Python-basert verktøy for analyse av resultater fra strekkmaskintester

Solveig Engebretsen, Tom Thorvaldsen og Bendik Sagsveen

Forsvarets forskningsinstitutt (FFI)

22. oktober 2012

FFI-rapport 2012/01944

380201

P: ISBN 978-82-464-2153-7
E: ISBN 978-82-464-2154-4

Emneord

Strekkmaskin

ASTM-standarder

Python

Strekktest

Bøyetest

Godkjent av

Einar Willassen	Prosjektleder
Jan Ivar Botnan	Avdelingssjef

Sammendrag

Denne rapporten er en brukermanual for analyse av tester utført på strekkmaskinen av typen MTS 810. Rapporten inneholder informasjon om hvordan man installerer Python-programvare som er nødvendig for å kjøre analysene, hvordan man bruker programvaren, samt informasjon om hvert enkelt program. I tillegg er det gitt en beskrivelse av hvordan man legger tabellen som programmet genererer, inn i en Windows Excel-tabell.

English summary

This report is a user guide for analysis of tests performed on the MTS 810 tensile test machine. The report contains information on how to install the Python software necessary for the analyses, how to use the software, as well as information about each Python program. In addition, a description of how to convert the table generated by the program to a Windows Excel table is provided.

Innhold

1 Innledning

Programvareverktøyet som beskrives i denne FFI-rapporten, er laget for å automatisere arbeidet med analyse av resultater fra tester utført på FFIs strekkmaskin av typen Material Testing Systems MTS 810, med tilhørende programvare fra MTS. Automatisering reduserer antall potensielle feil og effektiviserer analysearbeidet for hver test og serier av tester.

Programmene er skrevet i Python. Python har en klar syntaks, og koden er dermed rask og ukomplisert å skrive, samtidig som den er oversiktlig for andre å lese – også for de som ikke kjenner programmeringsspråket fra tidligere. Python har videre innebygde funksjoner for å lese og skrive til fil, og det er mulighet for å lagre informasjonen i lister. Dette gjør programmeringsspråket meget hensiktsmessig for omorganisering av data til et ønsket format. Dessuten er det utviklet moduler for å utføre matematiske operasjoner (også på lister), som gjør dette verktøyet relevant for analyse av dataene fra strekkmaskintester. Det legges også til at det i Python er relativt enkelt å lage et brukergrensesnitt (GUI) som gjør selve analyseverktøyet mer brukervennlig, slik at programmene kan brukes av operatører som ikke er kjent med Python og/eller som ikke ønsker/har behov for å forstå detaljene i programmet.

En introduksjon til bruk av selve strekkmaskinen, etter oppgradering av tilhørende programvare, er gitt i [1]. Det antas at leseren av denne rapporten er kjent med hvordan testene på strekkmaskinen utføres. Videre antas det at man har tilgang til resultatfiler fra en testserie. For øvrig er standardene som det henvises til i dokumentet, listet i Appendix A. Disse er tilgjengelige ved FFI. Kildekoden til de ulike Python-programmene er gitt i 0.

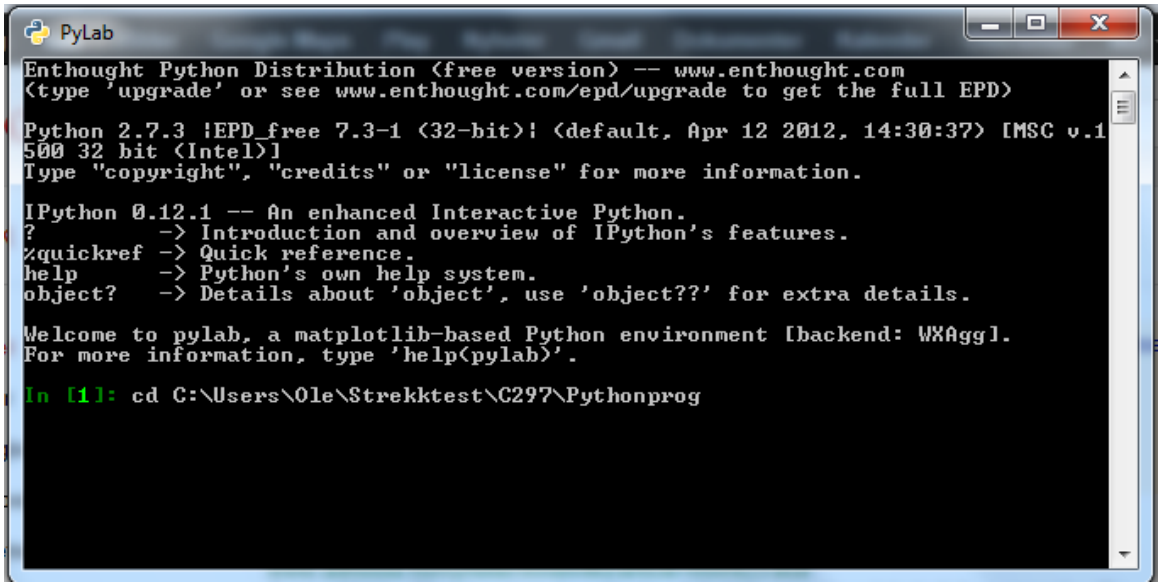
Merk at programvareverktøyet er laget for tester utført på strekkmaskinen fra og med oktober 2012. Dersom formatet på datafilen fra strekkmaskintester avviker fra dette formatet, må denne endres manuelt. Hvordan datafilene skal se ut, er beskrevet i seksjonen for de spesifikke testene.

2 Installasjon av Python-programvare

2.1 PyLab

PyLab er en variant av iPython, som er et interaktivt Python-terminalvindu (vanligvis kalt ”skall”; engelsk ”shell”), hvor man kan skrive kommandoer direkte og kjøre Python-programmer. [1] Et eksempel på et slikt vindu er vist i Figur 2.1.

[1] Tilsvarende kommandolinjevindu har man i operativsystemene Windows DOS og Unix/Linux.



Figur 2.1 Et eksempel på et PyLab-vindu

2.2 Installasjon

Det finnes mange måter å installere Python på. Én mulighet er å installere "Enthought Python Distribution Free" som er en gratisdistribusjon av Python hvor man får med PyLab. Enthought-distribusjonen av Python inneholder også en del nyttige gratispakker. Enthought-distribusjonen kan lastes ned fra nettsiden http://www.enthought.com/repo/free.

For å installere programmet på en Windows-maskin, gjør man som følger:

1) Dobbeltklikk på filen epd_free-7.3-1-win-x86.msi. Velg deretter *Run* eller *Save*. Det vil nå komme opp et installasjonsvindu.
2) Akseptér betingelsene i *Review the Install Agreement*
3) I *Select Destination Directory*, velg hvor på datamaskinen (lokalt) programmet skal lagres; forslaget fra installasjonsrutinene, det vil si "default" plassering, er som regel mest hensiktsmessig[2]
4) Velg om og hvor du vil ha snarveier for programmet (ofte hensiktsmessig for å få et ikon på skrivebordet)
5) Start installasjonen, og vent på at den gjør seg ferdig

Dersom man velger å installere Python på en annen måte enn beskrevet over, kan det hende at man må installere tilleggspakker, som for eksempel matplotlib, for å få programmene til å fungere.

I resten av denne manualen antas det at man har installert Enthought-distribusjonen av Python og kjører programmene fra et PyLab-vindu.

[2] Problemer med installasjonen kan oppstå dersom man prøver å opprette en egen katalog.

3 Hvordan kjøre et Python-program

Programmet startes ved å åpne PyLab. Dobbeltklikk på ikonet på skjermen din (så sant du har valgt å lage en snarvei). Det kommer da opp et PyLab-terminalvindu, som vist i Figur 2.1.

For å få kjørt et Python-program, må man i PyLab-vinduet befinne seg i samme mappe som Python-programmet. For å komme seg til rett mappe i sitt eget filsystem, brukes kommandoen cd (”change directory”). Dersom programmet for eksempel befinner seg under C:\Users\Ole\Strekktest\C297\Pythonprog, blir kommandoen (som også vist i Figur 2.1)

```
cd C:\Users\Ole\Strekktest\C297\Pythonprog
```

For å kjøre programmet, skriver man deretter

```
execfile("<Programnavn>.py ")
```

der <Programnavn> da er navnet på Python-programfilen. Samme kommandoer benyttes for alle testene beskrevet videre i denne rapporten.

4 Filer fra strekkmaskintester

Hver gang det gjøres en test i strekkmaskinen, får man en datafil som legges i en egen mappe for akkurat den prøven. Denne filen vil alltid hete **specimen.dat**. Et eksempel på de første åtte linjene fra en typisk **specimen.dat**-fil, er som følger:

```
1. MTS793|MPT|ENG|1|2|.|/|:|44|1|1|A
2.
3. Data  Acquisition  DataFangst  Time: 151.53809  Sec  08/05/2012 13:15:37
4. Time      Ch 1 Displacement      Ch 1 Force
5. Sec       mm                     N
6. 0.11425781          8.5681677e-05          1.2050189
7. 0.21386719          0.00091046095          2.7285168
8. 0.31347656          0.0022672117           -0.089243524
```

Filen skal alltid inneholde fem linjer med tekst/informasjon før talldataene kommer. Linje tre inneholder dato som nest siste informasjon. Linje fire inneholder kolonnenavnene, men kan inneholde annen informasjon i tillegg. Rekkefølgen på kolonnene i datafilen er vilkårlig.

Kolonnenavnene må stemme overens med kolonnenavnene spesifisert for hver spesifikke test, som beskrevet nærmere i Seksjon 6. Dette er viktig, da det er disse (nøkkel-) ordene programmene bruker for å kunne tolke hvilken kolonne som inneholder hvilken informasjon. Linje fem angir kun enhetene på tallene i kolonnene (det vil si ”N Sec mm”). Det må imidlertid være samsvar mellom kolonnenavn og enheter.

5 Organisering av filer

For hver standard finnes det to ulike Python-programmer:

1) Program for å få dataene over på et mer ”riktig” format – kalt **<Standard>test.py**.
2) Program for å kjøre analyse på testene – kalt **<Standard>resulttable.py**.

<Standard> er navnet på standarden som ligger til grunn for testen. Programmene er laget med utgangspunkt i den gitte standarden for hver test, men tilpasset FFIs bruk.

5.1 Fil over på ønsket format

For hver test kjører man først Python-programmet **<Standard>test.py**, for å få dataene over på ønsket format. Hvordan man kjører et Python-program er beskrevet i Seksjon 3.

Hver gang man kjører et av Python-programmene, vil det komme opp et vindu, eller en GUI (Graphical User Interface). I GUIen som dukker opp, vil det spørres etter en del parametere som er karakteristiske for den gitte testen; se seksjonene for de spesifikke testene i Seksjon 6. Felles for alle testene er at programmet hver gang vil spørre etter navnet på den spesifikke prøven (*Material_Specimen #*), sti (*Path*) og filnavn (*Filename*). I navnet på den spesifikke prøven skriver man materialet og prøvenummer (for eksempel ”Karbon_1”). Dette navnet kan velges fritt og trenger ikke samsvare med eventuelle tidligere navn gitt ved testing på strekkmaskinen. I stifeltet skriver man fullstendig navn på lokaliseringen av mappen hvor den aktuelle **specimen.dat**-filen ligger. I filnavnfeltet angir man det man ønsker som filnavn. Filnavnet vil da bli på formen **ASTM_<Standard>_<filnavn>.pdat**, der **pdat** indikerer at dette er en resultatfil fra Python-programanalysen. Dersom man for eksempel skriver ”test1” i filnavnfeltet, og det er en test som følger C297-standarden, vil man få generert en fil med navnet **ASTM_C297_test1.pdat**. Dersom man ønsker at **<filnavn>** skal være det samme som navnet på den spesifikke prøven, kan man la filnavnfeltet stå tomt. Dersom den spesifikke prøven ble kalt ”Karbon_1”, vil det da altså genereres en fil med navn **ASTM_C297_Karbon_1.pdat**. Resultatfilen vil havne i samme mappe som **specimen.dat**-filen.

5.2 Analyse av flere testprøver

Når man er ferdig med å kjøre det første programmet for alle prøvene (det vil si alle **specimen.dat**-filene for testserien) og har fått generert en ***.pdat**-fil for hver prøve, som beskrevet i Seksjon 5.1, må man samle alle ***.pdat**-filene i én og samme mappe for videre analyser av dataserien. I denne mappen kan det ikke ligge resultatfiler fra tidligere tester, da programmet vil lese gjennom alle filene i mappen av typen ***.pdat**. [3]

Analyseprogrammet startes som beskrevet i Seksjon 3. I dette tilfellet vil det spørres om stien til mappen med alle ***.pdat**-filene.

[3] Dersom man ønsker å sammenlikne flere serier samtidig, lagres disse i samme katalog.

Her vil programmet lese seg gjennom filene i mappen og generere to datafiler:

1) Fil som inneholder tabellen med resultatene
2) Fil med en samling av rådataene til testene.

Den første filen (som inneholder resultattabellen) vil hete **ASTM_<Standard>_table.dat**. Den andre filen (med rådataene) vil hete **ASTM_<Standard>_data.dat**, hvor **<Standard>**, som over, representerer navnet på standarden. Disse filene vil havne i samme mappen som alle ***.pdat**-filene.

6 Tester

I denne seksjonen vil de ulike testene det er laget analyseverktøy for, beskrives i mer detalj. Det henvises til tidligere seksjoner for de delene av analysene og bruken av programmene som er felles for alle testene.

6.1 ASTM C 297

6.1.1 Generelt

Programmene **C297test.py** og **C297resulttable.py** er laget som analyseverktøy for standarden ASTM C 297 – Standard Test Method for Flatwise Tensile Strength of Sandwich Constructions. Testen brukes for å bestemme strekkfastheten til sandwich-konstruksjoner.

6.1.2 Spesielt om data fra testene

Formatet på de åtte første linjene i en typisk inndatafil for denne testen er vist under.

1. MTS793|MPT|ENG|1|2|.|/|:|44|1|1|A
2.
3. Data Acquisition DataFangst Time: 151.53809 Sec 08/05/2012 13:15:37
4. Time Ch 1 Displacement Ch 1 Force
5. Sec mm N
6. 0.11425781 8.5681677e-05 1.2050189
7. 0.21386719 0.00091046095 2.7285168
8. 0.31347656 0.0022672117 -0.089243524

Inndatafilene fra disse testene heter alle **specimen.dat**. Filene består av tre kolonner: 1) Tid, 2) kraft og 3) deformasjon. Linje fire inneholder kolonnenavnene, men kan også inneholde flere ord. Linje fire må imidlertid inneholde nøkkelordene ”Time”, ”Force” og ”Displacement”; rekkefølgen kan dog være vilkårlig, avhengig av hvordan testen er utført. Nøkkelordene er viktige, da programmet bruker disse ordene for å gjenkjenne hvilken kolonne som inneholder hvilken informasjon. Linje fem lister kun de tilhørende enhetene, det vil si ”Sec”, ”N” eller ”kN”

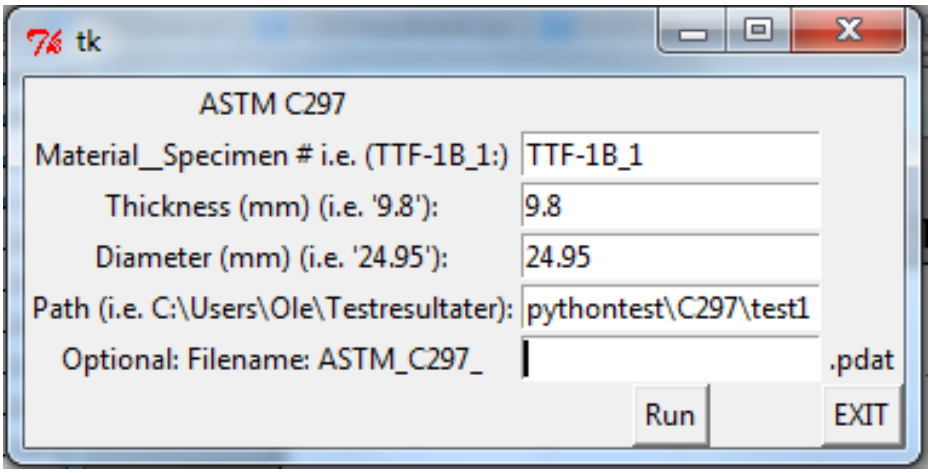
og ”mm”. Dersom kraften er gitt i kN , blir alle verdiene multiplisert med 1000 , og programmet jobber videre med N .

6.1.3 Fil over på ønsket format

Åpne PyLab-vinduet og flytt til katalogen hvor Python-programmet **C297test.py** ligger på maskinen. For å kjøre programmet skriver man

```
execfile("C297test.py")
```

Det vil nå komme opp en GUI, hvor programmet ber om inndata. GUIen for denne testen er vist i Figur 6.1.



*Figur 6.1 GUI for **C297test.py***

Inndataene som må fylles inn er:

- Navn på den spesifikke prøven (*Material_Specimen #*)
- Tykkelse (*Thickness (mm)*)
- Diameter (*Diameter (mm)*)
- Stien, det vil si hvor datafilen (**specimen.dat**) ligger (*Path*)
- Del av filnavn som man vil skal stå mellom ”ASTM_C297_” og ”.pdat” (*Filename*), i det som blir resultatfilen fra kjøringen.

Merk følgende:

- Dersom man vil at det man fyller inn for navn på den spesifikke prøven skal være det samme som skal stå mellom ”ASTM_C297_” og ”.pdat”, lar man filnavnfeltet stå tomt.
- Tykkelsen og diameteren må gis som rene tall uten benevning. Flyttall representeres ved punktum (”.”) og ikke komma (”,”).
- Det man fyller inn for *Material_Specimen #*, er det som kommer til å stå i tabellen og i figurene, for å kunne skille prøvene fra hverandre.

Etter at man har skrevet inn alle verdiene, trykker man *Run*.

Det vil nå komme opp et plott for sammenhengen mellom kraft og deformasjon for prøven, og programmet har videre lagret en fil med data fra prøven i mappen hvor **specimen.dat**- filen ligger, altså den angitte stien. Denne filen vil hete **ASTM_C297_<filnavn>.pdat**, hvor **<filnavn>** er det man skrev inn i *Filename*-boksen, eller i *Material_Specimen #* dersom man lot *Filename* stå tomt.

Filen **ASTM_C297_<filnavn>.pdat** inneholder i dette tilfellet dato for testen, tabell med navn på prøven, tykkelse, maksimal kraft, diameter, tverrsnittareal og strekkfasthet, samt kraft- og deformasjonsdataene.

Hvis man ønsker å lagre plottet, trykker man på lagreknappen under plottet. Her kan man også zoome inn om det er ønskelig, og man kan også lagre de zoomede versjonene. Merk at plottene må lukkes mellom hver gang for å forhindre at neste graf tegnes i samme vindu. Noen ganger vil ikke plottet lukke seg, men for å forhindre at neste graf tegnes i samme vindu, holder det at man har forsøkt å lukke plottet.

Det er da klart for å kjøre programmet på nytt for neste prøve. Det gjør man ved å skrive inn nye verdier i GUIen, og igjen trykke *Run.* Når man har kjørt programmet for alle prøvene, avsluttes GUIen ved å trykke på *Exit*.

6.1.4 Analyse av flere testprøver

Når man er ferdig med å kjøre alle prøvene med programmet **C297test.py**, må man gå inn i hver mappe og kopiere (eller flytte) **ASTM_C297_<filnavn>.pdat** -filene inn i en mappe hvor man vil ha resultattabellen. Her kan det ikke ligge resultatfiler fra tidligere tester, da programmet vil lese gjennom alle filene i mappen som slutter med ***.pdat** (såfremt man ikke ønsker å analysere flere sett med prøver i samme analyse).

Deretter kjører man programmet **C297resulttable.py**, ved å skrive

```
execfile("C297resulttable.py ")
```

Det vil da komme opp et vindu hvor man må angi stien til mappen hvor man har lagt alle resultatfilene. Deretter trykker man *Run*. Programmet leser så gjennom dataene i filene, og lager et kraft-deformasjons-plott for alle prøvene. Plottet kan nå lagres og zoomes i.

Det blir også laget to filer:

- **ASTM_C297_table.dat** – inneholder resultattabellen
- **ASTM_C297_data.dat** – inneholder kraft- og deformasjonsdata for alle prøvene

Filen med resultattabellen inneholder datoen for analysen, navn på de spesifikke prøvene, tykkelse, diameter, maksimal kraft, tverrsnittareal, strekkfasthet, samt gjennomsnittene av disse og de empiriske standardavvikene.

De to ***.dat**-filene ligger i mappen hvor resultatfilene fra kjøringene av **C297test.py** ble lagt.

6.1.5 Utregninger for ASTM C 297

Følgende utregninger er benyttet i analysen. Uttrykkene er i utgangspunktet hentet fra standarden.

6.1.5.1 Strekkfasthet

Maksimal kraft er største verdi i kolonnen med kraftverdier fra datafilen.

Programmet regner ut strekkfastheten ved hjelp av formelen

$$F_z^{ftu} = P^{\max} / A \tag{6.1}$$

fra likning (1) i standarden, hvor F_z^{ftu} er bruddstrekkfasthet (MPa), $P^{\max}$ er maksimal kraft (N) og A (mm^2) er tverrsnittarealet regnet ut fra diameteren gitt av bruker; $A = \pi(D/2)^2$.

