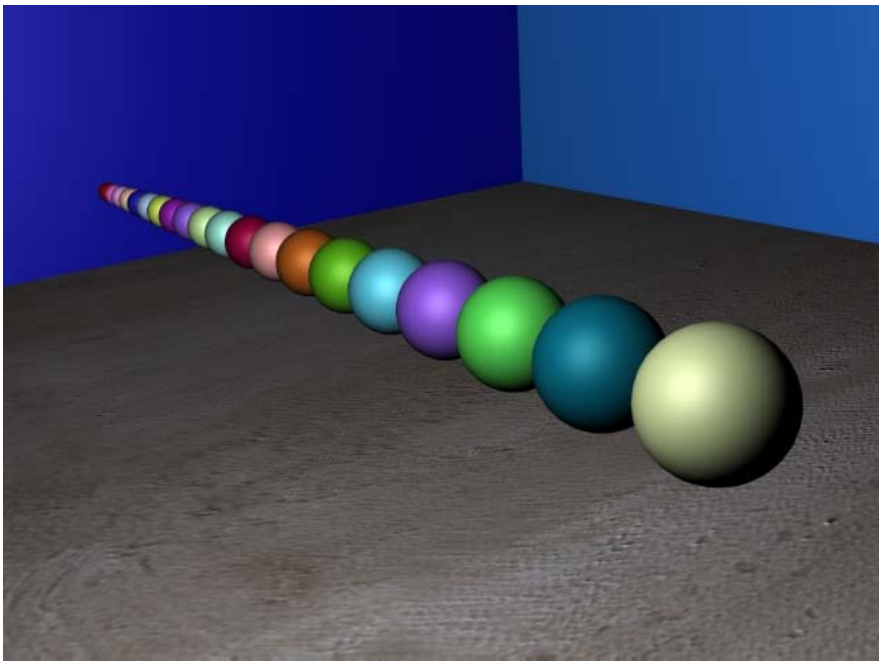
Taper Chain 2009

A quick reference



Adam Sokolow

1. Short Introduction:

The purpose of this program is to allow for the easy manipulation of the Taper Chain code with no C++ knowledge. The program now takes in a file of parameters, and from those values, performs the corresponding numerical simulation, including what outputs are desired.

2. The Basics:

Technically, no "parameter.txt" file is needed for the program to run. In this case, however, the program will execute with the default parameters which may or may not give desirable or even useful information. This is a common feature of the program and should be noted that **if a parameter is absent from the parameter file, a default value is used**.

Here is a sample "parameter.txt" file, with the first few basic variables:

parameter.txt
N: 9 q: 1 w: 0.0

This parameter file tells the program to run with 9 Grains, with a tapering of 1%, and no restitution (dissipation).

Any keyword in the parameter file must be followed by a colon (:) for the interpreter to understand what it is you would like it to do.

The keyword interpreter is case insensitive. This means that "n:" and "N:" are equivalents.

Other keywords that follow this same format are as follows:

~~D~~	~~A Material Parameter (mm^2/N)~~ [no longer works May 2009]
dt	Time step used by Velocity Verlet Algorithm (1e-5 musec default)
LargeInitV	Large Grain (one on Left) Initial Velocity (mm/musec)
Nsteps	Number of steps (of size dt) in the Simulation
exponential	The exponential of the Potential (added: July 2007, in v.2)
alpha	The shape parameter (added: May 2009)
precision	The decimal precision of the output files (integer)
Rho	The Density of the Material (mg/mm^3)
poisson	The default Poisson Ratio (0,0.5] - replaces D - May 2009
youngs	The default Youngs Modulus (kN/mm^2) - replaces D May 2009
Rlarge	The Radius of the Large Grain (mm)

SmallInitV	The Small Grain Initial Velocity (mm/musec)
timeMax	The Last Time Desired in the Output (musec)
timeMin	The First Time Desired in Output (musec)
timeSpits	The time step at which output is generated (musec)
preload	The initial loading force of the chain (kN) added October 2008

