

Modeling mechanical properties of single layer graphene sheet

Graphene:

Graphene is an allotrope of carbon in the form of a single layer of atoms in a two-dimensional hexagonal lattice in which one atom forms each vertex.

Graphene has a special set of properties which set it apart from other allotropes of carbon. In relation to its thickness, it is about 100 times stronger than the strongest steel. Yet its density is dramatically lower than any steel, with a surfacic mass of 0.763 mg per square meter. It conducts heat and electricity very efficiently and is nearly transparent.

Mechanical Properties:

Graphene is the strongest material ever tested, with an intrinsic tensile strength of 130 GPa (19,000,000 psi) and a Young's modulus (stiffness) of 1 TPa (150,000,000 psi). The Nobel announcement illustrated this by saying that a 1 square meter graphene hammock would support a 4 kg cat but would weigh only as much as one of the cat's whiskers, at 0.77 mg (about 0.001% of the weight of 1 m² of paper).

Graphene is also relatively brittle, with a fracture toughness of about 4 MPa√m. This indicates that imperfect graphene is likely to crack in a brittle manner like ceramic materials, as opposed to many metallic materials which tend to have fracture toughness in the range of 15–50 MPa√m.

Objective:

- Graphene is known for its strength, toughness and less weight. In this report we try to predict 'Fracture Strain', 'Fracture Strength' and 'Young's Modulus', the three important mechanical properties, using the parameters of graphene 'X0Y1', '% defect', 'strain rate in armchair direction' and 'Temperature'.

Implementation:

Step1 - Dataset:

Browse dataset of 'Single layer Graphene sheet' with parameters and Mechanical properties for training Machine Learning algorithm on it.

raw_data.csv - in this folder is the Raw data containing 'X0Y1', 'Defect', 'Strain rate', 'Temperature', 'Fracture strain', 'Fracture strength' and 'Young's Modulus' as columns of 1440 samples single layer Graphene sheets.

	A	B	C	D	E	F	G
1	X0Y1	Defect	StrainRate	Temperature	FractureStrain	FractureStrength	YoungsModulus
2	0	0	5E-05	1	0.17712	98.877191	973.289014
3	0	0	5E-05	1	0.176935	98.882547	973.408565
4	0	0	5E-05	1	0.17632	98.862375	973.365905
5	0	0	5E-05	1	0.17683	98.868643	973.412343
6	0	0	5E-05	1	0.176565	98.874732	973.397867
7	0	0	5E-05	10	0.1678	98.419275	970.760188
8	0	0	5E-05	10	0.16805	98.416075	970.395561
9	0	0	5E-05	10	0.16869	98.451972	970.45452
10	0	0	5E-05	10	0.168105	98.437869	971.078661
11	0	0	5E-05	10	0.168275	98.447568	971.113801
12	0	0	5E-05	100	0.14887	95.508419	961.236295
13	0	0	5E-05	100	0.149635	95.575465	956.740517
14	0	0	5E-05	100	0.14851	95.493324	961.454229
15	0	0	5E-05	100	0.14877	95.477374	962.299686
16	0	0	5E-05	100	0.147745	95.32401	963.1477