6.1.5.2 Statistikk

Gjennomsnittsverdien av en størrelse x er regnet ut med utgangspunkt i likning (2) i standarden[4]

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.2}$$

hvor $\overline{x}$ er gjennomsnittet av alle x $(i = 1, n)$ og n er antall tester.

Standardavviket til en størrelse x er regnet som et empirisk standardavvik, og er gitt fra likning (3) i standarden,

$$S_{n-1} = \sqrt{(\sum_{i=1}^{n} x_i^2 - n\overline{x}^2) / (n-1)} \tag{6.3}$$

hvor S_{n-1} er det empiriske standardavviket, $\overline{x}$ er gjennomsnittet av x og n er antall tester.

[4] Likning (2) i standarden er sannsynligvis feil, da den ikke deler på antall tester. I (6.2) og i Python-programmet er derfor denne faktoren lagt til.

6.2 Test ASTM D 3039

6.2.1 Generelt

Programmene **D3039test.py** og **D3039resulttable.py** er laget som analyseverktøy for standarden ASTM D 3039 – Standard Test Method for Tensile Properties of Fiber-Resin Composites. Testen brukes for å bestemme strekkegenskaper (det vil si strekkspenning, strekktøyning og strekkmodul) til fiberarmerte polymerbaserte kompositter. For å måle tøyningen brukes et biaksielt ekstensometer.

6.2.2 Spesielt om data fra testene

Formatet på de åtte første linjene i inndatafilene for denne testen er vist under.

1. MTS793|MPT|ENG|1|2|.|/|:|44|1|1|A
2.
3. Data Acquisition Data Time: 74.220703 Sec 08/10/2012 14:23:54
4. Time Ch 1 Displacement Ch 1 Force Ch 1 Strain Ch 1 Transvers
5. Sec mm N mm/mm mm
6. 0.5078125 0.1392588 575.52295 0.00060729444 0
7. 1.0078125 0.30559599 1428.2423 0.0016341815 0
8. 1.5078125 0.47432929 2151.8872 0.0028530061 0

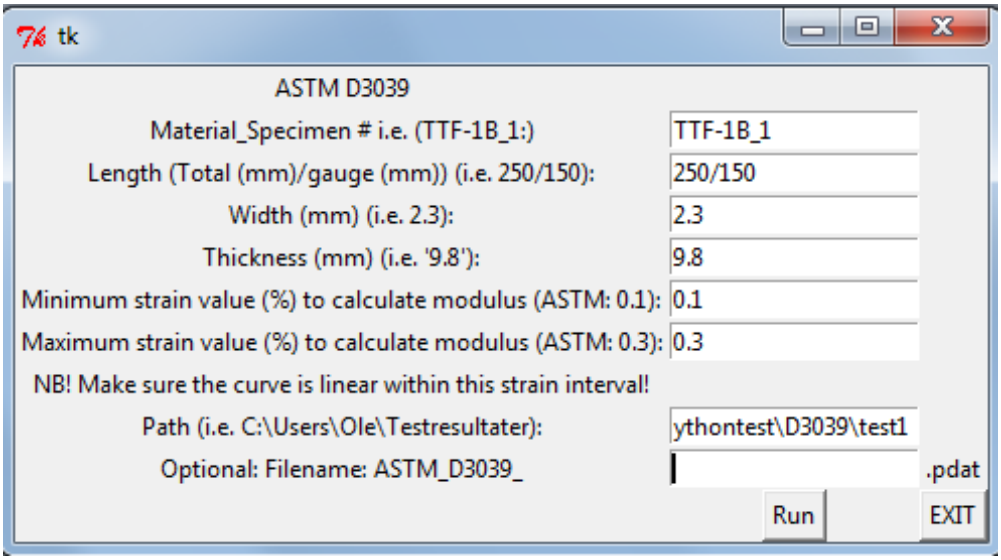
Inndatafilene fra disse testene heter alle **specimen.dat**. Filene består alle av fem kolonner: 1) Tid, 2) kraft, 3) deformasjon i lengderetningen, 4) tøyning og 5) deformasjon i tverretningen. Linje fire inneholder kolonnenavnene, men kan inneholde flere ord. Linje fire må imidlertid inneholde nøkkelordene ”Time”, ”Displacement”, ”Force”, ”Strain” og ”Transvers”; rekkefølgen kan dog være vilkårlig, avhengig av hvordan testen er utført. Nøkkelordene er viktige, da programmet bruker disse ordene for å gjenkjenne hvilken kolonne som inneholder hvilken informasjon. Linje fem lister kun de tilhørende enhetene, det vil si ”Sec”, ”mm”, ”N” eller ”kN”, ”%” eller ”mm/mm” og ”mm”. Dersom kraften er gitt i kN, blir alle verdiene multiplisert med 1000, og programmet jobber videre med N. Dersom tøyning er gitt i mm/mm multipliseres alle verdiene med 100, og programmet jobber videre i prosent.

6.2.3 Fil over på ønsket format

Åpne PyLab-vinduet og flytt til katalogen hvor Python-programmet **D3039test.py** ligger på maskinen. For å kjøre programmet skriver man

```
execfile("D3039test.py")
```

Det vil nå komme opp en GUI, hvor programmet ber om inndata. GUIen er vist i Figur 6.2.



*Figur 6.2 GUI for **D3039test.py***

Inndataene som må fylles inn er:

- Navn på den spesifikke prøven (*Material_Specimen #*)
- Lengde – total og målelengde (*Length (Total (mm)/Gauge (mm))*)
- Bredde (*Width (mm)*)
- Tykkelse (*Thickness (mm)*)
- Minste tøyningsverdi (*Minimum strain value (%)*)
- Største tøyningsverdi (*Maximum strain value (%)*)
- Stien, det vil si hvor datafilen (**specimen.dat**) ligger (*Path*)
- Det man vil skal stå mellom ”ASTM_D3039_” og ”.pdat” (*Filename*), som blir resultatfilen fra kjøringen.

Merk følgende:

- Dersom man vil at det man fyller inn for navn på den spesifikke prøven skal være det samme som skal stå mellom ”ASTM_D3039_” og ”.pdat”, lar man filnavnfeltet stå tomt.
- Lengdene, bredden, tykkelsen og tøyningsverdiene må gis som rene tall uten benevning. Flyttall representeres ved punktum (”.”) og ikke komma (”,”).
- *Material_Specimen #* er det som kommer til å stå i tabellen og i figurene, for å skille prøvene fra hverandre.

Etter at man har skrevet inn alle verdiene, klikker man *Run*.

Man vil nå få opp et plott av sammenhengen mellom spenning og tøyning for prøven. For å regne ut modulen brukes intervallet fra minste til største tøyningsverdi. Derfor må spenning-tøyningskurven være lineær innenfor dette intervallet. For å sjekke om kurven er lineær innenfor

dette intervallet, ser man på plottet programmet genererer. Dersom kurven ikke er lineær innenfor intervallet, kjører man programmet på nytt, og da med et nytt intervall hvor kurven er lineær. Programmet har videre lagret en fil med data fra prøven i mappen hvor **specimen.dat**- filen ligger, altså den angitte stien. Denne filen vil hete **ASTM_D3039_<filnavn>.pdat**, hvor **<filnavn>** er det man skrev inn i *Filename*-boksen, eller i *Material_Specimen #* dersom man lot *Filename* stå tom.

Filen **ASTM_D3039_<filnavn>.pdat** inneholder i dette tilfellet dato for testen, tabell med navn på prøven, lengde (total/målelengde), bredde, tykkelse, strekkfasthet, bruddstrekktøyning, strekkmodul, deformasjon og maksimal kraft, samt data om deformasjon i lengderetning, kraft, tøyning, deformasjon i tverretning og spenning.

Hvis man ønsker å lagre plottet, trykker man på lagreknappen under plottet. Her kan man også zoome inn om det er ønskelig, og man kan også lagre de zoomede versjonene. Merk at plottene må lukkes mellom hver gang for å forhindre at neste graf tegnes i samme vindu. Noen ganger vil ikke plottet lukke seg, men for å forhindre at neste graf tegnes i samme vindu, holder det at man har forsøkt å lukke plottet.

Det er da klart for å kjøre programmet på nytt for neste prøve. Det gjør man ved å skrive inn nye verdier i GUIen, og igjen trykke *Run.* Når man har kjørt programmet på alle prøvene, avsluttes GUIen ved å trykke på *Exit*.

6.2.4 Analyse av flere testprøver

Når man er ferdig med å kjøre alle prøvene med programmet **D3039test.py**, må man gå inn i hver mappe og kopiere (eller flytte) **ASTM_D3039_<filnavn>.pdat**-filene inn i en mappe hvor man vil ha resultattabellen. Her kan det ikke ligge resultatfiler fra tidligere tester, da programmet vil lese gjennom alle filene i mappen som slutter med ***.pdat** (såfremt man ikke ønsker å analysere flere sett med prøver i samme analyse).

Deretter kjører man programmet **D3039resulttable.py** ved å skrive

```
execfile("D3039resulttable.py ")
```

Man vil da få opp et vindu hvor man må angi stien til mappen hvor man har lagt alle resultatfilene. Deretter trykker man *Run*. Programmet leser så gjennom dataene i filene, og lager et spenning-tøynings-plott av alle prøvene. Plottet kan nå lagres og zoomes i.

Det blir også laget to filer:

- **ASTM_D3039_table.dat –** inneholder resultattabellen
- **ASTM_D3039_data.dat** – inneholder data om deformasjon i lengderetning, kraft, tøyning, deformasjon i tverretning og spenning for alle prøvene.

Filen med resultattabellen inneholder datoen for analysen, navn på de spesifikke prøvene, lengde (total/målelengde), bredde, tykkelse, strekkfasthet, maksimal strekktøyning og strekkmodul, samt gjennomsnittene av disse og de empiriske standardavvikene.

De to ***.dat**-filene ligger i mappen med resultatfilene fra kjøringene av **D3039test.py**.

6.2.5 Utregninger for ASTM D 3039

Følgende utregninger er benyttet i analysen. Uttrykkene er i utgangspunktet hentet fra standarden.

6.2.5.1 Maksimal kraft, deformasjon og maksimal strekktøyning

Maksimal kraft er største verdi i kolonnen med kraftverdier fra datafilen. Deformasjonen er gitt som deformasjonen i punktet hvor maksimal kraft inntreffer.

Maksimal strekktøyning er den maksimale verdien i kolonnen med tøyningsverdier fra datafilen.

6.2.5.2 Strekkfasthet

Programmet regner ut strekkfastheten ved hjelp av følgende formel fra standarden

$$S = P / bd \tag{6.4}$$

hvor S er maksimal strekkfasthet (MPa), P er maksimal kraft (N) og bd (mm^2) er tverrsnittarealet, regnet ut fra tykkelse og bredde gitt av bruker.

6.2.5.3 Spenning

Spenningen regnes ut ved

$$\sigma = P / bd \tag{6.5}$$

hvor σ er spenningen (MPa), P er kraften (N) og bd (mm^2) er tverrsnittarealet. Programmet lager en liste over spenningsverdier regnet ut fra kolonnen med kraftverdier.

6.2.5.4 Elastisitetsmodulen

Elastisitetsmodulen regnes ut ved å finne de to punktene i kolonnen med tøyningsverdier som ligger nærmest henholdsvis minste og største tøyningsverdi, som ble gitt som inndata. Deretter finner programmet de tilhørende punktene i spenningslisten, og vi får

$$E = 100 \frac{\Delta\sigma}{\Delta\epsilon} \tag{6.6}$$

hvor E er elastisitetsmodulen (MPa), $\Delta\sigma$ er differansen i spenningen (MPa) og $\Delta\epsilon$ er differansen i tøyningen (%) over det gitte intervallet. Faktoren 100 kommer av at tøyningen er gitt i prosent.

6.2.5.5 Statistikk

Gjennomsnittsverdien av en størrelse x er regnet ut ved

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.7}$$

hvor $\overline{x}$ er gjennomsnittet av alle x $(i = 1, n)$ og n er antall tester.

Standardavviket til en størrelse x er regnet som et empirisk standardavvik, gitt ved formelen

$$S_{n-1} = \sqrt{(\sum_{i=1}^{n} x_i^2 - n\overline{x}^2)/(n-1)} \tag{6.8}$$

hvor S_{n-1} er det empiriske standardavviket, $\overline{x}$ er gjennomsnittet av x og n er antall tester.

6.3 Test ASTM D 3518

6.3.1 Generelt

Programmene **D3518test.py** og **D3518resulttable.py** er laget som analyseverktøy for standarden ASTM D 3518 – Standard Test Method for In-Plane Shear Response of Polymer Matrix Composite Materials by Tensile Test of a ± 45º Laminate. Testen brukes for å bestemme skjæregenskaper (det vil si maksimal skjærspenning, maksimal skjærtøyning, skjærmodul og forskjøvet skjærstyrke (engelsk ”offset shear strength”)) til fiberarmerte polymerer. For å måle tøyningen brukes et biaksielt ekstensometer.

6.3.2 Spesielt om data fra testene

Formatet på de åtte første linjene i inndatafilene for denne testen er vist under.

1. MTS793|MPT|ENG|1|2|.|/|:|44|1|1|A
2.
3. Data Acquisition Data Time: 74.220703 Sec 08/10/2012 14:23:54
4. Time Ch 1 Displacement Ch 1 Force Ch 1 Strain Ch 1 Transvers
5. Sec mm N mm/mm mm
6. 0.5078125 0.1392588 575.52295 0.00060729444 0
7. 1.0078125 0.30559599 1428.2423 0.0016341815 0
8. 1.5078125 0.47432929 2151.8872 0.0028530061 0

Inndatafilene fra disse testene heter alle **specimen.dat**. Filene består av fem kolonner: 1) Tid, 2) kraft, 3) deformasjon i lengderetningen, 4) tøyning og 5) deformasjon i tverretningen. Linje fire inneholder kolonnenavnene, men kan inneholde flere ord. Linje fire må imidlertid inneholde nøkkelordene ”Time”, ”Displacement”, ”Force”, ”Strain” og ”Transvers”; rekkefølgen kan dog være vilkårlig, avhengig av hvordan testen er utført. Nøkkelordene er viktige, da programmet

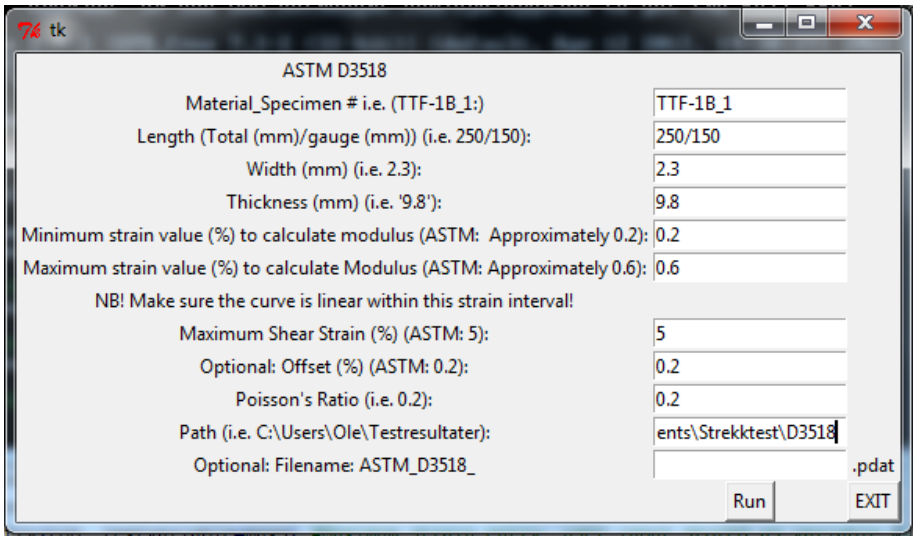
bruker disse ordene for å gjenkjenne hvilken kolonne som inneholder hvilken informasjon. Linje fem lister kun de tilhørende enhetene, det vil si ”Sec”, ”mm”, ”N” eller ”kN”, ”%” eller ”mm/mm” og ”mm”. Dersom kraften er gitt i kN, blir alle verdiene multiplisert med 1000, og programmet jobber videre med N. Dersom tøyning er gitt i mm/mm multipliseres alle verdiene med 100, og programmet jobber videre i prosent.

6.3.3 Fil over på ønsket format

Åpne PyLab-vinduet og flytt til katalogen hvor Python-programmet **D3518test.py** ligger på maskinen. For å kjøre programmet skriver man

```
execfile("D3518test.py")
```

Det vil nå komme opp en GUI, hvor programmet ber om inndata. GUIen er vist i Figur 6.3.



*Figur 6.3 GUI for **D3518test.py***

Inndataene som må fylles inn er:

- Navn på den spesifikke prøven (*Material_Specimen #*),
- Lengde – total og målelengde (*Length (Total (mm)/Gauge (mm))*),
- Bredde (*Width (mm)*),
- Tykkelse (*Thickness (mm)*),
- Minste tøyningsverdi (*Minimum strain value (%)*),
- Største tøyningsverdi (*Maximum strain value (%)*),
- Maksimal skjærtøyning (*Maximum Shear Strain (%)*),
- Poisson-tallet (*Poisson's Ratio*),
- Forskyvning (*Offset*),
- Stien, det vil si hvor datafilen (**specimen.dat**) ligger (*Path*)
- Det man vil skal stå mellom ”ASTM_D3518_” og ”.pdat” (*Filename*), som blir resultatfilen fra kjøringen

Merk følgende:

- Dersom man vil at det man fyller inn for navn på den spesifikke prøven skal være det som skal stå mellom ”ASTM_D3518_” og ”.pdat”, lar man filnavnfeltet stå tomt
- Lengdene, bredden, tykkelsen, minste skjærtøyningsverdi, største skjærtøyningsverdi, maksimal skjærtøyning, forskyvning og Poisson-tallet må gis som rene tall uten benevning. Flyttall representeres ved punktum (”.”) og ikke komma (”,”)
- *Material_Specimen #* er det som kommer til å stå i tabellen og i figurene, for å skille prøvene fra hverandre
- Maksimal skjærtøyning er den skjærtøyningen man vil stoppe testen ved dersom den ikke har stoppet før. I standarden er denne satt til 5 %.
- Forskyvning er der den rette linjen med stigningstall lik modulen starter på skjærtøyningsaksen. Skjæringspunktet mellom denne linjen og skjærspenning-skjærtøyningskurven er forskjøvet skjærstyrke (”offset shear strength”). Dersom man ikke ønsker å rapportere forskjøvet skjærstyrke, lar man *Offset*-feltet stå tomt.

Etter at man har skrevet inn alle verdiene, klikker man *Run*.

Man vil nå få opp et plott av skjærspenning-skjærtøyning for prøven. For å regne ut modulen, brukes intervallet fra minste til største tøyningsverdi. Derfor må spenning-tøyningskurven være lineær innenfor dette intervallet. Om kurven er lineær eller ikke, kan man se ut fra plottet programmet genererer. Dersom man ga et intervall hvor kurven ikke er lineær, kan man kjøre programmet på nytt med et intervall hvor man vet at kurven er lineær.

Videre har programmet lagret en fil med data fra prøven i mappen hvor **specimen.dat**- filen ligger, altså den angitte stien. Denne filen vil hete **ASTM_D3518_<filnavn>.pdat**, hvor **<filnavn>** er det man skrev inn i *Filename*-boksen, eller i *Material_Specimen #* dersom man lot *Filename* stå tom. Filen **ASTM_D3518_<filnavn>.pdat** inneholder i dette tilfellet dato for testen, tabell med navn på prøven, lengde (total/målelengde), bredde, tykkelse, maksimal skjærspenning, maksimal skjærtøyning, skjærmodul, deformasjon, maksimal kraft, forskjøvet skjærstyrke, såfremt man ikke lot *Offset*-feltet stå tomt, og Poisson-tallet, samt data om deformasjon i lengderetning, kraft, tøyning i lengderetning, deformasjon i tverretning, skjærtøyning og skjærspenning.

Hvis man ønsker å lagre plottet, trykker man på lagreknappen under plottet. Her kan man også zoome inn om det er ønskelig, og man kan også lagre de zoomede versjonene. Merk at plottene må lukkes mellom hver gang for å forhindre at neste graf tegnes i samme vindu. Noen ganger vil ikke plottet lukke seg, men for å forhindre at neste graf tegnes i samme vindu, holder det at man har forsøkt å lukke plottet.

Det er da klart for å kjøre programmet på nytt for neste prøve. Det gjør man ved å skrive inn nye verdier i GUIen, og igjen trykke *Run*. Når man har kjørt programmet på alle prøvene, avsluttes GUIen ved å trykke på *Exit*.

6.3.4 Analyse av flere testprøver

Når man er ferdig med å kjøre alle prøvene med programmet **D3518test.py**, må man gå inn i hver mappe og kopiere (eller flytte) **ASTM_D3518_<filnavn>.pdat** -filene inn i en mappe hvor man vil ha resultattabellen. Her kan det ikke ligge resultatfiler fra tidligere tester, da programmet vil lese gjennom alle filene i mappen som slutter med ***.pdat** (såfremt man ikke ønsker å analysere flere sett med prøver i samme analyse).

