For sekvensiell multiplisering tok jeg utgangspunkt i koden gitt i oppgaveteksten for ukesoppgaven for oppgave 3, og fulgte oppskriften på å lage en metode som gjorde sekvensielle biten med transponering av b, så endte opp med å ta utganspunkt i at alle matriser er transponert. Ville også ha en metode jeg kunne bruke til den parallelle biten også og har derfor med venstre og høyre som brukes for å dele inn biter til de forskjellige trådene og avgrense deres arbeidsområde til antall elementer per tråd.

For den sekvensielle biten lagde jeg en klasse kalt Para som implementerer Runnable interfacet, lagde en konstruktør til denne klassen som tar vare på viktig informasjon om matrisene og arbeidsområdet til selve tråden. Jeg valgte å bruke cyclicBarrier som synkroniseringsverktøy, siden trådene her leser og skriver av et objekt er det viktig at de ikke gjør det samtidig for å ikke få kluss og oppdatering/lagringsfeil underveis, når alle trådene er ferdig kjøres en testmetode før tiden i ms returneres.

Jeg klarte ikke å få speedup ved lav n, men det var ikke noe jeg forventet i utgangspunktet, det er kanskje mulig å få til noe bedre resultat, men her vil lagingen av tråder ta opp mye av tiden som blir brukt uansett, ellers var jeg veldig overrasket over hvor raskt det faktisk gikk i parallell etter at jeg hadde gjort ferdig hele den sekvensielle delen av oppgaven først forventet jeg meg noe nærmere 3-4x speedup, jeg så også markant økning ved transponering av arrayen selv her hvor vi bruker kvadratiske matriser noe jeg egentlig ikke hadde forestilt meg, dataene for kjøring uten transponering er ikke med i rapporten siden jeg ikke har lagt inn løsning for bytte i det endelige programmet mitt. Økningen i speedup ved parallellisering og høyere N kommer vel av mer effektiv traversering, altså ikke like mye leting i kolonner, mye av arbeidet kan utføres samtidig mens én enkelt tråd som hopper mye fram og tilbake i kolonner vil måte lese mye fra forskjellige deler av minnet.

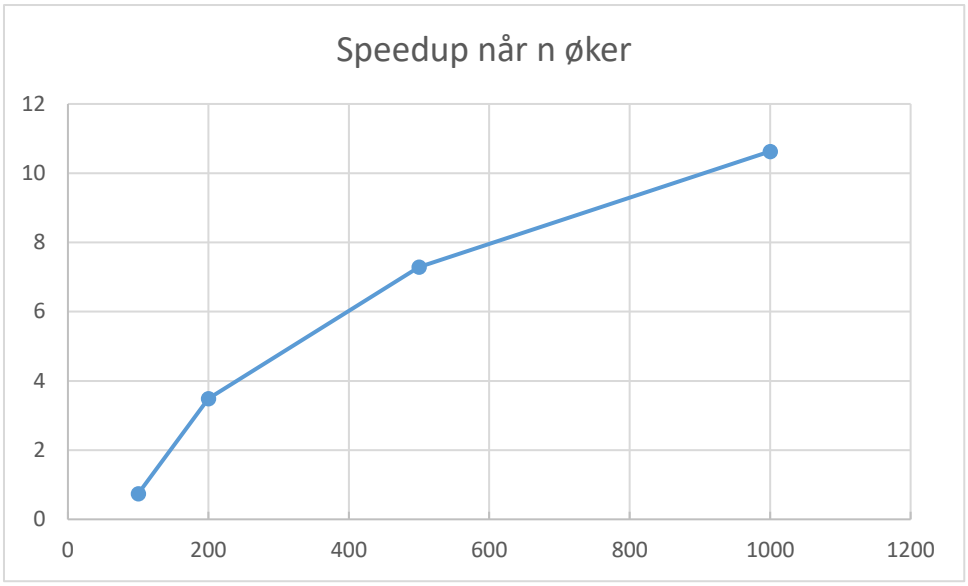
Jeg har også valgt å fjerne sjekker for at matrise a rader matcher b kolonner siden vi eksplisitt blir bedt om å bruke kvadratiske matriser, men å endre programmet for å fasilitere alle matriser ville ikke vært veldig vanskelig.

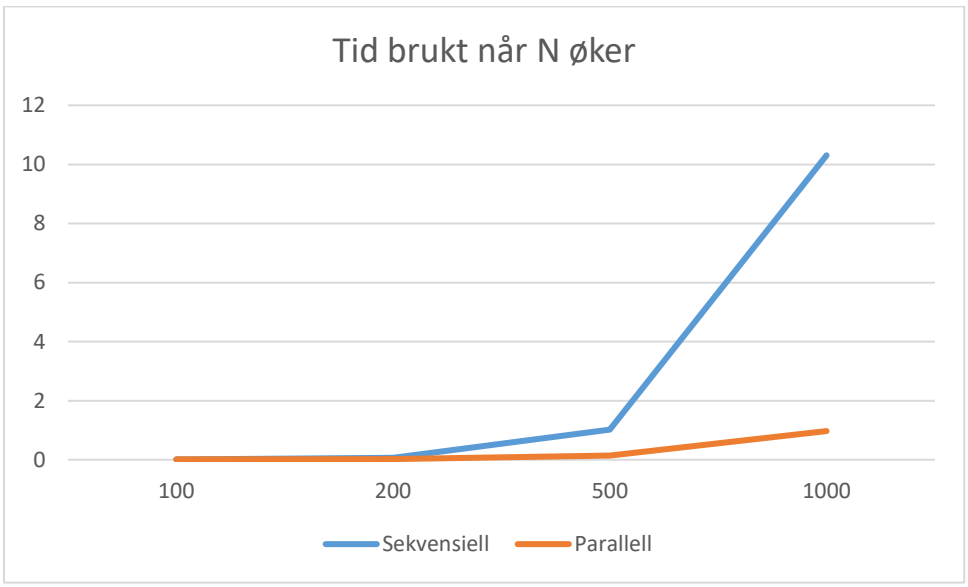
Kompiler med javac *.java

Kjør med java Oblig2 n eks: java Oblig2 1000

CPU : Intel® Core™ i7-4790 CPU @ 3.60GHz

| N | Sekvensiell | Parallell | speedup |
|---|---|---|---|
| 100 | 0.934568 | 1.260724 | 0.74 |
| 200 | 6.767202 | 1.974907 | 3.43 |
| 500 | 102.427195 | 14.057282 | 7.29 |
| 1000 | 1030.693319 | 96.976313 | 10.63 |


Speedup når n øker
12
10
8
6
4
2
0
0
200
400
600
800
1000
1200


Tid brukt når N øker
12
10
8
6
4
2
0
100
200
500
1000
Sekvensiell
Parallell