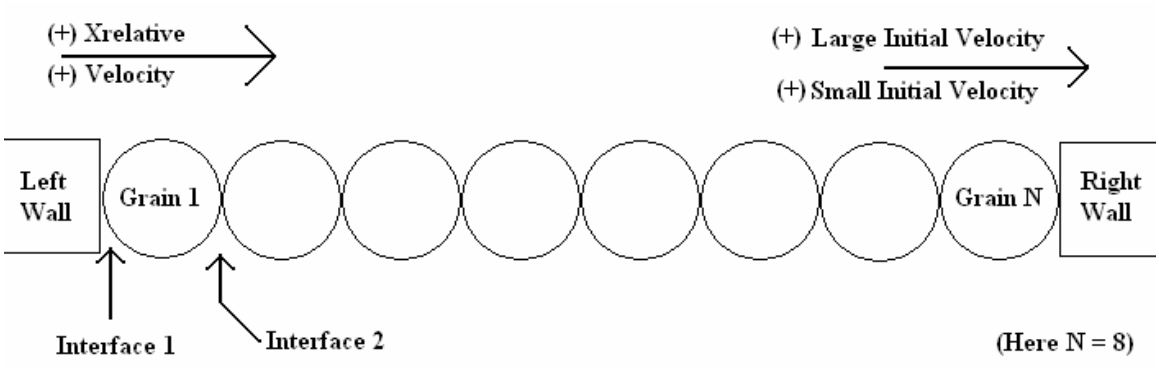
Exponential and Alpha set the same parameter just in different ways - the last one used in the parameter text file is the one that will be used in the computation.

3. Conventions:

There are some basic conventions used in this program. The first are the units of measurement:

Length	(mm)
Time	(musec) 1e-6s
Velocity	(mm/musec)
Mass	(mg)
Energy	(J)
Force	(kN)

Second there are direction/counting conventions:



Note: If we want Grain N to be set to initially head toward the chain, we need to assign it a negative velocity.

The Left/Right Walls used to take the limit as R-> infinity, but now they are set to the grain radius that is next to them. This allows for a more generic definition of the wall being a grain of infinite mass, but not necessarily flat. - May 2009.

3. String-String Commands:

There are two "String-String" commands, this simply means the keyword is followed by a string instead of a decimal value. The two commands are: **files**, and **filename**.

The **Files** command lets you specify what output files you would like the code to generate for you. The available files include A Relative Position file, a Velocity file, a Kinetic Energy file, a Force file, Total System energy, Force Applied file (FAf added May 2007).

The use of the command is as follows:

files: XVKFTA

Will provide you with all 6 output files. **No Spaces** are allowed between the letters, however **not all letters need to be used,** and their **order does not matter.** The default value for this command is all output files on.

The **filename** command allows you to change the "secondary root name" of the files created. For example, instead of the program creating a file called "Force_.dat" it could create "ForceN16q3w0.01.dat".

The use of this command is as follows:

filename: N16q3w0.01

Once again no spaces are allowed in the string.

3. The Wall Command:

In some runs it may be desired for there to only be one wall, or no walls at all, this can be achieved by using the **wall** command.

Four values are allowed for the wall command, they are:

11
01
10
00

A "1" represents a wall turned on, and a "0" a wall that is off. So "11" means left wall on, and right wall on, where as "10" means left wall on, right wall off.

For example:

Wall: 01

Results in a right wall only.

3. The Grains Command:

The **grains** command allows you to specify the range of the grains to be outputted. This can be very useful if you have a chain of over 100 grains and you only care about the central interactions. The grains command is followed by **two natural numbers** {1,2,3...N}, which specify the **range of the grains to be outputted in all the output files**.

Use of this command is as follows:

Grains: 5 10

This example will only output the information for grains 5, 6, 7, 8, 9, 10 (except in the total system energy file, where all grain information is used).

One grain can be specified by the command

Grains: 5 5

Note: In the Force file, forces are recorded at the interfaces of grains, so specifying "Grains: 5 5" Will provide you with the **force felt at the Interfaces between grains** 4&5, and 5&6.

Also: **If a grain you are specifying is an edge grain, but the wall it is next to is turned off, only one force will be reported for this grain.**

Do Not Specify a Range Larger than the Number of Grains

4. Delta Velocity Function:

The purpose of the **deltav** is to specify a particular grain to have at a specified **time** a specific **additional velocity.** Note this function will introduce a discontinuity into the simulation if used for any times after zero.

Its use is as follows:

deltav: 2 0.5 0.1
deltav: 8 0.5 -0.1

This gets interpreted as Grain 2, at Time = 0.5 mu-sec gets an additional velocity of 0.1, and Grain 8 gets at that same time, the opposite velocity.