Deretter kan man kjøre programmet **D3518resulttable.py**, ved å skrive

execfile("D3518resulttable.py ")

Man vil da få opp et vindu hvor man må angi stien til mappen hvor man har lagt alle resultatfilene. Deretter trykker man *Run*. Programmet leser så gjennom dataene i filene, og lager et skjærspenning-skjærtøynings-plott av alle prøvene. Plottet kan nå lagres og zoomes i.

Det blir også laget to filer:

- **ASTM_D3518_table.dat** – inneholder resultattabellen
- **ASTM_D3518_data.dat** – inneholder data om deformasjon i lengderetning, kraft, tøyning i lengderetning, deformasjon i tverretning, skjærtøyning og skjærspenning for alle prøvene

Filen med resultattabellen inneholder datoen for analysen, navn på de spesifikke prøvene, lengde (total/målelengde), bredde, tykkelse, maksimal skjærspenning, maksimal skjærtøyning, skjærmodul og forskjøvet skjærstyrke, såfremt man ikke lot *Offset*-feltet stå tomt, samt gjennomsnittene av disse og de empiriske standardavvikene.

De to ***.dat**-filene ligger i mappen med resultatfilene fra kjøringen av **D3518test.py**.

6.3.5 Utregninger for ASTM D 3518

Følgende utregninger er benyttet i analysen. Uttrykkene er i utgangspunktet hentet fra standarden.

6.3.5.1 Skjærtøyning

Skjærtøyningen regnes ut ved hjelp av Poisson-tallet

$$\gamma_{12} = \epsilon_l(1+\nu) \tag{6.9}$$

hvor γ_{12} er skjærtøyning (%), ϵ_l er tøyning i lengderetning (%) og ν er Poisson-tallet.

Formelen kommer fra

$$\nu = -\frac{\epsilon_t}{\epsilon_l} \Rightarrow \epsilon_t = -\nu\epsilon_l \tag{6.10}$$

$$\gamma_{12} = \epsilon_l - \epsilon_t = \epsilon_l(1+\nu) \tag{6.11}$$

hvor

$$\gamma_{12} = \epsilon_l - \epsilon_t \tag{6.12}$$

er likning (3) i standard, γ_{12} er skjærtøyning (%), ϵ_l er tøyning i lengderetning (%), ϵ_t er tøyning i tverretning (%) og ν er Poisson-tallet. Programmet lager en liste over skjærtøyningsverdier ut fra kolonnen med tøyningsverdier fra datafilen.

6.3.5.2 Skjærspenning

Skjærspenningen regnes ut fra likning (2) i standarden

$$\tau_{12} = P / 2A \tag{6.13}$$

hvor τ_{12} er skjærspenningen (MPa), P er kraften (N) og A (mm^2) er tverrsnittarealet gitt ved bredde multiplisert med tykkelse.

Programmet lager en liste over skjærspenningsverdier regnet ut fra kolonnen med kraftverdier fra datafilen.

6.3.5.3 Skjærmodulen

Skjærmodulen er regnet ut ved at programmet finner de to skjærtøyningsverdiene som ligger nærmest henholdsvis minste skjærtøyning og største skjærtøyning gitt av bruker. Deretter finner programmet de tilhørende skjærspenningsverdiene, og vi får fra likning (5) i standarden,

$$G_{12}^{chord} = 100\frac{\Delta\tau_{12}}{\Delta\gamma_{12}} \tag{6.14}$$

hvor G_{12}^{chord} er skjærmodulen (MPa), τ_{12} er skjærspenningen (MPa) og γ_{12} er skjærtøyningen (%). Faktoren 100 kommer av at skjærtøyningen er gitt i prosent.

6.3.5.4 Forskjøvet skjærstyrke

Forskjøvet skjærstyrke regnes ut ved å lage en linje

$$y = G_{12}^{chord}(\gamma_{12} - offset)/100 \tag{6.15}$$

hvor G_{12}^{chord} er skjærmodulen (*MPa*), γ_{12} er skjærtøyningen (%) og *offset* er forskyvningen gitt av bruker. Faktoren $1/100$ kommer av at skjærtøyningen er gitt i prosent. Deretter regner programmet ut skjæringen mellom denne linjen og skjærspenning-skjærtøynings-kurven ved å finne ved hvilke skjærtøyningspunkter vi har at linjen ligger *under* skjærspenningen i ett punkt, og *over* skjærspenningen i det påfølgende punktet. Deretter tas gjennomsnittet av skjærspenningene i disse to punktene. Dette gjennomsnittet rapporteres som forskjøvet skjærfasthet.

6.3.5.5 Maksimumsverdier

For å finne maksimal skjærtøyning, maksimal skjærspenning, maksimal kraft og maksimal deformasjon, bruker programmet følgende fremgangsmåte:

1) Finner maksimal skjærtøyningsverdi i listen med de utregnede skjærtøyningsverdiene

Dersom maksimal skjærtøyning er *større enn* angitt maksimal skjærtøyning:

2) Maksimal kraft blir satt til den maksimale kraften i kolonnen med kraftverdier opp til og med det punktet hvor skjærtøyning større enn angitt maksimal skjærtøyning inntreffer for første gang
3) Maksimal skjærspenning blir satt til skjærspenningen ved denne maksimale kraften
4) Maksimal deformasjon blir satt til deformasjonen ved denne maksimale kraften
5) Maksimal skjærtøyning blir satt til den minste verdien av angitt maksimal skjærtøyning og skjærtøyningen ved denne maksimale skjærspenningen
6) Kurvene og dataene som skrives ut til filene blir stoppet i det punktet hvor skjærtøyningen overstiger maksimal angitt skjærtøyning for første gang

Dersom maksimal skjærtøyning derimot er *mindre enn* angitt maksimal skjærtøyning:

2) Maksimal kraft er maksimal kraft i kolonnen med kraftverdier
3) Maksimal skjærspenning er maksimal spenning i listen med utregnede skjærspenningsverdier
4) Maksimal deformasjon blir satt til å være deformasjonen ved maksimal kraft
5) Maksimal skjærtøyning blir satt til tøyningen ved maksimal skjærspenning
6) Kurvene og dataene stoppes ikke

6.3.5.6 Statistikk

Gjennomsnittsverdien av en størrelse x er regnet ut fra likning (7) i standarden

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (6.16)$$

hvor $\overline{x}$ er gjennomsnittet av alle x $(i = 1, n)$ og n er antall tester.

Standardavviket til en størrelse x er regnet som et empirisk standardavvik, gitt fra likning (8) i standarden,

$$S_{n-1} = \sqrt{(\sum_{i=1}^{n} x_i^2 - n\overline{x}^2)/(n-1)} \tag{6.17}$$

hvor S_{n-1} er det empiriske standardavviket, $\overline{x}$ er gjennomsnittet av x og n er antall tester.

6.4 Test ASTM D 790

6.4.1 Generelt

Programmene **D790test.py** og **D790resulttable.py** er laget som analyseverktøy for standarden ASTM D 790 – Standard Test Method for Flexural Properties of Unreinforced and Reinforced Plastics and Electrical Insulating Materials. Testen brukes for å bestemme bøyningsegenskaper (det vil si bøyefasthet, maksimal tøyning som følge av bøyning og tangentmodul) til armerte og uarmerte plaster og elektrisk isolerende materialer.

6.4.2 Spesielt om data fra testene

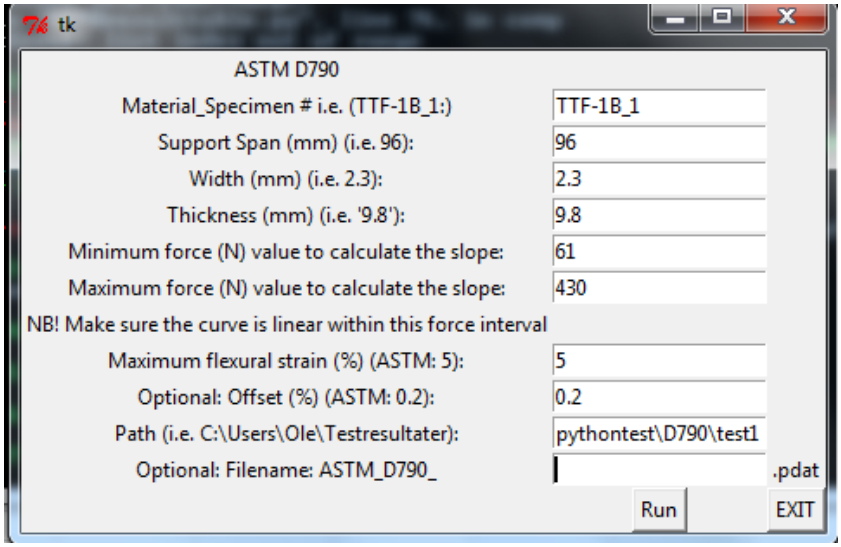
Formatet på de første åtte linjene i inndatafilene for denne testen er vist under.

1. MTS793|MPT|ENG|1|2|.|/|:|44|1|1|A
2.
3. Data Acquisition DataFangst Time: 151.53809 Sec 08/05/2012 13:15:37
4. Time Ch 1 Displacement Ch 1 Force
5. Sec mm N
6. 0.11425781 8.5681677e-05 1.2050189
7. 0.21386719 0.00091046095 2.7285168
8. 0.31347656 0.0022672117 -0.089243524

Inndatafilene fra disse testene heter alle **specimen.dat**. Filene består av tre kolonner: 1) Tid, 2) kraft og 3) deformasjon. Linje fire inneholder kolonnenavnene, men kan inneholde flere ord. Linje 4 må imidlertid inneholde nøkkelordene ”Time”, ”Displacement” og ”Force”; rekkefølgen kan dog være vilkårlig, avhengig av hvordan testen er utført. Nøkkelordene er viktige, da programmet bruker disse ordene for å gjenkjenne hvilken kolonne som inneholder hvilken informasjon. Linje fem lister kun de tilhørende enhetene, det vil si ”Sec”, ”mm” og ”N” eller ”kN”. Dersom kraften er gitt i kN, blir alle verdiene multiplisert med 1000, og programmet jobber videre med N.

6.4.3 Fil over på ønsket format

Åpne PyLab-vinduet og flytt til katalogen hvor Python-programmet **D790test.py** ligger på maskinen. For å kjøre programmet skriver man execfile("D790test.py")
Det vil nå komme opp en GUI, hvor programmet ber om inndata. GUIen er vist i Figur 6.4.



*Figur 6.4 GUI for **D790test.py***

Inndataene som må fylles ut er:

- Navn på den spesifikke prøven (*Material_Specimen #),*
- Spennvidde (*Support Span (mm)*),
- Bredde (*Width (mm)*),
- Tykkelse (*Thickness (mm)*),
- Minste kraftverdi (*Minimum force (N)*),
- Største kraftverdi (*Maximum force (N)*),
- Maksimal tøyning (*Maximum flexural strain (%)*), forskyvning (*Offset (%)*),
- Stien, det vil si hvor datafilen (**specimen.dat**) ligger (*Path*)
- Det man vil skal stå mellom ”ASTM_D790_” og ”.pdat” (*Filename*), som blir resultatfilen fra kjøringen

Merk følgende:

- Dersom man vil at det man fyller inn for navn på den spesifikke prøven skal være det som skal stå mellom ”ASTM_D790_” og ”.pdat”, lar man filnavnfeltet stå tomt
- Spennvidde, bredde, minste kraft, største kraft, maksimal tøyning, forskyvning og tykkelse må gis som rene tall uten benevning. Flyttall representeres ved punktum (”.”) og ikke komma (”,”)
- *Material_Specimen #* er det som kommer til å stå i tabellen og i figurene, for å skille prøvene fra hverandre
- Maksimal tøyning er den tøyningen man vil stoppe testen ved dersom den ikke har stoppet før. I standarden er denne satt til 5 %
- Forskyvningen er der den rette linjen med stigningstall lik modulen starter på skjærtøyningsaksen. Skjæringspunktet mellom denne linjen og skjærspenning-skjærtøynings-kurven er forskjøvet flytstyrke (engelsk ”offset yield strength”). Dersom man ikke vil rapportere forskjøvet flytstyrke lar man *Offset*-feltet stå tomt.

Etter at man har skrevet inn alle verdiene, klikker man *Run*.

Man vil nå få opp et plott av sammenhengen mellom kraft og nedbøyning for prøven. For å regne ut modulen, brukes stigningstallet på kraft-nedbøynings-plottet. Intervallet fra minste til største kraft brukes for å regne ut dette stigningstallet. Derfor må kurven være lineær innenfor dette intervallet. Om kurven er lineær innenfor intervallet, sjekker man ved å se på plottet programmet genererer. Dersom man valgte et intervall hvor kurven ikke er lineær, kan man kjøre på nytt med et nytt intervall hvor kurven er lineær.

Programmet har videre lagret en fil med data fra prøven i mappen hvor **specimen.dat**- filen ligger, altså den angitte stien. Denne filen vil hete **ASTM_D790_<filnavn>.pdat**, hvor **<filnavn>** er det man skrev inn i *Filename*-boksen, eller i *Material_Specimen #* dersom man lot *Filename* stå tomt.

Filen **ASTM_D790_<filnavn>.pdat** inneholder i dette tilfellet dato for testen, tabell med navn på prøven, spennvidde, bredde, tykkelse, bøyefasthet, maksimal tøyning som følge av bøyning, tangentmodul, nedbøyning, maksimal kraft, forskjøvet flytstyrke, såfremt man ikke lot *Offset*-feltet stå tomt, og stigningstallet på kraft-nedbøyningsplottet, samt nedbøynings- og kraftdata.

Hvis man ønsker å lagre plottet, trykker man på lagreknappen under plottet. Her kan man også zoome inn om det er ønskelig, og man kan også lagre de zoomede versjonene. Merk at plottene må lukkes mellom hver gang for å forhindre at neste graf tegnes i samme vindu. Noen ganger vil ikke plottet lukke seg, men for å forhindre at neste graf tegnes i samme vindu, holder det at man har forsøkt å lukke plottet.

Det er da klart for å kjøre programmet på nytt for neste prøve. Det gjør man ved å skrive inn nye verdier i GUIen, og trykker *Run.* Når man har kjørt programmet på alle prøvene, avsluttes GUIen ved å trykke på *Exit*.

6.4.4 Analyse av flere testprøver

Når man er ferdig med å kjøre alle prøvene med programmet **D790test.py**, må man gå inn i hver mappe og kopiere (eller flytte) **ASTM_D790_<filnavn>.pdat** -filene inn i en mappe hvor man vil ha resultattabellen. Her kan det ikke ligge resultatfiler fra tidligere tester, da programmet vil lese gjennom alle filene i mappen som slutter med ***.pdat** (såfremt man ikke ønsker å analysere flere sett med prøver i samme analyse).

Deretter kan man kjøre programmet **D790resulttable.py**, ved å skrive

```
execfile("D790resulttable.py ")
```

Man vil da få opp et vindu hvor man må angi stien til mappen hvor man har lagt alle resultatfilene. Deretter trykker man *Run*. Programmet leser så gjennom dataene i filene, og lager et kraft-nedbøyningsplott for alle prøvene. Plottet kan nå lagres og zoomes i.

Det blir også laget to filer:

- **ASTM_D790_table.dat** – inneholder resultattabellen
- **ASTM_D790_data.dat** – inneholder nedbøynings- og kraftdata fra alle prøvene

Filen med resultattabellen inneholder datoen for analysen, navn på de spesifikke prøvene, spennvidde, bredde, tykkelse, bøyefasthet, maksimal tøyning som følge av bøyning, tangentmodul og forskjøvet flytstyrke, såfremt man ikke lot *Offset*-feltet stå tomt, samt gjennomsnittene av disse og de empiriske standardavvikene.

De to ***.dat**-filene ligger i mappen med resultatfilene fra kjøringen av **D790test.py**.

6.4.5 Utregninger for ASTM D 790

Følgende utregninger er benyttet i analysen. Uttrykkene er i utgangspunktet hentet fra standarden.

6.4.5.1 Fortegn

Programmet endrer fortegnet på nedbøyning og kraft dersom nedbøyningen i både linje 2 og 21 i kolonnen med nedbøyningsverdier i datafilen er negative.

6.4.5.2 Bøyespenningen

Hvordan bøyespenningen regnes ut avhenger av forholdet mellom spennvidde og tykkelse. Dersom spennvidden dividert med tykkelsen er mindre enn eller lik 16, brukes formelen fra likning (3) i standarden

$$\sigma_f = 3PL / 2bd^2 \tag{6.18}$$

hvor σ er spenning (MPa), P er kraft (N), L er spennvidden (mm), b er bredden (mm) og d er tykkelsen (mm).

Dersom spennvidden dividert med tykkelsen er større enn 16, brukes formelen fra likning (4) i standarden

$$\sigma_f = (3PL / 2bd^2)[1 + 6(D / L)^2 - 4(d / L)(D / L)] \tag{6.19}$$

hvor σ er spenning (MPa), P er kraft (N), L er spennvidden (mm), b er bredden (mm), d er tykkelsen (mm) og D er nedbøyning (mm).

Programmet lager en liste med bøyespenningsverdier ut fra kraftverdiene fra datafilen og eventuelt også nedbøyningsverdiene fra datafilen.

6.4.5.3 Tøyning

Tøyningen som følge av bøyning er gitt fra likning (5) i standarden

$$\epsilon_f = 100 * 6Dd / L^2 \tag{6.20}$$

hvor ϵ_f er tøyningen som følge av bøyning (%), D er nedbøyning (mm), L er spennvidden (mm) og d er tykkelsen (mm). Faktoren 100 kommer for å få tøyningen som følge av bøyning gitt i prosent.

Programmet lager en liste over tøyningsverdier ut fra nedbøyningsverdiene fra datafilen.

6.4.5.4 Maksimumsverdier

For å finne maksimal kraft, maksimal nedbøyning, maksimal tøyning og bøyefasthet gjør programmet følgende:

1) Finner maksimal kraft i kolonnen med kraftverdier fra datafilen
2) Finner maksimal nedbøyning som nedbøyningen ved maksimal kraft
3) Finner maksimal tøyning, som følge av bøyning, fra (6.20) ved å la D være nedbøyning ved maksimal kraft
4) Sjekker om denne maksimale tøyningen er større enn angitt maksimal tøyning i GUIen

Dersom maksimal tøyning er *større enn* angitt maksimal tøyning:

5) Maksimal tøyning blir satt til angitt maksimal tøyning
6) Maksimal kraft blir satt til den maksimale kraften i kraftkolonnen opp til og med det punktet hvor tøyning større enn angitt maksimal tøyning inntreffer for første gang
7) Bøyefasthet er gitt ved (6.18) eller (6.19), avhengig av forholdet mellom spennvidden og tykkelsen. Bøyefastheten bestemmes da ved å la P være denne maksimale kraften. Dersom bøyefastheten er gitt av (6.19), er D nedbøyning ved denne maksimale kraften.
8) Maksimal nedbøyning blir satt til å være nedbøyningen ved maksimal kraft
9) Kurvene og dataene stoppes der hvor tøyning større enn angitt maksimal tøyning inntreffer for første gang

Dersom maksimal tøyning er *mindre enn* angitt maksimal tøyning:

5) Maksimal kraft, maksimal nedbøyning, maksimal tøyning som følge av bøyning endres ikke
6) Bøyefasthet blir gitt ved (6.18) eller (6.19), avhengig av forholdet mellom spennvidden og tykkelsen. Bøyefastheten bestemmes da ved å la P være maksimal kraft. Dersom bøyefastheten er gitt av (6.19), er D nedbøyning ved maksimal kraft.
7) Kurvene og dataene stoppes ikke

6.4.5.5 Tangentmodulen

Tangentmodulen regnes ut fra likning (6) i standarden

$$E_B = L^3 m / 4bd^3 \tag{6.21}$$

hvor E_B er tangentmodulen i bøyning (MPa), L er spennvidden (mm), b er bredden (mm), d er tykkelsen (mm) og m er stigningstall på kurven i kraft-nedbøynings-plottet. Stigningstallet regnes ut ved hjelp av formelen

$$m = \frac{1}{9}\left(\frac{\sum_{i=\max p-4}^{\max p+4} P_i - \sum_{i=\min p-4}^{\min p+4} P_i}{D_{\max p} - D_{\min p}}\right) \tag{6.22}$$

hvor m er stigningstallet (N / mm), $\max p$ er største kraft angitt av bruker, $\min p$ er minste kraft angitt av bruker, P er kraft (N) og D er nedbøyning (mm).

6.4.5.6 Forskjøvet teknisk strekkgrense

Forskjøvet flytstyrke regnes ut ved å lage en linje

$$y = E_B(\epsilon_f - offset)/100 \tag{6.23}$$

hvor E_B er elastisitetsmodulen (MPa), ϵ_f er tøyningen som følge av bøyning (%) og $offset$ er forskyvningen gitt av bruker. Faktoren $1/100$ kommer av at tøyningen er gitt i prosent. Deretter regner programmet ut skjæringen mellom denne linjen og bøyespenning-tøynings-kurven ved å finne ved hvilke tøyningspunkter vi har at linjen ligger *under* bøyningsspenningen i ett punkt, og *over* bøyningsspenningen i det påfølgende punktet. Deretter tas gjennomsnittet av bøyningsspenningene i disse to punktene. Dette gjennomsnittet rapporteres som forskjøvet flytstryke.

6.4.5.7 Statistikk

Gjennomsnittsverdien av en størrelse x er regnet ut ved

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.24}$$

hvor $\bar{x}$ er gjennomsnittet av alle x $(i = 1, n)$ og n er antall tester.

