Granular Fracture Model – Cluster Analysis

Mini Project

PH-312 – Bachelor of Technology, IIT Guwahati

Department of Physics – IIT Guwahati

Student Name

Priyam Bhavsar

Roll Number – 200121041

Under the Supervision of

Prof. Sitangshu Bikas Santra

Department of Physics

Indian Institute of Technology Guwahati

Acknowledgement

The path to achievement involves groundwork, diligence, and taking lessons from missteps. I express my sincere appreciation to Professor Dr. Sitangshu Santra for bestowing upon me the chance to work and acquire knowledge under his tutelage. I consider myself lucky to have received his expert supervision, valuable counsel, and guidance throughout the project, which enabled us to learn and shape the work we presented. His leadership, unwavering support, and recommendations were instrumental in making this project a reality. I would also like to extend my gratitude to the Department of Physics at IIT Guwahati for providing me with the opportunity to work on this Mini Project and contribute to making a significant impact on the world.

Certification of Guidance

I hereby certify that the project report entitled " Granular Fracture Model – Cluster Analysis" submitted by Priyam Bhavsar (Roll No: 200121041) to the Department of Physics, Indian Institute of Technology Guwahati, Assam, has been carried out under my supervision. Additionally, I confirm that the work presented in this report is a result of their independent investigation during the semester-long project work. As such, I recommend the submission of this report as partial fulfilment of the requirements for the Bachelor of Technology mini project for the current semester.

Signed by:

Prof. Sitangshu Bikas Santra

Professor

Department of Physics

Indian Institute of Technology, Guwahati

North Guwahati - 781039, Assam

Introduction

Rock fragmentation is a phenomenon of great interest in geology, engineering, and materials science. A solid rock will fracture into smaller pieces when put under stress that is too great to withstand. Rock fragmentation is a process that is still not fully understood, despite how important it is. In this term paper, we present and investigate a model of granular fracture that aims to imitate the dynamics of rock fragmentation.

According to the model, a rock is made up of interacting "grains" that change over time in accordance with Newtonian dynamics. The model has a history-dependent attractive potential between pairs of grains, one of its main characteristics. After the pair moves past a predetermined threshold distance, this potential is reset to zero, reflecting that once a crack has developed between two grains, it is unlikely to reappear.

Our research focuses on the system's cluster size distribution's distinctive features, which we contrast with those of the percolation problem. A mathematical framework called percolation theory explains how connected clusters in random networks behave. To understand the basic mechanisms underlying rock fragmentation, we compare the outcomes of our model to those predicted by percolation theory.

One of our study's main findings is a decline in the number of clusters with sample size. This observation agrees with earlier theoretical and experimental studies of rock fragmentation. In addition, we find a breakdown of the percolation theory property of hyper-scaling, which links the scaling exponents of various observables.

We also discover that critical exponents depend on the system's typical initial kinetic energy. According to this observation, the behavior of rock fragmentation may be significantly influenced by the energy input into the system.

Overall, our study presents a novel granular fracture model that captures the key characteristics of rock fragmentation. We improve our knowledge of the underlying mechanisms of rock fragmentation through our analysis of the distribution of cluster sizes and comparison with percolation theory.

Background

For many fields, including oil recovery, mining, and materials science, research on rock crack systems and the distribution of fragment sizes is crucial. Various mechanisms, from continuous explanations to lattice-type explanations, have been put forth to explain the development and spread of cracks. These models do not fully capture the dynamics of natural fragmentation, such as the correlations into the crack systems that appear natural.

As a result, we introduce a novel model in this paper that mimics the dynamics of natural fragmentation. We anticipate that our model will introduce correlations that appear natural in the crack systems and give details on the geometry of the crack. In contrast to what is possible on a lattice, we describe our model as a continuum, which may allow us to gather more geometrical data.

We will first examine the body of knowledge on crack systems and fragmentation models to accomplish our goals. Then, we'll outline our new model's theoretical underpinnings. The parameters used in the simulations and our methodology will be discussed next. The outcomes of our simulations will then be presented, along with information about the geometry of the crack systems and the distribution of fragment sizes. The relevance of our findings to various industries, including oil recovery, mining, and materials science, will be covered in our final discussion.

In summary, our research offers a fresh model to explain the dynamics of natural fragmentation in crack systems. We can better understand the behavior of rocks and materials under stress and improve our capacity to predict and control fracture propagation in various applications by comprehending the distribution of fragment sizes and the geometry of the crack systems.

Implementation

According to the granular fracture model (GFM), which assumes that visible grains make up rocks, two interactions between grains are hard-core while a continuous force gives the system cohesion. This study employs a numerical method to resolve Newton's equations of motion with an initial distribution of velocities in order to ascertain the fracture time and behavior of the grain system. The GFM process describes the evolution of the system, with particle scattering occurring when the distance between their centers is Rmin and a continuous force between particles that varies with their distance. Assuming that the grain has mass 1 (or m), the numerical solution employs a specific time step. Results of the study reveal a system of grains that cracks after 160 time steps, which is regarded as the fracture time. The strategy used in this study to comprehend the behavior of the grain system and establish the fracture time relies heavily on the GFM. The plan calls for using the force described in the model as a starting point for a numerical approach to solving Newton's equations of motion. Conclusive results are obtained by carefully choosing dimensionless parameters and free boundary conditions. Our knowledge of rock fracture has been aided by the findings, which offer insight into how grain systems behave.

We define the force equations based on the distance constraints between the particles:

$$F_{ij} = \begin{cases} \infty & \text{if } r_{ij} \leqslant R_{\min} \\ -[F_1(r_{ij} - R_0) - F_2(r_{ij} - R_0)^2]\hat{e}_{ij} & \text{if } R_{\min} < r_{ij} < R_{\max} \\ 0 & \text{otherwise} \end{cases}$$

In this equation, rij is the distance between the centres of the i and j grains, eij is a unit vector pointing from ri to rj, Ro (Rmin Ro Rmax) is the point at which the force changes from being repulsive to being attracted, Rmax is the point at which the force disappears, and F1, and F2 are constants. Due to short-range interactions on the grain surfaces, it is hypothesised that this force only acts between grains that were closest neighbours in the original lattice structure. Furthermore, once rl has increased past Rmax, it is history-dependent and is set to zero for all subsequent times. The interaction's "glue" quality illustrates how, once a macroscopic break has formed, it may be geometrically closed.