Note: There is a redundancy in syntax here. A command (if we assume 10 grains) of: "deltav: 10 0 -0.1" can replace the command "smallinitv: -0.1" However if you convert in the next simulation to a new number of grains, they will no longer be equivalents.

Note: If you specify both a deltav function *and* a SmallInitV or LargeInitV, they will be added if the deltav corresponds to time zero.

5. AddForce Command (added in May 2007 Version 1 and Later):

The purpose of the "AddForce:" command is to directly apply a force to the Left Most Grain. Use of the "AddForce:" command is as follows:

AddForce: c 2.3 4 3.14

That is to say "AddForce: char # # #". **Writing multiple AddForce commands results in their addition.**

There are **two forces** that have slightly different syntax than the others. All functions described below in the table:

Command Symbol	**Meaning**	**Usage**	**Translation**
s	sine	Amp Freq Phase	Amp*sin(Freq*t-Phase)
c	cosine	Amp Freq Phase	Amp*cos(Freq*t-Phase)
t	triangle wave	Amp Freq Phase	
w	saw wave	Amp Freq Phase	
q	square wave	Amp Freq Phase	
r	random force	Max Min Freq	
k	constant force	Value	

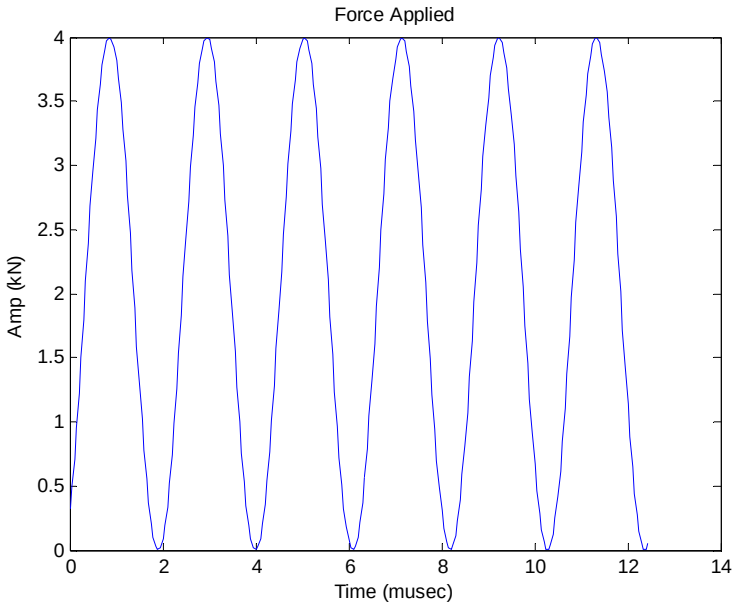
All Frequencies are defined with 2*Pi as the default period. So a frequency of 2 means two full waves in a period of 2Pi. This goes for all the waves.

The random force has a refresh frequency as specified, this way the random force is not changed every time step, but instead the user has some control over how frequently it gets changed to a new value.

So for example:

addforce: s 2 3 1
addforce: k 2

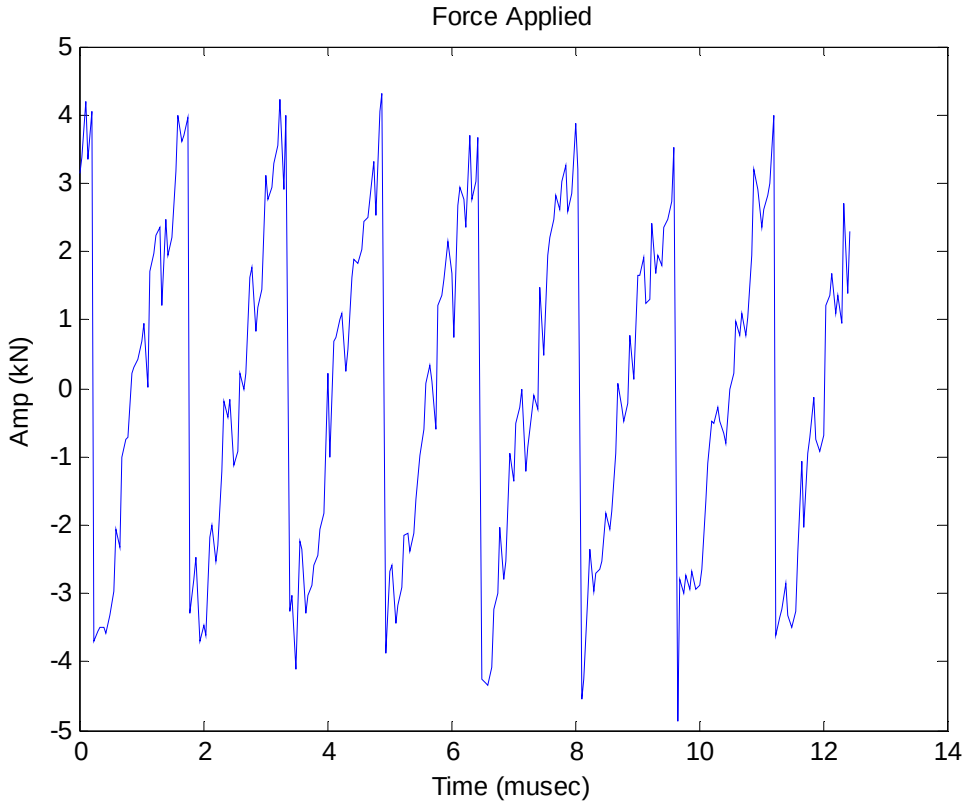
Adds the force F(t) = 2 + 2*sin(3*t-1), which you can verify in the DrivingForce file:



Or as another example:

addforce: r 1 -1 100
addforce: w 4 4 1

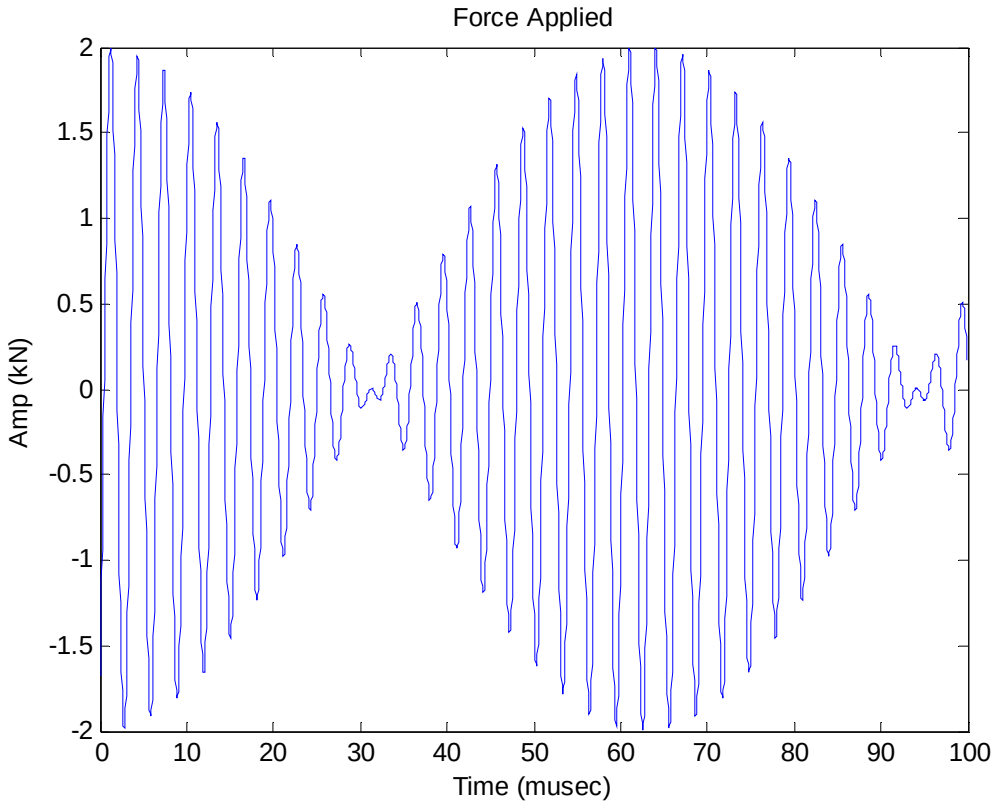
Adds a random force between -1 and 1 with a frequency of 100, to a Saw Wave with amplitude 4, frequency 4, and phase delay 1.