Standardavviket til en størrelse x er regnet som et empirisk standardavvik, gitt fra likning (8) i standarden,

$$S_{n-1} = \sqrt{(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2)/(n-1)} \tag{6.25}$$

hvor S_{n-1} er det empiriske standardavviket, $\bar{x}$ er gjennomsnittet av x og n er antall tester.

6.5 Test ASTM D 2344

6.5.1 Generelt

Programmene **D2344test.py** og **D2344resulttable.py** er laget som analyseverktøy for standarden D 2344 – Standard Test Method for Short-Beam Strength of Polymer Matrix Composite Materials and their Laminates. Testen brukes for å bestemme skjærstyrken (”short-beam strength”) til polymerkompositter.

6.5.2 Spesielt om data fra testene

Formatet på inndatafilene for denne testen er vist under.

1. MTS793|MPT|ENG|1|2|.|/|:|44|1|1|A
2.
3. Data Acquisition DataFangst Time: 151.53809 Sec 08/05/2012 13:15:37
4. Time Ch 1 Displacement Ch 1 Force
5. Sec mm N
6. 0.11425781 8.5681677e-05 1.2050189
7. 0.21386719 0.00091046095 2.7285168
8. 0.31347656 0.0022672117 -0.089243524

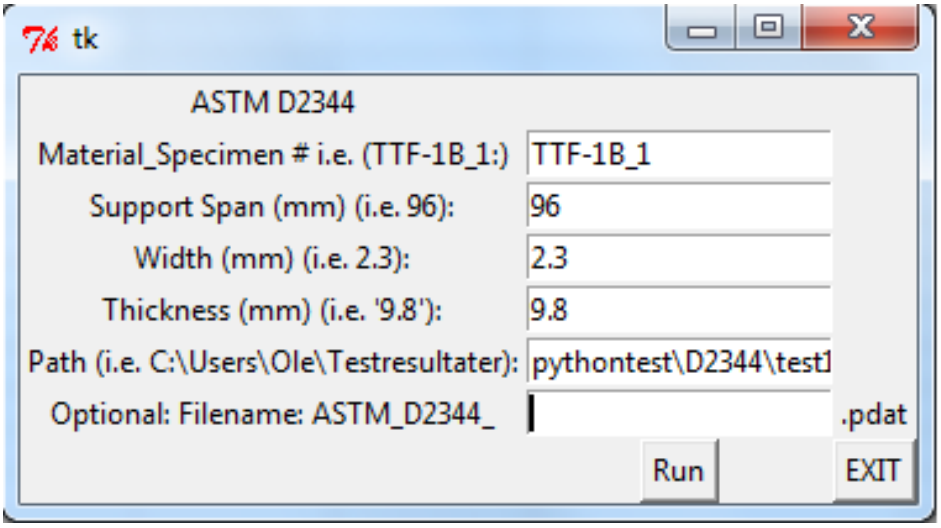
Inndatafilene fra disse testene heter alle **specimen.dat**. Filene består av tre kolonner: 1) Tid, 2) kraft og 3) deformasjon. Linje fire inneholder kolonnenavnene, men kan inneholde flere ord. Linje 4 må imidlertid inneholde nøkkelordene ”Time”, ”Displacement” og ”Force”; rekkefølgen kan dog være vilkårlig, avhengig av hvordan testen er utført. Nøkkelordene er viktige, da programmet bruker disse ordene for å gjenkjenne hvilken kolonne som inneholder hvilken informasjon. Linje fem lister kun de tilhørende enhetene, det vil si ”Sec”, ”mm” og ”N” eller ”kN”. Dersom kraften er gitt i kN, blir alle verdiene multiplisert med 1000, og programmet jobber videre med N.

6.5.3 Fil over på ønsket format

Åpne PyLab-vinduet og flytt til katalogen hvor Python-programmet **D2344test.py** ligger på maskinen. For å kjøre programmet skriver man

```
execfile("D2344test.py")
```

Det vil nå komme opp en GUI, hvor programmet ber om inndata. GUIen er vist under, se Figur 6.5.



*Figur 6.5: GUI for **D2344test.py***

Inndataene som må fylles ut er:

- Navn på den spesifikke prøven (*Material_Specimen #)*
- Spennvidde (*Support Span (mm)*)
- Bredde (*Width (mm)*)
- Tykkelse (*Thickness (mm)*)
- Stien, det vil si hvor datafilen (**specimen.dat**) ligger (*Path*)
- Det man vil skal stå mellom ”ASTM_D2344_” og ”.pdat” (*Filename*), som blir resultatfilen fra kjøringen

Merk følgende:

- Dersom man vil at det man fyller inn for navn på den spesifikke prøven skal være det som skal stå mellom ”ASTM_D2344_” og ”.pdat”, lar man filnavnfeltet stå tomt
- Spennvidden, bredden og tykkelsen må gis som rene tall uten benevning. Flyttall representeres ved punktum (”.”) og ikke komma (”,”).
- *Material_Specimen #* er det som kommer til å stå i tabellen og i figurene, for å skille prøvene fra hverandre

Etter at man har skrevet inn alle verdiene, trykker man *Run*.

Man vil nå få opp et plott av sammenhengen mellom kraft og deformasjon for prøven. Programmet har videre lagret en fil med data fra prøven i mappen hvor **specimen.dat**-filen ligger, altså den angitte stien. Denne filen vil hete **ASTM_2344_<filnavn>.pdat**, hvor **<filnavn>** er det man skrev inn i *Filename*-boksen, eller i *Material_Specimen #* dersom man lot *Filename* stå tomt.

Filen **ASTM_D2344_<filnavn>.pdat** inneholder i dette tilfellet dato for testen, tabell med navn på prøven, spennvidde, bredde, tykkelse, skjærstyrke og maksimal kraft, samt deformasjons- og kraftdata.

Hvis man ønsker å lagre plottet, trykker man på lagreknappen under plottet. Her kan man også zoome inn om det er ønskelig, og man kan også lagre de zoomede versjonene. Merk at plottene må lukkes mellom hver gang for å forhindre at neste graf tegnes i samme vindu. Noen ganger vil ikke plottet lukke seg, men for å forhindre at neste graf tegnes i samme vindu, holder det at man har forsøkt å lukke plottet.

Det er da klart for å kjøre programmet på nytt for neste prøve. Det gjør man ved å skrive inn nye verdier i GUIen, og igjen trykke *Run.* Når man har kjørt programmet på alle prøvene, avsluttes GUIen ved å trykke på *Exit*.

6.5.4 Analyse av flere testprøver

Når man er ferdig med å kjøre alle prøvene med programmet **D2344test.py**, må man gå inn i hver mappe og kopiere (eller flytte) **ASTM_D2344_<filnavn>.pdat** -filene inn i en mappe hvor man vil ha resultattabellen; **<filnavn>** er det man skrev inn i *Filename*-boksen, eller i *Material_Specimen #* dersom man lot *Filename* stå tomt. Her kan det ikke ligge resultatfiler fra tidligere tester, da programmet vil lese gjennom alle filene i mappen som slutter med ***.pdat** (såfremt man ikke ønsker å analysere flere sett med prøver i samme analyse).

Deretter kan man kjøre programmet **D2344resulttable.py**, ved å skrive

```
execfile("D2344resulttable.py ")
```

Man vil da få opp et vindu hvor man må angi stien til mappen hvor man har lagt alle resultatfilene. Deretter trykker man *Run*. Programmet leser så gjennom dataene i filene, og lager et kraft-deformasjons-plott av alle prøvene. Plottet kan nå lagres og zoomes i.

Det blir også laget to filer:

- **ASTM_D2344_table.dat** – inneholder resultattabellen
- **ASTM_D2344_data.dat** – inneholder deformasjons- og kraftdata for alle prøvene

Filen med resultattabellen inneholder datoen for analysen, navn på de spesifikke prøvene, spennvidde, bredde, tykkelse og skjærstyrke, samt gjennomsnittene av disse og de empiriske standardavvikene.

De to ***.dat**-filene ligger i mappen med resultatfilene fra kjøringen av **D2344test.py**.

6.5.5 Utregninger for ASTM D 2344

Følgende utregninger er benyttet i analysen. Uttrykkene er i utgangspunktet hentet fra standarden.

6.5.5.1 Fortegn

Programmet endrer fortegnet på nedbøyning og kraft dersom nedbøyningen i både linje 2 og 21 i kolonnen med nedbøyningsverdier i datafilen er negative.

6.5.5.2 Skjærstyrke

Skjærstyrken regnes ut fra likning (1) i standarden

$$F^{sbs} = 0.75 * \frac{P_m}{b * h} \tag{6.26}$$

hvor F^{sbs} er skjærstyrken (MPa), P_m er maksimal kraft (N), b er bredden (mm) og h er tykkelsen (mm).

6.5.5.3 Statistikk

Gjennomsnittsverdien av en størrelse x er regnet ut fra likning (2) i standarden

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{6.27}$$

hvor $\overline{x}$ er gjennomsnittet av alle x $(i = 1, n)$ og n er antall tester.

Standardavviket til en størrelse x er regnet som et empirisk standardavvik, fra likning (3) i standarden

$$S_{n-1} = \sqrt{(\sum_{i=1}^{n} x_i^2 - n\overline{x}^2) / (n-1)} \tag{6.28}$$

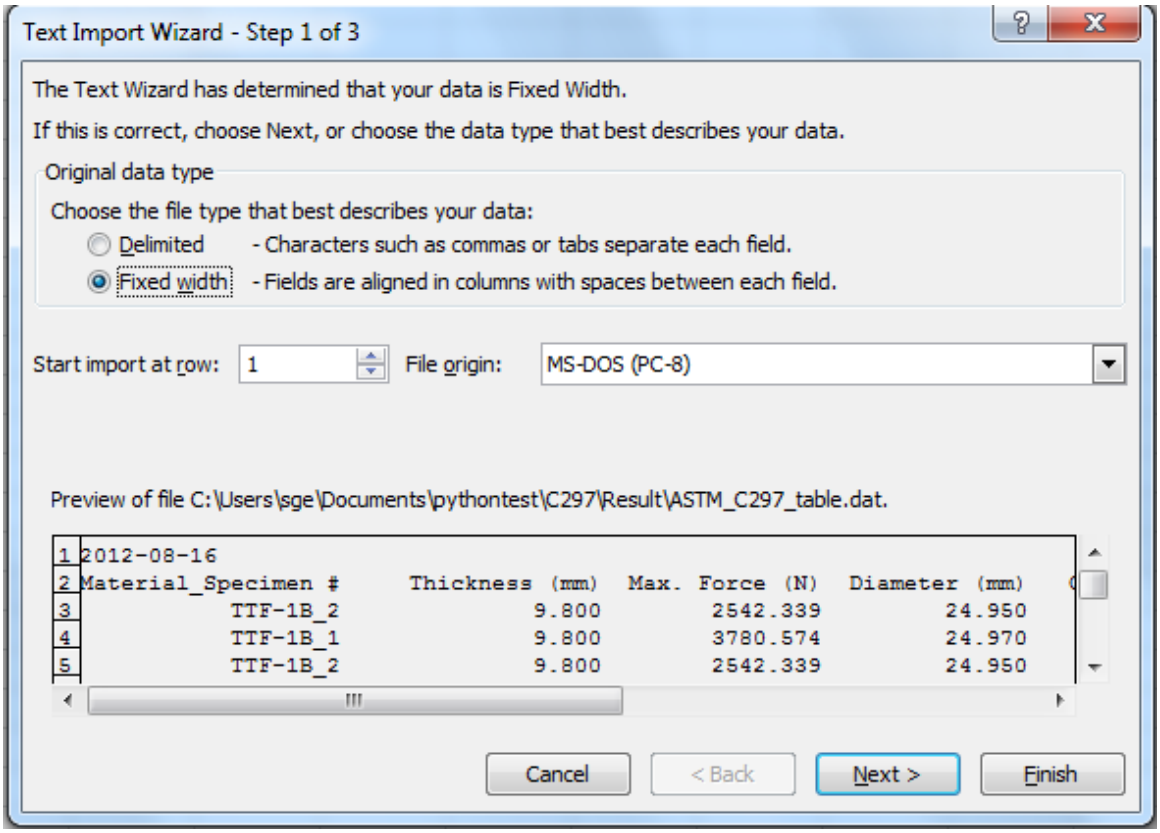
hvor S_{n-1} er det empiriske standardavviket, $\overline{x}$ er gjennomsnittet av x og n er antall tester.

7 Legge resultattabell/datafil inn i Excel

For å legge resultattabellen eller filen med rådataene fra alle testene inn i Excel, kan man bruke følgende fremgangsmåte (norske ord står i parentes):

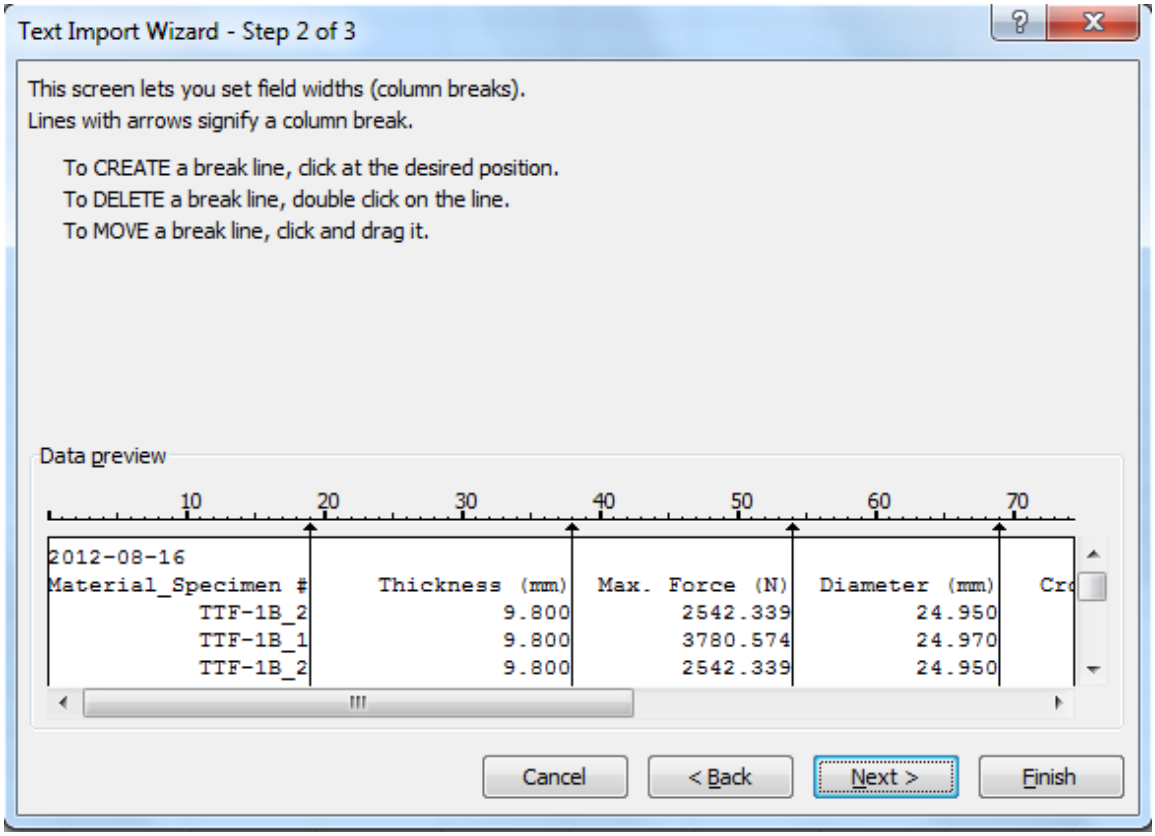
1) Åpne Microsoft Excel
2) Trykk på *Data* (*Data*) i menylinjen i Excel. Du får da opp en undermeny.
3) Trykk på *From Text* (*Fra tekst*) i undermenyen til *Data*. Du får da opp et vindu hvor du blir bedt om å velge fil.
4) Velg filen du vil legge inn i Excel. For at filen skal vises, må man klikke der hvor det står *Text files* (*Tekstfiler*), og i stedet velge *All files* (*Alle filer*). Man vil nå få opp et nytt vindu.

5) Under *Choose the file type that best describes the data* (*Velg filtypen som best beskriver dataene*), velg *Fixed Width* (*Data med fast bredde*) og klikk *Next >* (*Neste >*). Se Figur 7.1.



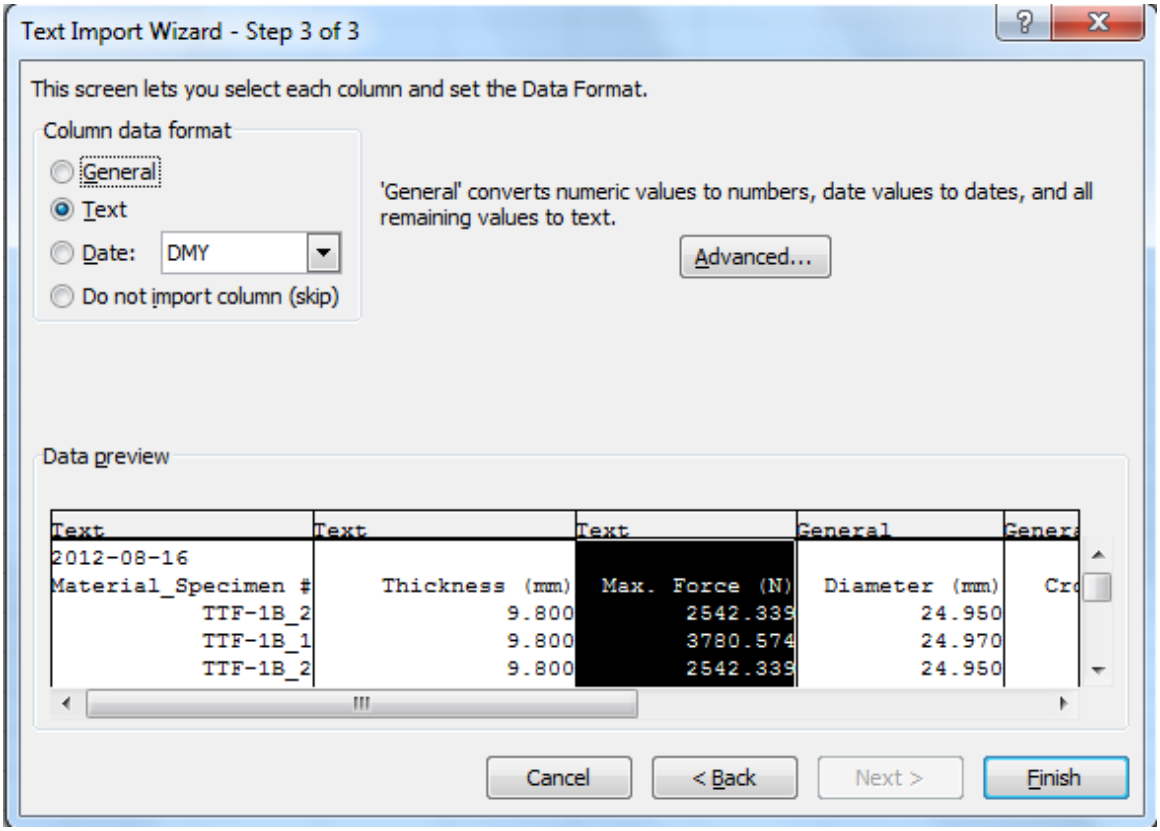
Figur 7.1 Choose the file type that best describes your data.

6) På neste steg under *Data Preview* (*Forhåndsvisning av data*), dra linjene slik at de passer med kolonnene. Pass på at alle tall ligger innenfor sin respektive kolonne. Klikk så på *Next >* (*Neste >*). Se Figur 7.2.



Figur 7.2 Data Preview.

7) Under *Column Data Format* (*Kolonnedataformat*) på neste steg, velg *Text* (*Tekst*) på samtlige kolonner for å unngå at Excel gjør om tallene til datoer eller liknende. Klikk så på *Finish* (*Fullfør*). Se Figur 7.3.



Figur 7.3 Column data format

8) Plassér tabellen der man vil ha den.

8 Oppsummering

Python-programmene er laget som analyseverktøy for analyse av data fra MTS-strekkmaskinen for ASTM-standardene C297, D3039, D3518, D790 og D2344. For å bruke programmene må man ha installert Python. Python kan installeres som beskrevet i Seksjon 2. Beskrivelse av hvordan man kjører et Python-program, er gitt i Seksjon 3. Det finnes to programmer for hver test. Det første programmet kjøres for å analysere hver enkelt prøve og for å få inndataene over på et ønsket format. Videre analyser gjøres i program nummer to, der det blir generert en resultattabell for alle prøvene, samt en samling av alle rådataene i én fil. Denne resultattabellen og rådatafilen kan så legges inn i Excel, som beskrevet i Seksjon 7. Programmene lager også plott av dataene i hvert tilfelle.

Takk

Takk til Jan Rune Nilssen, Kristin Lippe, Bernt Brønmo Johnsen og Torbjørn Olsen (alle ved FFI) for bistand underveis og nyttige kommentarer til denne rapporten og analyseprogrammene.

Referanser

[1] K. Lippe and Ø. Frøyland, "Kurs i bruk av MTS strekkmaskin 29. juni - 01. juli 2010," FFI Notat 2011/00351 (Ugradert), 2011.