With the force represented in the equation above, we numerically solve Newton's equations of motion. Then, we perform a straightforward scattering method that

involves exchanging the velocities of the two scattered particles when rij = Rmin. We assume that every grain has the same mass and is situated on a square, 2D lattice with rij = R0 at time t = 0. The fundamental presumption here is that randomness eventually takes over, thus the initial square lattice—important as it is for numerical calculations, labelling, etc.—will have little bearing on the structure of fractures in the future.

Real materials are influenced by outside forces, which in turn cause certain random velocities to occur within the material. In order to account for that, we provide a distribution. The velocity distribution we choose for the particles is:

$$p(\boldsymbol{v}) = (\pi v_0^2)^{-1}\Theta(v_0 - |\boldsymbol{v}|)$$

Where the v0 is the cut off velocity that we will be studying and experimenting on.

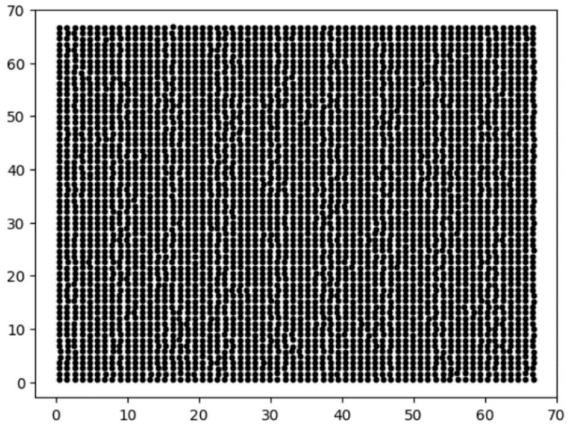
We set the parameters as F1 = 100, F2 = 50, Rmax = 1.1, Rmin = 1.0, R0 = 1.05. We assume the mass of the grain to be 1. The time step assumed for this simulation follows the condition:

$$10v0\Delta t \ \leq Rmax - Rmin$$

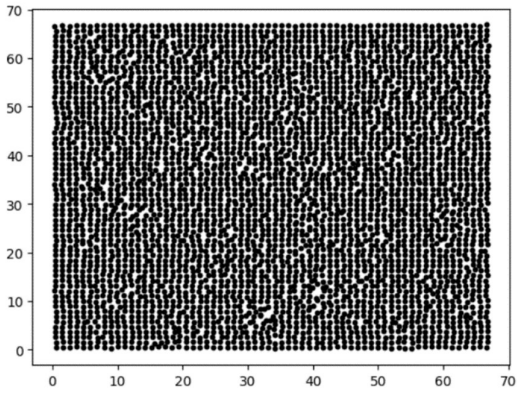
The process outlined above is how the assembling of grains evolved through time.

Output of the above formulation:

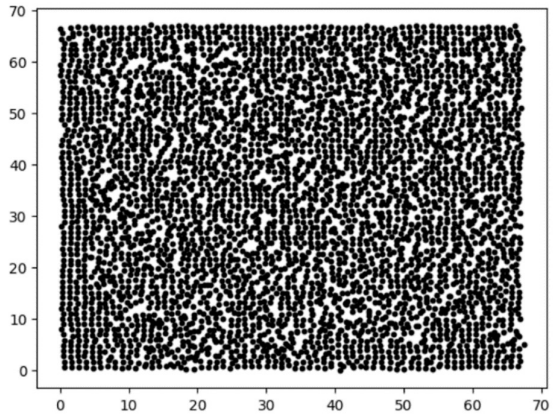
The simulation is performed at T = 160 timesteps and the state of the system is shown at various instances.



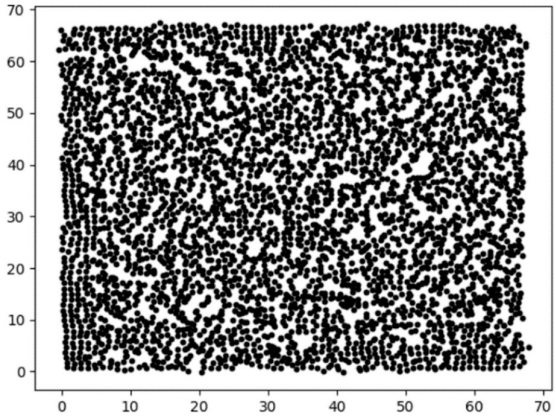
In panel 1, we observe the system's state at time T = 10, where the particles tilt slightly. This is due to the distribution of velocities between the particles. At this stage, cracks have already started appearing in the system, and these small cracks will eventually lead to larger ones.



In Figure (b), the system is at time T = 50, where the particles have considerable motion. This movement causes cracks in the system, and the distance between the particles increases beyond the initial limit we set at R0.

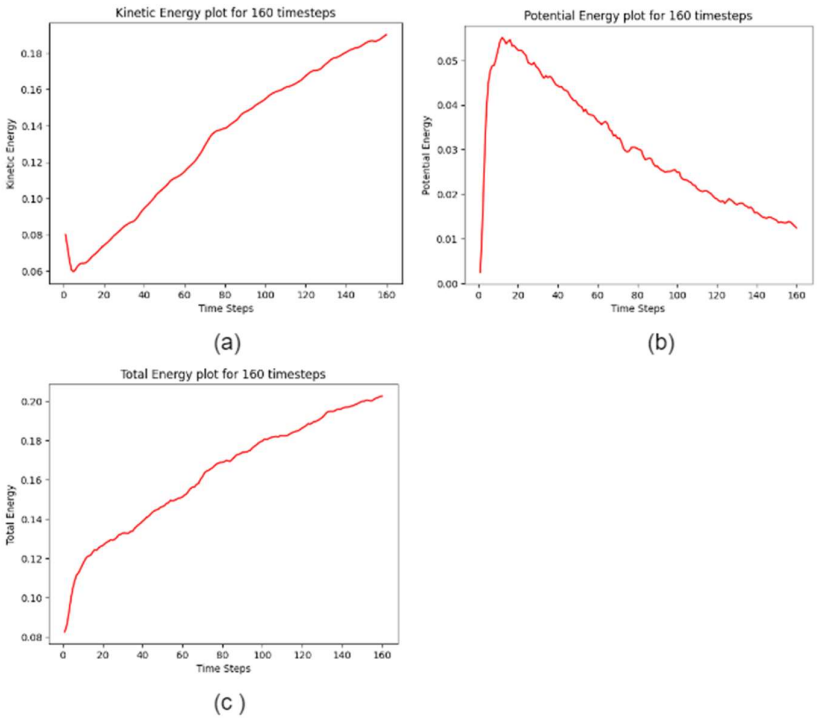


We have passed the halfway initialization at step T = 100 (Figure c). The system transitions from forming small cracks in the 20th century to the formation of apparently large cracks. Cracks have grown to define clusters in the system, resulting in exponential particle aggregation.