Another example:

addforce: s 1 2 1
addforce: s 1 2.1 1

Adds two sine functions that have slightly different frequencies, and we see the beats show up in the force applied file:



6. Chain Patterning and Subunit Commands (Added In the May 2009 Version):

The purpose of the "ChainPattern:" command along with the "Subunit:" command are to allow for a complete control of how a chain will be structured. For instance if the program user wanted to make a chain of steel alternating by brass beads and repeat this for 200 grains, this would require two 'subunit' commands to define a brass and a steel subunit, followed by the specification of the pattern of these two, ie alternating sequence repeated 100 times.

The "ChainPattern:" command is a symbolic way of doing arithmetic where the syntax is extremely important to get right, or the code will not work as you expect.

The important characters for a chain pattern are: the vertical line |, the pound symbol #, the left and right braces [,], and the hat ^.

The vertical line - this is used to separate subunits in your chain pattern string.

The pound sign # - this is used to specify how many of these subunits to put in the chain (not specified means 1)

The braces [] - these symbols must be used together to specify a grouping of a pattern.
The hat ^ - this is used in conjunction with the braces [], to specify how many times the grouped pattern will be repeated

Let's start easy and work our way up in complexity of the chain patterning.
The first example is if you wanted 5 steel grains, followed by 5 brass (where we assume that you have defined the subunits separately [to be explained later]). You would use the following command:

Example 1:

ChainPattern: steel#5|brass#5

This is literally translated to: "steel steel steel steel steel brass brass brass brass brass"

The important thing is that when we change subunits in the pattern, we need to include a vertical line |. The first thing the program looks for is the name. Any character before the # becomes the name of the subunit.

Example 2:

ChainPattern: thisnameismeanttobeincrediblylong#2|A#2

Translates: "thisnameismeanttobeincrediblylong thisnameismeanttobeincrediblylong a a"

You can assign the subunit any name you want, as long as it is one word - so you can make it as meaningful as 'steel' or as vague as 'a'.

Example 3:

ChainPattern: A|B#2|C

Translates: a b b c

You do not always have to use the pound sign - if you omit it, the program assumes you mean repeat once. As you can see above from Example 3, that both 'a' and 'c' are inserted in the chain once while 'b' is repeated twice. Example 4 shows how to use the pattern repeat command.

Example 4:

ChainPattern: [A|B^4]

Translates: a b a b a b a b

Here we can see that we've told the code we want the pattern a|b to be repeated four times. Note we did not use the pound sign, so each was assumed to be wanted only once, and note the braces location surrounding the pattern to be repeated. The power 4 applies to the braces it is inside.

Example 5:

ChainPattern: [A#2|B|C#2^3]

Translates: a a b c c a a b c c a a b c c

With this said, you can only use the braces to repeat one level, ie this example will fail:

Example 6:

ChainPattern: [[A|B^2]^2]

Translates: FAILURE

But you can use multiple pattern repeats:

Example 7:

ChainPattern: [A#2|B^2]|[A|C#2^3]

Translates: a a b a a b a c c a c c a c c

Example 8:

ChainPattern: [A|B^2]|C|[B|A^2]

Translates: a b a b c b a b a

The chain pattern command needs to be used in conjunction with the subunit command. If no subunits are specified the code uses the default values - so you will have just made a monodisperse chain of length equal to the length you wanted.

Specifying a subunit is easy, you need to first name the subunit (and this name must match the pattern name - but capitalization does not matter) - then you have the option of specifying its Density, Poisson Ratio, Youngs Modulus, its 'Width' Parameter, its Volume, and its RadiusPlus, RadiusMinus (where plus and minus refer to right or left side of grain).

Example: 9

ChainPattern: A#20
subunit: A|rho#2.0|youngs#0.01|poisson#0.3

Makes a monodisperse chain with radius equal to 'rlarge' where each subunit, called 'a' has density (rho) of 2.0(mg/mm^3), a Youngs Modulus of 0.01(kN/mm^2), and a Poisson Ratio of 0.3.

Similar to the chain pattern command, the subunit command requires that each field being specified needs to be separated by a vertical line |.