normalized_data.csv - in this folder is the normalized form of the raw_data.csv

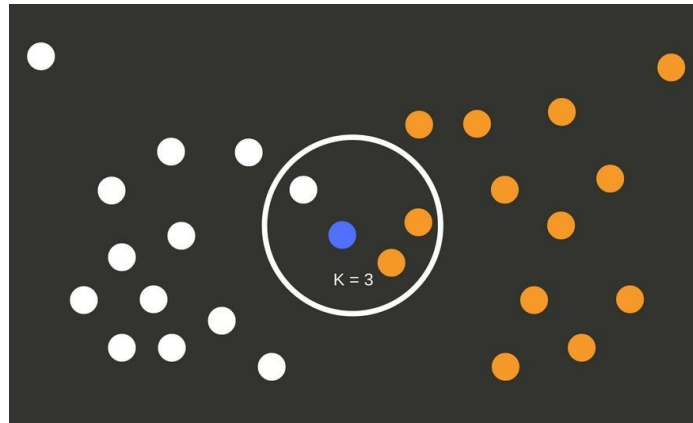
	A	B	C	D	E	F	G
1	X0Y1	Defect	StrainRate	Temperature	FractureStrain	FractureStrength	YoungsModulus
2	0	0	0	0	0.17712	98.877191	973.289014
3	0	0	0	0	0.176935	98.882547	973.408565
4	0	0	0	0	0.17632	98.862375	973.365905
5	0	0	0	0	0.17683	98.868643	973.412343
6	0	0	0	0	0.176565	98.874732	973.397867
7	0	0	0	0.00750626	0.1678	98.419275	970.760188
8	0	0	0	0.00750626	0.16805	98.416075	970.395561
9	0	0	0	0.00750626	0.16869	98.451972	970.45452
10	0	0	0	0.00750626	0.168105	98.437869	971.078661
11	0	0	0	0.00750626	0.168275	98.447568	971.113801
12	0	0	0	0.0825688	0.14887	95.508419	961.236295
13	0	0	0	0.0825688	0.149635	95.575465	956.740517
14	0	0	0	0.0825688	0.14851	95.493324	961.454229
15	0	0	0	0.0825688	0.14877	95.477374	962.299686
16	0	0	0	0.0825688	0.147745	95.32401	963.1477
17	0	0	0	1	0.074195	59.389012	873.373147
18	0	0	0	1	0.076525	59.047461	859.103574
19	0	0	0	1	0.0736	58.383299	865.852454

Step 2 - Machine Learning Algorithm :

Machine learning(ML) algorithm should be based on the type of data we are training it on. Observing the dataset, we concluded that data is in the form of clusters with 5 rows of the dataset falling within a smaller range.

K nearest neighbors:

Given a data point in the test set and a training set for which values are given, find the k nearest data points in the training set and target value is computed as the mean of the values of the k nearest neighbours.



Advantages: Simple to implement and robust to noise in the input data.

Implementation:

- Import raw_data.csv into Matlab workspace.
- 'X0Y1', 'Defect', 'Strain rate' and 'Temperature' in the raw data are considered the inputs (test set).
- fitcknn(X,Y,'NumNeighbors',5) an in-built function in matlab gives KNN-model as the output. X is the test set, Y is the target and k (no. of nearest neighbors) = 5
- After running knn.m
In the workspace
Model1 : KNN model for 'Fracture Strain'
Model2 : KNN model for 'Fracture Strength'
Model3 : KNN model for 'Young's Modulus'
- Computed the loss (error) of all the 3 Models on the dataset.

- Efficiency : K-fold Cross validation of the model can help us identify overfit of the model. If k-fold cross validation loss and loss on the dataset are comparable then the KNN model is efficient.

Model	Loss (error on Dataset)	K-fold Cross Validation Loss
Model1	0.7368	0.9035
Model2	0.8000	1.00
Model3	0.8000	1.00

We observed that both the losses are comparable and hence the Models are efficient.

Prediction example :

A defect free graphene at temperature 100 K under strain rate of 0.001 ps⁻¹ in the armchair direction can be represented by 0, 0, 0.001, 100.

The predicted fracture strain, fracture strength and Young's modulus are 0.1522, 95.7656 GPa and 958.4532 GPa, respectively. The Molecular Dynamics(MD) simulation results are 0.155, 96.2 GPa and 962.9 GPa, respectively.

