T)||^2}$$

No rotation $\Rightarrow$ R is equal to identity matrix.

$$\text{RMSD}(A,B) = \min_{T} \sqrt{\frac{1}{N}\sum_i ||A_i - (B_i - T)||^2}$$

Minimize with respect to T $\Rightarrow \partial_T \text{RMSD}(A,B) = 0$

$$\partial_T \sqrt{\frac{1}{N}\sum_i ||A_i - (B_i - T)||^2}$$

$$= \partial_T \sqrt{\frac{1}{N}\sum_i \left(\sqrt{\sum_i A_i - (B_i - T)}\right)^2}$$

$$= \partial_T \sqrt{\frac{1}{N}\sum_i A_i - (B_i - T)} = 0$$

$$\Rightarrow T = \frac{1}{N}\sum_i (B_i - A_i)$$

This is equivalent to moving the center of mass of A into the center of mass of B.

2 Molecular Dynamics – BALL

```
#include <BALL/FORMAT/PDBFile.h>
#include <BALL/STRUCTURE/peptides.h>
#include <BALL/STRUCTURE/peptideBuilder.h>
#include <BALL/STRUCTURE/fragmentDB.h>
#include <BALL/KERNEL/system.h>
#include <BALL/KERNEL/chain.h>
#include <BALL/KERNEL/protein.h>

using namespace std;
```

```
10 using namespace BALL;
11
12 int main(int argc, char* argv[]){
13
14   PDBFile sourceFile;
15   System mdSystem;
16   string url = "http://www.rcsb.org/pdb/files/";
17   string pdbid;
18
19   //Sanity checks for command-line arguments
20   if(argc == 2){
21     pdbid.append(argv[1] + string(".pdb"));
22     url.append(pdbid);
23   }else{
24     cout << "Wrong amount of Parameters\n\n Useage: prog pdbid" <<
       endl;
25     return 1;
26   }
27
28   // download pdb file (-N : if not exists or newer version online
       )
29   system((string("wget -N ") + url).c_str());
30   sourceFile.open(pdbid);
31   if(sourceFile.is_open()){
32     sourceFile.read(mdSystem);
33     sourceFile.close();
34   }
35
36   if (mdSystem.getProtein(0)){
37     Protein* protein = mdSystem.getProtein(0);
38     Chain* chain = protein->getChain(0);
39     String sequence = Peptides::GetSequence(*chain);
40     // test sequence of aminoacids
41     cout << sequence << endl;
42
43     // create new long sequence of aminoacids
44     String newSeq;
45     for (int i = 0; i < 10; i++){
46       newSeq.append(sequence);
47     }
48
49     Peptides::PeptideBuilder* pb = new Peptides::PeptideBuilder(
       newSeq);
50
51     FragmentDB fdb("");
52     pb->setFragmentDB(&fdb);
53
54     pb->setChainName("new_long_Chain");
55     pb->setProteinName("new_Protein");
```

```
56
57      Protein* newProt = pb->construct();
58
59      // test newProt
60      cout << Peptides::GetSequence(*newProt) << endl;
61
62      //System newSystem;
63      mdSystem.insert(*newProt);
64      PDBFile outFile(string("new_")+pdbid,ios::out);
65      outFile << mdSystem;
66      outFile.close();
67
68    }
69
70    return 0;
71  }
```

Blatt 10

SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Protein Docking

6/10

```
1
2
3  //Basic idea: dock one monomer after the other, buiding a longer
       chain with each docking.
4  //During each docking, check for clashes and pass the N (in the
       example its 4) possible
5  //candidates with the lowest energy without clashes to the next
       docking. This ensures
6  //that we get a wider array of possible solutions
7
8  define N 4 //number of possible candidates to pass on
9  define SIZE 5 //number of wanted size of monomer-1
10
11 ListOfDockedProtein dock2Proteins(base, add){
12   ListOfCandidates = execute rosetta base add;
13
14   ListOfEnergies = [];
15   for candidate in ListOfCandidates{
16
17     calculate energy;
18     check for clash;
19     if(clash in candidate){
20
21       set energy of candidate to +infinity;
22
23     }else{
24
25       set energy of candidate;
26
27     }
28
29   }
30
31   ListOfDockedProtein[N];
32   for energy in ListOfEnergies{
33
34     get N lowest energies of all possible energies;
35
36   }
37
38   return ListOfDockedProtein;
39
40 int main(arguments){
```

```
41
42  read monomer from file;
43
44  ListOfDockedProtein Current = dock2Proteins(monomer, monomer);
45
46  counter = 0;
47  repeat until counter = SIZE{
48
49    ListOfListOfDockedProteins = [];
50
51    i = 0;
52    for each protein in Current{
53
54      ListOfListOfDockedProteins[i] = dock2Proteins(protein,
    monomer);
55      i++;
56
57    }
58
59    ListOfDockedProtein Temp = [];
60    for each Protein in ListOfListOfDockedProteins{
61
62      check energies;
63      enter the four lowest energies into Temp;
64
65    }
66
67    Current = Temp;
68    counter++;
69
70  }
71
72  //Do a last energy check
73  for each protein in Current{
74
75    check energies;
76    return protein with lowest energy;
77
78  }
79
80 }
```