Finally, in Figure (d), we observe the system's final state at time T = 160. There is a clear difference between T = 10 and T = 160, as the cracks already define the clusters of the system leading to an exponential. particle agglomeration.

The plots for energy at timestep T = 160 are:



The figure above shows how the energy of the system changes at each time step.

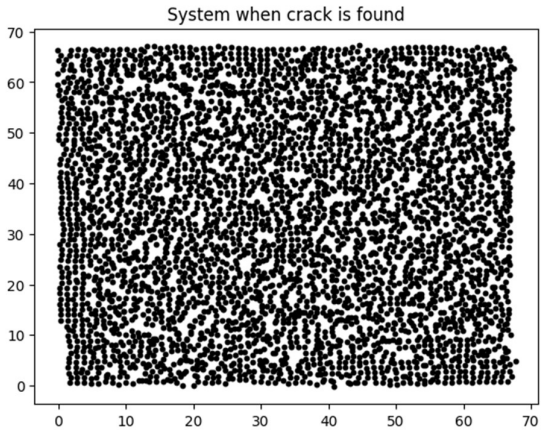
The plots were obtained by averaging over 160 independent simulations. Figure a) Shows the kinetic energy of the system, which is proportional to the temperature. Kinetic Energy decreases as particles lose speed due to collisions and attractions. Kinetic energy reaches a minimum when the particles are close together and form a particle stable configuration. Kinetic energy increases with particle velocity due to repulsive forces and thermal fluctuations.

Figure (b) shows the potential energy of the system, which is proportional to the intermediate particle power. The potential energy increases as the particles approach each other and experience a strong attraction. When the potential energy reaches a maximum The particles are at equilibrium distances and form stable configurations. this one As the particle travels on and survives less attractive.

Figure (c) shows the total energy of the system, which is equal to the sum of the kinetic energies and potential energy. Due to the numerical error of the integration algorithm

The total energy increases with each time step. The total energy is also reflected nonlinear behavior of kinetic and potential energies oscillating around their balance sheet value.

Now we determine the instance when the crack had propagated in the system.



This is the instance when the crack was detected by the program. This is the timestep T = 133 out of 160.

Using the cluster search algorithm which involves in the burning of connections with the nearest neighbours we calculate the number of clusters. We get 535 independent cluster formations with minimum cluster size as 1 and maximum cluster size as 353.

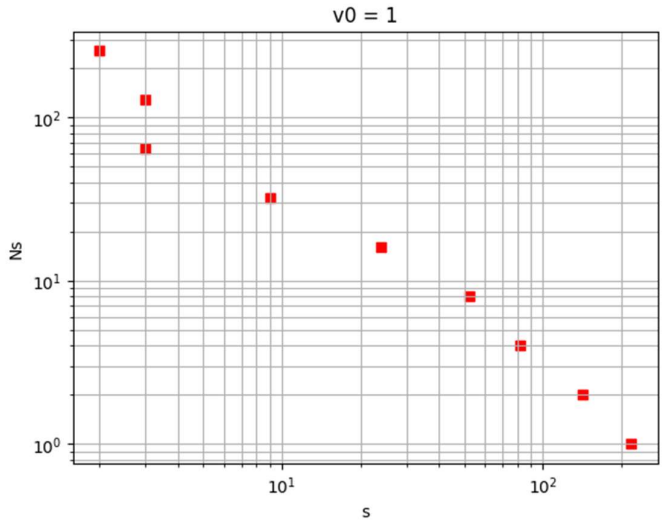
Due to the large number of clusters we use binning to separate out the clusters and place them in bins. We model the bins in the range of exponents of 2.

$$2^i \leq s \leq 2^{i+1}$$

Where i is an integer. We use this to calculate the number of clusters per bin and relate it to the size of the bin itself.

$$N_s(tc) = \sum_{s=2^i}^{s=2^{i+1}} n_s(t_c) \quad for\ s = 2^{i+1} - 1$$

We get the following graph :



This is a graph with both the graphs log-log plotted. This means that the relation between Ns and s is in the form of exponent:

$$N_s(t_c) \propto s^{-(\tau-1)}$$

The slope of the graph is the exponential relation between Ns and s. We get the exponential relation as:

$$\tau - 1 = 1.0475 \pm 0.01$$

The value we obtain is different from the value of percolation but we get this due to the fact that there is significant difference in the size between the simulation one our scale and the simulation on a very large scale.

It has been shown that the binning procedure used does not affect the results of a particular analysis. Specifically, if the division is performed at equal intervals of 10, the obtained slope values will still be almost the same as those obtained by the current division process. This means that the results of the analysis are stable and do not depend on the choice of the specific binning method.

The claim that the results are independent of the grouping procedure means that the choice of separation method does not affect the overall conclusions of the analysis. In other words, no matter which binning method is used, the same trend or correlation between the analyzed variables will be observed. This is an important feature of data and analytical methods, as it provides confidence in the robustness of the results and the validity of the conclusions drawn.

The Fractal Dimension

To analyze and compare the fractal dimension of percolation with that of the grain fracture model (GFM), we exploited the position of the particles on the initial grid of the GFM. Computing fractal dimension involves determining the average radius of gyration of the clusters at a particular bin, which we denote as Rs.

The process of calculating the fractal dimension involves first identifying the grain clusters in the GFM. We then calculated the radius of gyration of each cluster, which is a measure of the cluster's size and shape. The radius of gyration, Rs, was computed using the following equation:

$$R_s = \frac{1}{N}\sum_{i=1}^{N} r_i^2$$

where N is the number of grains in the cluster, ri is the distance of the ith grain from the cluster's center of mass.