Mechanical Properties	Prediction (KNN)	Actual (MD of Graphene)	Difference	%error
Fracture strain	0.1522	0.155	0.0028	1.80
Fracture Strength	95.7656 GPa	96.2 GPa	0.4344	0.45
Young's modulus	958.4532 GPa	962.9 GPa	4.4468	0.462

K Nearest Neighbors (KNN)

```
% 1,2,3,4 columns i.e X0Y1,Defect,Strain rate, Temperature of the
rawdata.csv are the parameters
X = [rawdata{:,1},rawdata{:,2},rawdata{:,3},rawdata{:,4}];

% column 5 i.e Fracture strain as the target-1
Y1 = rawdata{:,5};
% column 6 i.e Fracture strength as the target-2
Y2 = rawdata{:,6};
% column 5 i.e Young's Modulus as the target-3
Y3 = rawdata{:,7};

%random seed
rng(10);

% fitting knn using in-bulit function fitcknn
Model1 = fitcknn(X,Y1,'NumNeighbors',5);
Model2 = fitcknn(X,Y2,'NumNeighbors',5);
Model3 = fitcknn(X,Y3,'NumNeighbors',5);

% Loss computed for the Data
rloss1 = resubLoss(Model1);
disp('Loss Model1');
disp(rloss1);
rloss2 = resubLoss(Model2);
disp('Loss Model2');
disp(rloss2);
rloss3 = resubLoss(Model3);
disp('Loss Model3');
disp(rloss3);

% Cross Validation models
CV_model1 = crossval(Model1);
CV_model2 = crossval(Model2);
CV_model3 = crossval(Model3);

% k-fold cross validation error
kloss1 = kfoldLoss(CV_model1);
disp('k-fold loss CV_model1');
disp(kloss1);
kloss2 = kfoldLoss(CV_model2);
disp('k-fold loss CV_model2');
disp(kloss2);
kloss3 = kfoldLoss(CV_model3);
disp('k-fold loss CV_model3');
disp(kloss3);

o1 = predict(Model1,[0,0,0.001,100]);
disp('Fracture strain');
disp(o1);
o2 = predict(Model2,[0,0,0.001,100]);
```

```
disp('Fracture strength');  
disp(o2);  
o3 = predict(Model3,[0,0,0.001,100]);  
disp("Young's Modulus");  
disp(o3);
```

```
Loss Model1  
    0.7368
```

```
Loss Model2  
    0.8000
```

```
Loss Model3  
    0.8000
```

```
Warning: One or more folds do not contain points from all the groups.  
Warning: One or more folds do not contain points from all the groups.  
Warning: One or more folds do not contain points from all the groups.  
k-fold loss CV_model1  
    0.9035
```

```
k-fold loss CV_model2  
    1
```

```
k-fold loss CV_model3  
    1
```

```
Fracture strain  
    0.1522
```

```
Fracture strength  
    95.7656
```