Clashes?

Not very practical if you consider many poses...

2 Score Calculation

10/10 Very, very good!

```
1  ////md.cpp
2  #include <BALL/FORMAT/PDBFile.h>
3  #include <BALL/KERNEL/selector.h>
4  #include <BALL/STRUCTURE/defaultProcessors.h>
5  #include <BALL/STRUCTURE/residueChecker.h>
6  #include <BALL/STRUCTURE/fragmentDB.h>
7  #include <BALL/KERNEL/protein.h>
8  #include <BALL/KERNEL/system.h>
9
10 using namespace BALL;
11 using namespace std;
12
13 int main(int argc, char** argv)
14 {
15
16   //Sanity checks for command-line arguments
17   if(argc != 3){
18     cout << "Wrong amount of Parameters\n\n Useage: ";
19     cout << "prog pdbin pdbout" << endl;
20     return 1;
21   }
22
23   // open a PDB file with the name of the first argument
24   PDBFile file(argv[1]);
25
26   // create a system and read the contents of the PDB file
27   System S;
28   file >> S;
29   file.close();
30
31   cout << "read " << S.countAtoms() << " atoms." << endl;
32
33   // prepare the protein by adding missing information
34   FragmentDB fragment_db("");
35   S.apply(fragment_db.normalize_names);
36   S.apply(fragment_db.add_hydrogens);
37   S.apply(fragment_db.build_bonds);
38
39   ResidueChecker checker(fragment_db);
40   S.apply(checker);
41
42   if (!(S.getMolecule(0))){
43     cout << "no molecule found" << endl;
44     return 1;
45   }
46
47   Molecule* m0 = S.getMolecule(0);
48
49   System Smin;
```

```
50  Smin.insert(*m0);
51
52  PDBFile outfile(argv[2], File::MODE_OUT);
53  outfile << Smin;
54  outfile.close();
55
56  // md simulation with gromacs
57  system(("pdb2gmx -ff amber03 -water none -f " + string(argv[2]))
      .c_str());
58  system("editconf -f conf.gro -bt cubic -d 0.5 -o box.gro");
59  // add solvent
60  // system(("genbox -cp box.gro -p topol.top -o " + string(argv
      [2])).c_str());
61  system("grompp -f ../em.mdp -p topol.top -c box.gro -o em.tpr");
62  system("mdrun -v -deffnm em");
63  system("trjconv -f em.gro -s em.tpr -o em.pdb");
64
65  // get scores with clusco
66  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s rmsd -o cl_rmsd.txt").c_str());
67  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s drmsd -o cl_drmsd.txt").c_str());
68  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s gdt -o cl_gdt.txt").c_str());
69  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s gdtExt -o cl_gdtExt.txt").c_str());
70  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s tmscore -o cl_tmscore.txt").c_str());
71  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s maxsub -o cl_maxsub.txt").c_str());
72  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s CMO -o cl_CMO.txt").c_str());
73  system(("~/clusco/build/src/./clusco_cpu -e em.pdb -t " + string
      (argv[1]) + " -s CMOn -o cl_CMOn.txt").c_str());
74
75  return 0;
76 }
```

```
1 //em.mdp
2 integrator  = md-vv
3 coulombtype    = pme
4 tcoupl     = nose-hoover
5 tc-grps    = protein
6 tau-t    = -1
7 ref-t    = 300
8 dt     = 0.0001
9 nsteps     = 10
```