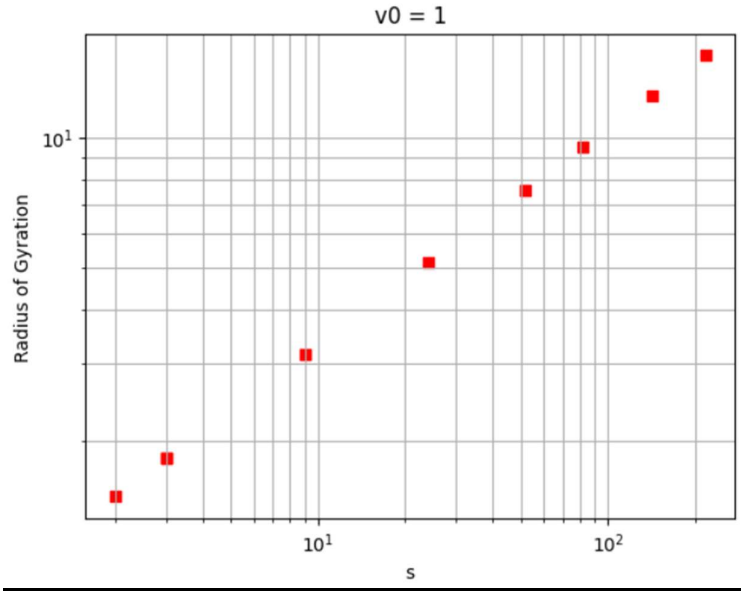
After calculating the gyration radius of each cluster, we plotted the logarithm of Rs against the cluster size window. The slope of the resulting graph is taken as an estimate of the fractal dimension of the GFM cluster. It should be noted that the fractal dimension calculated using this method is only approximate and may be affected by factors such as container size and GFM initial conditions. However, by using the mean gyration radius of the cluster, we can gain insight into the size and shape of the GFM clusters and compare them to the leaky clusters.

For fractal dimension, the Mass of the grains will be proportional to the radius to the fractal power

$$M_s \propto R_s^D$$

Where D is the fractal dimension.

We perform this analysis on our system and we find the following graph:



The slope of the graph is used to determine the dependence of D with mass and radius of gyration. From the graph we get the slope as:

$$\frac{1}{D} = 0.50 \pm 0.02$$

Thus the value of D comes out to be:

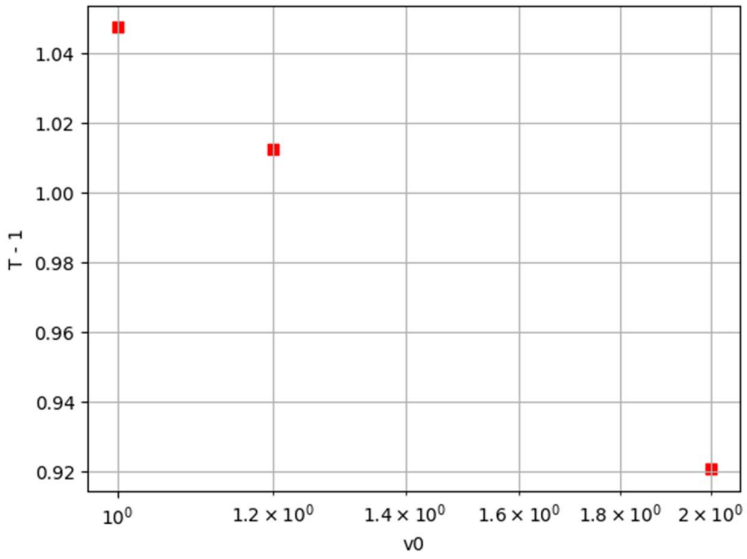
$$D = 1.99 \pm 0.07$$

Which is very close to the actual value of D = $\frac{91}{48}$.

Variance due to different velocity

In the context of the study, shear rate refers to the maximum rate at which grains can collide without breaking the bonds between them. The authors were interested in checking whether the results obtained in their simulations were consistent across different thresholds. To investigate this, they repeated the simulation with different cut-off values and plotted the exponent T-I against the cut-off value, as shown in Figure below. The plot shows that the exponential T-I peaks at a cut-off value of about 1 and that the T-I values approach the percolation values at very high and very low cutting speeds. In the case of high initial velocity, the effect of grain interaction is small and the velocity distribution is random. This results in the breaking of a permeant bond, a phenomenon that can also be observed in other physical processes, such as the splitting

of atomic nuclei into lighter nuclei when they are hit by high-energy charges. Overall, the results show that the results obtained in the study depend on the cutting speed parameter. The T-I behavior varies with different shear rates, but the percolation values are close at very high and very low shear rates. These results significantly affect the understanding of the dynamics of granular materials and the role of various parameters in their behavior.



The above graph shows the value of T – 1 for different values of velocity.

We can see that the graph peaks at v0 = 1 and it keeps on decreasing with increase in velocity. Same would be the case with velocity less than one. The graph peaks in the form of an extended delta function at v0 = 1.

Conclusion

In this study, the authors propose CFM as a model for modeling rock fragmentation. They showed that fragment analysis at a given time I, when the first fracture passes through the system, can be performed in the same way as the percolation method. The criticality exponent obtained from CFM was found to be different from that obtained by percolation. However, scaling is still observed but requires the dependence of n, the number of clusters, on the system size while hyperscaling is stopped. In addition, the authors note that the critical exponent depends on the kinetic energy of each particle of the system, i.e. cutting speed. The most significant penetration deviation occurs at v0=1. It is not clear whether the critical exponent has a continuous dependence on v0 or whether the observed exponent is simply the effective exponent, as there is a cross between the percolation exponent, which is valid for almost all v0, and a special exponent generated outside v0 = 1.

The results of this study reveal the behavior of rock fragmentation, and the differences in criticality exponents derived from percolation and CFM highlighted the need for a specific model of rock fragmentation. The dependence of the critical exponent on the kinetic energy of each particle means that model parameters must be carefully chosen to ensure accurate results. Further studies are needed to determine whether the indices are indeed persistently dependent on shear rate, or whether the observed indices are simply effective indices resulting from the crossover of different regimes. Overall, CFM is a useful tool for studying rock fragmentation and can be extended to other physical systems that fragment.

Basic information on what rock fragmentation is and why it is important in mining and engineering can be added to develop and expand this article. Also explain what percolation is and how it relates to rock fragmentation. It is also possible to give some examples of other models used to simulate rock fragmentation and compare them with CFM. It is also possible to discuss some limitations and challenges of using CFM and suggest some possible improvements or future directions.