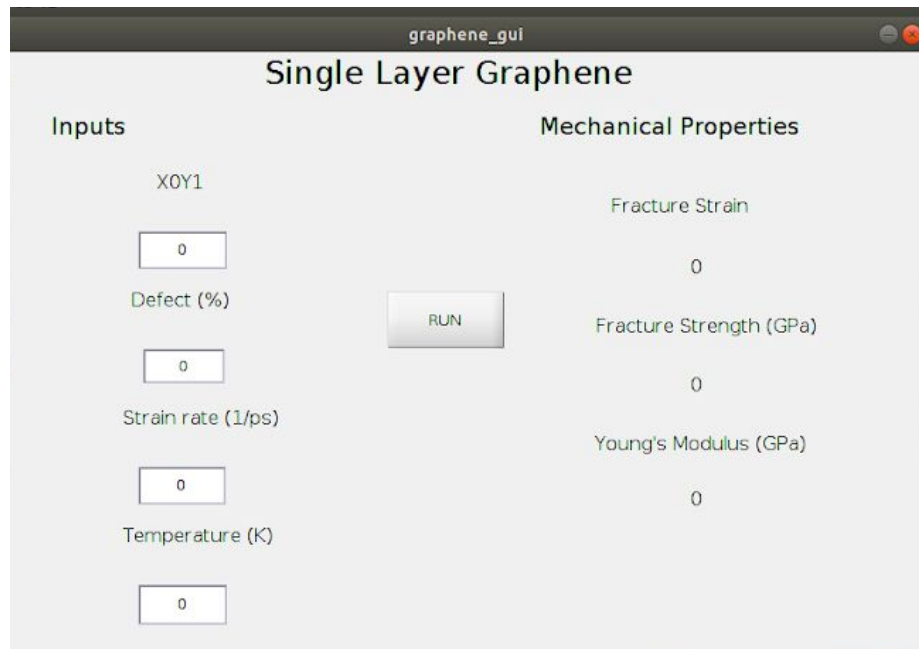
```
Young's Modulus  
    958.4532
```

Published with MATLAB® R2019a

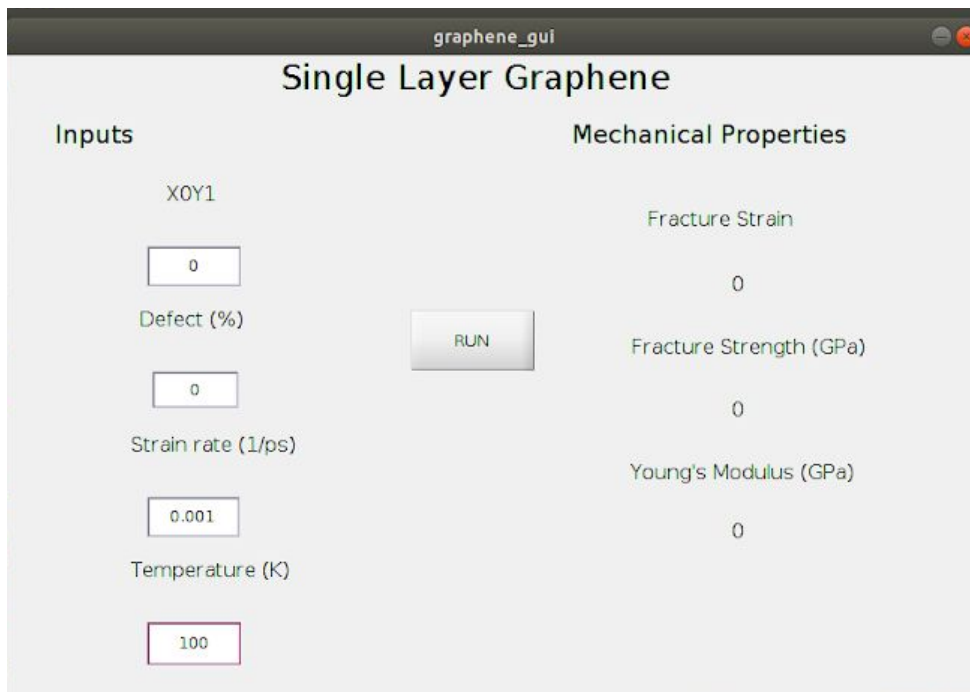
Step 3 - GUI :

Coded A GUI which takes 'X0Y1', 'Defect', 'Strain rate' and 'Temperature' as four inputs. On pressing the RUN button it displays the mechanical properties 'Fracture strain', 'Fracture Strength' and 'Young's Modulus' of the Single Layer Graphene sheet.

