**אנליזה נומרית תשע"ט**

**דו"ח סיכום האקטון**

קבוצה מס': 23

חברי הקבוצה הנוכחים: אור ממן, איוון רובינסון, סתיו לובל, שלי מירון

**סעיף א – הגדרת הבעיה –**

במאמר שניתן לנו ששמו:  
*“Two independent approaches used for estimating 2d contamination distribution on ground level- based on air monitoring information”*   
מוצגת בעיה מורכבת שדורשת פתרונות נומרים בכדי להביא אותה לידי מימוש תוכנתי.

הבעיה מתארת איזור מזוהם באיזוטופים רדיואקטיביים שיש לכמת, באמצעות הליקופטר עם גלאי/חיישן, שמטרתו למפות את רמות הקרינה באיזור.

הגלאי סורק את שטח ה- NXN המתואר, מגובה מסוים, לפי השיטה הבאה:

השטח מחולק לריבועים אחידים. מרכז כל "תא" כזה (=ריבוע) ייצג את רמת הקרינה בכל התא.

ההליקופטר מושפע מהקרינה בכל השטח בו-זמנית, אבל קיימות נוסחאות מתמטיות המאפשרות לבודד מתוך המדדים את רמות הקרינה בכל יחידת שטח. זאת מטרתינו – לפתור משוואות מתמטיות אלה באופן נומרי.

כדי לקבל תוצאה אמינה, יש לבצע את החישובים בשתי דרכים שונות. אנחנו נתמקד בדרך אחת – פתרון מטריצות.

הקבועים שאנחנו צריכים לקבל על מנת לבצע את החישובים:

* c = proportion coefficient of the monitor [cps m^2/gamma]
* k = radiation build up factor in air [m^-1]
* µ = radiation absorption coefficient in air [m^-1]
* גודל השטח המזוהם [m^2]
* רזולוציית חלוקת השטח לריבועים
* גובה ריחוף המסוק [m]
* המדידות שהתקבלו מהמסוק [cps]

יש לשים לב כי רמת דיוק הקבועים עלולה להשפיע על רמת דיוק התוצאה שנחשב; לכן נשתמש במספר שיטות שונות כדי להגיע אליהם.

כפל מטריצות מטבעו פעולה לא יציבה, לכן יש לשים לב לרמת הדיוק במטריצה שנחשב, ושנשתמש בשיטות הממזערות את רמת השגיאה.

**סעיף ב' – השיטה והצגת הכלים לפתרון –**

**סעיף ג' – הצגת הנתונים –**

**סעיף ד' – תוצאות –**

**סעיף ה' – סיכום –**

**נספח א’ –**

קוד שלנו:

* [Gauss Algorithm](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/GaussAlgo.py#L22)
* [Matrix Norma calculation](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/GaussAlgo.py#L4)
* [Matrix CondA calculation](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/GaussAlgo.py#L19)
* [Newton Repson Algorithm](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/NewtonRepson.py)

קוד שנעשה בו שימוש חוזר:

SciPy (<https://www.scipy.org/>) and NumPy (<http://www.numpy.org/>)

* numpy.tril
* numpy.dot
* numpy.array
* numpy.linalg.inv
* scipy.linalg.lu
* [Jacobi Algorithm](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/Jacobi.py) - <https://en.wikipedia.org/wiki/Jacobi_method#An_example_using_Python_and_Numpy>
* [Bisection Algorithm](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/BisectionAlgorithm.py) - <http://code.activestate.com/recipes/578417-bisection-method-in-python/>
* [Secant Algorithm](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/SecantAlgorithm.py) - <http://code.activestate.com/recipes/578420-secant-method-of-solving-equtions-in-python/>
* [Gauss-Seidel Algorithm](https://github.com/SCE-SWE-2018-G11/NumericAnalysis/blob/master/Gauss-Seidel.py) - <https://austingwalters.com/gauss-seidel-method/>