References

1] U Naftaly et al. 1991 J. Phys. A: Math. Gen. 24 L1175

2] Zhou, Wei Regueiro, Richard Wang, Qiao Chang, Xiaolin. (2016). Modeling the fragmentation of rock grains using computed tomography and combined FDEM. Powder Technology. 308. 10.1016/j.powtec.2016.11.046.

3] Gang Ma, Wei Zhou, Richard A. Regueiro, Qiao Wang, Xiaolin Chang, Modeling the fragmentation of rock grains using computed tomography and combined FDEM, Powder Technology, Volume 308

Appendix

This section shows the C code for this simulation

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "C:\Users\priya\Documents\Important stuff\6th sem\Sem - 6\try\MD\1-rn.c"

#define l0 64
#define L (l0*R0)
#define npart 4096
#define R0 1.05
#define sqrtn pow(npart,0.5)
#define tmax 10000
#define densi pow(sqrtn/(1.05*(sqrtn + 1)),2)
#define temp 0.5
#define v0 0.9
#define box pow(npart/densi,0.5)
#define dt 0.01
#define Rmin 1.0
#define Rmax 1.1


int neigh1[npart],neigh2[npart][4],neigh3[npart][4];
int NNcrack[npart][4];
int lattice[npart];
int ncl,t;
double vir,vir_sum,P,tmp;
double fx[npart],fy[npart];
double T0,rho,V,hL;
double x[npart],y[npart];
double xi[npart],yi[npart];
double xcm,ycm,xcm0,ycm0;
double xm[npart],ym[npart];
double vx[npart],vy[npart];
int find_crack();
int burn_nn();
void starting_position();
void create_neighbour();
void update_nearest_neighbors();
double force();
double integrate();
int burn_nn();
void crackspan();
double print_spanningcrack();
void printlatindex();
void CPC();


int main(void){

    clock_t t1,t2;
    t1 = clock();
```

```
    // creating the pointer variable to the files
    FILE *file1, *file2, *file3, *file4, *file5, *fp1, *fp2;

    file1 = fopen("posi.dat", "w"); //opening the files
    file2 = fopen("blender.xyz", "w");
    file3 = fopen("posf.dat", "w");
    file4 = fopen("Energy_1.dat", "w");
    file5 = fopen("withbond.dat", "w");
    fp1=fopen("FILE1.dat","w");
    fp2 = fopen("FILE2.dat","w");


    // variables
    double x[npart], y[npart], vx[npart], vy[npart], fx[npart], fy[npart],
vvx[npart], vvy[npart];

    // allocate memory for other arrays
    double Ep=0.0,Ec=0.0;
    for(int i=0;i<npart;i++){
    x[i]=0; y[i]=0; vx[i]=0; vy[i]=0; fx[i]=0; fy[i]=0; vvx[i]=0; vvy[i]=0;  //
zeroing of vectors
        }



    // initiating position
    starting_position();
    create_neighbour();

    fprintf(file2,"%d \n",npart);
    fprintf(file2,"Particle Center \n");
    for(int j=0;j<npart;j++){
        fprintf(file2,"Default      %lf      %lf      0 \n",xi[j],yi[j]);   //
blender
        fprintf(file1,"%lf      %lf\n",xi[j],yi[j]);    // initial position
        }

    int crackflag = 0;
    double sumv2,sumvx,sumvy;
    int i,j,ibond=0;
    for(int t=1;t<=tmax;t++){
        force();
        integrate();
        update_nearest_neighbors();
    if(crackflag==0){
    crackflag=find_crack(t);
    if(crackflag==1){
    int ncl=burn_nn();
    printf("%e %d\n",t*dt,burn_nn());
        //print_state_withbond_Lammpsdata(itime);
    printf("Crack found\n");
    for(int j=0;j<npart;j++){
        fprintf(file3,"%lf      %lf\n",xi[j],yi[j]);    // pos final
    }
    crackspan(t,ncl,NNcrack);
    for(i=0;i<npart;i++)
    for(j=0;j<4;j++)
        if(NNcrack[i][j]>0)
```

```
            fprintf(fp2,"%d %d %d\n",i*4+j+1,NNcrack[i][j],i+1);
            ibond++;
    print_spanningcrack(t,ncl,NNcrack);

    printf("%e crack = %d\n",t*dt,find_crack(t));
  fclose(fp1);
  fclose(fp2);
return 0;
      }
    }


        vir_sum+=vir;
        Ep = force();
        Ec = integrate();


        fprintf(file4,"%d      %lf      %lf \n",t,Ep,Ec);  // energy

    }


    // print_spanningcrack(t,ncl,NNcrack);
    // double temp2 = print_spanningcrack(t,ncl,NNcrack);


    t2 = clock();
    printf("%f",(((double)t2 - (double)t1) / 1000000.0F ));
    return 0;
}


//square grid
void starting_position(){
    int i,j,l3;
    double ix,iy,iz;
    double sumvx,sumvy,sumvz,sumv2,fs;


    for(i=0;i<npart;i++){
      x[i]=R0*(i/64)+R0/2.;
      y[i]=R0*(i%64)+R0/2.;
      xi[i]=x[i];    yi[i]=y[i];    //storing x(t=0)
      xcm0+=xi[i];    ycm0+=yi[i];    //cm for t=0;
    }

    xcm0/=npart;  ycm0/=npart; //Center of mass for the particles

    sumv2=0;sumvx=0;sumvy=0;sumvz=0;
    //uniform velocity distribution
    for (i=0;i<npart;i++){
      vx[i]=(ran2(&iseed)-0.5)*10;
      vy[i]=(ran2(&iseed)-0.5)*10;
    }
 }


void create_neighbour(){
```

```
  int i,j;
  double xr,yr,zr,r,r2;
  for(i=0;i<npart-1;i++){
    for(j=i+1;j<npart;j++){
      xr=xi[i]-xi[j];
      yr=yi[i]-yi[j];


        if (xr>=0.5*box){ xr=xr-box; }
        if (xr<=-0.5*box){ xr=xr+box; }
        if (yr>=0.5*box){ yr=yr-box; }
        if (yr<=-0.5*box){ yr=yr+box; }


