דו"ח הסבר שימוש במערכת:

במהלך חקירת הנושאים, יצרנו 3 מערכות של Lammps:

1. lammps- זוהי המערכת עם הקוד הבסיסי של אופק המפעיל את הפוטנציאל הנוסף על רבעיה אחת אקראית.
2. lammps_forsets- זוהי המערכת שאנו הוספנו בה את הקוד שלנו המפעיל את הפוטנציאל הנוסף על כל הרביעיות החוקיות שנמצאו בו זמנית (רביעיות חוקיות – אף רבעיה לא מכילה אטום משותף עם רבעיה אחרת).
3. lammps_gpu- זוהי המערכת שבעזרתה ניסינו להריץ סימולציות על כרטיס מסך GPU. במסגרת הזמן של הפרויקט לא הוספנו את קוד המטה-דינמיקה ל-GPU.

התקנה וקימפול של מערכת Lammps_forsets:

1. הורידו את Lammps מכאן.
2. העתיקו את fix_reaxc_checkFourset.cpp, fix_reaxc_checkFourset.h מהגיט אל תוך התיקייה: lammps/src
   העתיקו את pair_reaxc.cpp, pair_reaxc.h אל התיקייה: lammps/src/USER-REAXC
3. פתחו טרמינל בתיקייה lammps/src
   (כדי להשתמש בהרצה מקבילית וודאו כי OpenMP הוא מאפיין של הקומפיילר שלכם)
   הריצו פקודות אלו:

```
make yes-USER-REAXC
make yes-USER-OMP
make omp
```

4. הורידו את תיקיית הריצה מכאן.

הערה: עבור הרצה סיראלית והרצות יחידניות ניתן לפנות ל גיט של אופק.

אופן זרימת המערכת והרצתה:

1. הפעלת התכנית של הוצאות ריצות עם רביעיות במקביל למשך 500K צעדי זמן:
   בכל תיקיית ריצה ב Shira_Michal/before_delete יש 4 תיקיות אותם צריך לעדכן
   - הגדרת סדר גודל הריצה - מעתיקים אל תיקיית המin את קובץ הקלט של האטומים עליו רוצים להריץ את הסימולציה לדוג:
     epon16xdetda8_noPBC.dat ומשנים את שדה ה read_data בקובץ in.min לשם קובץ הקלט.

- מעדכנים את הקובץ Extra_Potential_Parameters בתיקייה: nvt_BB_real
עם צעדי זמן הרגיעה הרצויים, ערכי הf1 לrange - ערך התחלה, ערך סיום, קפיצות.
מגדירים בקובץ in.nvt בשורה האחרונה של הrun ל500000 צעדי זמן.
- בתיקייה Shira_Michal/ Our Python Code בוחרים את קוד הפייתון שמריץ עם רביעיות – create_runs_forsets.py
יש לוודא כי run_dir מעודכן לנתיב של תיקיית הריצה וה-cmd מעודכן לנתיב של הlammps שקמפלתם.
בפעם הראשונה שמריצים על סדר גודל מסוים יש להשאיר את שלושת הריצות המתבצעות בקוד create_runs_forsets.py ניתן לראות בחלק של if True # if suffix == 'nvt_BB_real'
לאחר שריצת הmin מסתיימת יש להעתיק את הקובץ restart.mini לתיקיית nvt1 ולהפעיל מחדש את הקוד.
לאחר שריצת הnvt1 מסתיימת יש להעתיק ידנית את הקובץ restart.nvt1 לתיקיית nvt_BB_real.
בפעמים הבאות בהרצה על סדר גודל זה יש להשתמש בחלק השני המסולש- # if suffix == 'nvt_BB_real'.
בשלב של save results of nvt__BB run ניתן לשמור קבצי פלט נוספים של הריצה.
- פותחים טרמינל בתיקייה Shira_Michal/ Our Python Code ומריצים את python create_runs_forsets.py.

2. כדי למצוא את הריצה עם הפרמטרים הטובים ביותר נפעיל את הקוד find_optimal_run_params_generic.py (יש לעדכן את הנתיב אל תיקיית הקבצי species שנשמרו בnvt_BB_real/ all_results_for_f1_f2)
הריצה תוציא את פרמטרי f1 האופטימליים.
בנוסף תיוצר תיקיית קבצי CSV בה מפורט עבור כל ריצה- שילוב פרמטרים פירוט על הצילובים והקרעים שנוצרו בצעדי זמן של סוף הריצה שנדגמו.

3. כעת מריצים ריצה של 2-4 מיליון צעדים (לפי הדרישה) על פרמטרים אלו.
מעדכנים את צעדי הזמן בקובץ in.nvt ואת שורת ה if True: בקובץ - create_runs_forsets.py לשורה המסולשת עם הפרמטרים האופטימליים שנמצאו.
ניתן לנתח את תוצאות הריצה הבודדת ע"י הקוד Analyze_Result_Of_Run.py שבתיקייה ב python_code בגיט.

4. בסיום הריצה אפשר להפעיל את התוכניות לויזואליזציית הנתונים והוצאת הגרפים:
- cal_cross_linking_percent.py- מציג את אחוז הצילוב כתלות בזמן, משתמש בקובץ bonds.reax.

- cal_cross_number.py- מחשב את כמות הצילובים כתלות בזמן, משתמש בקובץ species.
- read_res_nvt.py- קורא את הקובץ res_nvt (קובץ כבד) ושומר מתוכו רק שורות הרלוונטיות לקלט לתוכנית הבאה:
- display_num_foursets_for_crossover.py- מחשב את כמות הרביעיות התקינות עליהן מנסים להפעיל את הפוטנציאל הנוסף, כתלות בצעדי הזמן. משתמש בפלט התוכנית read_res_nvt.py, הקובץ- res_count.txt.
- species_reader.py- מחשב את המולקולות שנוצרו כתלות בצעד זמן, משתמש בקובץ species.out.

5. ריצות NPT:

create_NPTruns.py- תכנית להפעלת ריצות NPT,

- על פי תוצאת הגרף של התוכנית :cal_cross_linking_percent.py

ניתן לבחור את צעדי הזמן של אחוזי הצילוב בהם רוצים לבדוק, לאחר ריצת NPT, את הנפח (צפיפות ושאר פרמטרים), מעתיקים מתיקיית הdump בnvt_BB_real את הקבצים עבור צעדי זמן אלו.

- ממירים את קבצי ה-dump לקבצי dat, בעזרת תוכניות ה-matlab (נמצא בגיט + דף הסבר לאופן השימוש: Matlab_code).
- הפעלת הקוד להוצאת ריצות NPT:

לפני התחלת הריצה: יש להעתיק את הקבצים המומרים לתיקיית הnpt/input_files,

בקוד הפייתון, יש לעדכן את המשתנה TimeStep- למספר צעדי הזמן (2 מיליון, לפי דרישה), נתיב לתיקיית NPT ולעדכן את מערך שמות הקבצים []input_files.

הפעלת התכנית: Shira_Michal/our_python_code/create_NPTruns.py

ויזואליזצייה וניתוח נתוני הNPT:

- Isothermal.py- מחשב דחיסות איזותרמית, מודול ניפחי, נפח ממוצע וצפיפות ממוצעת. משתמש בקבצי הפלט- log של ריצת הNPT.