Example: 10

ChainPattern: A#20
subunit: A|rho#2.0|youngs#0.01|poisson#0.3|radiusminus#4|radiusplus#6|widthparam#2

Makes a chain with radii on the minus side (left) equal to 4mm, and a right side (plus) equal to 6mm, and an extra volume is added by specifying a cylindrical width of 2mm. where each subunit, called 'a' has density (rho) of 2.0(mg/mm^3), a Youngs Modulus of 0.01(kN/mm^2), and a Poisson Ratio of 0.3.

Example: 11

ChainPattern: A#20
subunit: A|radiusminus#4|radiusplus#6|volume#20

Makes a subunit where the radii parameters don't matter in the volume calculation. A volume is specified which is independent of the radii, so the mass of the grain can be tuned to avoid unwanted anomalies from a light grain in a chain due to different geometries. The other way of achieving the same thing is to play with the width parameter until the volume is what you want - though it is in general easier if you know the volume you want to just specify it. These patterns and subunits will show up in the readme file in a way that hopefully is understandable.

A standard tapered chain for example now shows up in the readme file as:

Tapering percent: 5
Restitution: 0
Epsilon (1-w): 1
Radius of Large Particle: 5 (mm)
Radii(-/a/+) of all Particles: (mm)
(5/0/5) (4.75/0/4.75) (4.5125/0/4.5125) (4.286875/0/4.286875)
(4.07253125/0/4.07253125) (3.868904687/0/3.868904687)
(3.675459453/0/3.675459453) (3.49168648/0/3.49168648)
(3.317102156/0/3.317102156) (3.151247049/0/3.151247049)
(2.993684696/0/2.993684696) (2.844000461/0/2.844000461)

(2.701800438/0/2.701800438) (2.566710416/0/2.566710416)
(2.438374896/0/2.438374896) (2.316456151/0/2.316456151)
(2.200633343/0/2.200633343) (2.090601676/0/2.090601676)
(1.986071592/0/1.986071592) (1.886768013/0/1.886768013)

Masses of all Particles: (mg)
2314.306588 1984.228611 1701.228005 1458.590361 1250.558911 1072.197946
919.2757141 788.1640154 675.7521227 579.3729762 496.7399055
425.8923764 365.1494763 313.0700322 268.4184189 230.1352419
197.312203 169.17055 145.0426003 124.3558995

Or a patterned chain as:

chainpattern: [A|B^2][C][B|A^2]
subunit: A|radiusminus#4|radiusplus#6|volume#2
subunit: B|radiusminus#6|radiusplus#4|volume#2
subunit: C|radiusminus#4|radiusplus#4|volume#3

Radius of Large Particle: 5 (mm)
Radii(-/a/+) of all Particles: (mm)
(4/0/6) (6/0/4) (4/0/6) (6/0/4) (4/0/4) (4/0/6) (6/0/4) (4/0/6) (6/0/4)
Youngs of all Particles: (kN/mm^2)
69.83796001 69.83796001 69.83796001 69.83796001 69.83796001 69.83796001
69.83796001 69.83796001 69.83796001
Poisson of all Particles: (-)
0.35 0.35 0.35 0.35 0.35 0.35 0.35 0.35 0.35
Masses of all Particles: (mg)
8.84 8.84 8.84 8.84 13.26 8.84 8.84 8.84 8.84
Interface Prefactor of all Particles: kN/(mm)^(xn-1)
47.14633336 57.74222998 47.14633336 57.74222998 47.14633336 47.14633336
57.74222998 47.14633336 57.74222998 47.14633336

You specified a patterned chain, which I've understood to be:
a b a b c a b a b

You specified a subunit definition, which I've understood to be (zeros mean you didn't specify, so I used the simulation defaults):
Subunit Name: a
R+: 6
R-: 4
a: 0
volume: 2
rho: 0
poisson: 0
youngs: 0