      r2=xr*xr+yr*yr;
      r=sqrt(r2);
      if(r<R0+0.01){
    neigh2[i][neigh1[i]]=j;
    neigh2[j][neigh1[j]]=i;

    neigh3[i][neigh1[i]]=1;
    neigh3[j][neigh1[j]]=1;

    neigh1[i]++;
    neigh1[j]++;
      }

    }
  }

}

void update_nearest_neighbors(){

    int i,j,k;
    double xr,yr,zr,r,r2;

    for(i=0;i<npart;i++){
    for(k=0;k<neigh1[i];k++){

      if(neigh3[i][k]==1){
        j=neigh2[i][k];
        xr=xi[i]-xi[j];
        yr=yi[i]-yi[j];

        if (xr>box/2)       xr-=box;
        else if (xr<-box/2) xr+=box;
        if (yr>box/2)       yr-=box;
        else if (yr<-box/2) yr+=box;

        r2=xr*xr+yr*yr;
        r=sqrt(r2);

        if(r>1.1)neigh3[i][k]=0;
      }
    }
  }
```

```
}

// interaction force between particles
double force(){
    double ene=0.0;
    int i,j,k;
    double xr,yr,zr,r,r2,ff = 0,Ep;
    for(i=0;i<npart;i++){
        fx[i]=fy[i]=0.0;
    }
    for(i=0;i<npart;i++){
        for(k=0;k<neigh1[k];k++){

        if(neigh3[i][k]>0){

            j=neigh2[i][k];

            xr=xi[i]-xi[j];
            yr=yi[i]-yi[j];

        if (xr>box/2)       xr-=box;
        else if (xr<-box/2) xr+=box;
        if (yr>box/2)       yr-=box;
        else if (yr<-box/2) yr+=box;

        r2=xr*xr+yr*yr;
        r=sqrt(r2);

        if(r-R0<10e-5){ // scattering: exchnage velocity
        tmp=vx[i];vx[i]=vx[j];vx[j]=tmp;
        tmp=vy[i];vy[i]=vy[j];vy[j]=tmp;
    }

    if(r>Rmin && r<Rmax){
      double rR0=r-R0;
      ff=-(100*rR0-50*rR0*rR0*rR0)/r;

      fx[i]=fx[i]+ff*xr;
      fx[j]=fx[j]-ff*xr;

      fy[i]=fy[i]+ff*yr;
      fy[j]=fy[j]-ff*yr;

      double e=(100*0.5*rR0*rR0-50*rR0*rR0*rR0*rR0/3.0);
      ene=ene+e;
      vir+=ff*r2;
    }

    }

    }
  }
    Ep=ene/(double)npart;
    return Ep;
}

// integration of particle positions
double integrate(){
```

```
    for(int i=0;i<npart;i++){
        vx[i]+=0.5*dt*fx[i];
        vy[i]+=0.5*dt*fy[i];
        xi[i]+=vx[i]*dt+0.5*fx[i]*dt*dt;
        yi[i]+=vy[i]*dt+0.5*fy[i]*dt*dt;
    }


    force();



    for(int i=0;i<npart;i++){
        vx[i]+=0.5*dt*fx[i];
        vy[i]+=0.5*dt*fy[i];
    }

    double sumv2,sumvx,sumvy,Ec;
    for (int i=0;i<npart;i++){
        sumvx+=vx[i];
        sumvy+=vy[i];
        sumv2+=(vx[i]*vx[i]+vy[i]*vy[i]);
    }
        Ec = 0.5*(sumv2/npart);
        return Ec;

}

// periodic boundary condition
void CPC(){
    for(int i=0;i<npart;i++){
        if (xi[i]>box/2)       xi[i]-=box;
        else if (xi[i]<-box/2) xi[i]+=box;
        if (yi[i]>box/2)       yi[i]-=box;
        else if (yi[i]<-box/2) yi[i]+=box;
    }
}

int  burn_nn()
{
  static int i,j,k,k1,k2,n,max=0;
  static int iroot,inow,rroot,sum=0;
  static double z;


  int *ip,*ns,*lattice;
  double *size;
  ip = malloc(npart* sizeof(int));
  ns = malloc(npart * sizeof(int));
  lattice = malloc(npart * sizeof(int));
  for(i=0;i<npart;i++)lattice[i]=0;

  size = malloc(npart * sizeof(double));
  for(i=0;i<npart;i++)size[i]=0;

  int ncl=0;
  for(i=0;i<npart;i++){//lattice scan for burning
    if(lattice[i]==0){
```

```
      ncl++;
      lattice[i]=ncl;
      n=1;
      ip[n]=i;
      k1=n;
      k2=k1;
    rr:
      for(j=k1;j<=k2;j++){
    iroot=ip[j];

    for(k=0;k<neigh1[iroot];k++){
      if(neigh3[iroot][k]==1){

        inow=neigh2[iroot][k];

        if(lattice[inow]==0){
          lattice[inow]=ncl;
          n++;
          ip[n]=inow;
        }
      }
    }
      }
      k1=k2+1;
      k2=n;
      if(k1<=k2)goto rr;
      if(n>max)max=n;
      ns[n]++;
      //printf("n=%d\n",n);
      for(k=1;k<=n;k++)
    ip[k]=0;

    size[ncl]=n;
    }//if(lattice[i]==0)
  }//end lattice scan

    FILE *fpc;
    FILE *fpp;
  char FILE1[50];
    int
sum1=0,sum2=0,sum4=0,sum8=0,sum16=0,sum32=0,sum64=0,sum128=0,sum256=0,sum512 = 0;
    int sum_1,sum_2,sum_4,sum_8,sum_16,sum_32,sum_64,sum_128,sum_256,sum_512 = 0;
    int
count1=0,count2=0,count4=0,count8=0,count16=0,count32=0,count64=0,count128=0,count2
56=0,count512 = 0;
    double
rad1=0,rad2=0,rad4=0,rad8=0,rad16=0,rad32=0,rad64=0,rad128=0,rad256=0,rad512 = 0;

  fpc=fopen("FILE3.dat","w");
  fpp=fopen("FILE_10.dat","w");
  for(i=1;i<=ncl;i++){
    if(size[i]>0){
        fprintf(fpc,"%f\n",size[i]);
      if((size[i]) ==1)
      {

        sum1 = sum1 + size[i];
        count1++;
```

```
      }
      else if (size[i] < 4 && size[i] >= 2 )
      {
        sum2 = sum2 + size[i];
        count2++;
      }
      else if (size[i] < 8 && size[i] >= 4 )
      {
        sum4 = sum4 + size[i];
        count4++;
      }
      else if (size[i] < 16 && size[i] >= 8 )
      {
        sum8 = sum8 + size[i];
        count8++;
      }
      else if (size[i] < 32 && size[i] >= 16 )
      {
        sum16 = sum16 + size[i];
        count16++;
      }
      else if (size[i] < 64 && size[i] >= 32 )
      {
        sum32 = sum32 + size[i];
        count32++;
      }
      else if (size[i] < 128 && size[i] >= 64 )
      {
        sum64 = sum64 + size[i];
        count64++;
      }
      else if (size[i] < 256 && size[i] >=128 )
      {
        sum128 = sum128 + size[i];
        count128++;
      }
      else if (size[i] < 512 && size[i] >=128 )
      {
        sum256 = sum256 + size[i];
        count256++;
      }