1. RUN graphene_gui.m file in this folder (Initial screen):



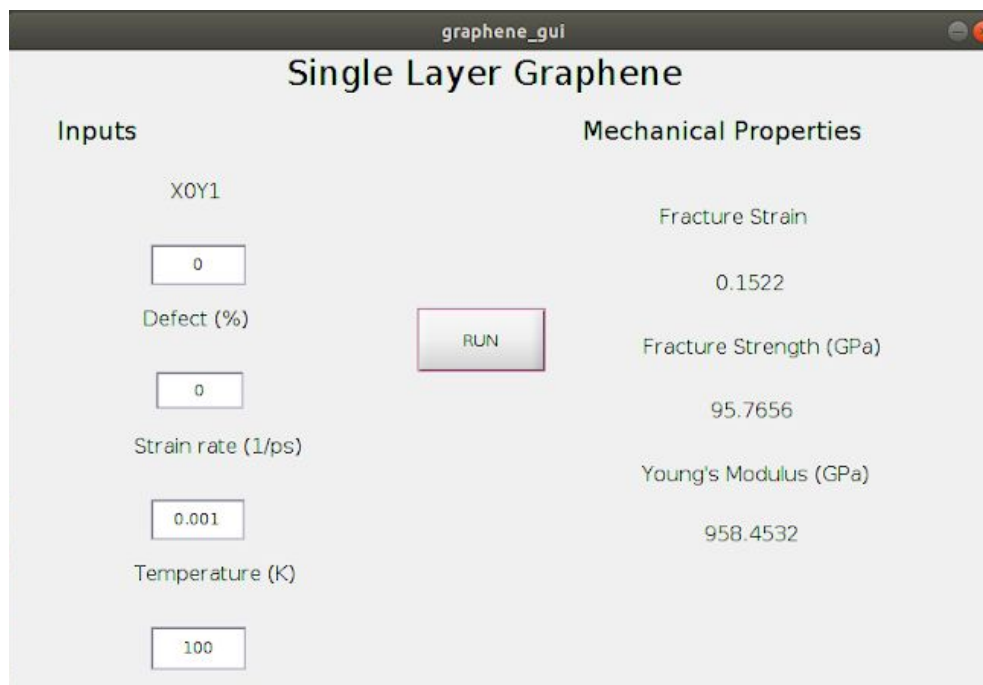
2. Enter the Inputs



The screenshot shows a window titled 'graphene_gui' with the subtitle 'Single Layer Graphene'. It is divided into two main sections: 'Inputs' on the left and 'Mechanical Properties' on the right. In the 'Inputs' section, there are four input fields: 'XOY1' with value '0', 'Defect (%)' with value '0', 'Strain rate (1/ps)' with value '0.001', and 'Temperature (K)' with value '100'. A 'RUN' button is located between the input and output sections. The 'Mechanical Properties' section currently shows all values as '0'.

Inputs	Mechanical Properties
XOY1	Fracture Strain
0	0
Defect (%)	Fracture Strength (GPa)
0	0
Strain rate (1/ps)	Young's Modulus (GPa)
0.001	0
Temperature (K)	
100	

3. Press the 'RUN' button on the window to display the outputs



The screenshot shows the same 'Single Layer Graphene' GUI window after the 'RUN' button has been pressed. The 'Mechanical Properties' section now displays calculated values: 'Fracture Strain' is 0.1522, 'Fracture Strength (GPa)' is 95.7656, and 'Young's Modulus (GPa)' is 958.4532. The 'Inputs' section remains unchanged.

Inputs	Mechanical Properties
XOY1	Fracture Strain
0	0.1522
Defect (%)	Fracture Strength (GPa)
0	95.7656
Strain rate (1/ps)	Young's Modulus (GPa)
0.001	958.4532
Temperature (K)	
100	

Conclusion:

- The mechanical properties are generally studied using sophisticated softwares like LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator), vmd (visual molecular dynamics) which directly involve visualising the atoms and drawing conclusions from it.
- Machine Learning provides us the flexible way, where the proficiency in understanding these sophisticated softwares and intrinsic behavior of the graphene structure is not necessary.
- Here with adequate data, using machine learning algorithms we trained it on a dataset and developed models which can predict the mechanical properties of Graphene.
- We developed a user-friendly GUI which can take inputs from the user and outputs the required mechanical properties of Graphene.