Appendix A Standarder

Oversikt over standarder:

- ASTM C 297 – "Standard Test Method for Flatwise Tensile Strenght of Sandwich Constructions" – 04
- ASTM D 3039 – "Standard Test Method for Tensile Properties of Fiber-Resin Composites" – 76 (Reapproved 1989)
- ASTM D 3518 – "Standard Test Method for In-Plane Shear Response of Polymer Matrix Composite Materials by Tensile Test of a ± 45º Laminate" – 94 (Reapproved 2007)
- ASTM D 790 – "Standard Test Method for Flexural Properties of Unreinforced and Reinforced Plastics and Electrical Insulating Materials" – 03
- ASTM D 2344 – "Standard Test Method for Short-Beam Strength of Polymer Matrix Composite Materials and their Laminates" – 00

Appendix B Programkode

B.1 ASTM C 297

B.1.1 Filen ”C297test.py”

```
from Tkinter import *
import matplotlib.pyplot as plt
import os
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
#Variables attached to widgets
s=StringVar() #Specimen name variable
t=StringVar() #Thickness variable
d=StringVar() #Diameter variable
p=StringVar() #Path variable
f=StringVar() #Filename variable
standard=Label(top, text="ASTM C297") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
specimen=Label(top, text="Material__Specimen # i.e. (TTF-1B_1:)") #Specimen box text
specimen.grid(row=4, column=0) #Place the specimen box text at row 4, column 0
sp=Entry(top, textvariable=s) #Specimen entry, user input stored as variable s
sp.grid(row=4, column=1) #Place the specimen entry box at row 4, column 1
thickness=Label(top, text="Thickness (mm) (i.e. '9.8'):") #Thickness box text
thickness.grid(row=6, column=0) #Place the thickness box text at row 6, column 0
th=Entry(top, textvariable=t)          #Thickness entry, user input stored as variable t
th.grid(row=6, column=1)               #Place the thickness entry box at row 6, column 1
diameter=Label(top, text="Diameter (mm) (i.e. '24.95'):") #Diameter box text
diameter.grid(row=8, column=0) #Store diameter box text at row 8, column 0
dia=Entry(top,textvariable=d)          #Diameter entry, user input stored as variable d
dia.grid(row=8, column=1)              #Place the diameter entry box at row 8, column 1
paway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
paway.grid(row=10, column=0) #Place the path box text at row 10, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=10, column=1) #Place the path entry box at row 10, column 1
filename=Label(top, text="Optional: Filename: ASTM_C297_") #Filename box text
filename.grid(row=12, column=0) #Place the filename box text at row 12, column 0
file=Entry(top, textvariable=f) #Filename entry, user input stored as variable f
file.grid(row=12, column=1) #Place the filename entry box at row 12, column 1
filetyp=Label(top, text=".pdat") #End of filename
filetyp.grid(row=12, column=2) #Place the end of the filename at row 12, column 2
```

```
def comp():
        original_folder=os.getcwd() #Get the original folder
        S, T, D, P, F=s.get(), float(t.get()), float(d.get()), p.get(), f.get() #Get input from entry
                                                                                #boxes
        os.chdir(P) #Change directory to the input directory
        C=pi*(D/2)**2 #Cross sect. area
        infile=open("specimen.dat", "r")        #Open data file
        infile.readline() #Skip the first and second line
        infile.readline()
        lines=infile.readlines()
        dateline=lines[0] #Store the third line to get the date
        info=dateline.split()
        date=info[-2] #Date
        del lines[0] #Delete the line with the date
        infile.close() #Close file
        names=lines[0].split()      #Store line with names of columns
        del lines[0]   #Delete line with names
        Units=lines[0].split() #Store units
        del lines[0] #Delete line with units, now there are only numbers in lines
        indexlist=[] #List to store column names in correct order
        for j in range(len(names)):
                if names[j]=="Displacement" or names[j]=="Force" or
names[j]=="Time": #Find out the order of the columns
                        indexlist.append(names[j])  #Append the column names
                                                    #in the right order
        mindex=indexlist.index("Displacement")      #Find out which column has
                                                    #the respective info
        findex=indexlist.index("Force")
        sindex=indexlist.index("Time")
        Displacement=[] #Empty lists to store the data
        Time=[]
        Force=[]
        for line in lines:          #Loop over the data, storing data in lists
                words=line.split() #Splits the line at the spaces
                Displacement.append(float(words[mindex])) #Append the number
corresponding to the index of the word Displacement in the indexlist list
                Time.append(float(words[sindex])) #Append the number
corresponding to the index of the word Time in the indexlist list
                Force.append(float(words[findex])) #Append the number
corresponding to the index of the word Force in the indexlist list
        if Units[findex]=="kN": #Multiply by 1000 if the force is given in kN instead of N, so
that Force is given in N.
                Force=1000*array(Force)
```

```
		plt.plot(Displacement, Force)		#Plot force vs. displacement
		plt.title("ASTM C-297 %s" % S) #Plot title
		plt.ylabel("N")		#name on y-axis
		plt.xlabel("%s" % Units[mindex])	#name on x-axis
		plt.grid(True) #Put grid on plot
		Fmax=(array(Force)).max()		#Find max force
		ten=Fmax/float(C)	#Ultimate Flatwise Tensile Strength - maximum force
divided by area. From equation (1) in standard
		test="ASTM_C297_"	#Beggining of filename
		fi=".pdat" #End of filename
		if F=="" or F=="No" or F=="no":
			Strings=[test, S, fi] #List to create filename if "filename"-entry is empty
		else:
			Strings=[test,F, fi] #list to create filename
		Filename="".join(Strings) #creating filename
		outfile=open(Filename, "w") #Create a file in the folder where the data file is
		outfile.write("%s" % date) #Write out the test date
		outfile.write("\n") #New line
		outfile.write("Material_Specimen #    Thickness (mm)  Max. Force (N)     Diameter
(mm)  Cross. sect (mm^2) Flatwise Tensile Strength (MPa)")
		outfile.write("\n") #New line
		outfile.write("%19s %17.3f %15.3f %17.3f %19.3f %31.3f" % (S, T, Fmax, D, C, ten))
		#Write results to file
		outfile.write("\n") #New line
		outfile.write("   Displacement      Force")
		outfile.write("\n") #New line
		for i in range(len(Force)): #Write Displacement and Force table to the file
			outfile.write("%14.8f   %14.8f" % (Displacement[i], Force[i]))
			outfile.write("\n") #New line
		outfile.close()
		os.chdir(original_folder) #Change directory to the original folder
def quit(event=None): #Quit button function
			root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=14, column=1) #Place the Run-button at row 14, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=14, column=2) #Place the EXIT-button at row 14, column 2
root.mainloop()
```

B.1.2 Filen ”C297resulttable.py”

```
import os
from numpy import sqrt
import matplotlib.pyplot as plt
import glob
from datetime import date
import itertools
from Tkinter import *
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
p=StringVar() #path variable
standard=Label(top, text="ASTM C297") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
pway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
pway.grid(row=4, column=0) #Place the path box text at row 4, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=4, column=1) #Place the path entry box at row 4, column 1
def comp():
        P=p.get() #Get the path
        original_folder=os.getcwd() #Store the original folder
        os.chdir(P)   #Change directory to the given path
        today=date.today()          #Store today's date
        filelist=glob.glob("*.pdat") #List all pdat files in the folder
        number=len(filelist) #Number of files/Number of tests
        Speci=[] #Empty lists to store the data to calculate mean value and standard
deviation
        Thick=[]
        Maxf=[]
        Dia=[]
        Cross=[]
        Tens=[]
        outfile=open("ASTM_C297_table.dat", "w") #Write the results to the file
"ASTM_C297_table.dat"
        outfile.write("%s" % today) #Write out the date
        outfile.write("\n") #New line
        outfile.write("Material_Specimen    # Thickness (mm)  Max. Force (N)  Diameter
                                            # (mm)  Cross sect. (mm^2) Flatwise tensile
                                            # strength (MPa)")
        outfile.write("\n") #New line
        ForcesDis=[] #List for Force vs. Displacement data
        for i in range(number): #Loop over the different files
```

```
                Displacement=[] #Empty lists to store the Displacement and Force data
from each test
                Force=[]
                infile=open(filelist[i], "r")
                lines=infile.readlines() #Read the lines in the specific file
                words=lines[2].split() #Line with table data
                infile.close()
                del lines[0] #deleting all the lines without force/displacement data
                del lines[0]
                del lines[0]
                del lines[0]
                for line in lines: #Appending displacement and force for plotting
                        val=line.split()
                        Displacement.append(float(val[0]))
                        Force.append(float(val[1]))
                ForcesDis.append(Displacement)
                ForcesDis.append(Force)
                ax = plt.subplot(111)
                ax.plot(Displacement, Force, label=words[0]) #Name of the respective
                                                        #plot is the input in "Specimen #"
                                                        #in the GUI
                ax.legend(loc='center left',bbox_to_anchor=(1, 0.5)) #Move legend box
                                                                #outside plot
                ax.grid(True) #Put a grid on the plot
                plt.xlabel("mm") #Name x-axis
                plt.ylabel("N") #Name y-axis
                plt.title("ASTM C-297") #Plot title
                legend()
                Speci.append(words[0])          #Append the table values to their
                                                #respective list
                Thick.append(float(words[1]))
                Maxf.append(float(words[2]))
                Dia.append(float(words[3]))
                Cross.append(float(words[4]))
                Tens.append(float(words[5]))
                outfile.write("%19s %18s %15s %14s %20s %31s" % (words[0],words[1],
words[2], words[3], words[4], words[5])) #Write out the values in the table dat file
                outfile.write("\n") #New line

        meanth=sum(Thick)/len(Thick) #Mean thickness
        meanf=sum(Maxf)/len(Maxf) #Mean max force
        meanDia=sum(Dia)/len(Dia) #Mean diameter
        meanc=sum(Cross)/len(Cross) #Mean cross section
```

```
        meanTens=sum(Tens)/len(Tens) #Mean ultimate flatwise tensile strength
        outfile.write("Mean %33.3f %15.3f %14.3f %20.3f %31.3f" % (meanth, meanf,
meanDia, meanc, meanTens)) #Write out mean values to the table
        n=number #number of values for each variable
        xth=0 #Variables to calculate standard deviation
        xf=0
        xd=0
        xc=0
        xt=0
        for i in range(number):
                xth+=(Thick[i])**2          #Adding the squares of the different values for
                                            #the respective variable from the different tests
                xf+=(Maxf[i])**2
                xd+=(Dia[i])**2
                xc+=(Cross[i])**2
                xt+=(Tens[i])**2
        if number==1:
                print "You need more than one test from C297test.py to calculate
statistics"
                thavvik=0
                favvik=0
                davvik=0
                cavvik=0
                tavvik=0
        else:
                thavvik=sqrt((xth-n*(meanth**2))/(n-1)) #Sample standard deviations.
From equation (3) in standard.
                favvik=sqrt((xf-n*(meanf**2))/(n-1))
                davvik=sqrt((xd-n*(meanDia**2))/(n-1))
                cavvik=sqrt((xc-n*(meanc**2))/(n-1))
                tavvik=sqrt((xt-n*(meanTens**2))/(n-1))
        datafile=open("ASTM_C297_data.dat", "w") #Create file "ASTM_C297_data.dat" to
write in the displacement/force data from each test
        datafile.write("                                ".join(Speci))   #Write out the specific
                                                                         #test names joined by spaces
        datafile.write("\n") #New line
        titnames=["    Displacement", "Force  "] #The names of the different columns
        coltit=n*titnames #Adding the names of all the columns i.e. "Displacement Force
Displacement Force ..." n times
        datafile.write("            ".join(coltit)) #Joining these names with spaces
        datafile.write("\n") #New line
        for t in itertools.izip_longest(*ForcesDis): #Write out the data as columns
```

```
                                datafile.write("            ".join("%15s" % x if x is not None else "               "
for x in t)) #Join the data by spaces, if there is no data, just write spaces
                                datafile.write("\n") #New line
                datafile.close()
                outfile.write("\n") #New line
                outfile.write("Std Dev. %29.3f %15.3f %14.3f %20.3f %31.3f" % (thavvik, favvik,
davvik, cavvik, tavvik)) #Write out the standard deviation values to the table
                outfile.close()
                os.chdir(original_folder) #Change directory to the original folder

def quit(event=None): #Quit button function
                                root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=6, column=1) #Place the Run-button at row 6, column 1

widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=6, column=2) #Place the EXIT-button at row 6, column 2

root.mainloop()
```

B.2 ASTM D 3039

B.2.1 Filen ”D3039test.py”

```
from Tkinter import *
import matplotlib.pyplot as plt
import os
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
#Variables attached to widgets
s=StringVar() #Specimen name variable
l=StringVar() #Length (total/gauge) variable
w=StringVar() #Width variable
t=StringVar() #Thickness variable
mins=StringVar() #Minimum strain variable
maxs=StringVar() #Maximum strain variable
p=StringVar() #Path variable
f=StringVar() #Filename variable
standard=Label(top, text="ASTM D3039") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
```

```
specimen=Label(top, text="Material_Specimen # i.e. (TTF-1B_1:)") #Specimen box text
specimen.grid(row=4, column=0) #Place specimen box text at row 4, column 0
sp=Entry(top, textvariable=s) #Specimen entry, user input stored as variable s
sp.grid(row=4, column=1) #Place specimen entry box at row 4, column 1
length=Label(top, text="Length (Total (mm)/gauge (mm)) (i.e. 250/150):") #Length box text
length.grid(row=6, column=0) #Place length box text at row 6, column 0
leng=Entry(top, textvariable=l) #Length entry, user input stored as variable l
leng.grid(row=6, column=1) #Place length entry box at row 6, column 1
width=Label(top, text="Width (mm) (i.e. 2.3):") #Width box text
width.grid(row=8, column=0) #Place width box text at row 8, column 0
wid=Entry(top, textvariable=w) #Width entry, user input stored as variable w
wid.grid(row=8, column=1) #Place width entry box at row 8, column 1
thickness=Label(top, text="Thickness (mm) (i.e. '9.8'):") #Thickness box text
thickness.grid(row=10, column=0) #Place thickness box at row 10, column 0
th=Entry(top, textvariable=t)              #Thickness entry, user input stored as variable t
th.grid(row=10, column=1) #Place thickness entry box at row 10, column 1
minimumstrain=Label(top, text="Minimum strain value (%) to calculate modulus (ASTM: 0.1):")
#Minimum strain box text
minimumstrain.grid(row=12, column=0) #Place the miniumum strain box text at row 12, column
0
minstr=Entry(top, textvariable=mins) #Minimum strain entry, user input stored as variable mins
minstr.grid(row=12, column=1) #Place the minimum strain entry at row 12, column 1
maximumstrain=Label(top, text="Maximum strain value (%) to calculate modulus (ASTM: 0.3):")
#Maximum strain box text
maximumstrain.grid(row=14, column=0) #Place the maximum strain box text at row 14, column
0
maxstr=Entry(top, textvariable=maxs) #Maximum strain entry, user input stored as variable
maxs
maxstr.grid(row=14, column=1) #Place the maximum strain entry box at row 14, column 1

warning=Label(top, text="NB! Make sure the curve is linear within this strain interval!") #NB box
text
warning.grid(row=15, column=0) #Place the NB box text at row 15, column 0
paway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
paway.grid(row=17, column=0) #Place path box text at row 17, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=17, column=1) #Place path entry box at row 17, column 1
filename=Label(top, text="Optional: Filename: ASTM_D3039_") #Filename box text
filename.grid(row=19, column=0) #Place filename box text at row 19, column 0
file=Entry(top, textvariable=f) #Filename entry, user input stored as variable f
file.grid(row=19, column=1) #Place filename entry box at row 19, column 1
filetyp=Label(top, text=".pdat") #Filename end
filetyp.grid(row=19, column=2) #Place filename end at row 19, column 2
```

```
def comp():
        original_folder=os.getcwd() #Get the original folder
        S, L, W, T, MINS, MAXS, P, F=s.get(), l.get(), float(w.get()),
float(t.get()),float(mins.get()), float(maxs.get()), p.get(), f.get() #Get input from entry boxes
        os.chdir(P) #Change directory to the input directory
        C=W*T #Cross sect. area width*thickness
        infile=open("specimen.dat", "r")          #Open data file
        infile.readline() #Skip the first and second line
        infile.readline()
        lines=infile.readlines()
        dateline=lines[0] #Store the third line to get the date
        info=dateline.split()
        date=info[-2] #Date
        del lines[0]  #delete line with date
        infile.close() #Close file
        names=lines[0].split()      #Store line with names of columns
        del lines[0]   #Delete line with names
        Units=lines[0].split() #Store units
        del lines[0] #Delete line with units, now there are only numbers in lines
        indexlist=[]
        for j in range(len(names)):
                if names[j]=="Displacement" or names[j]=="Force" or names[j]=="Time"
or names[j]=="Strain" or names[j]=="Transvers": #Find out the order of the columns
                        indexlist.append(names[j])
        lindex1=indexlist.index("Displacement") #Find out which column has the respective
info
        findex=indexlist.index("Force")
        tindex=indexlist.index("Time")
        sindex=indexlist.index("Strain")
        lindex2=indexlist.index("Transvers")
        Length1=[] #Empty lists to store the data
        Force=[]
        Time=[]
        Strain=[]
        Length2=[]
        for line in lines:              #Loop over the data, storing data in lists
                words=line.split() #Splits the line at the spaces
                Length1.append(float(words[lindex1])) #Append the number
corresponding to the index of the word Displacement in the indexlist to the list Length1
                Force.append(float(words[findex])) #Append the number
corresponding to the index of the word Force in the indexlist to the list Force
```

```
                Time.append(float(words[tindex])) #Append the number
corresponding to the index of the word Time in the indexlist to the list Time
                Strain.append(float(words[sindex])) #Append the number
corresponding to the index of the word Strain in the indexlist to the list Strain
                Length2.append(float(words[lindex2]))          #Append the
number corresponding to the index of the word Transvers in the indexlist to the list Length_2
        force=array(Force) #Make array of Force-list
        if Units[findex]=="kN": #Multiply by 1000 if the force is given in kN instead of N, so
that Force is given in N.
                force=1000*force
        if Units[sindex]=="mm/mm": #Multiply by 100 if the strain is given in mm/mm
instead of %, so that Strain is given in %.
                Strain=100*array(Strain)
                Strain=list(Strain)
        Stress=force/C          #Stress=P/A divides every element in the force list by the
area
        minstrain=1000 # To help calculate the number closest to the given minimum strain
point in the Strain list for modulus calculations
        minvalue=0 #The number closest to minimum strain
        maxstrain=1000 # To help calculate the number closest to the maximum strain point
in the Strain list for modulus calculations
        maxvalue=0 #The number closest to maximum strain
        for i in range(len(Strain)): #Loop to find minvalue and maxvalue.
                if abs(Strain[i]-MINS)<minstrain:
                        minstrain=abs(Strain[i]-MINS)
                        minvalue=Strain[i]
                if abs(Strain[i]-MAXS)<maxstrain:
                        maxstrain=abs(Strain[i]-MAXS)
                        maxvalue=Strain[i]
        mini=Strain.index(minvalue) #Get the index of minvalue and maxvalue
        maxi=Strain.index(maxvalue)
        Modulus=100*(Stress[maxi]-Stress[mini])/(Strain[maxi]-Strain[mini]) #Modulus as
delta stress/delta strain for strain between approximately given minimum and given maximum.
Multiplied by 100 since strain is in %

        plt.plot(Strain, Stress)    #Plot stress vs. strain
        plt.title("ASTM D-3039 %s" % S) #Plot title
        plt.ylabel("Tensile Stress, MPa")       #name on y-axis
        plt.xlabel("Tensile Strain, %")         #name on x-axis
        plt.grid(True) #Put grid on plot
        Fmax=force.max()        #Find max force
        if Units[findex]=="kN": #Find the index of the maximum force
                fnumb=Force.index(Fmax/1000)
```

```
		else:
			fnumb=Force.index(Fmax)
		ten=Fmax/float(C)		#Ultimate Flatwise Tensile Strength - maximum force divided by area. From formula, see standard.
		failstrain=array(Strain).max() #Find ultimate tensile strain
		test="ASTM_D3039_"	#Beggining of filename
		fi=".pdat" #End of filename
		if F=="" or F=="No" or F=="no":
			Strings=[test, S, fi]  #List to create filename if "filename"-entry is empty
		else:
			Strings=[test,F, fi] #list to create filename
		Filename="".join(Strings) #creating filename
		outfile=open(Filename, "w") #Create a file in the given folder with the specimen.dat-file
		outfile.write("%s" % date) #Write today's date
		outfile.write("\n") #New line
		outfile.write("Material_Specimen #	Length (Total (mm)/gauge (mm))	Width (mm)	Thickness (mm)	Maximum Tensile Stress (MPa)	Ultimate Tensile Strain (%%)	Tensile Chord Modulus (MPa)	Displacement (%s)	Max. Force (N)" % Units[lindex1])
		outfile.write("\n") #New line
		outfile.write("%19s %35s %12.3f %15.3f %31.3f %32.3f %29.3f %20.3f %19.3f" % (S, L, W, T,ten, failstrain, Modulus, Length1[fnumb], Fmax))		#Write results to file
		outfile.write("\n") #New line
		outfile.write("	%s		Force		Strain_long	Displ_trans	Stress" % Units[lindex1])
		outfile.write("\n") #New line
		for i in range(len(Length1)): #Write the column data to the file
			outfile.write("%14.8f	%14.8f	%14.8f	%14.8f	%14.8f" % (Length1[i], force[i], Strain[i], Length2[i], Stress[i]))
			outfile.write("\n") #New line
		outfile.close()
		os.chdir(original_folder) #Change directory to the original folder

def quit(event=None): #Quit button function
			root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function comp()
compute.grid(row=21, column=1) #Place the Run-button at row 21, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function quit()
widget.grid(row=21, column=2) #Place the EXIT-button at row 21, column 2
root.mainloop()
```