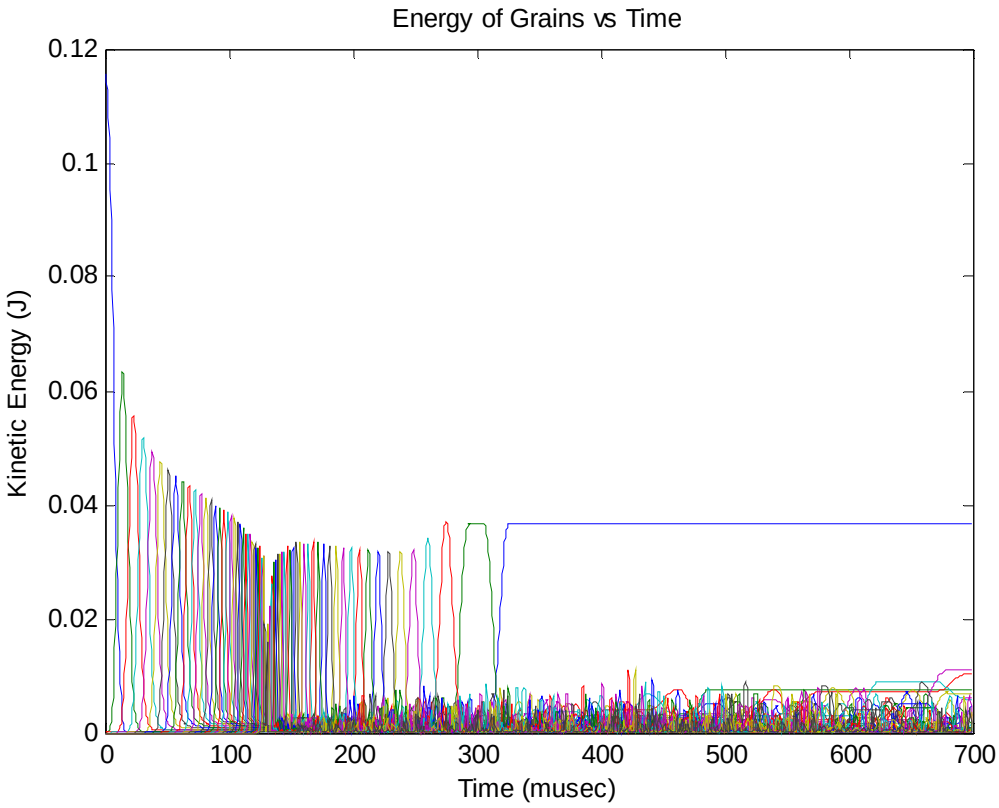
Subunit Name: b
R+: 4
R-: 6
a: 0
volume: 2
rho: 0
poisson: 0
youngs: 0

Subunit Name: c
R+: 4
R-: 4
a: 0
volume: 3
rho: 0
poisson: 0
youngs: 0

7. Sample Program Results/Use:

Using the following "parameter.txt" file:

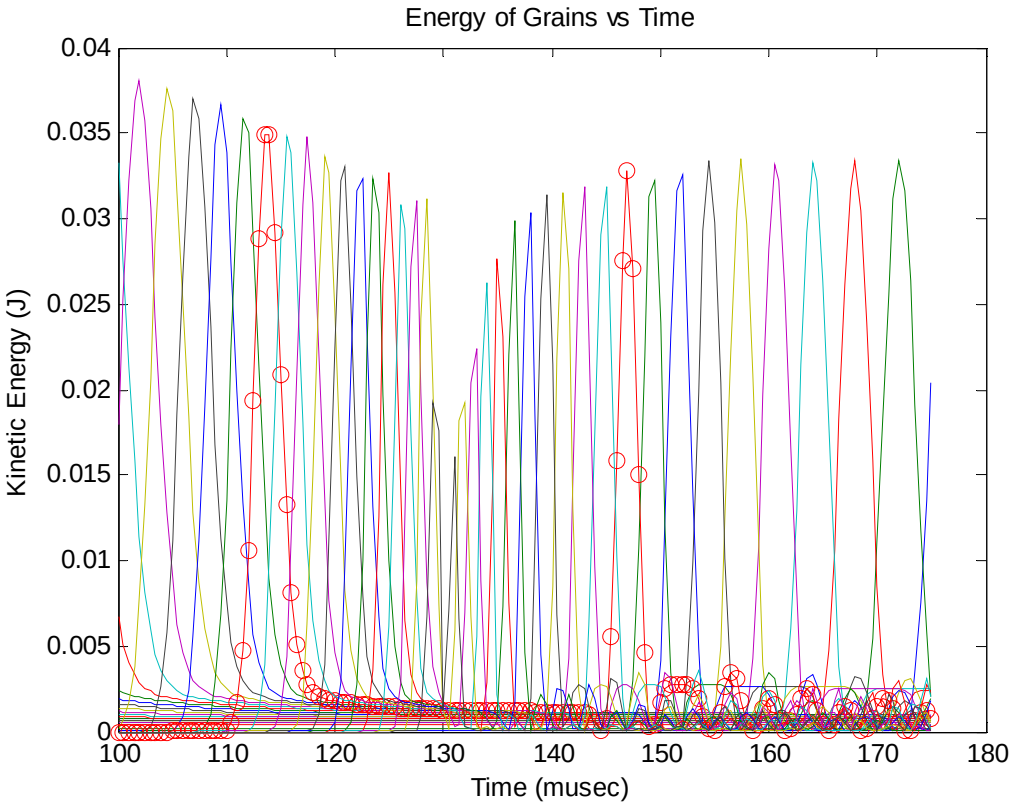
```
N: 35
q: 5
dt: 0.01
nsteps: 70000
files: K
rho: 4.42
D: 0.01206
wall: 01
rlarge: 5
timeSpits: 0.5
SMALLINITV: 0.0
LargeInitV: 0.01
filename: N35
```