  }}
//   sum_2 = sum2 - sum1;
//   sum_4 = sum4 - sum_2;
//   sum_8 = sum8 - sum_4;
//   sum_16= sum16 - sum_8;
//   sum_32 = sum32 - sum_16;
//   sum_64= sum64 - sum_32;
//   sum_128 = sum128 - sum_64;
//   sum_256 = sum256 - sum_128;

rad1 = pow(count1,0.5)*R0;
rad2 = sqrt(count2*(1.05)*(1.05));
rad4 = sqrt(count4*(1.05)*(1.05));
rad8 = sqrt(count8*(1.05)*(1.05));
```

```
rad16 = sqrt(count16*(1.05)*(1.05));
rad32 = sqrt(count32*(1.05)*(1.05));
rad64 = sqrt(count64*(1.05)*(1.05));
rad128 = sqrt(count128*(1.05)*(1.05));
rad256 = sqrt(count256*(1.05)*(1.05));


  fprintf(fpp,"1    %d      %d      %f\n",sum1,count1,rad1);
  fprintf(fpp,"2    %d      %d      %f\n",sum2,count2,rad2);
  fprintf(fpp,"4    %d      %d      %f\n",sum4,count4,rad4);
  fprintf(fpp,"8    %d      %d      %f\n",sum8,count8,rad8);
  fprintf(fpp,"16   %d      %d      %f\n",sum16,count16,rad16);
  fprintf(fpp,"32   %d      %d      %f\n",sum32,count32,rad32);
  fprintf(fpp,"64   %d      %d      %f\n",sum64,count64,rad64);
  fprintf(fpp,"128  %d      %d      %f\n",sum128,count128,rad128);
  fprintf(fpp,"256  %d      %d      %f\n",sum256,count256,rad256);


fclose(fpp);
  fclose(fpc);

  return ncl;
}

int  find_crack(int t)
{
  static int i,j,k,k1,k2,n,max=0;
  static int iroot,inow,rroot,sum=0;
  static double z;
  int l1=64-1,ll1=64*l1,spannedcrack=0;

  int NNcrack[npart][4];
  for(i=0;i<npart;i++)
    for(j=0;j<4;j++)
      NNcrack[i][j]=0;


  int xcls[npart],ycls[npart];
  for(i=0;i<npart;i++){xcls[i]=0;ycls[i]=0;}


  int *ip,*ns,*lattice,*size;
  ip = malloc(npart * sizeof(int));
  ns = malloc(npart * sizeof(int));
  lattice = malloc(npart * sizeof(int));
  for(i=0;i<npart;i++)lattice[i]=0;

  size = malloc(npart * sizeof(int));
  for(i=0;i<npart;i++)size[i]=0;

  //Here I do bond percolation analysis of broken bond . If the span
  //of the cluster O(L): system spanning crack found and the function
  //return 1 else 0.
  for(i=0;i<npart;i++)lattice[i]=0;
    //{
    //  lattice[i]=-1;
    //  for(k=0;k<neigh1[i];k++){
    //  if(neigh3[i][k]==0)lattice[i]=0;
```

```
  //  }
  //}

int startcounting;
int ncl=0;
for(i=0;i<npart;i++){//lattice scan for burning

  startcounting=0;
  for(k=0;k<neigh1[i];k++)
    if(neigh3[i][k]==0)startcounting=1;


  if(lattice[i]==0 &&  startcounting==1){
    ncl++;
    lattice[i]=ncl;
    n=1;
    ip[n]=i;
    k1=n;
    k2=k1;

    xcls[i]=0;ycls[i]=0;

    //printf("ncl=%d\n",ncl);
    //printf("xstart=%d ystart=%d\n",xstart,ystart);

  rr:
    for(j=k1;j<=k2;j++){
  iroot=ip[j];

  for(k=0;k<neigh1[iroot];k++){
    //for(k=0;k<4;k++){
    //inow=iroot+iv[k];
    //if(iroot%l0==l1 && k==0)inow=iroot-l1;
    //if(iroot/l0==l1 && k==1)inow=iroot-ll1;
    //if(iroot%l0==0 && k==2)inow=iroot+l1;
    //if(iroot/l0==0 && k==3)inow=iroot+ll1;

    inow=neigh2[iroot][k];

      if(lattice[inow]==0 && neigh3[iroot][k]==0){
        lattice[inow]=ncl;
        n++;
        ip[n]=inow;

        if(k==0){
      xcls[inow]=xcls[iroot]+1;
      ycls[inow]=ycls[iroot];
        }
        if(k==1){
      xcls[inow]=xcls[iroot];
      ycls[inow]=ycls[iroot]+1;
        }
        if(k==2){
      xcls[inow]=xcls[iroot]-1;
      ycls[inow]=ycls[iroot];
        }
        if(k==3){
      xcls[inow]=xcls[iroot];
```

```
    ycls[inow]=ycls[iroot]-1;
      }

    }//if lattice[inow]=0



    //storing crack information
    if(lattice[inow]==ncl && neigh3[iroot][k]==0)
      NNcrack[iroot][k]=ncl;


}//loop over neighbour
  }
  k1=k2+1;
  k2=n;
  if(k1<=k2)goto rr;
  if(n>max)max=n;
  ns[n]++;

  for(k=1;k<=n;k++)
ip[k]=0;

  size[ncl]=n;