B.2.2 Filen ”D3039resulttable.py”

```
import os
from numpy import sqrt
import matplotlib.pyplot as plt
import glob
from datetime import date
import itertools
from Tkinter import *
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
p=StringVar() #path variable
standard=Label(top, text="ASTM D3039") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
pway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
pway.grid(row=4, column=0) #Place the path box text at row 4, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=4, column=1) #Place the path entry box at row 4, column 1
def comp():
        P=p.get() #Get the path
        original_folder=os.getcwd() #Store the original folder
        os.chdir(P)   #Change directory to the given path
        today=date.today()          #Store today's date
        filelist=glob.glob("*.pdat") #List all files ending with pdat in the folder
        number=len(filelist) #Number of files/Number of tests
        Speci=[] #Empty lists to store the data to calculate mean value and standard
deviation
        Length=[]
        Width=[]
        Thick=[]
        Stress=[]
        Strain=[]
        Modulus=[]
        Totlen=[]
        Galen=[]
        outfile=open("ASTM_D3039_table.dat", "w") #Write the results to the file
"ASTM_D3039_table.dat" in the folder provided by the user
        outfile.write("%s" % today) #Write out the date
        outfile.write("\n") #New line
        outfile.write("Material_Specimen #     Length (Total (mm)/gauge (mm))   Width
(mm)  Thickness (mm)    Maximum Tensile Stress (MPa)     Ultimate Tensile Strain (%)   Tensile
Chord Modulus (MPa)")
```

```
outfile.write("\n") #New line
Datalist=[] #List for data from all the tests
for i in range(number): #Loop over the different files
        Length1=[] #Lists to store the raw number data
        Force1=[]
        Strain1=[]
        Length2=[]
        Stress1=[]
        infile=open(filelist[i], "r") #Open the files in the list
        lines=infile.readlines() #Read the lines in the specific file
        words=lines[2].split() #Line with table data
        infile.close()
        del lines[0] #deleting all the lines without raw number data
        del lines[0]
        del lines[0]
        del lines[0]
        for line in lines: #Appending data to lists
                val=line.split()
                Length1.append(float(val[0]))
                Force1.append(float(val[1]))
                Strain1.append(float(val[2]))
                Length2.append(float(val[3]))
                Stress1.append(float(val[4]))
        Datalist.append(Length1) #Append the data from each test to Datalist,
so that Datalist contains all the data
        Datalist.append(Force1)
        Datalist.append(Strain1)
        Datalist.append(Length2)
        Datalist.append(Stress1)
        ax = plt.subplot(111)
        ax.plot(Strain1, Stress1, label=words[0]) #Name of the respective plot is
the input in "Specimen#" in the GUI
        ax.legend(loc='center left',bbox_to_anchor=(1, 0.5)) #Move legend box
outside plot
        plt.xlabel("Tensile Strain, %") #Name x-axis
        plt.ylabel("Tensile Stress, MPa") #Name y-axis
        plt.title("ASTM D-3039") #Plot title
        plt.grid(True) #Put a grid on the plot
        legend()
        Speci.append(words[0]) #Append the values to their respective list
        Length.append(words[1])
        Width.append(float(words[2]))
        Thick.append(float(words[3]))
```

```
                    Stress.append(float(words[4]))
                    Strain.append(float(words[5]))
                    Modulus.append(float(words[6]))
                    ltlg=words[1].split("/")
                    Totlen.append(float(ltlg[0]))
                    Galen.append(float(ltlg[1]))
                    outfile.write("%19s %35s %12s %15s %31s %32s %29s" %
(words[0],words[1], words[2], words[3], words[4], words[5], words[6])) #Write out the values in
the table dat file
                    outfile.write("\n") #New line
          meanxlt=sum(Totlen)/len(Totlen) #Mean total length
          meanxlg=sum(Galen)/len(Galen) #Mean gauge length
          meanw=sum(Width)/len(Width) #Mean width
          meanth=sum(Thick)/len(Thick) #Mean thickness
          meanstress=sum(Stress)/len(Stress) #Mean maximum tensile stress
          meanstrain=sum(Strain)/len(Strain) #Mean ultimate tensile strain
          meanm=sum(Modulus)/len(Modulus) #Mean tensile chord modulus of elasticity
          outfile.write("Mean %42.3f/%5.3f %12.3f %15.3f %31.3f %32.3f %29.3f" %
(meanxlt,meanxlg,meanw, meanth, meanstress, meanstrain, meanm)) #Write out mean values to
the table
          n=number #number of values for each variable
          xlt=0 #Variables to calculate standard deviation
          xlg=0
          xw=0
          xth=0
          xstress=0
          xstrain=0
          xm=0
          for i in range(number):
                    xlt+=((Totlen[i]))**2 #Adding the squares of the different values for the
respective variable from the different tests
                    xlg+=((Galen[i]))**2
                    xw+=((Width[i]))**2
                    xth+=((Thick[i]))**2
                    xstress+=((Stress[i]))**2
                    xstrain+=((Strain[i]))**2
                    xm+=((Modulus[i]))**2
          if number==1:
                    print "You need more than one resultfile from D3039test.py to calculate
statistics"
                    xltavvik=0
                    xlgavvik=0
                    wavvik=0
```

```
                thavvik=0
                stressavvik=0
                strainavvik=0
                mavvik=0
        else:
                #Sample standard deviations. From formula, see standard:
                xltavvik=sqrt((xlt-n*(meanxlt**2))/(n-1))
                xlgavvik=sqrt((xlg-n*(meanxlg**2))/(n-1))
                wavvik=sqrt((xw-n*(meanw**2))/(n-1)) #Width deviation
                thavvik=sqrt((xth-n*(meanth**2))/(n-1))          #Thickness
deviation
                stressavvik=sqrt((xstress-n*(meanstress**2))/(n-1))  #Stress deviation
                strainavvik=sqrt((xstrain-n*(meanstrain**2))/(n-1)) #Strain deviation
                mavvik=sqrt((xm-n*(meanm**2))/(n-1)) #Modulus deviation
        datafile=open("ASTM_D3039_data.dat", "w") #File to write in the raw number data
data from each test
        datafile.write("                                                ".join(Speci)) #Write
out the specific test names joined by spaces
        datafile.write("\n") #New line
        titnames=["    mm        ", "        Force", "    Strain_long", "    Displ_trans", "
Stress"] #Names of the columns for each test
        coltit=n*titnames #Multiplies the names with the number of tests to get the names of
all the columns for all the tests
        datafile.write("   ".join(coltit)) #Write the names of the columns, joined by spaces
        datafile.write("\n") #New line
        for t in itertools.izip_longest(*Datalist): #Write out the data as columns
                datafile.write("   ".join("%15s" % x if x is not None else "            " for x
in t)) #Join the data by spaces, if there is no data, just write spaces
                datafile.write("\n") #New line
        datafile.close()
        outfile.write("\n") #New line
        outfile.write("Std Dev. %40.3f/%5.3f %12.3f %15.3f %31.3f %32.3f %29.3f" %
(xltavvik, xlgavvik, wavvik, thavvik, stressavvik, strainavvik, mavvik)) #Write out the standard
deviations to the table
        outfile.close()
        os.chdir(original_folder) #Change directory to the original folder
def quit(event=None): #Quit button function
                root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=6, column=1) #Place the Run-button at row 6, column 1
```

```
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=6, column=2) #Place the EXIT-button at row 6, column 2
root.mainloop()
```

B.3 ASTM D 3518

B.3.1 Filen ”D3518test.py”

```
from Tkinter import *
import matplotlib.pyplot as plt
import os
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
#Variables attached to widgets
s=StringVar() #Specimen name variable
l=StringVar() #Length (total/gauge) variable
w=StringVar() #Width variable
t=StringVar() #Thickness variable
mins=StringVar() #Minimum strain variable
maxs=StringVar() #Maximum strain variable
ms=StringVar() #Max Shear Strain variable
of=StringVar() #Offset variable
pr=StringVar() #Poisson's ratio variable
p=StringVar() #Path variable
f=StringVar() #Filename variable
standard=Label(top, text="ASTM D3518") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
specimen=Label(top, text="Material_Specimen # i.e. (TTF-1B_1:)") #Specimen box text
specimen.grid(row=4, column=0) #Place specimen box text at row 4, column 0
sp=Entry(top, textvariable=s) #Specimen entry, user input stored as variable s
sp.grid(row=4, column=1) #Place specimen entry box at row 4, column 1
length=Label(top, text="Length (Total (mm)/gauge (mm)) (i.e. 250/150):") #Length box text
length.grid(row=6, column=0) #Place length box text at row 6, column 0
leng=Entry(top, textvariable=l) #Length entry, user input stored as variable l
leng.grid(row=6, column=1) #Place length entry box at row 6, column 1
width=Label(top, text="Width (mm) (i.e. 2.3):") #Width box text
width.grid(row=8, column=0) #Place width box text at row 8, column 0
wid=Entry(top, textvariable=w) #Width entry, user input stored as variable w
wid.grid(row=8, column=1) #Place width entry box at row 8, column 1
thickness=Label(top, text="Thickness (mm) (i.e. '9.8'):") #Thickness box text
thickness.grid(row=10, column=0) #Place thickness box at row 10, column 0
th=Entry(top, textvariable=t)          #Thickness entry, user input stored as variable t
```

```
th.grid(row=10, column=1) #Place thickness entry box at row 10, column 1
minimumstrain=Label(top, text="Minimum strain value (%) to calculate modulus (ASTM:
Approximately 0.2):") #Minimum strain box text
minimumstrain.grid(row=12, column=0) #Place the miniumum strain box text at row 12, column
0
minstr=Entry(top, textvariable=mins) #Minimum strain entry, user input stored as variable mins
minstr.grid(row=12, column=1) #Place the minimum strain entry at row 12, column 1
maximumstrain=Label(top, text="Maximum strain value (%) to calculate Modulus (ASTM:
Approximately 0.6):") #Maximum strain box text
maximumstrain.grid(row=14, column=0) #Place the maximum strain box text at row 14, column
0
maxstr=Entry(top, textvariable=maxs) #Maximum strain entry, user input stored as variable
maxs
maxstr.grid(row=14, column=1) #Place the maximum strain entry box at row 14, column 1

warning=Label(top, text="NB! Make sure the curve is linear within this strain interval!") #NB box
text
warning.grid(row=15, column=0) #Place the NB box text at row 15, column 0
stopstr=Label(top, text="Maximum Shear Strain (%) (ASTM: 5):") #Maximum Shear Strain box
text
stopstr.grid(row=17, column=0) #Place maximum shear strain box at row 17, column 0
mst=Entry(top, textvariable=ms) #Maximum shear strain entry, user input stored as variable ms
mst.grid(row=17, column=1) #Place maximum shear strain entry box at row 17, column 1
offset=Label(top, text="Optional: Offset (%) (ASTM: 0.2):") #Offset box text
offset.grid(row=19, column=0) #Place the offset box text at row 19, column 0
ofs=Entry(top, textvariable=of) #Offset entry box, user input stored as variable of
ofs.grid(row=19, column=1) #Place the offset entry box at row 19, column 1
poissonratio=Label(top, text="Poisson's Ratio (i.e. 0.2):") #Poisson's ratio box text
poissonratio.grid(row=21, column=0)  #Place Poisson's ratio box text at row 21, column 0
po=Entry(top, textvariable=pr) #Poisson's ratio entry box, user input stored as variable pr
po.grid(row=21, column=1)              #Place Poisson's ratio entry box at row 21, column 1

paway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
paway.grid(row=23, column=0) #Place path box text at row 23, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=23, column=1) #Place path entry box at row 23, column 1
filename=Label(top, text="Optional: Filename: ASTM_D3518_") #Filename box text
filename.grid(row=25, column=0) #Place filename box text at row 25, column 0
file=Entry(top, textvariable=f) #Filename entry, user input stored as variable f
file.grid(row=25, column=1) #Place filename entry box at row 25, column 1
filetyp=Label(top, text=".pdat") #Filename end
filetyp.grid(row=25, column=2) #Place filename end at row 25, column 2
```

```
def comp():
        original_folder=os.getcwd() #Get the original folder
        S, L, W, T, MINS, MAXS, MS, OF, PR, P,F=s.get(), l.get(), float(w.get()),
float(t.get()),float(mins.get()), float(maxs.get()), eval(ms.get()), of.get(), float(pr.get()), p.get(),
f.get() #Get input from entry boxes
        os.chdir(P) #Change directory to the input directory
        C=W*T #Cross sect. area width*thickness
        infile=open("specimen.dat", "r")        #Open data file
        infile.readline() #Skip the first and second line
        infile.readline()
        lines=infile.readlines()
        dateline=lines[0] #Store the third line to get the date
        info=dateline.split()
        date=info[-2] #Date
        del lines[0]
        infile.close() #Close file
        names=lines[0].split()      #Store line with names of columns
        del lines[0]   #Delete line with names
        Units=lines[0].split() #Store units
        del lines[0] #Delete line with units, now there are only numbers in lines
        indexlist=[]
        for j in range(len(names)):
                if names[j]=="Displacement" or names[j]=="Force" or names[j]=="Time"
or names[j]=="Strain" or names[j]=="Transvers": #Find out the order of the columns
                        indexlist.append(names[j])
        lindex1=indexlist.index("Displacement") #Find out which column has the respective
info
        findex=indexlist.index("Force")
        tindex=indexlist.index("Time")
        sindex=indexlist.index("Strain")
        lindex2=indexlist.index("Transvers")
        Length1=[] #Empty lists to store the data
        Force=[]
        Time=[]
        Strain=[]
        Length2=[]
        for line in lines:          #Loop over the data, storing data in lists
                words=line.split() #Splits the line at the spaces
                Length1.append(float(words[lindex1])) #Append the number
corresponding to the index of the word Displacement in the indexlist to the list Length1
                Force.append(float(words[findex])) #Append the number
corresponding to the index of the word Force in the indexlist to the list Force
```

```
                Time.append(float(words[tindex])) #Append the number
corresponding to the index of the word Time in the indexlist to the list Time
                Strain.append(float(words[sindex])) #Append the number
corresponding to the index of the word Strain in the indexlist to the list Strain
                Length2.append(float(words[lindex2]))          #Append the
number corresponding to the index of the word Transvers in the indexlist to the list Length_2

        force=array(Force) #Make array of Force-list
        if Units[findex]=="kN": #Multiply by 1000 if the force is given in kN instead of N, so
that Force is given in N.
                force=1000*force
        if Units[sindex]=="mm/mm": #Multiply by 100 if the strain is given in mm/mm
instead of %, so that Strain is given in %.
                Strain=100*array(Strain)
                Strain=list(Strain)
        ShearStress=force/(2*C)  #Shear Stress=P/(2A) divides every element in the force
list by the 2*area. From equation (1) in standard
        ShearStrain=(PR+1)*array(Strain) #Shear strain (1+v)*Strain, see User Guide.
        minstrain=100000 # To help calculate the number closest to the user input minimum
in the Shear Strain list for modulus calculations
        minvalue=0 #The number closest to the user minimum
        maxstrain=100000 # To help calculate the number closest to the user input maximum
in the Shear Strain list for modulus calculations
        maxvalue=0 #The number closest to user maximum
        for i in range(len(ShearStrain)): #Loop to find minvalue and maxvalue.
                if abs(ShearStrain[i]-MINS)<minstrain:
                        minstrain=abs(ShearStrain[i]-MINS)
                        minvalue=ShearStrain[i]
                if abs(ShearStrain[i]-MAXS)<maxstrain:
                        maxstrain=abs(ShearStrain[i]-MAXS)
                        maxvalue=ShearStrain[i]
        mini=list(ShearStrain).index(minvalue) #Get the index of minvalue and maxvalue
        maxi=list(ShearStrain).index(maxvalue)
        Modulus=100*(ShearStress[maxi]-ShearStress[mini])/(ShearStrain[maxi]-
ShearStrain[mini]) #Modulus as delta stress/delta strain for strain between approximately user
minimum and user maximum. Multiplied by 100 since strain is in %
        maxshearstrain=ShearStrain.max() #Find the maximum shear strain value
        if maxshearstrain>MS: #MS is the limit for max shear strain, if max shear strain>MS,
then max shear strain is minimum of MS and shear strain at max shear stress in the list up to the
point where shear strain=MS occurs
                for i in range(len(ShearStrain)):
                        if ShearStrain[i]>MS:
                                stopnumb=i
```

```
                                        Fmax=force[0:i+1].max()
                                        maxshearstress=Fmax/(2*C)

        indexmss=list(ShearStress).index(maxshearstress)
                                        maxshearstrain=ShearStrain[indexmss]
                                        break
                if          maxshearstrain>MS:
                                        maxshearstrain=MS
        else:                                                                   #if
maxshearstrain <MS:
                Fmax=force.max()                    #then maximum force is the
largest number in the force list
                maxshearstress=ShearStress.max() # max shear stress is the largest
number in the shear stress list
                indexmss=list(ShearStress).index(ShearStress.max()) #Find the index of
the maximum value in the Shear Stress list
                maxshearstrain=ShearStrain[indexmss] #Set max shear strain to the
strain at this point
                stopnumb=len(ShearStrain)-1 #Do not stop

        if Units[findex]=="kN": #Find the index of the maximum force
                fnumb=Force.index(Fmax/1000)
        else:
                fnumb=Force.index(Fmax)
        if OF=="" or OF=="No" or OF=="no":
                OffsetSS="N/A"
        else:
                OF=float(OF)
                OffsetSS=0   #If there is no intersection, it will write 0.
                Offsetline=Modulus*(ShearStrain-OF)/100 #Offsetline is a straight line
that starts in the given offset, with slope equal to the modulus. Divided by 100, since strain is in
%.
                for i in range(stopnumb):
                        if Offsetline[i]<ShearStress[i] and
Offsetline[i+1]>ShearStress[i+1]: #Finds out the points where the line crosses the shear stress vs.
shear strain curve.

                OffsetSS=(ShearStress[i]+ShearStress[i+1])/2.0 #Finds the average between the point
on the line under the curve, and the point on the line over the curve.
                plt.plot(ShearStrain[0:stopnumb+1], ShearStress[0:stopnumb+1])          #Plot
shear stress vs. shear strain, max shear strain=MS.
                plt.title("ASTM D-3518 %s" % S) #Plot title
                plt.ylabel("Shear Stress, MPa")         #name on y-axis
```

```
		plt.xlabel("Shear Strain, %")		#name on x-axis
		plt.grid(True) #Put a grid on the plot
		test="ASTM_D3518_"	#Beggining of filename
		fi=".pdat" #End of filename
		if F=="" or F=="No" or F=="no":
			Strings=[test, S, fi]  #List to create filename if "filename"-entry is empty
		else:
			Strings=[test,F, fi] #list to create filename
		Filename="".join(Strings) #creating filename
		outfile=open(Filename, "w") #Create a file in the given folder with the specimen.dat-
file
		outfile.write("%s" % date) #Write today's date
		outfile.write("\n") #New line
		if OffsetSS=="N/A":
			outfile.write("Material_Specimen #	Length (Total (mm)/gauge
(mm))	Width (mm)	Thickness (mm)	Max Shear Stress (MPa)	Max Shear Strain (%%)
Chord Shear Modulus (MPa)	Displacement (%s)	Max. Force (N)	Offset Shear Strength
(MPa)	Poisson's ratio" % (Units[lindex1]))
		else:
			outfile.write("Material_Specimen #	Length (Total (mm)/gauge
(mm))	Width (mm)	Thickness (mm)	Max Shear Stress (MPa)	Max Shear Strain (%%)
Chord Shear Modulus (MPa)	Displacement (%s)	Max. Force (N)	Offset Shear Strength at
%4.2f %% (MPa)	Poisson's ratio" % (Units[lindex1], OF))
		outfile.write("\n") #New line
		if OffsetSS=="N/A":
			outfile.write("%19s %35s %12.3f %15.3f %25.3f %25.3f %28.3f %20.3f
%19.3f %29.3s %17.3f" % (S, L, W, T, maxshearstress,maxshearstrain, Modulus, Length1[fnumb],
Fmax, OffsetSS, PR))	#Write results to file
		else:
			outfile.write("%19s %35s %12.3f %15.3f %25.3f %25.3f %28.3f %20.3f
%19.3f %39.3f %17.3f" % (S, L, W, T, maxshearstress,maxshearstrain, Modulus, Length1[fnumb],
Fmax, OffsetSS, PR))	#Write results to file
		outfile.write("\n") #New line
		outfile.write("	%s	Force	Strain_long	Displ_trans	Shear_Strain
Shear_Stress" % Units[lindex1])
		outfile.write("\n") #New line
		for i in range(0,stopnumb+1): #Write the column data to the file, stop when strain=5
			outfile.write("%14.8f  %14.8f  %14.8f  %14.8f  %14.8f  %14.8f" %
(Length1[i], force[i], Strain[i], Length2[i], ShearStrain[i], ShearStress[i]))
			outfile.write("\n") #New line
		outfile.close()
		os.chdir(original_folder) #Change directory to the original folder
```

```
def quit(event=None): #Quit button function
                        root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=27, column=1) #Place the Run-button at row 27, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=27, column=2) #Place the EXIT-button at row 27, column 2
root.mainloop()
```