This run, for argument sake is too long, and we only care about the region where we collided with the wall. So now we can specify the lower and upper time limits. With our new parameter file:

N: 35
q: 5
dt: 0.01
nsteps: 17500
files: K
rho: 4.42
D: 0.01206
wall: 01
rlarge: 5
timeSpits: 0.5
SMALLINITV: 0.0
LargeInitV: 0.01
filename: N35
timemin: 100
timemax: 175

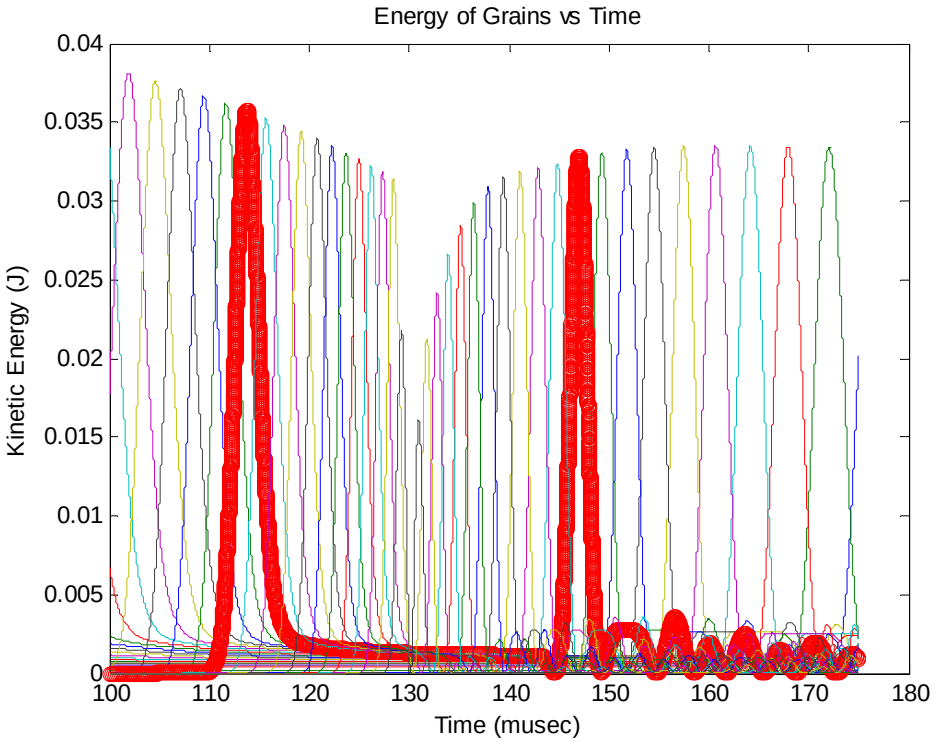
With the new result:



From this plot it is obvious that the number of points used is insufficient, and as we want stable results we must also shrink our stepsize dt.

N: 35
q: 5
dt: 0.0001
nsteps: 1750000
files: KT
rho: 4.42
D: 0.01206
wall: 01
rlarge: 5
timeSpits: 0.005
SMALLINITV: 0.0
LargeInitV: 0.01
filename: N35
timemin: 100
timemax: 175

Resulting in:



With the following Total System Energy plot, which shows conservation of energy: