Algorithm description:

For each iteration we'll do the following:

1. Calculate times with a given step-size
2. Translate the time point to Julian date format required by SGP4
3. Call SGP4 on array of object\_1 and object\_2 and get location vectors of the objects
4. For each time point, we'll calculate the relative distance between the objects
5. If a new minimal distance is received, save the distance and the time is happened, and save the vectors in a file
6. If minimal step-size is reached, or the minimum we received is no better than the previous 2, get out of the loop. Else, reduce the step-size and repeat 1 to 6.

Improvement for the Algorithm:

Working with large size of data can make errors, such as memory errors, it is easy to deal with if we limit the number of time point we measure to blocks with fixed size.

1. Calculate times with a given step-size
2. divide the times received into blocks
3. for each block do steps 3 to 5 in the previous algorithm
4. If minimal step-size is reached, or the minimum we received is no better than the previous 2, get out of the loop. Else, reduce the step-size and repeat A to D.

Pseudo code:

Input: tle1, tle2, t\_end, st\_min, p\_max, factor, init\_step\_size, name

1. Initialize min\_distance, step\_size, min\_time
2. Loop:
3. Set counter to 2
4. Calculate number of points and blocks
5. For each block:
   1. Generate time points
   2. Calculate jd, fr
   3. Initialize satellite objects sat1, sat2
   4. Assign jd, fr to sat1 and sat2
   5. Calculate r1, r2 for sat1 and sat2
   6. Calculate distance between the objects
   7. Save location vectors and times to a file
   8. Find the minimum distance between the objects in the current block
   9. If a new minimum distance is found, update min\_distance and min\_time
6. If no new minimum is found in any block, decrement the counter

while (step size is greater than st\_min and counter is greater than 0)

3. Return min\_distance and min\_time

תיאור אלגוריתם ללא מגבלת כמות מידע נשמרת

בכל איטרציה נבצע:

נחשב נקודות זמן בקפיצות בגודל הצעד הנתון

נעביר את נקודות הזמן לפורמט הזמן המתאים עבור SGP4

נקרא SGP4 על מערך הנקודות שלנו עבור אובייקט1

נקרא SGP4 על מערך הנקודות שלנו עבור אובייקט2

נקבל וקטורי מיקום של 2 אובייקטים, בכל נקודת זמן נבצע חישוב של גודל וקטור המרחק היחסי בניהם

אם המרחק שהתקבל קטן מהמינימום עד כה(באיטרציה זו) שמור אותו ואת הזמן שבו הוא התרחש במקום

שמור את הוקטורים בקובץ

נרוץ עד שקורה 1 מ2 אופציות, או שהגענו לגודל צעד זמן מינימלי או שהמינימום שמצאנו בסבב הנוכחי לא טוב יותר מהמינימום שמצאנו בסבב הקודם(כלומר סבבים נוספים יהיו מיותרים). במידה והלולאה עדיין מקיימת את 2 הדרישות הנ"ל, הקטן את צעד הזמן בפקטור כלשהי נתון וחזור להתחלה

תיאור האלגוריתם עם גודל מערך מינימלי אפשרי

פרמטרים: גודל בלוק מקסימלי, גודל צעד מינימלי, 2 נתוני לווינים, גודל זמן נמדד בשניות, גודל שינוי שכבר לא אכפת לנו ממנו

בכל איטרציה נבצע:

פתח קובץ חדש עבור הרצה בגודל צעד זה

חשב את מספר המערכים הנדרש כדי לאחסן את כלל הנקודות בסבב זה.

עבור כל בלוק מ 0 עד k חשב את זמן ההתחלה שלו וזמן הסיום שלו

לולאה, עבור כל בלוק מ0 עד K בצע:

4.1. נחשב את נקודות הזמן מנקודת ההתחלה בקפיצות של גודל צעד עד לזמן הסיום

4.2. נעביר את נקודות הזמן לפורמט הזמן המתאים עבור SGP4

4.3. נעביר את נקודות הזמן לפורמט הזמן המתאים עבור SGP4

4.4. נקרא SGP4 על מערך הנקודות שלנו עבור אובייקט1

4.5. נקרא SGP4 על מערך הנקודות שלנו עבור אובייקט2

4.6. נקבל וקטורי מיקום של 2 אובייקטים, בכל נקודת זמן נבצע חישוב של גודל וקטור המרחק היחסי בניהם

4.7. אם המרחק שהתקבל קטן מהמינימום עד כה(באיטרציה זו) שמור אותו ואת הזמן שבו הוא התרחש במקום

4.8. שמור את הוקטורים בקובץ

שמור את הקובץ של הסבב בגודל צעד זה עם נקודת הזמן והמרחק המינימלי שנמצאו

נרוץ עד שקורה 1 מ2 אופציות, או שהגענו לגודל צעד זמן מינימלי או שהמינימום שמצאנו בסבב הנוכחי לא טוב יותר מהמינימום שמצאנו בסבב הקודם(כלומר סבבים נוספים יהיו מיותרים). במידה והלולאה עדיין מקיימת את 2 הדרישות הנ"ל, הקטן את צעד הזמן בפקטור כלשהי נתון וחזור להתחלה

Propagator points generating algorithm

TLE = two lines elements

T\_end = we need points in time from epoch time up to T\_end seconds later.

St\_min = לא מחלקים לנקודת זמן קטנה יותר

B\_max = we cant save one array\list bigger than that size, divide into smaller arrays – צריך לחשב איכשהו

e1, e2 are the TLE(two line elements) of 2 sattelites

We try to get the min from Epoch up to T\_end seconds later

St\_min is the minimal size of step in seconds(can be a fraction)

P\_max is the max amount of points we can calculate in one call of sgp4\_array

Factor is the factor we divide the time step by

initial step size is the size of the initial step size in seconds

Input: e1,e2,T\_end, St\_min, P\_max, Factor, initial step size

stepSize = initial step size (in seconds)

Do:

1. Calculate the number of inner iteration and number of points in each iteration
2. for each inner iteration do:
   1. For each point in the iteration
      1. Calculate the Time From Epoch(Jf +Fr)
   2. Run sgp4\_array on the array of time points
   3. For each time point in the inner iteration do:
      1. Calculate the distance(size of relative distance vector)
      2. If the distance is smaller than the minimal distance points save the current distance + time
   4. Save the data points in file
3. stepSize = stepSize/Factor

while stepSize > St\_min AND | current minimum – last iteration minimum | > 0

save the minimal point, the final step size

main –

מקבלים TLE של 2 לווינים

מקבלים כמות נקודות בכל מקטע עבור ancas + catch

קובץ נוסף שבו מבוצעים חישוב נקודות בקפיצות זמן(האלגוריתם למעלה) עם פונקציה ראשית של האלגוריתם שמקבלת קלט כפי שהוגדר

קובץ פונקציות לחישובי עזר כמו חישוב שניות לjd fr וכו

נתחיל בגודל בלוק התחלתי

נמדוד זמן לכל איטרציה

בכל איטרציה נבנה נתונים עבור sgp4 array ונריץ אותו

את התוצאות נכתוב לקובץ.

נעצור את מדידת הזמן

לכל מדידת זמן כזאת נוציא את הזמן שלקח לאומת מספר הנקודות(לקובץ)

הגדלת נקודות + הוספה של מספר קבוע(חזקה של 2 למשל 1024)

אחר כך:

נעשה פונקציה\גרף

נמצא את הנקודת שבר(בה כבר לא משתלם להגדיל את מספר הנקודות), נמצא את הנקודה בה הקוד קורס(קריטית)