B.3.2 Filen ”D3518resulttable.py”

```
import os
from numpy import sqrt
import matplotlib.pyplot as plt
import glob
from datetime import date
import itertools
from Tkinter import *
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
standard=Label(top, text="ASTM D3518") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
p=StringVar() #path variable
pway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
pway.grid(row=4, column=0) #Place the path box text at row 4, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=4, column=1) #Place the path entry box at row 4, column 1
def comp():
        P=p.get() #Get the path
        original_folder=os.getcwd() #Store the original folder
        os.chdir(P)   #Change directory to the given path
        today=date.today()          #Store today's date
        filelist=glob.glob("*.pdat") #List all files ending with pdat in the folder
        number=len(filelist) #Number of files/Number of tests
        Speci=[] #Empty lists to store the data to calculate mean value and standard
deviation
        Length=[]
        Width=[]
        Thick=[]
        MShearStress=[]
```

```
MShearStrain=[]
Modulus=[]
Offset=[]
Totlen=[]
Galen=[]
outfile=open("ASTM_D3518_table.dat", "w") #Write the results to the file
"ASTM_D3518_table.dat" in the folder provided by the user
outfile.write("%s" % today) #Write out the date
outfile.write("\n") #New line
outfile.write("Material_Specimen #     Length (Total (mm)/gauge (mm))   Width
(mm)  Thickness (mm)    Max Shear Stress (MPa)     Max Shear Strain (%)   Chord Shear Modulus
(MPa) Offset Shear Strength")
Datalist=[] #List for data from all the tests
for i in range(number): #Loop over the different files
        Length1=[] #Lists to store the raw number data
        Force1=[]
        Strain1=[]
        Length2=[]
        ShearStrain1=[]
        ShearStress1=[]
        infile=open(filelist[i], "r") #Open the files in the list
        lines=infile.readlines() #Read the lines in the specific file
        coln=lines[1].split() #Find the given offset strain
        if i==0: #For the first file, write out the offset strain in the column name
in the table.
                test=lines[2].split()
                if test[-2]=="N/A":
                        outfile.write(" (MPa)")
                else:
                        offsetname=coln[-5]
                        outfile.write(" at %s %% (MPa)" %
offsetname)
                outfile.write("\n") #New line
        words=lines[2].split() #Line with table data
        infile.close()
        del lines[0] #deleting all the lines without raw number data
        del lines[0]
        del lines[0]
        del lines[0]
        for line in lines: #Appending data to lists
                val=line.split()
                Length1.append(float(val[0]))
                Force1.append(float(val[1]))
```

```
                                        Strain1.append(float(val[2]))
                                        Length2.append(float(val[3]))
                                        ShearStrain1.append(float(val[4]))
                                        ShearStress1.append(float(val[5]))
                        Datalist.append(Length1) #Appending all the lists to the datalist
                        Datalist.append(Force1)
                        Datalist.append(Strain1)
                        Datalist.append(Length2)
                        Datalist.append(ShearStrain1)
                        Datalist.append(ShearStress1)
                        ax = plt.subplot(111)
                        ax.plot(ShearStrain1, ShearStress1, label=words[0]) #Name of the
respective plot is the input in "Specimen#" in the GUI
                        ax.legend(loc='center left',bbox_to_anchor=(1, 0.5)) #Move legend box
outside plot
                        plt.xlabel("Shear Strain, %") #Name x-axis
                        plt.ylabel("Shear Stress, MPa") #Name y-axis
                        plt.title("ASTM D-3518") #Plot title
                        plt.grid(True) #Put a grid on the plot
                        legend()
                        Speci.append(words[0]) #Append the values to their respective list
                        Length.append(words[1])
                        Width.append(float(words[2]))
                        Thick.append(float(words[3]))
                        MShearStress.append(float(words[4]))
                        MShearStrain.append(float(words[5]))
                        Modulus.append(float(words[6]))
                        if words[9]!="N/A":
                                Offset.append(float(words[9]))
                        ltlg=words[1].split("/")
                        Totlen.append(float(ltlg[0]))
                        Galen.append(float(ltlg[1]))
                        outfile.write("%19s %35s %12s %15s %25s %25s %27s %27s" %
(words[0],words[1], words[2], words[3], words[4], words[5], words[6], words[9])) #Write out the
values in the table dat file
                        outfile.write("\n") #New line
                pr=words[-1] #Poisson's Ratio
                meanxlt=sum(Totlen)/len(Totlen) #Mean total length
                meanxlg=sum(Galen)/len(Galen) #Mean gauge length
                meanw=sum(Width)/len(Width) #Mean width
                meanth=sum(Thick)/len(Thick) #Mean thickness
                meanshearstress=sum(MShearStress)/len(MShearStress) #Mean maximum shear
stress
```

```
		meanshearstrain=sum(MShearStrain)/len(MShearStrain) #Mean maximum shear
strain
		meanm=sum(Modulus)/len(Modulus) #Mean chord shear modulus of elasticity
		if len(Offset)!=0:
				meano=sum(Offset)/len(Offset) #Mean offset shear strength
				outfile.write("Mean %42.3f/%5.3f %12.3f %15.3f %25.3f %25.3f %27.3f
%27.3f" % (meanxlt,meanxlg,meanw, meanth, meanshearstress, meanshearstrain,
meanm,meano)) #Write out mean values to the table
		else:
				meano="N/A"
				outfile.write("Mean %42.3f/%5.3f %12.3f %15.3f %25.3f %25.3f %27.3f
%27.3s" % (meanxlt,meanxlg,meanw, meanth, meanshearstress, meanshearstrain,
meanm,meano)) #Write out mean values to the table
		n=number #number of values for each variable
		xlt=0 #Variables to calculate standard deviation
		xlg=0
		xw=0
		xth=0
		xshstress=0
		xshstrain=0
		xm=0
		xo=0
		for i in range(number):
				xlt+=((Totlen[i]))**2 #Adding the squares of the different values for the
respective variable from the different tests
				xlg+=((Galen[i]))**2
				xw+=((Width[i]))**2
				xth+=((Thick[i]))**2
				xshstress+=((MShearStress[i]))**2
				xshstrain+=((MShearStrain[i]))**2
				xm+=((Modulus[i]))**2
		for j in range(len(Offset)):
				xo+=(Offset[j])**2
		if number==1:
				print "You need more than one resultfile from D3518test.py to calculate
statistics"
				xltavvik=0
				xlgavvik=0
				wavvik=0
				thavvik=0
				shstressavvik=0
				shstrainavvik=0
				mavvik=0
```

```
            else:
                        #Sample standard deviations. From equation (8) in standard:
                        xltavvik=sqrt((xlt-n*(meanxlt**2))/(n-1))
                        xlgavvik=sqrt((xlg-n*(meanxlg**2))/(n-1))
                        wavvik=sqrt((xw-n*(meanw**2))/(n-1)) #Width deviation
                        thavvik=sqrt((xth-n*(meanth**2))/(n-1))                #Thickness
deviation
                        shstressavvik=sqrt((xshstress-n*(meanshearstress**2))/(n-1))  #Shear
Stress deviation
                        shstrainavvik=sqrt((xshstrain-n*(meanshearstrain**2))/(n-1)) # Shear
Strain deviation
                        mavvik=sqrt((xm-n*(meanm**2))/(n-1)) #Modulus deviation
            if len(Offset)==0 or len(Offset)==1:
                        ofavvik="N/A"
            else:
                        ofavvik=sqrt((xo-len(Offset)*(meano**2))/(len(Offset)-1)) #Offset
deviation
            datafile=open("ASTM_D3518_data.dat", "w") #File to write in the raw number data
data from each test
            datafile.write("Poisson's Ratio: %s" % pr)
            datafile.write("\n") # New line
            datafile.write("
".join(Speci)) #Write out the specific test names joined by spaces
            datafile.write("\n") #New line
            titnames=["    mm          ", "      Force     ", "      Strain_long      ",
"Deformation_trans", "    Shear Strain","     Shear Stress    "] #Names of the columns for each test
            coltit=n*titnames #Multiplies the names with the number of tests to get the names of
all the columns for all the tests
            datafile.write(" ".join(coltit)) #Write the names of the columns, joined by spaces
            datafile.write("\n") #New line
            for t in itertools.izip_longest(*Datalist): #Write out the data as columns
                        datafile.write("     ".join("%15s" % x if x is not None else "              " for x
in t)) #Join the data by spaces, if there is no data, just write spaces
                        datafile.write("\n") #New line
            datafile.close()
            outfile.write("\n") #New line
            if len(Offset)==0 or len(Offset)==1:
                        outfile.write("Std Dev. %40.3f/%5.3f %12.3f %15.3f %25.3f %25.3f
%27.3f %27.3s" % (xltavvik, xlgavvik, wavvik, thavvik, shstressavvik, shstrainavvik, mavvik,
ofavvik)) #Write out the standard deviations to the table
            else:
```

```
                    outfile.write("Std Dev. %40.3f/%5.3f %12.3f %15.3f %25.3f %25.3f
%27.3f %27.3f" % (xltavvik, xlgavvik, wavvik, thavvik, shstressavvik, shstrainavvik, mavvik,
ofavvik)) #Write out the standard deviations to the table
        outfile.close()
        os.chdir(original_folder) #Change directory to the original folder
def quit(event=None): #Quit button function
                    root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=6, column=1) #Place the Run-button at row 6, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=6, column=2) #Place the EXIT-button at row 6, column 2
root.mainloop()
```

B.4 ASTM D 790

B.4.1 Filen ”D790test.py”

```
from Tkinter import *
import matplotlib.pyplot as plt
import os
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
#Variables attached to widgets
s=StringVar() #Specimen name variable
l=StringVar() #Support span variable
w=StringVar() #Width variable
t=StringVar() #Thickness variable
minf=StringVar() #Minimum force variable
maxf=StringVar() #Maximum force variable
lims=StringVar() #Maximum flexural strain variable
of=StringVar() #Offset flexural strain variable
p=StringVar() #Path variable
f=StringVar() #Filename variable
standard=Label(top, text="ASTM D790") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
specimen=Label(top, text="Material_Specimen # i.e. (TTF-1B_1:)") #Specimen box text
specimen.grid(row=4, column=0) #Place specimen box text at row 4, column 0
sp=Entry(top, textvariable=s) #Specimen entry, user input stored as variable s
```

```
sp.grid(row=4, column=1) #Place specimen entry box at row 4, column 1
length=Label(top, text="Support Span (mm) (i.e. 96):") #Length box text
length.grid(row=6, column=0) #Place length box text at row 6, column 0
leng=Entry(top, textvariable=l) #Length entry, user input stored as variable l
leng.grid(row=6, column=1) #Place length entry box at row 6, column 1
width=Label(top, text="Width (mm) (i.e. 2.3):") #Width box text
width.grid(row=8, column=0) #Place width box text at row 8, column 0
wid=Entry(top, textvariable=w) #Width entry, user input stored as variable w
wid.grid(row=8, column=1) #Place width entry box at row 8, column 1
thickness=Label(top, text="Thickness (mm) (i.e. '9.8'):") #Thickness box text
thickness.grid(row=10, column=0) #Place thickness box at row 10, column 0
th=Entry(top, textvariable=t)              #Thickness entry, user input stored as variable t
th.grid(row=10, column=1) #Place thickness entry box at row 10, column 1
Minimumforce=Label(top, text="Minimum force (N) value to calculate the slope:") #Minimum
force box text
Minimumforce.grid(row=12, column=0) #Place minimum force box at row 12, column 0
minfor=Entry(top, textvariable=minf) #Minimum force entry, user input stored as variable minf
minfor.grid(row=12, column=1) #Place the minimum force entry box at row 12, column 0
Maximumforce=Label(top, text="Maximum force (N) value to calculate the slope:") #Maximum
force box text
Maximumforce.grid(row=14, column=0) #Place the maximum force box text at row 14, column
maxfor=Entry(top, textvariable=maxf) #Maximum force entry, user input stored as variable maxf
maxfor.grid(row=14, column=1) #Place the maximum force entry box at row 14, column 1
warning=Label(top, text="NB! Make sure the curve is linear within this force interval") # NB box
text
warning.grid(row=15, column=0) #Place the NB box text at row 15, column 0
stopstr=Label(top, text="Maximum flexural strain (%) (ASTM: 5):") #Maximum flexural strain
box text
stopstr.grid(row=17, column=0) #Place maximum flexural strain box at row 17, column 0
mst=Entry(top, textvariable=lims) #Maximum flexural strain entry, user input stored as variable
lims
mst.grid(row=17, column=1) #Place maximum flexural strain entry box at row 17, column 1
offset=Label(top, text="Optional: Offset (%) (ASTM: 0.2):") #Offset box text
offset.grid(row=19, column=0) #Place the offset box text at row 19, column 0
ofs=Entry(top, textvariable=of) #Offset entry box, user input stored as variable of
ofs.grid(row=19, column=1) #Place the offset entry box at row 19, column 1
paway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
paway.grid(row=21, column=0) #Place path box text at row 21, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=21, column=1) #Place path entry box at row 21, column 1
filename=Label(top, text="Optional: Filename: ASTM_D790_") #Filename box text
filename.grid(row=23, column=0) #Place filename box text at row 23, column 0
file=Entry(top, textvariable=f) #Filename entry, user input stored as variable f
```

```
file.grid(row=23, column=1) #Place filename entry box at row 23, column 1
filetyp=Label(top, text=".pdat") #Filename end
filetyp.grid(row=23, column=2) #Place filename end at row 23, column 2

def comp():
        original_folder=os.getcwd() #Get the original folder
        S, L, W, T, MINF, MAXF, MS, OF, P,F=s.get(), float(l.get()), float(w.get()),
float(t.get()), float(minf.get()), float(maxf.get()), float(lims.get()), of.get(), p.get(), f.get() #Get input
from entry boxes
        os.chdir(P) #Change directory to the input directory
        infile=open("specimen.dat", "r")        #Open data file
        infile.readline() #Skip the first and second line
        infile.readline()
        lines=infile.readlines()
        dateline=lines[0] #Store the third line to get the date
        info=dateline.split()
        date=info[-2] #Date
        del lines[0] #Delete the line with the date
        infile.close() #Close file
        names=lines[0].split()      #Store line with names of columns
        del lines[0]   #Delete line with names
        Units=lines[0].split() #Store units
        del lines[0] #Delete line with units, now there are only numbers in lines
        indexlist=[]
        for j in range(len(names)):
                if names[j]=="Displacement" or names[j]=="Force" or
names[j]=="Time": #Find out the order of the columns
                        indexlist.append(names[j])
        lindex=indexlist.index("Displacement") #Find out which column has the respective
info
        findex=indexlist.index("Force")
        tindex=indexlist.index("Time")
        Length=[] #Empty lists to store the data
        Force=[]
        Time=[]
        for line in lines:              #Loop over the data, storing data in lists
                words=line.split() #Splits the line at the spaces
                Length.append(float(words[lindex])) #Append the number
corresponding to the index of the word Displacement in the indexlist to the list Length
                Force.append(float(words[findex])) #Append the number
corresponding to the index of the word Force in the indexlist to the list Force
                Time.append(float(words[tindex])) #Append the number
corresponding to the index of the word Time in the indexlist to the list Time
```

```
force=array(Force) #Make array of Force list
length=array(Length) #Make array of Length list
if length[1] and length[20]<0: #Change sign of force and displacement if the displacement at point 1 and 20 are negative
force=(-1)*force
length=(-1)*length
if Units[findex]=="kN": #Multiply by 1000 if the force is given in kN instead of N, so that Force is given in N.
force=1000*force
Fmax=force.max() #Max force in force list
fnumb=list(force).index(Fmax) #Index of max force
if L/T<=16: #Different stress formulas depending on the support span-to-depth ratio
Stress=3*force*L/(2*W*T**2) #From equation (3) in standard
Maxstress=Stress.max()
else:
Stress=3*force*L/(2*W*T**2)*(1+6*(length/L)**2-4*(T/L)*(length/L)) #Flexural Stress. From equation (4) in standard
Maxstress=3*Fmax*L/(2*W*T**2)*(1+6*(length[fnumb]/L)**2-4*(T/L)*(length[fnumb]/L))
Maxstrain=100*6*length[fnumb]*T/L**2
Strain=100*6*length*T/L**2 #Flexural Strain. From equation (5) in standard
if Maxstrain>MS:
Maxstrain=MS
for i in range(len(Strain)): #If max flexural strain> MS, then max force is the maximum force in the force list up to the point where strain larger than 5 first occurs, and max flexural stress is also at this point.
if Strain[i]>MS:
stopnumb=i
Fmax=force[0:i+1].max()
fnumb=list(force).index(Fmax)
if L/T<=16:
Maxstress=3*Fmax*L/(2*W*T**2) #From equation (3) in standard
else:
Maxstress=3*Fmax*L/(2*W*T**2)*(1+6*(length[fnumb]/L)**2-4*(T/L)*(length[fnumb]/L)) #From equation (4) in standard
break
else:
stopnumb=len(Strain)-1
minforce=1000 # To help calculate the number closest to minimum input in the force list for tangent modulus calculations
```

```
        minvalue=0 #The number closest to minimum input
        maxforce=1000 # To help calculate the number closest to maximum input in the force list for tangent modulus calculations
        maxvalue=0 #The number closest to maximum input
        for i in range(fnumb): #Loop to find minvalue and maxvalue.
                if abs(force[i]-MINF)<minforce:
                        minforce=abs(force[i]-MINF)
                        minvalue=force[i]
                if abs(force[i]-MAXF)<maxforce:
                        maxforce=abs(force[i]-MAXF)
                        maxvalue=force[i]
        MINFindex=list(force).index(minvalue) #Get the index of minvalue and maxvalue
        MAXFindex=list(force).index(maxvalue)
        slope=(sum(force[MAXFindex-4:MAXFindex+5])/9.0-sum(force[MINFindex-4:MINFindex+5])/9.0)/(length[MAXFindex]-length[MINFindex]) #Average force from the index of maximum input-4 up to and including maximum input+4 minus average force from the index of minimum input-4 up to and including minimum input+4 divided by displacement at maximum input minus displacement at minimum input
        Tangentmod=L**3*slope/(4*W*T**3) #Tangent Modulus. From equation (6) in standard
        if OF=="" or OF=="No" or OF=="no":
                OffsetFS="N/A"
        else:
                OF=float(OF)
                Offsetline=Tangentmod*(Strain-OF)/100 #Offsetline is a straight line that starts in the given offset, with slope equal to the modulus. Divided by 100, since strain is in %.
                OffsetFS=0 #If there is no intersection, it will write 0.
                for i in range(stopnumb):
                        if Offsetline[i]<Stress[i] and Offsetline[i+1]>Stress[i+1]: #Finds out the points where the line crosses the flexural stress vs. flexural strain curve.
                                OffsetFS=(Stress[i]+Stress[i+1])/2.0 #Finds the average between the point on the line under the curve, and the point on the line over the curve.
        plt.plot(length[0:stopnumb+1], force[0:stopnumb+1])             #Plot force vs. displacement
        plt.title("ASTM D-790 %s" % S) #Plot title
        plt.ylabel("N")             #name on y-axis
        plt.xlabel("mm")            #name on x-axis
        plt.grid(True) #Put a grid on the plot
        test="ASTM_D790_"       #Beggining of filename
        fi=".pdat" #End of filename
        if F=="" or F=="No" or F=="no":
```

```
                Strings=[test, S, fi]  #List to create filename if "filename"-entry is empty
        else:
                Strings=[test,F, fi] #list to create filename
        Filename="".join(Strings) #creating filename
        outfile=open(Filename, "w") #Create a file in the given folder with the specimen.dat-
file
        outfile.write("%s" % date) #Write today's date
        outfile.write("\n") #New line
        if OffsetFS=="N/A":
                outfile.write("Material_Specimen #      Length (mm)/Width (mm)
Thickness (mm)    Max Flexural Stress (MPa)      Max Flexural Strain (%%)    Tangent Modulus
(MPa)    Deflection (%s)      Max. Force (N)   Offset Yield Strength (MPa)     Slope (N/mm)" %
(Units[lindex]))
                outfile.write("\n") #New line
                outfile.write("%19s %20.3f/%5.3f %15.3f %28.3f %28.3f %24.3f %18.3f
%19.3f %29.3s %16.3f" % (S, L, W, T, Maxstress,Maxstrain, Tangentmod, length[fnumb], Fmax,
OffsetFS, slope))          #Write results to file
        else:
                outfile.write("Material_Specimen #      Length (mm)/Width (mm)
Thickness (mm)    Max Flexural Stress (MPa)      Max Flexural Strain (%%)    Tangent Modulus
(MPa)    Deflection (%s)      Max. Force (N)   Offset Yield Strength at %4.2f %% (MPa)     Slope
(N/mm)" % (Units[lindex], OF))
                outfile.write("\n") #New line
                outfile.write("%19s %20.3f/%5.3f %15.3f %28.3f %28.3f %24.3f %18.3f
%19.3f %39.3f %16.3f" % (S, L, W, T, Maxstress,Maxstrain, Tangentmod, length[fnumb], Fmax,
OffsetFS, slope))          #Write results to file
        outfile.write("\n") #New line
        outfile.write("     %s          Force" % Units[lindex])
        outfile.write("\n") #New line
        for i in range(len(Force)): #Write the column data to the file, stop when strain=5
                outfile.write("%14.8f   %14.8f" % (length[i], force[i]))
                outfile.write("\n") #New line
        outfile.close()
        os.chdir(original_folder) #Change directory to the original folder

def quit(event=None): #Quit button function
                root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=25, column=1) #Place the Run-button at row 25, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
```

```
widget.grid(row=25, column=2) #Place the EXIT-button at row 25, column 2
root.mainloop()
```