  //printf("ncl=%d n=%d\n",ncl,n);

  int xmin=0,xmax=0,ymin=0,ymax=0;
  for(k=0;k<npart;k++){

if(lattice[k]==ncl){
  //printf("x[k]=%d y[k]=%d\n",xcls[k],ycls[k]);
  if(xcls[k]<xmin)xmin=xcls[k];
  if(xcls[k]>xmax)xmax=xcls[k];

  if(ycls[k]<ymin)ymin=ycls[k];
  if(ycls[k]>ymax)ymax=ycls[k];
}
  }


  //printf("xmin=%d ymin=%d\n",xmin,ymin);
  //printf("xmax=%d ymax=%d\n",xmax,ymax);
  //printf("xdiff=%d ydiff=%d\n",xmax-xmin,ymax-ymin);


  if(xmax-xmin>=64 || ymax-ymin>=64)spannedcrack=1;




  //printf("ncl=%d\n",ncl);
  //printlatindex(lattice);
```

```
    }//if lattice[]=0
  }//end lattice scan

//   printlatindex(lattice);

  //printf("ncl=%d\n",ncl);

  if(spannedcrack)
    {
          printlatindex(lattice);
      crackspan(t,ncl,NNcrack);


  }
  return spannedcrack;
}

void printlatindex(int lattice[])
{
    FILE *fpn;
    fpn= fopen("FILE20.dat","w");
  int i,j;
  int
count1,count2,count4,count8,count16,count32,count64,count128,count256,count512 = 0;
  for(i=64-1;i>=0;i--){
    for(j=0;j<64;j++)
      fprintf(fpn," %d\n",i,lattice[i*64+j]+1);
  }

  fclose(fpn);
}
void crackspan(int t, int ncl, int NNcrack[][4])
{
    FILE *fp2;
    fp2 = fopen("FILE2.dat","w");
    int i,j,ibond = 0;
    for(i=0;i<npart;i++)
    for(j=0;j<4;j++)if(NNcrack[i][j]>0)
        fprintf(fp2,"%d %d %d\n",i*4+j+1,NNcrack[i][j],i+1);
        ibond++;
    fclose(fp2);
}
double print_spanningcrack(int t,int ncl,int NNcrack[][4])
{
  FILE *fp1;
  char FILE1[50];
  int i,j,ibond=0;

  for(i=0;i<npart;i++)
    for(j=0;j<4;j++)if(NNcrack[i][j]>0)ibond++;



  fp1=fopen("FILE.dat","w");
  fprintf(fp1,"%s LAMMPS data file\n","#");
  fprintf(fp1,"%d atoms\n",npart);
```

```
  fprintf(fp1,"%d bonds\n",ibond);
  fprintf(fp1,"1 atom types\n");
  fprintf(fp1,"%d bond types\n",ncl+1);
  fprintf(fp1,"0.0 %e xlo xhi\n",L);
  fprintf(fp1,"0.0 %e ylo yhi\n",L);
  fprintf(fp1,"0.0 0.001 zlo zhi\n");

  fprintf(fp1,"\n");
  fprintf(fp1,"Atoms %s bond\n","#");
  fprintf(fp1,"\n");
    for(i=0;i<npart;i++)
    fprintf(fp1,"%d 1 1 %g %g\n",i+1,xi[i],yi[i]);


  fprintf(fp1,"\n");
  fprintf(fp1,"Bonds\n");
  fprintf(fp1,"\n");


  for(i=0;i<npart;i++)
    for(j=0;j<4;j++)
      if(NNcrack[i][j]>0)
    fprintf(fp1,"%d %d %d %d\n",i*4+j+1,NNcrack[i][j],i+1,neigh2[i][j]+1);
  fclose(fp1);
}
```

Miniproject_Priyam_Bhavsar_200121041

ORIGINALITY REPORT

13%	9%	11%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	U Naftaly, M Schwartz, A Aharony, D Stauffer. "The granular fracture model for rock fragmentation", Journal of Physics A: Mathematical and General, 1991 Publication	4%
2	svn.monitord.de Internet Source	1%
3	cpp.hotexamples.com Internet Source	1%
4	research.coe.drexel.edu Internet Source	1%
5	eprints.kfupm.edu.sa Internet Source	1%
6	manualzz.com Internet Source	1%
7	hrcak.srce.hr Internet Source	1%
8	people.sc.fsu.edu Internet Source	<1%

9	Martin Oliver Steinhauser. "Computer Simulation in Physics and Engineering", Walter de Gruyter GmbH, 2013 Publication	<1 %
10	"Direct Simulation Monte Carlo", Computational Granular Dynamics, 2005 Publication	<1 %
11	CAGLAR, ATTILA, and MICHAEL GRIEBEL. "ON THE NUMERICAL SIMULATION OF FULLERENE NANOTUBES: $C_{100.000.000}$ AND BEYOND !", Molecular Dynamics On Parallel Computers, 2000. Publication	<1 %
12	www.cognex.com Internet Source	<1 %
13	Hoover, . "Computer Programming", Advanced Series in Nonlinear Dynamics, 2006. Publication	<1 %
14	raw.githubusercontent.com Internet Source	<1 %
15	www.uniroma2.it Internet Source	<1 %
16	Ruth Chabay, Bruce Sherwood, Aaron Titus. "A unified, contemporary approach to teaching energy in introductory physics", American Journal of Physics, 2019 Publication	<1 %

17	www.utupub.fi Internet Source	<1 %
18	acikbilim.yok.gov.tr Internet Source	<1 %
19	epdf.pub Internet Source	<1 %
20	hal.archives-ouvertes.fr Internet Source	<1 %
21	root.cern Internet Source	<1 %
22	www.creatis.insa-lyon.fr Internet Source	<1 %
23	www.scribd.com Internet Source	<1 %
24	Masanori Hanada, So Matsuura. "MCMC from Scratch", Springer Science and Business Media LLC, 2022 Publication	<1 %
25	boiling.seas.ucla.edu Internet Source	<1 %
26	users.df.uba.ar Internet Source	<1 %
27	www.slideshare.net Internet Source	<1 %

28	www2.fizik.usm.my Internet Source	<1%
29	"Computational Granular Dynamics", Springer Science and Business Media LLC, 2005 Publication	<1%
30	open.ptit.edu.vn Internet Source	<1%
31	theses.gla.ac.uk Internet Source	<1%

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		

Miniproject_Priyam_Bhavsar_200121041

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28