

forceMat (generic function with 1 method)

```

1 #####calculate force matrix#####
2 function forceMat(parPosTmp::Matrix{Float64},Rs::Float64)
3   parPos=deepcopy(parPosTmp);
4   δh=0.0005;
5   dimension=3;
6   dimension2=2;
7   rndDigNo=9;
8   pNo=length(parPos[:,1]);
9   kMat=Matrix{Float64}(undef,pNo*dimension2,pNo*dimension2);
10  for iTmp in 1:pNo
11    for jTmp in 1:dimension2
12      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]+2*δh;
13      f1=round.(forcePackHigh(Rs,parPos); digits=rndDigNo);
14      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]-2*δh;
15      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]+δh;
16      f2=round.(forcePackHigh(Rs,parPos); digits=rndDigNo);
17      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]-δh;
18      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]-2*δh;
19      f3=round.(forcePackHigh(Rs,parPos); digits=rndDigNo);
20      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]+2*δh;
21      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]-δh;
22      f4=round.(forcePackHigh(Rs,parPos); digits=rndDigNo);
23      parPos[iTmp,jTmp]=parPos[iTmp,jTmp]+δh;
24      kk1=(-f1.+8*f2.-8*f4.+f3)/12/δh;
25      if dimension2==dimension
26        kk2=copy(kk1);
27      else
28        kk2=Array{Float64}(undef,pNo*dimension2);
29        countTmp=1;
30        for kTmp in 1:pNo*dimension
31          if kTmp%3==0
32            continue
33          else
34            kk2[countTmp]=kk1[kTmp];
35            countTmp+=1;
36          end
37        end
38      end
39      kMat[:,(iTmp-1)*dimension2+jTmp]=kk2;
40    end
41    println("particle "*string(iTmp)*" done")
42  end
43  return kMat
44 end

```

sweepRs (generic function with 1 method)

```

1 #####sweep particle radius and output force matrix to file#####
2 function sweepRs(parPosTmp::Matrix{Float64},Rs::Float64)
3 parPos=deepcopy(parPosTmp);
4 dRs=0.0001;
5 println("Input Data file name and press Enter:");
6 fileName=readline();
7 outputData=open(fileName*".txt","w");
8 totRndNo=10;
9 rndDigNo=9;
10 for iTmp in 1:totRndNo
11     write(outputData,"round"*string(iTmp));
12     println("round"*string(iTmp));
13     write(outputData,"\n");
14     write(outputData,"Rs: "*string(Rs));
15     write(outputData,"\n");
16     write(outputData,"initial positions: ");
17     write(outputData,"\n");
18     writedlm(outputData,parPos,'\t');
19     write(outputData,"\n");
20     parPos=md(parPos,Rs);
21     write(outputData,"equilibrium positions: ");
22     write(outputData,"\n");
23     writedlm(outputData,parPos,'\t');
24     write(outputData,"\n");
25     kMat=forceMat(parPos,Rs);
26     write(outputData,"force matrix: ");
27     write(outputData,"\n");
28     writedlm(outputData,kMat,'\t');
29     println(kMat);
30     write(outputData,"\n");
31     egv=eigen(kMat);
32     write(outputData,"eigenvalues: ");
33     write(outputData,"\n");
34     writedlm(outputData,round.(egv.values;digits=rndDigNo),'\t');
35     println(round.(egv.values,digits=rndDigNo));
36     write(outputData,"\n");
37     write(outputData,"eigenvectors: ");
38     write(outputData,"\n");
39     writedlm(outputData,round.(egv.vectors;digits=rndDigNo),'\t');
40     write(outputData,"\n");
41     write(outputData,"\n");
42     Rs=Rs+dRs;
43     flush(outputData);
44 end
45 close(outputData);
46 return 0;
47 end

```