B.4.2 Filen ”D790testtable.py”

```
import os
from numpy import sqrt
import matplotlib.pyplot as plt
import glob
from datetime import date
import itertools
from Tkinter import *
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
standard=Label(top, text="ASTM D790") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
p=StringVar() #path variable
pway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
pway.grid(row=4, column=0) #Place the path box text at row 4, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=4, column=1) #Place the path entry box at row 4, column 1
def comp():
        P=p.get() #Get the path
        original_folder=os.getcwd() #Store the original folder
        os.chdir(P)   #Change directory to the given path
        today=date.today()          #Store today's date
        filelist=glob.glob("*.pdat") #List all files ending with pdat in the folder
        number=len(filelist) #Number of files/Number of tests
        Speci=[] #Empty lists to store the data to calculate mean value and standard
deviation
        Lenwid=[]
        Thick=[]
        MFlexStress=[]
        MFlexStrain=[]
        TanModulus=[]
        Offset=[]
        Length=[]
        Width=[]
        outfile=open("ASTM_D790_table.dat", "w") #Write the results to the file
"ASTM_D790_table.dat" in the folder provided by the user
        outfile.write("%s" % today) #Write out the date
        outfile.write("\n") #New line
```

```
		outfile.write("Material_Specimen #	Length (mm)/Width (mm)	Thickness (mm)	Max Flexural Stress (MPa)	Max Flexural Strain (%)	Tangent Modulus (MPa) Offset Yield Strength")
		Datalist=[] #List for data from all the tests
		for i in range(number): #Loop over the different files
			Disp=[] #Lists to store the raw number data
			Force=[]
			infile=open(filelist[i], "r") #Open the files in the list
			lines=infile.readlines() #Read the lines in the specific file
			coln=lines[1].split() #Find the given offset strain
			if i==0: #For the first file, write out the offset strain in the column name in the table.
				test=lines[2].split()
				if test[-2]=="N/A":
					outfile.write(" (MPa)")
				else:
					offsetname=coln[-5] #Offset value
					outfile.write(" at %s %% (MPa)" % offsetname)
				outfile.write("\n") #New line
			words=lines[2].split() #Line with table data
			infile.close()
			del lines[0] #deleting all the lines without raw number data
			del lines[0]
			del lines[0]
			del lines[0]
			for line in lines: #Appending data to lists
				val=line.split()
				Disp.append(float(val[0]))
				Force.append(float(val[1]))
			Datalist.append(Disp) #Appending all the lists to the datalist
			Datalist.append(Force)
			ax = plt.subplot(111)
			ax.plot(Disp, Force, label=words[0]) #Name of the respective plot is the input in "Specimen#" in the GUI
			ax.legend(loc='center left',bbox_to_anchor=(1, 0.5)) #Move legend box outside plot
			ax.grid(True) #Put a grid on the plot
			plt.xlabel("mm") #Name x-axis
			plt.ylabel("N") #Name y-axis
			plt.title("ASTM D-790") #Plot title
			legend()
			Speci.append(words[0]) #Append the values to their respective list
```

```
                        Lenwid.append(words[1])
                        Thick.append(float(words[2]))
                        MFlexStress.append(float(words[3]))
                        MFlexStrain.append(float(words[4]))
                        TanModulus.append(float(words[5]))
                        if words[8]!="N/A":
                                Offset.append(float(words[8]))
                        lnwd=words[1].split("/")
                        Length.append(float(lnwd[0]))
                        Width.append(float(lnwd[1]))
                        outfile.write("%19s %27s %15s %28s %28s %23s %27s" %
(words[0],words[1], words[2], words[3], words[4], words[5], words[8])) #Write out the values in
the table dat file
                        outfile.write("\n") #New line
        meanxl=sum(Length)/len(Length) #Mean support span
        meanw=sum(Width)/len(Width) #Mean width
        meanth=sum(Thick)/len(Thick) #Mean thickness
        meanflexstress=sum(MFlexStress)/len(MFlexStress) #Mean maximum flexural
stress
        meanflexstrain=sum(MFlexStrain)/len(MFlexStrain) #Mean maximum flexural
strain
        meantm=sum(TanModulus)/len(TanModulus) #Mean modulus of elasticity in
bending
        if len(Offset)!=0:
                        meano=sum(Offset)/len(Offset) #Mean offset shear strength
                        outfile.write("Mean %35.3f/%5.3f %15.3f %28.3f %28.3f %23.3f %27.3f"
% (meanxl,meanw, meanth, meanflexstress, meanflexstrain, meantm, meano)) #Write out mean
values to the table
        else:
                        meano="N/A"
                        outfile.write("Mean %35.3f/%5.3f %15.3f %28.3f %28.3f %23.3f %27.3s"
% (meanxl,meanw, meanth, meanflexstress, meanflexstrain, meantm, meano)) #Write out mean
values to the table
        n=number #number of values for each variable
        xl=0 #Variables to calculate standard deviation
        xw=0
        xth=0
        xfstress=0
        xfstrain=0
        xtm=0
        xo=0
        for i in range(number):
```

```
                xl+=((Length[i]))**2 #Adding the squares of the different values for the
respective variable from the different tests
                xw+=((Width[i]))**2
                xth+=((Thick[i]))**2
                xfstress+=((MFlexStress[i]))**2
                xfstrain+=((MFlexStrain[i]))**2
                xtm+=((TanModulus[i]))**2
        for j in range(len(Offset)):
                xo+=(Offset[j])**2
        if number==1:
                print "You need more than one resultfile from D790test.py to calculate
statistics"
                xlavvik=0
                wavvik=0
                thavvik=0
                fstressavvik=0
                fstrainavvik=0
                tmavvik=0
        else:
        #Sample standard deviations. From equation (8) in standard:
                xlavvik=sqrt((xl-n*(meanxl**2))/(n-1)) #Support span deviation
                wavvik=sqrt((xw-n*(meanw**2))/(n-1)) #Width deviation
                thavvik=sqrt((xth-n*(meanth**2))/(n-1))          #Thickness
deviation
                fstressavvik=sqrt((xfstress-n*(meanflexstress**2))/(n-1))  #Flexural
Stress deviation
                fstrainavvik=sqrt((xfstrain-n*(meanflexstrain**2))/(n-1)) # Flexural
Strain deviation
                tmavvik=sqrt((xtm-n*(meantm**2))/(n-1)) #Tangent Modulus deviation
        if len(Offset)==0 or len(Offset)==1:
                ofavvik="N/A"
        else:
                ofavvik=sqrt((xo-len(Offset)*(meano**2))/(len(Offset)-1)) #Offset
deviation
        datafile=open("ASTM_D790_data.dat", "w") #File to write in the raw number data
data from each test
        datafile.write("\n") # New line
        datafile.write("                ".join(Speci)) #Write out the specific test names
joined by spaces
        datafile.write("\n") #New line
        titnames=["    mm", "        Force"] #Names of the columns for each test
        coltit=n*titnames #Multiplies the names with the number of tests to get the names of
all the columns for all the tests
```

```
		datafile.write("	".join(coltit)) #Write the names of the columns, joined by spaces
		datafile.write("\n") #New line
		for t in itertools.izip_longest(*Datalist): #Write out the data as columns
			datafile.write("	".join("%15s" % x if x is not None else "		" for x
in t)) #Join the data by spaces, if there is no data, just write spaces
			datafile.write("\n") #New line
		datafile.close()
		outfile.write("\n") #New line
		if len(Offset)==0 or len(Offset)==1:
			outfile.write("Std Dev. %32.3f/%5.3f %15.3f %28.3f %28.3f %23.3f
%27.3s" % (xlavvik, wavvik, thavvik, fstressavvik, fstrainavvik, tmavvik, ofavvik)) #Write out the
standard deviations to the table
		else:
			outfile.write("Std Dev. %32.3f/%5.3f %15.3f %28.3f %28.3f %23.3f
%27.3f" % (xlavvik, wavvik, thavvik, fstressavvik, fstrainavvik, tmavvik, ofavvik)) #Write out the
standard deviations to the table
		outfile.close()
		os.chdir(original_folder) #Change directory to the original folder
def quit(event=None): #Quit button function
			root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=6, column=1) #Place the Run-button at row 6, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=6, column=2) #Place the EXIT-button at row 6, column 2
root.mainloop()
```

B.5 ASTM D 2344

B.5.1 Filen ”D2344test.py”

```
from Tkinter import *
import matplotlib.pyplot as plt
import os
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
top.pack()
#Variables attached to widgets
s=StringVar() #Specimen name variable
l=StringVar() #Support span variable
w=StringVar() #Width variable
```

```
t=StringVar() #Thickness variable
p=StringVar() #Path variable
f=StringVar() #Filename variable
standard=Label(top, text="ASTM D2344") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
specimen=Label(top, text="Material_Specimen # i.e. (TTF-1B_1:)") #Specimen box text
specimen.grid(row=4, column=0) #Place specimen box text at row 4, column 0
sp=Entry(top, textvariable=s) #Specimen entry, user input stored as variable s
sp.grid(row=4, column=1) #Place specimen entry box at row 4, column 1
length=Label(top, text="Support Span (mm) (i.e. 96):") #Length box text
length.grid(row=6, column=0) #Place length box text at row 6, column 0
leng=Entry(top, textvariable=l) #Length entry, user input stored as variable l
leng.grid(row=6, column=1) #Place length entry box at row 6, column 1
width=Label(top, text="Width (mm) (i.e. 2.3):") #Width box text
width.grid(row=8, column=0) #Place width box text at row 8, column 0
wid=Entry(top, textvariable=w) #Width entry, user input stored as variable w
wid.grid(row=8, column=1) #Place width entry box at row 8, column 1
thickness=Label(top, text="Thickness (mm) (i.e. '9.8'):") #Thickness box text
thickness.grid(row=10, column=0) #Place thickness box at row 10, column 0
th=Entry(top, textvariable=t)              #Thickness entry, user input stored as variable t
th.grid(row=10, column=1) #Place thickness entry box at row 10, column 1
paway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
paway.grid(row=12, column=0) #Place path box text at row 12, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=12, column=1) #Place path entry box at row 12, column 1
filename=Label(top, text="Optional: Filename: ASTM_D2344_") #Filename box text
filename.grid(row=14, column=0) #Place filename box text at row 14, column 0
file=Entry(top, textvariable=f) #Filename entry, user input stored as variable f
file.grid(row=14, column=1) #Place filename entry box at row 14, column 1
filetyp=Label(top, text=".pdat") #Filename end
filetyp.grid(row=14, column=2) #Place filename end at row 14, column 2

def comp():
        original_folder=os.getcwd() #Get the original folder
        S, L, W, T, P,F=s.get(), float(l.get()), float(w.get()), float(t.get()), p.get(), f.get() #Get
input from entry boxes
        os.chdir(P) #Change directory to the input directory
        infile=open("specimen.dat", "r")          #Open data file
        infile.readline() #Skip the first and second line
        infile.readline()
        lines=infile.readlines()
        dateline=lines[0] #Store the third line to get the date
        info=dateline.split()
```

```
date=info[-2] #Date
del lines[0] #Delete the line with the date from lines
infile.close() #Close file
names=lines[0].split()      #Store line with names of columns
del lines[0]   #Delete line with names from lines
Units=lines[0].split() #Store units
del lines[0] #Delete line with units from lines, now there are only numbers in lines
indexlist=[]
for j in range(len(names)):
        if names[j]=="Displacement" or names[j]=="Force" or
names[j]=="Time": #Find out the order of the columns
                indexlist.append(names[j])
mindex=indexlist.index("Displacement") #Find out which column has the respective
info
findex=indexlist.index("Force")
sindex=indexlist.index("Time")
Displacement=[] #Empty lists to store the data
Time=[]
Force=[]
for line in lines:          #Loop over the data, storing data in lists
        words=line.split() #Splits the line at the spaces
        Displacement.append(float(words[mindex])) #Append the number
corresponding to the index of the word Displacement in the name list
        Time.append(float(words[sindex])) #Append the number
corresponding to the index of the word Time in the name list
        Force.append(float(words[findex])) #Append the number
corresponding to the index of the word Force in the name list
if Units[findex]=="kN": #Multiply by 1000 if the force is given in kN instead of N, so
that Force is given in N.
        Force=1000*array(Force)
if Displacement[1] and Displacement[20]<0: ##Change sign of force and
displacement if the displacement at point 1 and 20 are negative
        Force=(-1)*array(Force)
        Displacement=(-1)*array(Displacement)
plt.plot(Displacement, Force)          #Plot force vs. displacement
plt.title("ASTM D-2344 %s" % S) #Plot title
plt.ylabel("N")          #name on y-axis
plt.xlabel("%s" % Units[mindex])      #name on x-axis
plt.grid(True) #Put a grid on the plot
Fmax=(array(Force)).max()          #Find max force
Sbs=0.75*Fmax/(T*W) #Short-beam strength, from equation (1) in standard
test="ASTM_D2344_"      #Beggining of filename
fi=".pdat"
```

```
            if F=="" or F=="No" or F=="no":
                        Strings=[test, S, fi]  #List to create filename if "filename"-entry is empty
            else:
                        Strings=[test,F, fi] #list to create filename
            Filename="".join(Strings) #creating filename
            outfile=open(Filename, "w") #Create a file in the folder where the data file is
            outfile.write("%s" % date) #Write out the test date
            outfile.write("\n") #New line
            outfile.write("Material_Specimen #     Length (mm)/Width (mm)   Thickness (mm)
Short-beam strength (MPa)    Max Force (N)")
            outfile.write("\n") #New line
            outfile.write("%19s %17.3f/%f %16.3f %29.3f %16.3f" % (S, L, W, T, Sbs, Fmax))
            #Write results to file
            outfile.write("\n") #New line
            outfile.write("   Displacement      Force")
            outfile.write("\n") #New line
            for i in range(len(Force)): #Write Displacement and Force table to the file
                        outfile.write("%14.8f   %14.8f" % (Displacement[i], Force[i]))
                        outfile.write("\n") #New line
            outfile.close()
            os.chdir(original_folder) #Change directory to the original folder
def quit(event=None): #Quit button function
                        root.destroy()

compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=16, column=1) #Place the Run-button at row 16, column 1
widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=16, column=2) #Place the EXIT-button at row 16, column 2
root.mainloop()
```

B.5.2 Filen ”D2344testtable.py”

```
import os
from numpy import sqrt
import matplotlib.pyplot as plt
import glob
from datetime import date
import itertools
from Tkinter import *
import sys
root=Tk() #root (main) window
top=Frame(root) #create frame
```

```
top.pack()
standard=Label(top, text="ASTM D2344") #GUI title
standard.grid(row=0, column=0) #Place the GUI title at row 0, column 0
p=StringVar() #path variable
pway=Label(top, text="Path (i.e. C:\Users\Ole\Testresultater):") #Path box text
pway.grid(row=4, column=0) #Place the path box text at row 4, column 0
pa=Entry(top, textvariable=p) #Path entry, user input stored as variable p
pa.grid(row=4, column=1) #Place the path entry box at row 4, column 1
def comp():
        P=p.get() #Get the path
        original_folder=os.getcwd() #Store the original folder
        os.chdir(P)   #Change directory to the given path
        today=date.today()          #Store today's date
        filelist=glob.glob("*.pdat") #List all files in the folder
        number=len(filelist) #Number of files/Number of tests
        Speci=[] #Empty lists to store the data to calculate mean value and standard
deviation
        Length=[]
        Width=[]
        Thick=[]
        Sbs=[]
        Maxf=[]
        outfile=open("ASTM_D2344_table.dat", "w") #Write the results to the file
"ASTM_D2344_table.dat"
        outfile.write("%s" % today) #Write out the date
        outfile.write("\n") #New line
        outfile.write("Material_Specimen #      Length (mm)/Width (mm)   Thickness (mm)
Short-Beam Strength (MPa)")
        outfile.write("\n") #New line
        ForcesDis=[] #List for Force vs. Displacement data
        for i in range(number): #Loop over the different files
                Displacement=[] #Empty lists to store the Displacement and Force data
from each test
                Force=[]
                infile=open(filelist[i], "r")
                lines=infile.readlines() #Read the lines in the specific file
                words=lines[2].split() #Line with table data
                infile.close()
                del lines[0] #deleting all the lines without force/displacement data
                del lines[0]
                del lines[0]
                del lines[0]
                for line in lines: #Appending displacement and force for plotting
```

```
                                        val=line.split()
                                        Displacement.append(float(val[0]))
                                        Force.append(float(val[1]))
                                ForcesDis.append(Displacement)
                                ForcesDis.append(Force)
                                ax = plt.subplot(111)
                                ax.plot(Displacement, Force, label=words[0]) #Name of the respective
plot is the input in "Specimen#" in the GUI
                                ax.legend(loc='center left',bbox_to_anchor=(1, 0.5)) #Move legend box
outside plot
                                ax.grid(True) #Put a grid on the plot
                                plt.xlabel("mm") #Name x-axis
                                plt.ylabel("N") #Name y-axis
                                plt.title("ASTM D-2344") #Plot title
                                legend()
                                Speci.append(words[0]) #Append the table values to their respective
list
                                Lenwid=words[1].split("/")
                                Length.append(float(Lenwid[0]))
                                Width.append(float(Lenwid[1]))
                                Thick.append(float(words[2]))
                                Sbs.append(float(words[3]))
                                Maxf.append(float(words[4]))
                                outfile.write("%19s %27s %16s %27s" % (words[0],words[1], words[2],
words[3])) #Write out the values in the table dat file
                                outfile.write("\n") #New line
                        meanlen=sum(Length)/len(Length) #Mean support span
                        meanwid=sum(Width)/len(Width) #Mean width
                        meanth=sum(Thick)/len(Thick) #Mean thickness
                        meansbs=sum(Sbs)/len(Sbs) # Mean short-beam strength
                        meanf=sum(Maxf)/len(Maxf) #Mean max force
                        outfile.write("Mean     %31.3f/%5.3f %16.3f %27.3f" % (meanlen, meanwid, meanth,
meansbs)) #Write out mean values to the table
                        n=number #number of values for each variable
                        xl=0 #Variables to calculate standard deviation
                        xw=0
                        xth=0
                        xsbs=0
                        xf=0
                        for i in range(number):
                                xl+=((Length[i]))**2 #Adding the squares of the different values for the
respective variable from the different tests
                                xw+=((Width[i]))**2
```

```
                        xth+=((Thick[i]))**2
                        xsbs+=((Sbs[i]))**2
                        xf+=((Maxf[i]))**2
                if number==1:
                        print "You need more than one resultfile from D2344test.py to calculate
statistics"
                        lenavvik=0
                        widavvik=0
                        thavvik=0
                        sbsavvik=0
                        favvik=0
                else:
                        lenavvik=sqrt((xl-n*(meanlen**2))/(n-1)) #Sample standard deviations.
From equation (3) in standard.
                        widavvik=sqrt((xw-n*(meanwid**2))/(n-1))
                        thavvik=sqrt((xth-n*(meanth**2))/(n-1))
                        sbsavvik=sqrt((xsbs-n*(meansbs**2))/(n-1))
                        favvik=sqrt((xf-n*(meanf**2))/(n-1))
                datafile=open("ASTM_D2344_data.dat", "w") #Create file "ASTM_D2344_data.dat"
to write in the displacement/force data from each test
                datafile.write("                                        ".join(Speci)) #Write out the specific test
names joined by spaces
                datafile.write("\n") #New line
                titnames=["     Displacement", "Force  "] #The names of the different columns
                coltit=n*titnames #Adding the names of all the columns i.e. "Displacement Force
Displacement Force ..." n times
                datafile.write("              ".join(coltit)) #Joining these names with spaces
                datafile.write("\n") #New line
                for t in itertools.izip_longest(*ForcesDis): #Write out the data as columns
                        datafile.write("            ".join("%15s" % x if x is not None else "              "
for x in t)) #Join the data by spaces, if there is no data, just write spaces
                        datafile.write("\n") #New line
                datafile.close()
                outfile.write("\n") #New line
                outfile.write("Std Dev. %31.3f/%5.3f %16.3f %27.3f" % (lenavvik,
widavvik,thavvik,sbsavvik)) #Write out the standard deviation values to the table
                outfile.close()
                os.chdir(original_folder) #Change directory to the original folder

def quit(event=None): #Quit button function
                        root.destroy()
```

```
compute=Button(top, text='Run', command=comp) #Button to run program, calls the function
comp()
compute.grid(row=6, column=1) #Place the Run-button at row 6, column 1

widget=Button(top, text='EXIT', command=quit) #Button to quit program, calls the function
quit()
widget.grid(row=6, column=2) #Place the EXIT-button at row 6, column 2

root.mainloop()
```